# Diversité par clustering pour la recherche d'images : étude expérimentale

Christian Antonio Kuoman Mamani




# THÈSE DE DOCTORAT DE L'UNIVERSITÉ PIERRE ET MARIE CURIE

École doctorale Informatique, télécommunications et électronique (Paris)
Laboratoire d'Informatique de Paris 6

pour obtenir le grade de :

**DOCTEUR DE L'UNIVERSITÉ PIERRE ET MARIE CURIE**
**Spécialité : INFORMATIQUE**

présentée par

Christian Antonio KUOMAN MAMANI

Titre de la thése :

# Diversité par Clustering pour la Recherche d'Images : Étude Expérimentale

**Soutenue le 31 août 2015**

devant le jury composé de :

| | |
|---|---|
| Matthieu CORD (LIP6 - UPMC, Paris) | Examinateur |
| Marcin DETYNIECKI (CNRS LIP6 - UPMC, Paris) | Directeur de thèse |
| Bogdan IONESCU (LAPI - UPB, Bucarest) | Rapporteur |
| Philippe MULHEM (CNRS LIG - UJF, Grenoble) | Rapporteur |
| Adrian POPESCU (CEA LIST, Paris) | Examinateur |
| Sabrina TOLLARI (LIP6 - UPMC, Paris) | Co-encadrante de thèse |

# Remerciements

Une thèse est un travail de longue haleine, nécessitant beaucoup d'investissement, du sérieux et surtout d'aide de personnes que je tiens tout particulièrement à remercier. Ainsi, en premier lieu, je tiens à remercier Sabrina Tollari et Marcin Detyniecki qui m'ont encadré durant ces trois années et m'ont apporté des conseils précieux. Ils ont su se montrer très patients et m'ont apporté la motivation nécessaire afin de mener à bien cet ambitieux projet.

Je tiens à remercier Philippe Mulhem et Bogdan Ionescu d'avoir accepté de rapporter mon travail de thèse. Je remercie également Matthieu Cord et Adrian Popescu pour leur participation à mon jury de thèse.

Mes plus grands remerciements s'adressent également à Cyril March qui a été aussi une personne indispensable pour le bon déroulement de ma thèse. Son aide, sa bonne humeur, sa gentillesse ont été en grande partie responsable du succès de ma thèse.

Je tiens à témoigner ma gratitude à Marie-Jeanne Lesot pour m'avoir aidé avec ses relectures et ses conseils.

La thèse n'aurait pas été ce qu'elle a été sans la présence de l'ensemble de l'équipe LFI et de l'équipe de Xilopix. Je tiens donc à leur exprimer ma gratitude et ma reconnaissance pour avoir partagé des moments inoubliables pendant ces trois années, pour ses encouragements et son soutien.

Un grand merci à ma famille qui m'a toujours soutenu et m'a poussé à être ambitieux. Je considère que c'est grâce à elle que j'en suis là. Je remercie enfin toutes les personnes que j'ai pu oublier et qui ont contribué à l'aboutissement de cette thèse.

# Sommaire

# Nomenclature

| | |
|---|---|
| $A_{i_1}$ | Ensemble des chemins de l'image $i_1$ |
| $A^j_{i_1}$ | Chemin de l'image $i_1$ dans l'univers $j$, $A^j_{i_1} = (a_{0,0}, a^j_1(i_1), \cdots, a^j_p(i_1))$ |
| $a^j_1(i_1)$ | Nœud racine dans l'univers $j$ de l'image $i_1$ |
| $a^j_p(i_1)$ | Feuille du chemin dans l'univers $j$ de l'image $i_1$ |
| $C$ | Ensemble des clusters $C = \{C_1, C_2, \cdots, C_k, \cdots, C_K\}$ |
| $C_k, C^{(i)}_k$ | le $k$ième cluster, le $k$ième cluster à l'itération $i$ |
| $D$ | Ensemble des documents $D = \{d_1, d_2, \cdots, d_i, \cdots, d_{n_D}\}$ |
| $d, d_i$ | Un document quelconque, le $i$ième document du corpus |
| $d^{C_k}_i$ | le $i$ ième document du $k$ième cluster |
| $\vec{d}$ | Le vecteur contenant les valeurs pour chaque descripteur de $d$ |
| $I$ | Nombre d'itérations |
| $i$,$i_1$ | Image, image $i_1$ |
| $J(i_1)$ | Ensemble des univers de l'image $i_1$ |
| $j$ | Univers |
| $K$ | Nombre de clusters |
| $m_X, \vec{m_X}$ | Medoïde des valeurs de $X$, vecteur medoïde de $X$ |
| $n_D$ | Nombre d'éléments dans l'ensemble $D$ |
| $p(i_1)$ | Profondeur de l'arborescence de l'image $i_1$ |
| $Q$ | Ensemble des requêtes |
| $q$ | requête |
| $sim$ | Mesure de similarité |
| $T$ | Espace de termes d'indexation $T = \{t_1, t_2, \cdots, t_j, \cdots, t_{n_T}\}$ |
| $t, t_j$ | Un terme quelconque, le $j$ième terme dans l'espace de termes d'indexation |
| $w_j(i)$ | Poids du terme $t_j$ dans le document $d_i$ |
| $\vec{x}, x_{1:n}$ | Le vecteur $\vec{x} = (x_1, x_2, ..., x_n)$ |
| $\delta$ | Mesure de distance |
| $\mu_X, \vec{\mu}_X$ | Moyenne des valeurs de $X$, vecteur moyen de $X$ |

## Résumé

Les moteurs de recherche d'images traditionnels offrent à l'utilisateur des résultats de plus en plus pertinents, mais, dans la plupart des cas, les résultats similaires ont tendance à se regrouper entre eux. L'utilisateur peut souhaiter retrouver des documents qui soient certes tous pertinents par rapport à sa requête, mais aussi qui soient différents les uns des autres.

Dans cette thèse, placée dans un cadre CIFRE avec l'entreprise Xilopix, nous considérons le problème de la diversité pour les systèmes de recherche d'images. Nous avons focalisé notre attention sur la diversité par l'exploitation du clustering, plus spécialement par une approche hiérarchique, car sa hiérarchie de clusters peut bien correspondre à la nature intrinsèquement hiérarchique de la diversité.

Pour pouvoir comparer les différentes approches existantes et aussi nos propositions, nous avons proposé un cadre théorique générique qui permet d'identifier précisément l'étape du processus différentiant. Côté expérimental, nous avons cherché à identifier des comportements les plus généraux possibles. En particulier, nous avons fait attention à ce que les résultats ne soient pas dépendants d'un benchmark, ni de certains cas particuliers. Ainsi, nous avons choisi de travailler sur trois benchmarks assez différents, en nombre de requêtes, d'images, en types de descripteurs et provenant de contextes différents.

Nous avons aussi exploité le cadre générique sur le benchmark MediaEval 2013. Ce qui nous a permis de montrer qu'incrémenter la pertinence par le texte, puis faire un clustering visuel est une bonne stratégie pour améliorer la diversité.

Les résultats expérimentaux montrent l'intérêt du clustering hiérarchique sur la diversité par rapport à des méthodes de clustering plat, mais aussi par rapport à une méthode de diversité par optimisation.

A l'intérieur du cadre des méthodes dites par clustering, nous proposons des solutions aux difficultés de son utilisation, comme sont le choix du nombre de clusters optimal et le réordonnancement des images à partir des clusters obtenus.

Côté descripteurs, nous avons aussi exploré les aspects hiérarchiques en étudiant l'utilisation d'une arborescence de concepts, car intuitivement une structure hiérarchique peut bien correspondre à la structure hiérarchique de la diversité. Ainsi, nous avons proposé un nouveau descripteur alternatif au texte ou visuel, basé sur l'utilisation de arborescence de concepts de Xilopix. De plus, nous avons développé des solutions afin d'utiliser cette approche dans tout autre méthode de diversité. Finalement nous avons montré expérimentalement l'intérêt d'utiliser le descripteur développé plutôt que l'arborescence de concepts directement.

Enfin, un autre résultat de notre thèse est le développement d'un prototype complet qui prends en compte les contraintes fortes de temps de calcul ce qui le rend adapté pour être utilisable dans le moteur de recherche de l'entreprise. Nous avons montré la robustesse du prototype en étudiant les temps de réponse en fonction du nombre d'images, de la dimension des descripteurs, du nombre de clusters et du nombre d'itérations.

## Abstract

Current search engines return relevant results but often the retrieved items are similar. Moreover, similar results tend to appear together. The user may be interested to find documents that are relevant and diverse at the same time.

In this thesis, carried out in the context of an industrial agreement (CIFRE) with the company Xilopix, we consider the problem of the diversity of the image retrieval systems. We have focused our attention on diversity by clustering, especially on an approach based on an agglomerative hierarchical clustering (AHC) because the hierarchy of clusters may well correspond to the hierarchical nature of the diversity.

In order to compare the different existing approaches and our propositions, we propose a theoretical framework that allows precisely identify steps making the difference. From an experimental point of view, we put a great effort on identifying the most general possible behaviors. In particular, in order to have results benchmark independent, we chose to work with three benchmarks that are different in terms of number of queries, of number of images, of descriptor types ; and that were collected in very different contexts

We also used our generic framework, on the Mediaeval 2013 benchmark, to explicit that an interesting strategy to improve diversity is to increase the relevance using the text, and then to exploit visual based clustering to diversify the results.

The experimental results show that a hierarchical exploitation of the results of the AHC increases the diversity in comparison with standard flat clustering methods and a method of diversity by optimization.

Moreover, for all clustering based approaches, we offer solutions to the difficulties which these approaches imply : First, how to obtain the ideal number of clusters in order to reflecting the right diversity ? Then, how to sort the images using the clusters ?

Furthermore, on the descriptors side we explored also a hierarchical property, based o the usage of a tree of concepts. Intuitively its hierarchical structure may well correspond to the hierarchical nature of the diversity. We proposed a novel approach for exploiting riche description resources, namely the Xilopix tree of concepts, as an alternative to text or visual description. In addition, we developed solutions to use this approach with any other methods for diversity. Finally, we showed experimentally that using the proposed descriptor is even better than using the tree of concepts directly.

Finally, one other result of this thesis is the development of a complete prototype that takes into account the strong constraints of response time, which makes it suitable to be used in the company's search engine. We have shown the robustness of the prototype in terms of response time by considering the number of images, the size of the descriptors, the number of clusters and the number of iterations.

# Chapitre 1

# Introduction

Actuellement, on assiste à une explosion du nombre d'images mises à la disposition du grand public à travers internet. Cette explosion est due à plusieurs facteurs comme le développement des outils numériques (appareils photos numériques, téléphones portables, scanners, etc.), de le partage des images grâce aux réseaux sociaux, etc. Pour une personne il est difficile de trouver l'information dont elle a besoin dans la grande masse d'informations disponible sur le web. Les systèmes de recherche d'information (SRI) ont généralement pour objectif d'aider les utilisateurs à retrouver les documents pertinents correspondants à leur besoin d'information exprimé sous la forme d'une requête. Cependant, dans la plupart des cas, les résultats pertinents sont similaires entre eux produisant ainsi des réponses répétitives qui ont tendance à se regrouper. Ce phénomène est particulièrement présent dans le cas de SRI orientés images.

Par exemple, dans la figure 1.1, si nous effectuons la requête "phare" dans un moteur de recherche en ligne, les premiers résultats fournis sont quasiment tous des images correspondant à des phares dans la mer par une belle journée ensoleillée. Un utilisateur devra parcourir de nombreux résultats afin de trouver des phares de voiture. De plus, même en ce qui concerne les phares (au sens système de signalisation), il existe différents niveaux de catégories de phares : les phares maritimes, les phares aéronautiques... qui se trouvent de plus dans différents types de scènes : de jour, de nuit, en pleine tempête... Pour éviter à l'utilisateur de parcourir de nombreux résultats dans la liste pour trouver le type d'images qu'il recherche, l'idéal serait d'obtenir une liste d'images qui soient les plus pertinentes et diverses à la fois.

Dans ce chapitre introductif, nous précisons d'abord la notion de diversité ainsi que le contexte de ce travail, avant de donner une vision synthétique de nos contributions et de présenter le plan général de la thèse.

## 1.1 Diversité

Les SRI permettent de retrouver un grand nombre d'images. L'utilisateur peut être intéressé par retrouver des documents qui soient certes tous pertinents par rapport à sa requête, mais aussi qui soient différents les uns des autres. Nous appelons **sous-thèmes** les différents aspects des résultats d'une requête. Le **problème de la diversité** peut être défini comme le fait d'essayer de retrouver les documents qui couvrent le plus de sous-thèmes possibles pour une requête donnée.

Par exemple, dans le cas de la recherche d'images, la requête "phare" peut avoir au moins deux sous-thèmes : "phare maritime" et "phare de voiture" (voir figure 1.1). Mais, la même requête peut avoir aussi d'autres sous-thèmes qui correspondent à différents types de scènes : "de jour", "de nuit", "en pleine tempête", etc.

Dans la notion de diversité, on distingue deux manières de voir la chose :

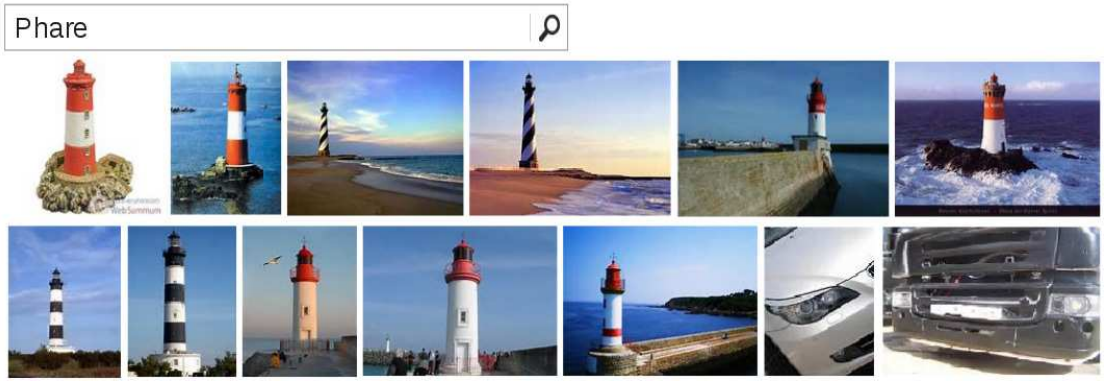



FIGURE 1.1 – Exemple du problème de la diversité avec la requête "Phare". Les images résultats ne sont pas diversifiées ; elles présentent beaucoup d'images de "phare maritime" et peu d'images de "phare de voiture"

— **la nouveauté** : les documents qui apportent des nouvelles informations sont intéressants pour l'utilisateur.
— **la redondance** : les documents redondants ne sont pas intéressants pour l'utilisateur.

Dans cette thèse, nous nous intéressons particulièrement à la diversité dans le cas de la recherche d'images qui est un cas particulier de diversité en recherche d'information.

### 1.1.1 Pourquoi diversifier les résultats ?

Montrer à l'utilisateur la diversité d'une requête donnée peut être intéressant dans les cas suivants.

**Répondre aux besoins de différents utilisateurs** Le principe de la plupart des moteurs de recherche est d'ordonner les documents en fonction de leur similarité à la requête. Cependant, cette approche ne prend pas en compte le fait qu'une requête peut avoir plusieurs sous-thèmes. Plusieurs utilisateurs peuvent poser la même requête, mais rechercher des documents différents. Dans ce cas, le système classique ne peut pas répondre aux besoins de tous les utilisateurs, car il retourne souvent des documents du sous-thème majoritaire de la requête. Dans ce cas, montrer la diversité (ou les sous-thèmes) de la requête peut mieux répondre aux besoins de différents utilisateurs.

**Réduire l'ambiguïté d'une requête** Bien souvent l'utilisateur ne sait pas bien exprimer son besoin d'information et émet des requêtes larges qui contiennent plusieurs sous-thèmes. Dans ce cas, montrer la diversité de la requête peut être intéressant afin de :

— aider à l'utilisateur à préciser sa recherche ;
— montrer les sous-thèmes disponibles dans la collection de documents ;
— traiter les requêtes ambiguës ;
— ne pas imposer à l'utilisateur de reformuler sa requête avec d'autres termes.

**Réduire le nombre de documents retournés** En général, les SRI classiques essaient de maximiser le nombre de documents pertinents retrouvés. Cependant, l'utilisateur n'est pas forcement intéressé par le fait de retrouver tous les documents pertinents pour cette requête, il peut lui suffire de trouver quelques documents bien choisis. De plus,

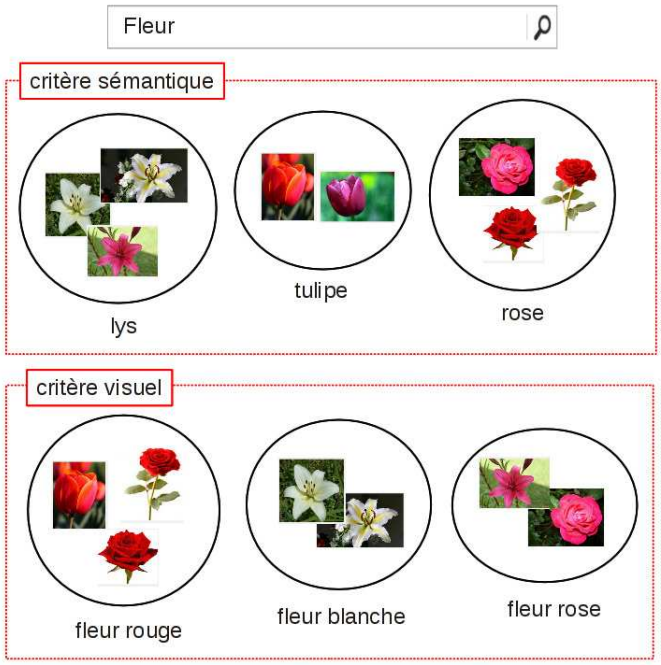



FIGURE 1.2 – Exemple des différents types de diversité avec la requête "fleur". On peut séparer les images par un critère sémantique en " lys", "tulipe", "rose" ou bien aussi séparer les images par un critère visuel en "fleur rouge", "fleur blanche", "fleur rose"

dans certains cas, retourner de nombreux documents pertinents peut réduire les chances de l'utilisateur de trouver les documents dont il a besoin. Dans ce cas, montrer la diversité de la requête en donnant un aperçu dès les premiers résultats de différents sous-thèmes de la requête peut être intéressant pour l'utilisateur, particulièrement quand il y a beaucoup de résultats pertinents.

**Éviter de montrer dès les premiers résultats des documents similaires** Les SRI classiques supposent que la pertinence d'un document est indépendante des autres documents. Cependant, un document peut être inutile à l'utilisateur s'il a déjà vu un similaire (avec un contenu similaire). Dans ce cas, montrer la diversité de la requête peut éviter de montrer dès les premiers résultats des documents semblables à ceux que l'utilisateur a déjà vus.

### 1.1.2 Critères de diversité

Une requête peut être diversifiée par différents critères de diversité. Ainsi dans le cas de la recherche d'images la requête peut être diversifiée par :

- critère **visuel** : séparer les images par la couleur, la forme ou séparer les images de jour ou de nuit, à l'intérieur ou à l'extérieur, etc.
- critère **géographique** : séparer les images provenant d'un pays différent ou d'une ville différente, etc.
- critère **social** : séparer les images par le nombre de personnes qui ont aimé ou non les images, par la date des images, par l'auteur, etc.
- critère **sémantique** : séparer les images possédant différents sens. Par exemple, en fonction d'une classification biologique (des animaux, des végétaux, etc.), des objets, etc.

La figure 1.2 montre un exemple de deux critères possibles de diversité dans la recherche d'images avec la requête "fleur". La requête peut être diversifiée par un critère


FIGURE 1.3 – Exemple des niveaux de diversité avec la requête "phare". Dans un premier niveau on peut séparer les images en "phare maritime" et "phare de véhicule", puis dans un deuxième niveau on peut séparer les images de "phare de véhicule" en "phare de voiture" et "phare de camion"

sémantique en séparant les images de "rose", les images de "tulipe", les images de "lys", etc. Cependant, la même requête peut être aussi diversifiée par un critère visuel en séparant les images de "fleur rouge", les images de "fleur blanche", etc.

Il faut noter que certains types des critères sont plus appropriés pour certains types de requêtes. Par exemple, dans la figure 1.2, on peut penser que pour la requête "fleur", faire une diversité sémantique peut être plus utile à la plupart des utilisateurs par rapport à une diversité visuelle et encore plus utile par rapport à une diversité par le texte.

### 1.1.3 Niveaux de diversité

Une requête peut être aussi diversifiée par différents niveaux de diversité où chaque niveau peut avoir une diversité plus ou moins spécifique à la requête. Par exemple, dans le cas de la recherche d'images, la figure 1.3 montre deux niveaux de diversité avec la requête "phare" en utilisant un critère sémantique. Dans cette figure, on voit que dans un premier niveau, la requête "phare" peut être diversifiée en séparant les images de "phare maritime" et les images de "phare de véhicule". Puis, dans un deuxième niveau, plus spécifique, la requête peut être diversifiée, par exemple, en séparant les images de "phare de véhicule" en "phare de voiture", "phare de camion", etc.

### 1.1.4 Méthodes de diversité

Dans la littérature, il existe différentes approches pour diversifier une requête. Un grand nombre d'entre elles s'appuient sur le partitionnement, tandis que les approches basées sur l'optimisation essayent de maximiser la diversité, ou simultanément la diversité et la pertinence.

**Diversité par optimisation** Le problème de la diversité peut être défini comme un problème d'optimisation où un document candidat à être choisi est divers s'il obtient une similarité minimale aux documents déjà choisis par rapport aux autres documents candidats non encore choisis.

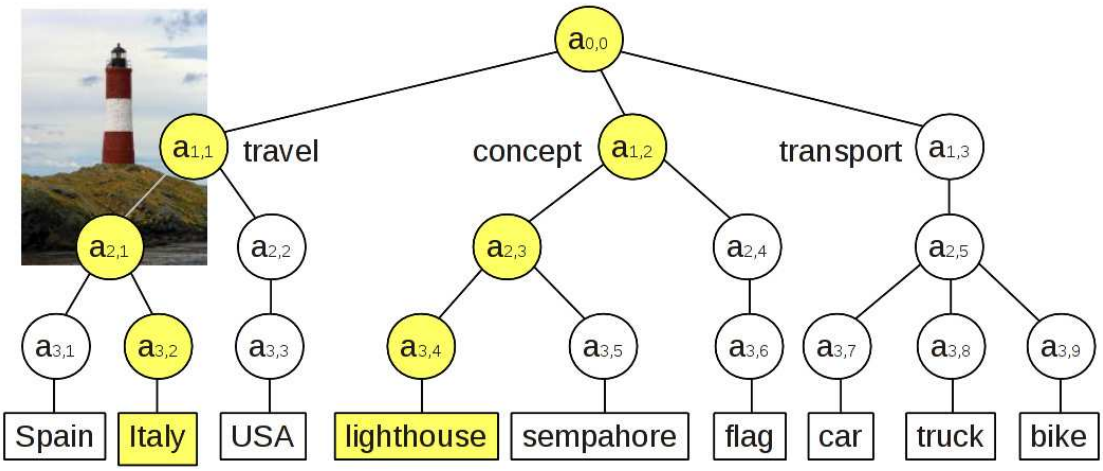



FIGURE 1.4 – Représentation graphique d'un exemple d'arborescence de concepts. L'image de gauche est associée à deux concepts : "Italy" et "lighthouse"

**Diversité par partitionnement** Une autre manière de diversifier les résultats consiste à utiliser une méthode de partitionnement, par exemple du clustering, qui permet de regrouper une liste de documents en générant plusieurs clusters. On fait alors l'hypothèse que les clusters retrouvés correspondent aux sous-thèmes de la requête ce qui n'est pas forcément le cas. Lorsqu'un utilisateur fait une requête, le SRI ne connaît pas le nombre de sous-thèmes correspondant, or de nombreux algorithmes de clustering demandent de connaître le nombre de clusters à générer. Une difficulté d'utilisation des méthodes de diversité par partitionnement est donc de déterminer le nombre de clusters qu'il faut choisir pour retrouver le maximum de sous-thèmes dans les premiers résultats.

## 1.2 Contexte

Ce travail de recherche s'est déroulé dans le contexte d'une thèse de type CIFRE (Conventions Industrielles de Formation par la REcherche) en collaboration entre l'entreprise Xilopix [1] et le laboratoire LIP6 [2] (Laboratoire d'Informatique de Paris 6).

Xilopix est une entreprise qui a développé un moteur de recherche d'images en ligne où les images sont décrites par différents types de descripteurs : des descripteurs textuels, visuels, d'autres provenant d'une arborescence de concepts, etc.

**Arborescence de concepts de Xilopix** L'un des points forts de cette entreprise est que les images sont associées à une arborescence de concepts (thésaurus). Contrairement à d'autres travaux où l'arborescence de concepts est construite automatiquement, celle de Xilopix a été construite manuellement par les documentalistes de l'entreprise. Cette arborescence est donc bien adaptée pour la recherche d'images. De plus, cela évite de rencontrer certains problèmes d'incompatibilité entre les différents niveaux de profondeur qui existent souvent dans les arborescences construites automatiquement.

L'arborescence de concepts Xilopix est composée en plusieurs univers (faune, flore, voyage, personne, etc.) où chaque univers a une structure d'arbre. Une image peut être associée à un ou plusieurs concepts. Par exemple, l'image de la figure 1.4 est associée

1. http ://fr.xilopix.com
2. http ://www.lip6.fr

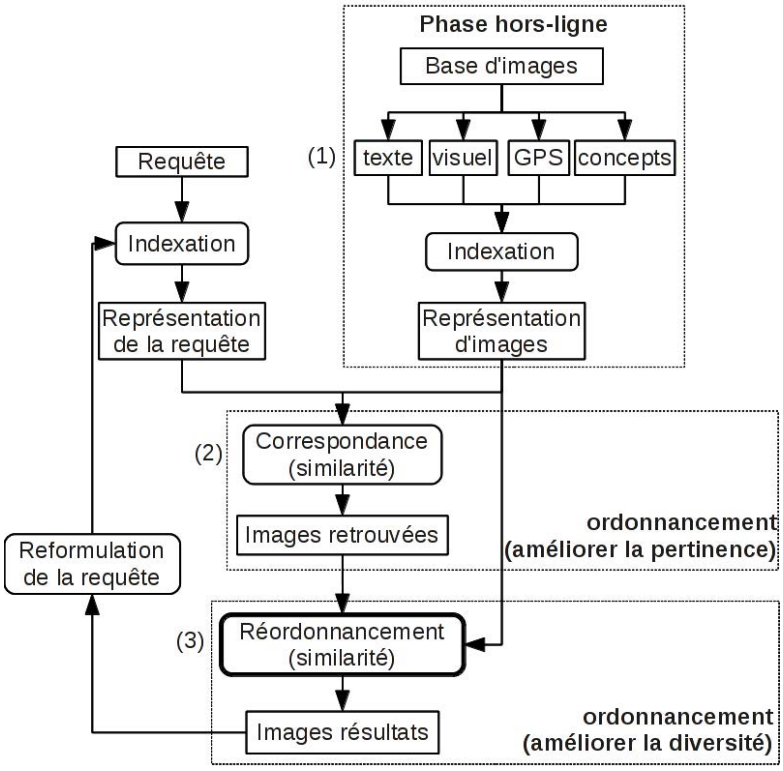



FIGURE 1.5 – Schéma d'un système proposé de recherche d'images avec une étape de réordonnancement des images afin d'augmenter la diversité

à deux concepts : le concept "Italy" qui appartient à l'univers "Travel" et le concept "Lighthouse" qui appartient à l'univers "Concept".

## 1.3 Proposition

Les travaux de thèse que nous avons réalisés se déclinent selon 4 axes concernant l'architecture du système, le cadre théorique de notre approche proposée, la prise en compte d'un descripteur basé sur une arborescence de concepts et une étude expérimentale, comme détaillé ci-dessous.

### 1.3.1 Architecture de l'approche proposée en termes de SRI

Afin de résoudre le problème de la diversité dans un système de recherche d'images (SRIm), nous proposons le schéma de la figure 1.5 : comme un système classique, il comporte une étape hors-ligne (représentée par le bloc (1)), qui indexe et représente les images, puis une étape de mise en correspondance avec la représentation de la requête par un calcul de similarité (bloc (2)). Nous ajoutons ensuite une étape supplémentaire de diversité par réordonnancement (représentée par le bloc (3)). Dans cette nouvelle étape, nous étudions différentes méthodes de diversité par réordonnancement des images retrouvées par le système, qui soient capables de générer une liste d'images diversifiées et puissent ainsi fournir un résultat qui répond mieux aux besoins de différents utilisateurs.

Dans notre schéma proposé, nous différencions deux étapes :

— **Ordonnancement pour améliorer la pertinence (2)** : cette étape provient de l'étape classique d'un SRI où le système renvoie la liste des images les plus similaires à la requête.

— **Ordonnancement pour améliorer la diversité (3)** : dans cette nouvelle étape nous étudions diverses méthodes qui réordonnent les images retrouvées dans l'étape précédente (2) afin de retourner une liste d'images la plus diversifiée possible.

L'intérêt de cette proposition vient de ce qu'elle permet de prendre en compte les aspects de la diversité suivants.

**Adaptabilité à la diversité de chaque requête** De manière générale, la diversité d'une requête possède des caractéristiques différentes par rapport à celle d'une autre requête. Par exemple, la granularité de diversité est souvent différente d'une requête à une autre. Dans ce cas, notre schéma proposé nous permet de mieux nous adapter à la diversité de chaque requête, car la méthode de diversité est exécutée après la mise en correspondance par similarité à la requête, en une phase de post-traitement.

**Prise en compte de différents critères de diversité** Dans un SRIm, une requête peut être diversifiée par différents critères comme décrit dans la section 1.1.2. En sachant qu'un utilisateur recherche des images représentées par un critère de diversité en particulier, l'idéal serait de pouvoir adapter ce dernier à chaque requête.

Cependant, le système ne peut pas savoir a priori le critère de diversité qui correspond au besoin de l'utilisateur.

Pour cette raison, dans cette thèse nous utilisons le même critère de diversité pour diversifier la requête, mais nous étudions l'influence de différents types de descripteurs sur la diversité afin d'identifier quel type de critère de diversité répond mieux aux besoins de la plupart des utilisateurs. Nous considérons en particulier les descripteurs textuels, visuels, géographiques et un descripteur basé sur une arborescence de concepts, comme indiqué dans le bloc (1) de la figure 1.5.

**Influence de la pertinence initiale sur la diversité** Dans la plupart des cas, diversifier directement les images résultats d'une requête fait diminuer le score de pertinence. En effet, le fait de montrer dès les premiers résultats les images les plus diverses risque aussi de ramener des images non-pertinentes à la requête. C'est pour cette raison que nous proposons un schéma qui sépare les 2 étapes de pertinence et de diversité (blocs (2) et (3) de la figure 1.5). Elle permet d'étudier dans quelle mesure l'augmentation de la pertinence initiale permet aussi d'augmenter la diversité.De plus, ce schéma nous permet d'étudier deux cas d'utilisation de la diversité : le cas idéal où toutes les images retrouvées sont pertinentes et le cas réel où parmi les images retrouvées un certain nombre est non-pertinentes.

### 1.3.2 Cadre théorique de l'approche proposée : diversité par clustering hiérarchique

Comme évoqué dans la section 1.1.4, un grand nombre des méthodes de diversité existantes s'appuient sur 2 approches : les approches par optimisation et les approches par clustering. Parmi ces méthodes de diversité, nous choisissons les méthodes de clustering, car le regroupement des images en plusieurs clusters peut nous donner un aperçu des sous-thèmes que peut avoir la requête.

Parmi les méthodes de clustering, nous choisissons une méthode de clustering hiérarchique, car elle s'adapte mieux au problème de la diversité que nous pensons aussi hiérarchique (car une requête peut avoir différents niveaux de diversité). L'originalité de notre approche est d'exploiter une structure hiérarchique de clusters, obtenus par la `AHC`.

L'intérêt de cette approche est l'adaptation naturelle aux niveaux de diversité introduits plus haut.

De plus, pour montrer l'intérêt du clustering hiérarchique sur la diversité, nous comparons cette méthode avec une méthode classique de clustering plat et pour rendre cette étude plus générale, nous comparons les méthodes de clustering avec une méthode de diversité par optimisation qui est une approche totalement différente de l'approche par clustering.

Il existe de nombreux travaux dans le domaine de la diversité. Certains travaux ont déjà travaillé avec les algorithmes de clustering que nous utilisons, cependant l'apport de cette thèse est d'avoir cherché à obtenir des résultats qui ne soient pas dépendants d'un benchmark. Pour cela, nous avons volontairement choisi de travailler sur trois benchmarks assez différents en nombre de requêtes, en nombre d'images, en type de descripteurs... et provenant de sources différentes.

Les défis posées par cette approche sont les suivants :

— Utiliser du clustering sur la diversité n'est pas vraiment évident à faire, car il y a un certain nombre de difficultés qui doivent être prises en compte pour obtenir des résultats optimaux. Par exemple, dans le clustering, comment choisir le nombre optimal de clusters pour chaque requête ? Comment réordonner les résultats après le clustering ? Le clustering ne fournit pas d'ordre ni sur les clusters ni sur les images, comment déterminer leur ordre et celui des images pour améliorer la diversité des résultats ? Pour les résoudre, nous proposons quelques solutions à ces problèmes (détaillés dans le chapitre 5).

— Dans un SRIm, une requête peut être diversifiée par différents niveaux de diversité où chaque niveau peut avoir une diversité plus ou moins spécifique à la requête. En sachant que l'utilisateur recherche des images qui sont dans un niveau spécifique de la diversité, l'idéal serait de choisir une méthode de diversité qui puisse mieux répondre ou s'adapter à ce problème. Dans les méthodes de clustering, les niveaux de diversité peut correspondre aux nombres de clusters. Cependant, le système ne peut pas savoir a priori le niveau de diversité idéal pour l'utilisateur et, donc, les méthodes de clustering ne peuvent pas savoir le nombre de clusters idéal pour chaque requête. Pour le résoudre, nous pensons que la méthode de clustering hiérarchique s'adapte mieux à ce problème, car sa hiérarchie des clusters peut bien correspondre aux différents niveaux de granularité de la diversité ce qui permet à l'utilisateur de changer facilement ce niveau sans avoir besoin de relancer la méthode.

### 1.3.3 Mise en œuvre pour des données complexes : descripteur basé sur une arborescence de concepts

De par le cadre de la CIFRE chez Xilopix, nous avons considéré des images, qui offrent une large gamme de descripteurs (des descripteurs visuels, des descripteurs textuels, des coordonnées GPS).

De plus, grâce à cette entreprise, nous disposons d'une arborescence de concepts (décrite plus haut) qui pour nous est une ressource très riche et intéressante pour augmenter la diversité, car sa structure hiérarchique peut se rapprocher de la nature intrinsèquement hiérarchique de la diversité.

L'originalité de notre approche est d'utiliser une arborescence de concepts, et plus particulièrement celle de Xilopix, comme descripteur alternatif au texte ou visuel dans nos méthodes proposées afin d'améliorer la pertinence et la diversité des résultats. De plus, peu de travaux se sont intéressés à l'utilisation de cette ressource comme descripteur de l'image pour améliorer la diversité dans le SRIm ce qui peut être une nouvelle approche de grande valeur à la communauté scientifique.

Les défis posées par cette approche sont les suivants :

— L'utilisation de ce type d'arborescence comme descripteur de l'image pose de nombreux problèmes. Par exemple, comment faire la correspondance entre les termes de la requête et les concepts dans l'arborescence ? Comment déterminer la similarité entre deux images décrites par plusieurs concepts ?, etc. Pour les résoudre, nous proposons quelques solutions à ces problèmes (détaillés dans le chapitre 5).

— Dans la littérature, il existe des ressources publiques (comme par exemple ImageNet [12]) où les images sont associées a une ontologie. Cependant, nous n'avons pas trouvé un corpus public où les images sont associées à plusieurs concepts de l'ontologie et qui soit annoté manuellement pour évaluer la diversité. Pour résoudre cette difficulté, nous avons construit, avec l'aide des documentalistes de l'entreprise Xilopix, le benchmark XiloDiv. Ce benchmark est composé d'images indexées manuellement par plusieurs concepts de l'arborescence de Xilopix. Ces images sont également annotées manuellement par les documentalistes pour permettre l'évaluation de la pertinence et la diversité.

### 1.3.4 Étude expérimentale comparative pour la diversité

Dans cette thèse, nous avons mis au point des protocoles expérimentaux afin d'étudier et comparer nos méthodes proposées pour la diversité. De plus, dans ce cadre expérimental, nous faisons une comparaison exhaustive avec toutes sortes de méthodes de diversité afin de comparer différentes approches de diversité dans les mêmes conditions expérimentales.

Il faut remarquer que faire ce type d'étude expérimentale est difficile car le comportement de la diversité peut être influencé par différents facteurs comme le nombre d'images, le type de descripteur, l'approche de la méthode, le nombre de clusters, etc. Pour cela, nous prenons en compte un certain nombre de difficultés qui sont :

— Certains travaux montrent des résultats qui sont dépendants d'un benchmark ce qui donne des comportements spécifiques à certains cas particuliers et qui peuvent fausser le vrai comportement des méthodes sur la diversité. Pour résoudre cette difficulté, nous avons choisi de travailler sur trois benchmarks assez différents, en nombre d'images, en type de descripteurs... afin d'obtenir des comportements les plus généraux possibles.

— Est très rare de trouver dans la littérature des benchmarks publics disponibles qui disposent des mêmes descripteurs permettant de comparer de manière globale les résultats obtenus dans ces benchmarks. Pour résoudre cette difficulté, nous avons généré un type de descripteur commun (détaillé dans l'annexe B) sur tous les benchmarks afin de comparer les résultats obtenus sur les trois benchmarks.

Outre ces contributions théoriques et expérimentales, notre thèse a pour résultats un prototype complet, en cours de déploiement dans le moteur de recherche de l'entreprise avec en particulier prise en compte des contraintes fortes de temps de calcul. En effet, la problématique industrielle du temps de calcul consiste à fournir une réponse rapide à une requête dans une application de recherche d'images en ligne. Dans ce cas, nous avons pris en compte un certain nombre des contraintes qui peuvent influencer le temps de calcul et la diversité. Par exemple, quel est le bon choix du nombre d'images que les méthodes de diversité doivent prendre en compte afin d'augmenter la diversité sans trop négliger son temps de calcul ? Pour les méthodes de clustering, le nombre de clusters influe-t-il beaucoup sur ce temps ?

## 1.4 Plan de la thèse

Ce document est composé de 2 parties principales complétées par 3 annexes.

**Partie I** Afin d'étudier les avantages et les inconvénients des méthodes classiques de diversité, dans une première partie nous décrivons un état de l'art de ces sujets abordés dans cette thèse.

Dans le chapitre 2, nous décrivons la recherche d'information, plus spécifiquement la recherche d'images, les différents descripteurs utilisés pour représenter une image et les mesures classiques de similarité. Ces descripteurs représentent les différents critères de diversité (critère visuel, textuel, de géolocalisation, sémantique) dans un SRIm. Nous décrivons aussi les métriques utilisées pour évaluer la pertinence et la diversité dans une liste d'images résultats.

Dans le chapitre 3, nous décrivons différentes méthodes classiques de clustering.

Dans le chapitre 4, nous présentons un état de l'art de différentes méthodes de diversité, en distinguant les approches basées sur l'utilisation du clustering et celle basée sur l'optimisation.

**Partie II** Dans la deuxième partie, nous présentons nos propositions.

Dans notre schéma proposé, nous ajoutons une nouvelle étape de diversité par réordonnancement et dans cette étape nous choisissons l'étude des méthodes de clustering sur la diversité. Dans le chapitre 5, nous détaillons cette étape supplémentaire : nous présentons d'abord notre cadre général étudié afin de nous placer dans un contexte spécifique de la diversité ; puis, nous décrivons la chaîne de traitement que nous avons développée afin d'améliorer la diversité avec le clustering tout en conservant une pertinence acceptable. Dans cette chaîne, nous décrivons les défis rencontrés ainsi que les solutions proposées de l'utilisation des méthodes de clustering sur la diversité et de l'utilisation d'une arborescence de concepts comme descripteur sur les méthodes de clustering. Enfin, nous présentons un cadre expérimental où nous citons les différents benchmarks utilisés ainsi que les différents cas d'étude de la diversité.

Dans le chapitre 6, nous décrivons les contextes ou les challenges retrouvés sur chaque benchmark utilisé pour améliorer la diversité, par exemple, l'intérêt d'utiliser une arborescence de concepts comme descripteur de l'image afin d'améliorer la diversité par rapport à l'utilisation directe de l'arborescence et à une approche visuelle.

Dans les chapitres 7 et 8, nous étudions deux méthodes classiques de clustering : une méthode de clustering hiérarchique `AHC` (chapitre 7) et deux méthodes de clustering plat : `K-Means` et `K-Medoids` (chapitre 8). Dans ces chapitres, nous montrons aussi le comportement de la valeur maximale de diversité sur les trois méthodes de clustering qui est lié au choix du nombre de clusters. Ce comportement de la valeur maximale est intéressant parce qu'il pourrait répondre à la difficulté du choix du bon nombre de clusters à adopter afin d'obtenir la valeur maximale de diversité.

Afin de rendre cette étude encore plus générale, dans le chapitre 9, nous comparons les méthodes de diversité par clustering avec une méthode de diversité par optimisation (`Maxmin`). De plus, dans ce chapitre, nous étudions aussi les contraintes fortes du temps de calcul sur les méthodes de diversité afin d'utiliser notre prototype développé dans un moteur de recherche en ligne d'une entreprise.

Dans le chapitre 10, nous concluons sur le travail réalisé.

Dans les annexes, nous décrivons d'abord les différentes expérimentations supplémentaires que nous avons faites. Puis, nous décrivons la génération d'un histogramme de couleurs. Enfin, nous donnons un aperçu de la structure du logiciel que nous avons

développé pour résoudre le problème de la diversité en prenant en compte la problématique industrielle du temps de calcul et les difficultés précitées dans cette thèse. Ce logiciel a été développé dans le langage utilisé dans l'entreprise et il est disponible pour être utilisable dans le moteur de recherche de l'entreprise.

# Première partie

# État de l'art

# Chapitre 2

# Recherche d'information

Actuellement, on assiste à une explosion du nombre d'images mises à disposition du grand public à travers internet. Cette explosion est due à plusieurs facteurs comme notamment ou entre autres : le développement des outils numériques (appareils photos numériques, téléphones portables, scanners, etc.), l'augmentation du stockage, l'ampliation des performances de transfert des réseaux, le partage des images grâce aux réseaux sociaux, etc. Les systèmes de recherche d'information sont des outils qui permettent à l'utilisateur de retrouver des images qu'il recherche parmi la grande masse d'information disponible.

Les systèmes de recherche d'information (SRI) comportent deux phases principales : une phase d'indexation et une phase de recherche (et éventuellement une phase de reformulation de la requête). Le processus commence quand un utilisateur formule son besoin d'information sous la forme d'une requête. Dans la **phase d'indexation**, la requête et les documents sont indexées par le système. L'indexation permet de construire le descripteur de la requête et le descripteur de l'ensemble de documents. Dans la **phase de recherche**, le système mesure la correspondance ou la similarité entre le descripteur de la requête et le descripteur de chaque document. Enfin, le système renvoie une liste de documents résultats considérés comme pertinents par rapport à la requête.

Les SRI traditionnels offrent à l'utilisateur des résultats de plus en plus pertinents, mais, dans la plupart des cas, les résultats similaires ont tendance à se regrouper entre eux. L'utilisateur peut être intéressé de retrouver des documents qui soient certes tous pertinents par rapport à sa requête, mais aussi qui soient différents les uns des autres (problème de la diversité). Pour résoudre le problème de la diversité, nous utilisons des méthodes de réordonnancement afin diversifier les résultats du SRI traditionnel.

Dans cette thèse, nous étudions aussi l'influence de différents types de descripteurs (du texte, du visuel, etc.) sur la diversité. De plus, nous proposons l'utilisation d'une arborescence de concepts comme descripteur pour augmenter la diversité. Dans ce chapitre, nous détaillons la génération de ces descripteurs à partir de l'information disponible sur l'image, quelques mesures de similarités classiques adaptées à ces descripteurs et les avantages et les difficultés d'utiliser ces descripteur dans la recherche d'images.

Dans la partie 2.1. nous décrivons de manière générale le système de recherche d'images. Dans la partie 2.2, nous décrivons les différents modèles classiques qui utilisent l'information textuelle disponible dans l'image pour la recherche d'images. Puis, dans la partie 2.3, 2.4 et 2.5 nous décrivons la recherche d'images par des autres attributs de l'image telle que l'information visuelle, les informations de géolocalisation associées qui lui sont attachées et une approche conceptuelle (par une arborescence de concepts) ainsi que la description de quelques mesures de similarités classiques. Enfin, dans la partie 2.6 nous citons quelques mesures classiques d'évaluation pour mesurer le problème de la pertinence et de la diversité.

## 2.1 Système de recherche d'images (SRIm)

Dans cette thèse nous nous plaçons dans le cadre de la recherche d'images en ligne qui a la contrainte du temps de calcul, car la réponse à une requête doit être rapide.

Sur le web, il existe différents types de SRIm : Le SRIm par le texte où les images sont indexées à partir du texte associé à chaque image et où la requête est composée de quelques mots-clés. Le SRIm par le contenu visuel où les images sont indexées à partir des attributs visuels de l'image (comme la couleur, la texture ou la forme) et la requête qui est composée sous la forme d'une image. Actuellement, les SRIm par le texte, sont souvent utilisées car ils obtiennent souvent des meilleurs résultats en pertinence. La raison c'est parce que le texte donne une signification sémantique à l'image, contrairement aux images qui donnent une signification plutôt numérique, alors que les utilisateurs sont intéressés par leur contenu sémantique.

## 2.2 Recherche d'images par le texte

Une procédure classique de la recherche d'une image se fait ou procède par une approche basée sur l'information textuelle. Dans ce cas, la requête est exprimée par des mots-clés.

Dans la phase d'indexation, on prend l'information textuelle associée à l'image pour générer le descripteur de l'image. Puis, dans la phase de recherche on calcule la similarité entre le descripteur de chacune d'entre elles et le descripteur de la requête afin d'ordonner les images en fonction de son score de similarité à la requête.

Dans cette section, nous détaillons d'abord l'information textuelle associée à une image. Puis, nous décrivons des modèles classiques par le texte qui diffèrent principalement de la façon dont ils représentent les images et de la façon de mesurer la correspondance entre la requête et chaque image.

### 2.2.1 Information textuelle

Dans la base d'image, chacune est souvent annotée par un ensemble de mots-clés (information textuelle). Cette information provient d'une annotation produite manuellement ou automatiquement. Dans le cas d'une annotation automatique, les images risquent d'être mal annotées, car normalement le texte associé provient de pages du web, de revues, etc. Ce texte n'est pas forcément pertinente pour l'image, ce qui occasionne, dans la plupart de cas, le problème de la pertinence. Dans le cas d'une annotation manuelle, les images sont mieux annotées que par une annotation automatique, car celle-ci est souvent réalisée par des documentalistes qui sont des experts dans la recherche d'information. Même si une image est bien annotée, cela n'assure pas que l'image soit pertinente à la requête, car l'utilisateur pourrait utiliser un autre mot (synonyme) pour chercher cette image. Par exemple, si une image est annotée avec le mot "voiture", l'utilisateur ne trouvera pas cette image s'il exprime sa requête en utilisant des mots synonymes comme "automobile" ou "véhicule".

Un autre problème est celui des homonymies. Exemple : avocat le fruit ou l'homme de loi. Même si les images sont bien annotées, le SRI ne peut pas savoir si la requête est le fruit ou l'homme de loi. Il y a aussi des problèmes avec les expressions composées de plusieurs mots qui font que, suivant le modèle utilisé, les résultats présentés n'ont pas de rapport avec la requête, même s'ils contiennent bien les mots de la requête.

Dans la recherche d'information, le texte associé à chaque image passe souvent par un processus de tokenization (aussi appelé d'indexation) qui consiste à enlever les mots-clés qui n'apportent rien à la recherche, par exemple des pronoms, des articles, etc.

Une procédure classique de tokenization est composée principalement en trois étapes : d'abord on recherche tous les mots-clés associés aux images dans la base d'images, puis on supprime les mots-clés non pertinents (par exemple en utilisant une liste de "StopWords") et enfin on traite de manière lexicale les mots-clés retenus. À la fin de cette procédure, les mots-clés retenus vont être appelées termes d'indexation qui vont servir pour générer le descripteur de chaque image.

### 2.2.2 Modèles classiques de recherche d'information (RI)

Plusieurs modèles de RI ont été présentées dans la littérature. Nous avons pu identifier trois types de modèles : le modèle booléen, le modèle vectoriel et le modèle probabiliste.

Nous définissons aussi les notations suivantes :

— Une collection de documents : $D = \{d_1, d_2, \cdots, d_i, \cdots, d_{n_D}\}$

— Un espace de termes d'indexation : $T = \{t_1, t_2, \cdots, t_j, \cdots, t_{n_T}\}$

#### 2.2.2.1 Modèle booléen

Le modèle booléen est basé sur la théorie des ensembles et l'algèbre de Boole. Dans ce modèle, le document $d_i$ est représenté par un ensemble de termes $t_j$ associés au document sans pondération. La requête $q$ est représenté par une expression logique, composée de termes reliées par des opérateurs logiques : ET, OU et NON. La correspondance entre la requête et un document est une correspondance exacte ; par exemple, si un document implique au sens logique la requête alors le document est pertinent. Sinon, il est considéré non pertinent.

#### 2.2.2.2 Modèle vectoriel

Le modèle vectoriel [51] est un modèle algébrique où les documents et la requête sont représentés par des vecteurs dans un espace multidimensionnel dont les dimensions sont les termes issus de l'indexation. Dans ce modèle, un document $d_i$ est représenté par un vecteur $w_i = (w_1(i), w_2(i), \cdots, w_j(i), \cdots, w_{n_T}(i))$ où $w_j(i)$ représente le poids du terme $t_j$ dans le document $d_i$. La requête est représentée par le vecteur $w_q = (w_1(q), w_2(q), \cdots, w_j(q), \cdots, w_{n_T}(q))$.

Dans le modèle vectoriel, il existe différents schémas de pondération pour calculer le poids de chaque terme associé au document. Ici, nous décrivons une pondération classique TfIdf et une pondération Social-TfIdf qui est une variante intéressante de la pondération TfIdf.

**Pondération TfIdf** L'un des schémas de pondération les plus utilisés est TfIdf [37] qui prend comme information la pondération locale et globale et se définit ainsi :

$$w_j(i) = tf(t_j, d_i) \times idf(t_j) \tag{2.1}$$

La pondération locale $tf(t_j, d_i)$ mesure la fréquence du terme $t_j$ dans le document $d_i$, exprimée ainsi : $tf(t_j, d_i) = \frac{df(t_j, d_i)}{|\vec{w}_i|}$ où $df(t_j, d_i)$ est le nombre de fois que le terme $t_j$ apparaît dans le document $d_i$

La pondération globale $idf(t_j)$ mesure la fréquence du terme $t_j$ dans la collection de documents et s'exprime ainsi : $idf(t_j) = log(\frac{|D|}{|\{d_i : t_j \in d_i\}|})$ où $|\{d_i : t_j \in d_i\}|$ est le nombre de documents $d_i$ où le terme $t_i$ apparaît dans la collection de documents $D$.

Puis, pour comparer la correspondance entre la requête et chaque document en utilisant cette pondération, plusieurs mesures de similarité ont été définies. Les mesures les

plus courantes sont le **produit scalaire** ($prod_{sca}$) et la **similarité cosinus** ($sim_{cos}$), définies ainsi :

$$prod_{sca}(q, d_i) = \sum_{j=1}^{T} w_j(q) \times w_j(i) \tag{2.2}$$

$$sim_{cos}(q, d_i) = \frac{\sum_{j=1}^{T} w_j(q) \times w_j(i)}{\sqrt{\sum_{j=1}^{T} w_j(q)^2 \times \sum_{j=1}^{T} w_j(i)^2}} \tag{2.3}$$

**Pondération Social-TfIdf** Dans le cas où l'on dispose de l'information concernant le nombre d'utilisateurs, on peut utiliser une pondération (inspirée de TfIdf) proposé par [46] appelée Social-TfIdf qui prend plus en compte l'information du nombre d'utilisateurs qui ont annoté un terme spécifique dans la collection de documents, exprimée ainsi :

$$w_j(i) = users(t_j) \times \frac{1}{log(pre(t_j))} \tag{2.4}$$

où $users(t_j)$ est le nombre d'utilisateurs qui ont annoté les images avec le terme $t_j$ et $pre(t_j)$ est le nombre d'utilisateurs qui ont annoté les images avec le terme $t_j$ dans la collection de documents $D$.

Puis, pour comparer la correspondance entre le vecteur de la requête et le vecteur de chaque document en utilisant cette pondération, [46] propose d'utiliser la formule suivante :

$$sim_{social}(q, d_i) = \frac{overlap(q, d_i)}{|\vec{w_q}|} \tag{2.5}$$

où $overlap(q, d_i)$ est le nombre de correspondances entre les termes de la requête $q$ et les termes du document $d_i$. S'il y a des documents avec les mêmes scores $sim_{social}$, on réordonne ces documents par le score $sim_{cos}$ en utilisant le descripteur généré par la pondération Social-TfIdf.

#### 2.2.2.3 Modèle probabiliste

Le modèle probabiliste de base [49] est fondé sur la théorie des probabilités. Ce modèle consiste à sélectionner les documents ayant à la fois une forte probabilité d'être pertinents et une faible probabilité d'être non pertinents à la requête.

Dans le modèle probabiliste, les **modèles de langue** sont très utilisés dans la RI. Ce modèle consiste à construire un modèle de langue $M_{d_i}$ pour chaque document, puis à calculer la probabilité qu'une requête $q$ puisse être générée par le modèle de langue du document, soit $P(q|M_{d_i})$. Comme le modèle de langue utilisé est souvent le modèle uni-gramme, la fonction de classement de ce modèle (qui devient aussi la correspondance entre la requête $q$ et le document $d_i$) est exprimée ainsi :

$$sim_{proba}(q, d_i) = P(q|M_{d_i}) = \prod_{t_q \in q} P(t_q|M_{d_i}) \tag{2.6}$$

$P(t_q|M_{d_i})$ est normalement estimée en se basant sur l'estimation maximale de vraisemblance qui est donnée par : $P(t_q|M_{d_i}) = \frac{df(t_q,d_i)}{|\vec{d_i}|}$ où $df(t_q, d_i)$ est la fréquence du terme $t_q$ dans le document $d_i$. Cependant, nous avons souvent le problème de l'absence de termes dans le document, mais présents dans la requête, et qui ont pour effet d'avoir une probabilité $P(q|M_{d_i})$ nulle. Dans ce cas, des techniques de lissage qui consiste à

assigner des probabilités non nulles aux termes qui n'apparaissent pas dans les documents sont utilisées. Une des techniques de lissage les plus employées est le **lissage de Dirichlet** [44] exprimée ainsi :

$$P(t_q|M_{d_i}) = \frac{df(t_q, d_i) + \mu * P(t_q|D)}{|\vec{d_i}| + \mu} \tag{2.7}$$

où $df(t_q, d_i)$ est la fréquence du terme $t_q$ dans le document $d_i$, $P(t_q|D)$ est la probabilité du terme $t_q$ dans la collection de documents $D$ et $\mu$ est un paramètre à calculer.

Selon Jacques Savoy [52], le modèle de recherche probabiliste est plus efficace que le modèle de recherche booléen, mais moins performant que le modèle de recherche vectoriel.

## 2.3 Recherche d'images par le contenu visuel

Dans le SRIm, nous pouvons aussi rechercher une image par une approche basée sur le contenu visuel de l'image. Il est aussi nommé la recherche d'images par le contenu (Content-Based Image Retrieval CBIR). Dans ce cas, la requête est représentée sous la forme d'une image.

Dans la phase d'indexation, on extrait les informations visuelles de chaque image afin de générer son descripteur. Un avantage de cette approche est que le descripteur d'une image peut être généré directement de leur contenu de façon automatique. Puis, dans la phase de recherche on calcule la similarité entre le descripteur de la requête et le descripteur de chaque image.

Dans cette section nous citons d'abord les différentes informations visuelles qu'on peut extraire d'une image, puis, nous citons quelques approches classiques pour la représenter (génération du descripteur visuel). Enfin, nous citons des mesures classiques pour calculer la similarité entre deux images.

### 2.3.1 Information visuelle

Une littérature abondante existe par rapport aux différents types d'informations visuelles qu'on peut extraire d'une image. Nous citons trois caractéristiques classiques et génériques qui sont la couleur, la texture et la forme.

#### 2.3.1.1 Couleur

La couleur est une des caractéristiques les plus employés pour représenter une image. Il existe plusieurs espaces colorimétriques, mais la littérature montre qu'il n'y a pas d'espace de couleur idéal pour représenter une image. Nous montrons quelques espaces colorimétriques :

- L'espace **RVB** (Red-Vert-Blue) est défini par les composantes suivantes : rouge, vert et bleu. Cet espace est très simple à utiliser et il est employé par de nombreux appareils numériques, car il est orienté au matériel. Par contre, les trois composantes sont fortement corrélées, par exemple : si on diminue le composant vert, l'image parait plus rouge.
- L'espace **HSV** (Hue-Saturation-Value) est défini par les composantes suivantes : la teinte (Hue) qui décrit la couleur, la saturation (Saturation) qui décrit l'intensité de la couleur et une valeur (Value) qui décrit la luminosité de la couleur. Cet espace est plus intuitif, car le composant Hue correspond à la façon dont nous percevons les couleurs.
- L'espace **Lab** est défini par les composants suivantes : la luminosité (L) et la chromaticité représentées par le couple (a,b) où l'axe "a" correspond au couple

vert-rouge et l'axe "b" correspond au couple bleu-jaune. Cet espace s'approche le plus de la perception de l'œil humain (espace perceptuellement uniforme).

Une étude [33] compare six espaces colorimétriques et montre que l'espace HSV est le plus efficace pour la recherche d'images.

#### 2.3.1.2 Texture

Il n'existe pas de définition de la texture, mais elle peut être définie comme la répétition d'éléments de base construits à partir de pixels qui respectent un certain ordre (par exemple des grilles, des murs, des tissus, etc.). Une texture peut avoir un aspect périodique, mais aussi un aspect aléatoire comme par exemple le sable, le nuage, l'herbe, etc.

De nombreuses méthodes sont utilisées dans la littérature pour extraire les caractéristiques de la texture. Entre les méthodes classiques les plus utilisées nous avons les filtres de Gabor, la décomposition paramétrique de Wold, etc. Les outils classiques de statistiques (la moyenne, la variance, le moment, l'énergie, l'entropie, ...) et les matrices de cooccurrences peuvent être aussi utilisées pour analyser les textures.

La texture pourrait apporter une information assez intéressante, car elle permet de différencier des parties d'images où les descripteurs de couleur sont identiques. Par exemple, le ciel et la mer vont avoir des descripteurs de couleur très similaires, mais leur texture pourrait faire la différence car le ciel a une texture unie ou nuageuse tandis que la mer a une texture de vagues.

#### 2.3.1.3 Forme

Comme la texture, la forme est une information complémentaire de la couleur. Par exemple, si l'utilisateur cherche des images de ballons, on ne peut pas utiliser la couleur, car on peut trouver des ballons de plusieurs couleurs, mais ils ont une forme très caractéristique. La forme est donc une information discriminante qui peut être utile dans la recherche d'images dans certains cas.

Nous distinguons deux catégories de descripteurs de formes : les descripteurs basés régions et les descripteurs basés frontières. Entre les descripteurs classiques utilisés pour extraire cette information nous avons les moments d'inertie, les moments invariants, les descripteurs de Fourrier, etc. (voir [65] pour plus de détails).

### 2.3.2 Indexation visuelle

Dans cette partie nous citons les différentes manières de représenter l'information obtenue d'une image. Pour cela, deux approches sont possibles : la première c'est l'extraction de descripteurs visuels sur l'image entière (descripteurs globaux) ; et la deuxième c'est l'extraction des descripteurs d'une partie de l'image (descripteurs locaux).

**Histogramme** Une représentation classique est l'histogramme qui est souvent rattaché à la caractéristique de couleur. La raison du succès de l'histogramme est due à sa simplicité de calcul et d'utilisation, assortie d'une relative robustesse à la rotation, aux changements d'échelles et aux occlusions. Par contre, l'inconvénient de l'histogramme est la perte de l'information spatiale.

Pour construire un histogramme de couleurs, on utilise généralement deux phases : d'abord, les couleurs sont quantifiées, puis les couleurs quantifiées sont dénombrées. La quantification a pour objectif d'obtenir une représentation compacte du contenu visuel sans pour autant réduire son efficacité. Chaque composante du vecteur ou "bin" donne la quantité d'une couleur présente dans l'image. Afin de rendre les histogrammes invariants

à la taille de l'image ou des régions, on peut normaliser l'histogramme par le nombre de pixels. Dans l'annexe B page 197 nous décrivons la procédure utilisée afin de générer des histogrammes de couleurs de 31 bins dans le space HSV.

**Segmentation et points d'intérêt** L'extraction de descripteurs globaux permet de réduire le nombre de calculs nécessaires ainsi que le calcul de la similarité entre images. Cependant, les descripteurs locaux sont plus efficaces, mais coûteux. Les descripteurs locaux peuvent être : des régions de l'image obtenues soit par segmentation de l'image entière, soit par recherche de régions d'intérêt ; et des points d'intérêt.

La **segmentation de l'image** consiste à séparer en régions homogènes les divers composants visibles dans une image. Ils existent deux grandes familles de méthodes de segmentation :Une approche par "contour" et une approche par "région".

Les **points d'intérêt** d'une image sont les points qui seront trouvés similaires dans les images similaires. Une manière de les déterminer est de prendre en compte les zones où le signal change. Par exemple, les points d'intérêt peuvent être les coins, les jonctions en T ou les points de fortes variations de texture.

Dans le contexte d'un CBIR "en ligne", dans [16], les auteurs représentent les images par des vecteurs de distributions statistiques de couleurs (comparaison de quatre spaces : RGB, HSV, CIELAB et CIELUV) et de textures (des filtres de Gabor). Dans [58], la représentation de l'image est modélisée en utilisant des modèles HMAX.

### 2.3.3 Mesures de similarité

Dans la phase de recherche, nous devons calculer la correspondance entre les descripteurs de deux images. Pour cela, ils existent divers mesures de similarité ou de distance : la distance Euclidienne, l'intersection entre les histogrammes [55], les distances quadratiques [18], etc. Nous décrivons quelques mesures classiques :

**Distance Euclidienne** Soit une image $i_1$ représentée par le vecteur $H_{i_1} = (h_1(i_1), h_2(i_1), \cdots, h_x(i_1), \cdots, h_{n_{H_{i_1}}(i_1)})$ où $h_x(i_1)$ représente la valeur du vecteur dans la position $x$ de l'image $i_1$, cette distance est exprimée ainsi :

$$\delta_{euc}(H_{i_1}, H_{i_2}) = \sqrt{\sum_{x=1}^{n_{H_{i_1}}} |h_x(i_1) - h_x(i_2)|^2} \tag{2.8}$$

**Distance entre histogrammes** Les techniques géométriques, telles que la distance euclidienne, peuvent être appliquées sur des histogrammes. Il existe cependant des mesures spécifiques aux histogrammes. L'intersection d'histogrammes [55] est définie ainsi :

$$inter_{histo}(H_{i_1}, H_{i_2}) = \frac{\sum_{x=1}^{n_{H_{i_1}}} \min(h_x(i_1), h_x(i_2))}{\sum_{x=1}^{n_{H_{i_1}}} h_x(i_1)} \tag{2.9}$$

## 2.4 Recherche d'images par la géolocalisation

Les coordonnées GPS est une information de géolocalisation, utilisé dans nombreux domaines comme ceux des systèmes de navigation par satellite, de navigation maritime, sur route, etc. Dans la recherche d'images, cette approche est peu souvent employée, car l'utilisateur préfère exprimer sa requête d'une autre façon (par exemple par le nom d'un endroit), car les coordonnées GPS sont une information difficile à retenir.

Dans cette section nous décrivons d'abord les coordonnées GPS et son association avec les images. Puis, nous citons une mesure classique pour calculer la correspondance entre deux images avec les coordonnées GPS.

### 2.4.1 Information de géolocalisation

Les coordonnées GPS sont composées principalement de deux coordonnées des valeurs angulaires : la latitude qui exprime le positionnement nord-sud d'un point sur Terre et la longitude qui exprime le positionnement est-ouest d'un point sur Terre.

Dans une image, les coordonnées GPS indiquent l'endroit où l'image a été prise. Cependant, cette information n'est pas souvent trouvée, car annoter cette information manuellement n'est pas évident à connaître par l'utilisateur. Heureusement, dans l'actualité de nombreux appareils numériques permettent d'associer automatiquement les coordonnées GPS à une image.

### 2.4.2 Mesures de similarité et de distance

Une fois citée les coordonnées GPS dans la phase de recherche, nous devons calculer la correspondance entre deux coordonnées GPS ce qui n'est pas très évidente à calculer dans la réalité, car les coordonnées représentent deux points sur une sphère. En effet, la distance entre deux points dans l'espace euclidien est la longueur d'une ligne droite entre les deux points, mais sur la sphère il n'y a pas de lignes droites mais des arcs.

Nous décrivons ci-dessous la distance de Haversine qui est une mesure classique très adaptée pour calculer la distance avec ce type d'information, car cette mesure utilise la trigonométrie sphérique pour calculer la longueur d'un arc dans une sphère.

**Distance de Haversine :** soit une image $i_1$ représentée par le vecteur $G_{i_1} = (\lambda_{i_1}, \gamma_{i_1})$ où $\lambda_{i_1}$ représente la latitude et $\gamma_{i_1}$ représente la longitude de l'image $i_1$, la distance de Haversine (aussi appelée distance du grand cercle) est exprimée ainsi :

$$\delta_{hav}(G_{i_1}, G_{i_2}) = 2 \times R \times \sqrt{\sin^2(\frac{\gamma_{i_1} - \gamma_{i_2}}{2}) + \cos(\gamma_{i_1}) \times \cos(\gamma_{i_2}) \times \sin^2(\frac{\lambda_{i_1} - \lambda_{i_2}}{2})} \quad (2.10)$$

où $R$ est le rayon de la Terre (environ 6366 km).

## 2.5 Recherche d'images par une arborescence de concepts

Dans la recherche d'images, l'approche conceptuelle consiste à utiliser une structure conceptuelle pour représenter un document et utiliser ce type de representation afin d'améliorer la diversité des résultats obtenus dans les SRIm classiques.

Dans cette section, nous décrivons d'abord les types de structures conceptuelles et les structures disponibles dans l'état de l'art. Puis, nous décrivons les mesures classiques de similarité qui calculent la distance entre deux concepts dans une structure conceptuelle. Enfin, nous décrivons l'utilisation classique de cette ressource dans la recherche d'images ainsi que les contraintes rencontrées.

### 2.5.1 Structures conceptuelles

Les structures conceptuelles servent comme des intermédiaires entre le langage libre utilisé par un utilisateur et un langage contrôlé. Ces structures peuvent être décrites dans une taxonomie de concepts, dans un thésaurus, dans une ontologie, etc.

— Une **taxonomie** est une liste de termes contrôlés organisés de façon hiérarchique (principalement avec des relations de type "Is-A"). Une taxonomie facilite la recherche de termes à partir de relations hiérarchiques.
— Un **thésaurus** est un réseau de termes contrôlés, enrichis par des relations associatives pré-définies (par exemple, des synonymes). Un thésaurus facilite la recherche de termes en fonction de différents types de relations (pas seulement hiérarchiques).
— Une **ontologie** est un modèle de description des connaissances basé sur des concepts avec des types, des propriétés et des relations. L'ontologie décrit les concepts avec un modèle formel, directement implémentable en machine. L'ontologie peut intégrer, a priori, n'importe quel type de relation.

#### 2.5.1.1 Types de structures par son contenu

Si on analyse le contenu de la structure conceptuelle, il peut être de type générique (cas de WordNet [38]) où spécifique à un domaine (cas de MESH [1] [40] pour le domaine médical). Nous citons l'ontologie WordNet qui est une des ressources publiques la plus utilisée dans la littérature.

**WordNet** WordNet [2] [38] représente une base de données lexicales construite par des linguistes de l'Université de Princeton afin de classifier et de mettre en relation le contenu sémantique et lexical de la langue anglaise. La structure du WordNet est composée par plusieurs concepts sous la forme d'une **ontologie**. Chaque concept est composé par d'un ensemble de termes qui sont des synonymes. Ces concepts sont reliées entre eux par différentes relations sémantiques (d'hyperonymie "Is-A", d'hyponymie, etc.) Par exemple, dans WordNet le mot "automobile" est lié au concept "car, auto, automobile, machine, motorcar". Ce concept est lié aussi par une relation d'hyperonymie au concept père "motor vehicle, automotive vehicle". De plus de ce fait nous pouvons avoir aussi un arbre de concepts de plus en plus généraux reliés par la relation d'hyperonymie :

— car, auto, automobile, machine, motorcar
  — motor vehicle, automotive vehicle
    — vehicle
      — conveyance, transport ...

WordNet a été utilisée dans un grand nombre d'applications comme la désambigüisation de sens, l'annotation sémantique de texte, l'extraction d'information et la recherche d'information textuelle [14]. Cependant, sa structure conceptuelle est peu adaptée aux domaines spécifiques comme la recherche d'images, car cette structure ne peut pas être réglée selon les particularités d'un domaine spécifique.

#### 2.5.1.2 Types de structures par sa construction

Les structures conceptuelles peuvent être aussi divisées selon leur construction : de manière manuelle ou automatique. Il existe plusieurs travaux qui ont généré de manière automatique une structure conceptuelle en utilisant, par exemple, l'information textuelle des documents [67]. Les résultats obtenus avec une construction automatique sont moins satisfaisants que avec une construction manuelle, car on rencontre souvent des problèmes d'incompatibilité entre les différents niveaux de profondeur dans la structure. Cependant, une construction manuelle est toujours très coûteuse à réaliser.

1. http ://www.nlm.nih.gov/mesh/
2. http ://wordnet.princeton.edu/

**Arborescence de concepts de Xilopix** L'arborescence de concepts de Xilopix a été construit manuellement par les documentalistes de l'entreprise qui sont des spécialistes de la recherche d'information. Cela garantit d'obtenir une arborescence de bonne qualité en évitant de rencontrer problèmes d'incompatibilité entre les différents niveaux de profondeur qui existe souvent dans les arborescences construites automatiquement.

Celle-ci se compose de plusieurs univers (faune, flore, voyage, personne, etc.) où chaque univers à une structure d'arbre. Dans la figure 1.4 page 7 nous montrons un extrait de sa representation graphique avec trois domaines : le domaine "travel" où les feuilles sont des pays, le domaine "concept" où les feuilles sont des objets et le domaine "transport" où les feuilles sont des moyens de transport. Les concepts de cette arborescence sont reliés aux autres concepts par différentes relations sémantiques (hyperonymie "Is-A", hyponymie). Cette arborescence contient 13 univers différents et la profondeur moyenne de l'arbre de concepts est de 6.54.

L'arborescence de concepts de Xilopix est très adaptée au domaine de la recherche d'images, car cette structure a été construite exclusivement pour être utilisée dans un SRIm en ligne.

#### 2.5.1.3 Mesures de similarité et de distance

Afin d'utiliser une structure conceptuelle comme descripteur, nous devons d'abord étudier la distance entre deux concepts dans une structure conceptuelle.

Soit un concept $c_1$ qui est représenté par le chemin $A_{c_1} = (a_{0,0}, a_1(c_1), ..., a_{p(c_1)}(c_1))$ où $a_0$ est le nœud racine de l'arborescence, $a_{p(c_1)}(c_1)$ est le nœud feuille du concept $c_1$ et $p(c_1)$ est la profondeur du chemin du concept $c_1$, nous citons trois mesures classiques de distance.

**Distance Tree-Editing** La distance "Tree-Editing" consiste à calculer le coût minimal pour transformer un arbre vers l'autre. Soit $S = (S_1, \cdots, S_i, \cdots, S_{n_S})$ l'ensemble de séquences possibles pour la même transformation, cette mesure est définie ainsi :

$$\delta_{TE}(A_1, A_2) = argmin_i\{\gamma(S_i)\}$$

où $\gamma(S_i)$ est le coût une séquence d'opérations pour transformer un arbre vers un autre.

Soit $S_i = (s_1, \cdots, s_{n_{S_i}})$ un ensemble d'opérations pour la séquence $S_i$, le coût $\gamma(S_i)$ de cette séquence est définie ainsi :

$$\gamma(S) = \sum_{j=1}^{n_S} \gamma(s_j)$$

où $s_(j)$ est une opération de base.

Soit $\lambda$ un nœud vide spécial, cette mesure transforme un arbre en considérant trois types d'opérations de base :

— la modification d'un nœud (abrégé en $a(c_1) \rightarrow a(c_2)$)
— l'insertion d'un nœud (abrégé en $\lambda \rightarrow a(c_1)$)
— la suppression d'un nœud (abrégé en $a(c_1) \rightarrow \lambda$)

Contrairement à la distance "Tree-Editing" qui est basée sur la transformation d'un arbre, il existe aussi, dans la littérature, des autres mesures de similarité basées sur la relation entre les concepts. Nous citons deux mesures classiques qui calculent la distance entre deux concepts dans une arborescence de concepts.

**Similarité de Wu-Palmer** La similarité de Wu-Palmer [68] mesure la similarité entre deux concepts d'une même arborescence, exprimée ainsi :

$$sim_{WP}(A_{c_1}, A_{c_2}) = \frac{2 \times z}{p(c_1) + p(c_2)} \tag{2.11}$$

où $z$ est la profondeur du nœud commun le plus profond.

**Similarité de Lin** La similarité de Lin [32] est une autre mesure de la similarité entre deux concepts dans une arborescence. Cette mesure a la particularité de prendre en compte la proportion du nombre de documents associés à un concept par rapport au nombre total de documents. Cette mesure est exprimée ainsi :

$$sim_{LIN}(A_{c_1}, A_{c_2}) = \frac{2 \times \log P(a_z(c_1))}{\log P(a_{p(c_1)}(c_1)) + \log P(a_{p(c_2)}(c_2))} \tag{2.12}$$

où $P(a_{p(c_1)}(c_1))$ (resp. $P(a_{p(c_2)}(c_2))$) est la probabilité calculée en fonction du nombre d'occurrences du nœud $a_{p(c_1)}(c_1)$ (resp. $a_{p(c_2)}(c_2)$) de profondeur $p(c_1)$ (resp. $p(c_2)$) par rapport à la collection total de documents, $z$ est la profondeur du nœud commun le plus profond et $a_z(c_1)$ est le noeud du concept $c_1$ de profondeur $z$.

### 2.5.2 Recherche d'images par l'approche conceptuelle

Dans la recherche d'images, cette approche pourrait être intéressante pour résoudre le problème de l'ambiguïté des termes souvent trouvée dans la recherche par le texte. Par exemple, si la requête textuelle est "voiture", le système donnera sûrement des images associées au terme "voiture", mais dans la base d'images nous pourrions avoir d'autres images pertinentes à la requête, mais liées à des synonymes comme "véhicule" ou "automobile".

Cependant, l'approche conceptuelle dépend beaucoup du bon choix de la structure conceptuelle et d'une bonne indexation entre les documents et les différents concepts de la structure choisie. Dans la littérature, il existe des travaux qui ont certes généré une structure conceptuelle, mais également connu des problèmes pour obtenir une structure conceptuelle correcte. Dans l'état de l'art nous avons trouvé des projets très utilisés et intéressants comme l'ontologie WordNet ou la base d'images ImageNet en utilisant WordNet. En général, l'idéal serait de construire manuellement une structure conceptuelle et de faire une indexation manuelle, mais ces procédures sont très coûteuses à faire. Heureusement, l'entreprise Xilopix a développée manuellement une arborescence de concepts où les images ont été associées manuellement aux concepts de cette arborescence ce qui la rend adaptée à la recherche d'images.

#### 2.5.2.1 Indexation conceptuelle

Une fois que nous avons construit une structure conceptuelle la plus adaptée à notre problème, nous devons d'associer la collection d'images dans les concepts de cette structure. Même si dans une structure conceptuelle, les images peuvent être associées manuellement ou automatiquement, une association automatique est toujours moins performante qu'une association manuelle, mais cette dernière est très coûteuse.

**ImageNet** ImageNet [3] [12] est une base d'images très connue utilisée dans la littérature. Dans cette base, les images ont été associées de manière automatique aux concepts

3. http ://www.image-net.org

de l'ontologie WordNet, en vérifiant manuellement cette association par plusieurs utilisateurs. Cette base d'images contient environ 14 millions d'images pour près de 22 mil concepts.

ImageNet est très utilisée avec un grand succès dans le domaine du traitement des images. Cependant, un inconvénient demeure dans ImageNet : les annotations sont univoques pour chaque image (une image est associée à un seul concept) tandis que dans la recherche d'images, une image peut être associée à plusieurs concepts. En effet, si on prend une image de voiture, cette image est normalement associée au concept "voiture", mais elle pourrait être associée à un autre concept comme par exemple "phare" si l'image met en premier-plan le phare de la voiture.

**Indexation de Xilopix** L'un des points forts de l'entreprise Xilopix est l'association manuelle des images à une arborescence de concepts construite aussi manuellement par les documentalistes de l'entreprise. Contrairement à ImageNet, dans l'arborescence de concepts de Xilopix on peut trouver des images qui sont associées à plusieurs concepts. Cette arborescence et ses associations avec les images sont donc bien adaptées pour la recherche d'images.

#### 2.5.2.2 Mesures de similarité

Une fois que les images sont associées aux concepts d'une structure conceptuelle, nous devons calculer la correspondance entre deux images en prenant en compte cette association. Nous avons vu qu'une image peut être associée à un ou plusieurs concepts d'une structure conceptuelle.

- Dans le cas où **l'image est associée à un seul concept**, nous pouvons utiliser les mesures de similarité classiques de Wu-Palmer et Lin décrites dans la section 2.5.1.3.
- Dans le cas où **l'image est associée à plusieurs concepts**, nous n'avons pas trouvé, dans la littérature, de mesures classiques de similarité. Pour cela, dans la section 5.1.4.1 page 55 nous proposons une méthode qui généralise les similarités classiques (de Wu-Palmer et Lin) dans le cas où l'image est associée à plusieurs concepts.

## 2.6 Métriques d'évaluation

Pour mesurer le **problème de la pertinence** dans les SRIm, on procède à la comparaison de combinaisons de documents pertinents, de documents retrouvés pour chaque requête. Il existe à cet effet de nombreuses mesures classiques et nous avons retenu les suivantes :

### 2.6.1 Précision (P)

La Précision mesure la proportion de documents pertinents retrouvés par le système parmi tous ceux retrouvés. Elle mesure la capacité du système à trouver exclusivement des documents pertinents. Cette mesure détermine également le bruit, c'est-à-dire la proportion de documents non pertinents retrouvés par le système.

$$P = \frac{nb\ documents\ pertinents\ retrouvés}{nb\ documents\ retrouvés}$$

La **précision à $n$ documents** (P@n) mesure le nombre de documents pertinents qui existent dans les $n$ premiers documents retrouvés par le système.

$$P@n = \frac{nb\ documents\ pertinents\ dans\ n\ premiers\ résultats}{n}$$

### 2.6.2 Rappel (R)

Le Rappel mesure la proportion de documents pertinents retrouvés par le système parmi tous les documents pertinents disponibles. Cette mesure permet de déterminer le silence, c'est-à-dire la proportion de documents pertinents non retrouvés par le système.

$$R = \frac{nb\ documents\ pertinents\ retrouvés}{nb\ documents\ pertinents}$$

De même que pour la précision, le **rappel à $n$ documents** (R@n) mesure le nombre de documents pertinents qui existent dans les $n$ premiers documents retrouvés par le système parmi tous les documents pertinents disponibles.

$$R@n = \frac{nb\ documents\ pertinents\ dans\ n\ premiers\ résultats}{nb\ documents\ pertinents}$$

### 2.6.3 Cluster rappel (CR)

Pour mesurer le **problème de la diversité**, on prend en compte le nombre de sous-thèmes retrouvés sur les requêtes. Pour une requête donnée, la vérité terrain donne le nombre de sous-thèmes différents. Par exemple, pour la requête "phare" les documents peuvent être regroupés par les sous-thèmes "phare maritime", "phare de voiture", "phare électrique", etc.

Le cluster rappel est une mesure classique de diversité citée dans [4, 9, 72] similaire à celle du rappel. Le cluster rappel à $n$ documents (CR@n) mesure le nombre de sous-thèmes qui existent dans les $n$ premiers documents retrouvés par le système parmi le nombre total de sous-thèmes différents de la vérité terrain.

$$CR@n = \frac{nb\ sousthèmes\ retrouvés\ dans\ n\ premiers\ résultats}{nb\ sousthèmes\ disponibles}$$

Quand le SRI restitue une liste de documents résultats on pourrait se trouver dans deux cas différents :

- si la précision (P) est faible, si dans les premiers résultats on trouve beaucoup d'images non-pertinentes, alors le cluster rappel (CR) est souvent également faible car les images non-pertinentes pénalisent la précision et la diversité.
- si le score de précision est fort, si dans les premiers résultats on trouve beaucoup d'images pertinentes, alors le score CR peut être variable. Le score CR peut être fort dans le cas où les premières images représentent différents sous-thèmes de la requête, mais le score CR peut être également faible dans le cas où les images pertinentes appartiennent au même sous-thème de la requête.

### 2.6.4 Moyenne de la précision moyenne (MAP)

MAP est une mesure de performance globale qui synthétise en une valeur la qualité d'un système. Cette mesure est calculée en deux étapes. D'abord, on calcule la précision moyenne $AP_q$ pour une requête donnée obtenue en calculant la moyenne des précisions sur un ensemble de points de rappel, exprimée ainsi :

$$AP_q = \frac{1}{n} \sum_{i=1}^{N} P@i * R(i)$$

où $R(i) = 1$ si le $i^{ème}$ document retrouvé est pertinent, $R(i) = 0$ si le $i^{ème}$ document retrouvé est non pertinent, $P@i$ la précision à $i$ documents retrouvés, $n$ le nombre de documents pertinents retrouvés et $N$ le nombre total de documents pour la requête $q$.

Dans la seconde étape, on calcule la précision moyenne pour un ensemble de requêtes, en effectuant la moyenne des précisions moyennes de chaque requête exprimée ainsi :

$$MAP = \frac{1}{M} \sum_{j=1}^{M} AP_{q_j}$$

où $AP_{q_j}$ dénote la précision moyenne pour la requête $q_j$ et $M$ représente le nombre de requêtes considérées.

### 2.6.5 Mesure F (F-mesure)

La "F-mesure" est une mesure de pondération globale entre la précision et le rappel où la formule générale est la suivante :

$$F_\beta@n = (1 + \beta^2) \frac{P@n * R@n}{\beta^2 * P@n + R@n}$$

où $\beta$ est une valeur réelle positive.

A partir de cette formule générale, nous avons des autres mesures couramment utilisées comme la mesure $F_1$ où le rappel et la précision sont uniformément pondérées, la mesure $F_2$ où le rappel a un poids plus élevé que la précision et de manière inverse pour la mesure $F_{0.5}$.

Enfin, pour obtenir une évaluation globale de la précision et la diversité, la mesure $F_1$ est normalement utilisée dans la littérature en calculant la moyenne harmonique de la Précision et le Cluster Rappel.

$$F_1@n = \frac{2 * P@n * CR@n}{P@n + CR@n}$$

# Chapitre 3

# Classification non supervisée

Dans ce chapitre nous allons décrire différentes méthodes classiques de clustering ou de classification non supervisée. Le clustering est une tâche dont l'objectif est de trouver des groupes dans un ensemble de documents selon un critère de similarité. Par exemple, si le critère visuel est la couleur, les méthodes de clustering pourront regrouper les documents qui possèdent la même couleur. Ce critère de similarité est représenté par une mesure de similarité qui est souvent liée au type de descripteur utilisé dans les documents. Par exemple, si les documents sont représentés par des histogrammes de couleur, la mesure de similarité la plus adaptée serait une similarité Euclidienne qu'une similarité Cosinus.

Dans la recherche d'images, la méthode de clustering pourrait être intéressante pour résoudre le problème de la diversité. En effet, si la requête textuelle est "fleur" le système retournera des images de "fleur". Pour diversifier les images retrouvées, nous pouvons utiliser une méthode de clustering pour regrouper d'abord ces images par un critère en particulier (la couleur par exemple), puis réordonner les images en alternant une image de chaque groupe générée par la méthode de clustering. De plus, le clustering s'adapte bien à notre cadre de la recherche d'images en ligne, car contrairement aux méthodes de classification supervisée, les méthodes de clustering n'ont pas besoin de savoir si une image est annotée comme pertinente ou pas pour les regrouper.

Dans la littérature, citons par exemple [66, 5], il existe de très nombreuses méthodes de clustering permettant de créer des clusters de manière automatique, chacune utilisant une approche propre pour construire les clusters. Traditionnellement les approches de clustering sont globalement divisées en partitionnement hiérarchique et partitionnement plat. Le clustering par partitionnement hiérarchique construit les clusters progressivement en générant une hiérarchie de clusters, tandis que le clustering par partitionnement plat apprend les clusters directement en essayant de découvrir les clusters par la relocalisation itérative de points dans les clusters.

Il existe aussi des autres méthodes comme le clustering basé sur la densité, le clustering basé sur l'utilisation de grilles, etc. Les méthodes basées sur la densité essaient d'identifier, dans l'espace, les régions de forte densité entourées par des régions de faible densité, qui formeront les clusters. Contrairement aux autres méthodes de clustering, le clustering par densité n'a pas besoin d'indiquer le nombre de clusters. De plus, cette approche est moins sensibles aux valeurs aberrantes et ils peuvent découvrir des formes irrégulières. Les méthodes de clustering basé sur l'utilisation de grilles, dont l'idée est d'utiliser une grille pour partitionner l'espace en un sous-ensemble, puis d'identifier les sous-ensembles denses connectées, qui formeront les clusters. Ils utilisent fréquemment l'agglomération hiérarchique comme une phase de traitement. Les algorithmes basés sur l'utilisation de grilles sont rapides et gèrent bien les valeurs aberrantes. Dans [71] les auteurs comparent plusieurs méthodes de clustering classiques et les résultats montrent que

le clustering par partitionnement hiérarchique semble être la méthode la plus intéressante pour augmenter la diversité.

Dans la section 3.1. nous décrivons de manière générale deux méthodes classiques de clustering par partitionnement plat, dans la section 3.2. une méthode classique de clustering par partitionnement hiérarchique ainsi que les méthodes utilisées pour regrouper les documents. Enfin, dans la section 3.3. nous montrons les avantages et inconvénients d'utiliser les deux approches de clustering.

Nous définissons les notations suivantes :

— Un ensemble de documents $D = (d_1, d_2, \cdots, d_i, \cdots, d_{n_D})$

— Un ensemble de $K$ clusters de la partition $C = \{C_1, C_2, \cdots, C_k, \cdots, C_K\}$

## 3.1 Clustering par partitionnement plat

Les méthodes de clustering par partitionnement plat consistent à faire une optimisation globale. Dans cette section nous citons deux approches de clustering par partitionnement plat. La première, la méthode `K-Means` qui utilise des centroides pour représenter les clusters. La deuxième, la méthode `K-Medoids` qui utilise des medoides pour représenter les clusters.

### 3.1.1 Méthode K-Moyennes (`K-Means`)

`K-Means` [20] est une méthode classique de clustering plat très connu et très utilisée dans la littérature. Cette méthode vise à répartir les documents en $K$ clusters homogènes afin de minimiser la distance entre les documents à l'intérieur de chaque cluster :

$$\underset{C}{\operatorname{argmin}} \sum_{k=1}^{K} \sum_{d \in C_k} \|\vec{d} - \vec{\mu}_k\|^2 \tag{3.1}$$

où $\vec{\mu}_k$ est le vecteur moyen de documents dans le cluster $C_k$

**Algorithme** L'algorithme classique de `K-Means` peut être résumé ainsi :

1. initialiser $K$ centres (centroïdes) des clusters : $\mu_1^{(0)}, \cdots, \mu_K^{(0)}$
2. associer chaque document $d$ au cluster $C_k^{(i)}$ le plus proche (à l'itération $i$)

$$C_k^{(i)} = \{d : ||\vec{d} - \mu_k^{\vec{(i-1)}}|| \leqslant ||\vec{d} - \mu_l^{\vec{(i-1)}}|| \ \forall\ l = k, 1 < l < K\}$$

3. recalculer les centroïdes $\mu_k^{(i)}$ de chaque cluster $C_k^{(i)}$ (à l'itération $i$)

$$\vec{\mu_k^{(i)}} = \frac{1}{n_{C_k^{(i)}}} \sum_{d \in C_k^{(i)}} \vec{d}$$

4. réitérer les étapes (2) et (3) jusqu'à la convergence où il n'y a plus de changement à l'étape (2).

L'algorithme de `K-Means` a besoin du calcul des centroïdes. Par contre, ce calcul n'est pas très adapté pour certains types de descripteurs. Par exemple, si les descripteurs sont des coordonnées GPS, le vecteur moyen ne correspond pas à la réalité.

Un avantage de l'algorithme de `K-Means` est sa complexité peu coûteuse $O(n_D \times K \times I)$ où $n_D$ est le nombre de documents à regrouper, $K$ est le nombre de clusters et $I$ est nombre d'itérations. Cet avantage nous permet d'utiliser cette méthode avec un grand nombre d'images dans un système de recherche d'images "en ligne". De plus, dans la

littérature il existe d'autres versions encore plus rapides de `K-Means` (par exemple le `K-Means` itérative).

Par contre, l'inconvénient de `K-Means` est que la qualité du résultat final qui dépend de l'initialisation des centroïdes des clusters ce qui n'assure pas une réponse unique. De plus, la sensibilité à l'initialisation devient plus grande quand le nombre de documents est également important. Pour obtenir toujours une réponse unique on peut initialiser les centroïdes en prenant toujours les $K$ premiers documents, mais probablement les $K$ premiers documents sont très similaires entre eux ce qui peut générer des clusters de mauvaise qualité (due à une fausse convergence). Dans la littérature, une autre initialisation classique est de prendre $K$ documents de manière aléatoire.

Un autre inconvénient de `K-Means` est que on doit indiquer le nombre de clusters avant de lancer cette méthode. Or comment savoir le nombre de connaissances présentes dans les documents ? Dans la littérature choisir le bon nombre de clusters est un problème sans réponse unique.

La méthode `K-Means` est utilisée dans plusieurs domaines : dans le partitionnement des données, la quantification vectorielle, la segmentation du marché, l'astronomie, la vision par ordinateur, etc.

### 3.1.2 Méthode K-Médoïdes (`K-Medoids`)

`K-Medoids` est une méthode classique de clustering de partitionnement plat qui au contraire de `K-Means` où les centres des clusters sont représentés par le vecteur moyen (centroïde), les centres des clusters sont représentés par un document appelé médoïde. Cette méthode vise à répartir les documents en $K$ clusters homogènes afin de minimiser la distance entre les documents à l'intérieur de chaque cluster.

$$\underset{C}{\operatorname{argmin}} \sum_{k=1}^{K} \sum_{d \in C_k} \|\vec{d} - \vec{m}_k\|^2 \tag{3.2}$$

où $\vec{m}_k$ est le medoïde de documents dans le cluster $C_k$

**Algorithme** PAM (Partitioning Around Medoids) est l'algorithme classique de `K-Medoids` qui est résumé ainsi :

1. initialiser $K$ medoïdes des clusters : $m_1^{(0)}, \cdots, m_k^{(0)}, \cdots, m_K^{(0)}$ ,
2. associer chaque document $d$ au cluster $C_k^{(i)}$ le plus proche (à l'itération $i$) :

$$C_k^{(i)} = \{d : ||\vec{d} - m_k^{(\vec{i-1})}|| \leqslant ||\vec{d} - m_l^{(\vec{i-1})}|| \ \forall\ l = k, 1 < l < K\}$$

3. calculer le coût de configuration de médoïdes,
4. **pour chaque** medoïde $m_k^{(i-1)}$
   (a) **pour chaque** document $d_o$ non medoïde de l'ensemble de documents $D$
      i. calculer le coût de configuration de l'échange de $m_k^{(i-1)}$ par $d_o$,
5. sélectionner la configuration qui donne le coût minimal,
6. réitérer les étapes (2), (3) et (4) jusqu'à la convergence où il n'y a pas changement dans les medoïdes

La complexité de l'algorithme PAM de `K-Medoids` est $O(K \times (n_D - K)^2 \times I)$ où $n_D$ est le nombre de documents, $K$ est le nombre de clusters et $I$ est nombre d'itérations. Contrairement au `K-Means`, l'algorithme PAM est très coûteux en complexité, car dans cet algorithme on cherche, pour chaque medoïde, le document non medoïde qui pourrait

donner une meilleure configuration. Dans la littérature d'autres versions du `K-Medoids` moins coûteuses existent comme l'algorithme CLARA (Clustering LARge Applications) [27] qui divise d'abord l'ensemble de documents en échantillons et pour chaque échantillon on utilise l'algorithme PAM.

Contrairement à l'algorithme `K-Means` qui n'est pas adapté à certains descripteurs, `K-Medoids` l'est pour la plupart des descripteurs, car le descripteur du medoïde est représenté par celui d'un document.

Également dans l'utilisation de `K-Means`, par rapport à `K-Medoids` la qualité du résultat final dépend également de l'étape d'initialisation ce qui n'assure pas toujours la même réponse. Dans la littérature, une initialisation classique c'est de prendre $K$ documents de manière aléatoire.

Un autre inconvénient de cette méthode est à noter ; en effet on doit indiquer le nombre de clusters à l'étape d'initialisation. Dans la littérature choisir le bon nombre de clusters est un problème sans réponse unique.

## 3.2 Clustering par partitionnement hiérarchique

Les méthodes de clustering hiérarchique visent à construire une hiérarchie de clusters pour regrouper les documents. Contrairement au clustering plat qui fait une optimisation globale, le clustering hiérarchique fait une optimisation locale ce qui lui permet d'obtenir toujours la même qualité des clusters. Ils existent deux façons de construire cette hiérarchie : de manière agglomérative et de manière divisive. Nous décrivons ci-dessous une méthode classique de clustering agglomératif hiérarchique (`AHC`) qui est une des méthodes hiérarchiques les plus couramment utilisée.

### 3.2.1 Clustering agglomératif hiérarchique (`AHC`)

Le clustering agglomératif hiérarchique (`AHC`) [31] est une méthode classique de clustering hiérarchique qui diffère des méthodes de clustering plat où on regroupe les données en $K$ clusters, la `AHC` regroupe les données de manière itérative (hiérarchique) jusqu'à regrouper tous les données dans un seul cluster.

**Algorithme** L'algorithme classique de la `AHC` peut être résumé ainsi :

1. initialiser $n_D$ clusters $C_n$ formes chacun par un document de l'ensemble de documents $D$ : $C_n = \{d_n\}$,
2. calculer la distance entre toutes les paires de clusters : $\delta^*(C_p, C_q) = \delta(d_p, d_q)$ où $\delta^*$ est la distance entre deux clusters et $\delta$ est la distance entre deux documents,
3. agréger les deux clusters $C_y$ et $C_z$ qui ont la plus petite distance pour former un nouveau cluster : $C_k = C_y \cup C_z$ tel que $\delta^*(C_y, C_z) = \min_{p,q} \delta^*(C_p, C_q)$,
4. calculer les distances $\delta^*(C_k, C_{k'})$ entre le nouveau cluster $C_k$ et les autres pour $k' \neq y, z$
5. répéter $(n_D - 1)$ fois les étapes (3) et (4) jusqu'à ce que tous les clusters soient regroupés dans un cluster.

Contrairement au `K-Means`, un inconvénient de la `AHC` est sa complexité de type quadratique $O(n_D^2)$ qui est très coûteuse ce qui peut limiter l'utilisation de cette méthode avec un grand nombre de documents dans un système de recherche "en ligne". Cependant, dans l'état de l'art ils existent aussi d'autres versions plus rapides de l'algorithme classique `AHC`. De plus, nous pouvons aussi trouver quelques astuces pour diminuer la complexité de cet algorithme. Par exemple, si on veut obtenir un nombre fixe de $K$ clusters, on peut arrêter l'algorithme jusqu'à obtenir les $K$ clusters demandés. On peut aussi

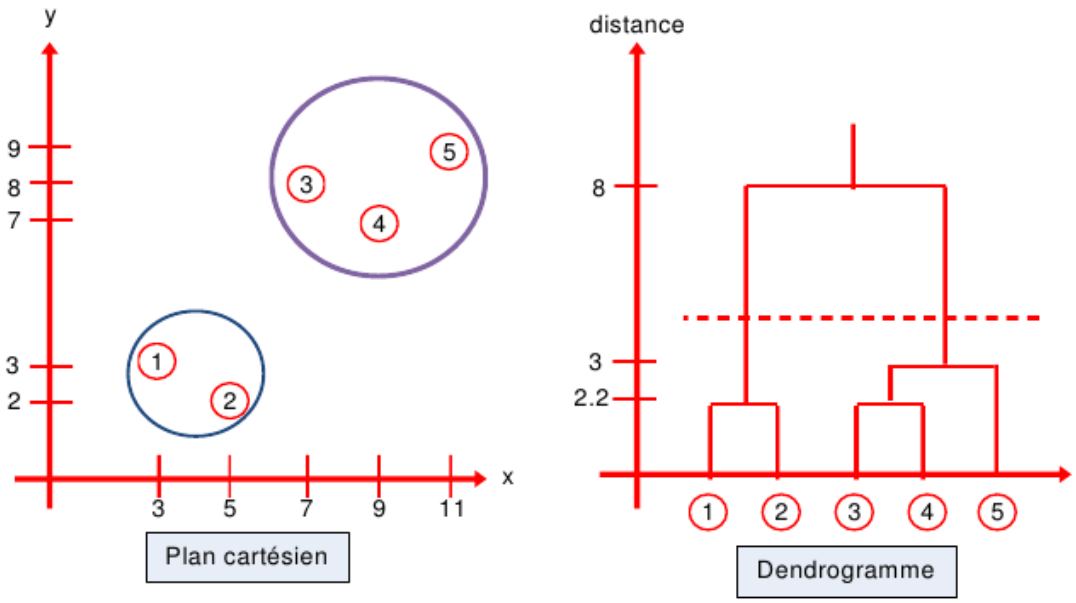



FIGURE 3.1 – (à gauche) L'ensemble de documents représentées dans un plan cartésien. (à droite) Exemple du dendrogramme ou hiérarchie de clusters généré par la `AHC` avec les 5 documents

fusionner les couples de documents qui ont une distance inférieure à un seuil déterminé au préalable afin de réduire le nombre d'itérations de l'algorithme.

**Hiérarchie de clusters** Le résultat de cet algorithme est la génération d'une hiérarchie de clusters ou dendrogramme. Cette hiérarchie est représentée sous la forme d'un arbre binaire, où les documents sont contenus dans les feuilles. La hauteur des nœuds de l'arbre indique généralement la distance entre les clusters. Cette forme de représentation facilite la visualisation de la hiérarchie de clusters. Dans la figure 3.1 nous montrons un exemple de cette hiérarchie en utilisant un ensemble de 5 documents.

Pour construire cette hiérarchie, l'algorithme utilise deux méthodes : la première est la mesure de similarité qui calcule la distance entre deux documents ($\delta$) ; et la deuxième est le critère d'agrégation qui calcule la distance entre deux clusters ($\delta^*$).

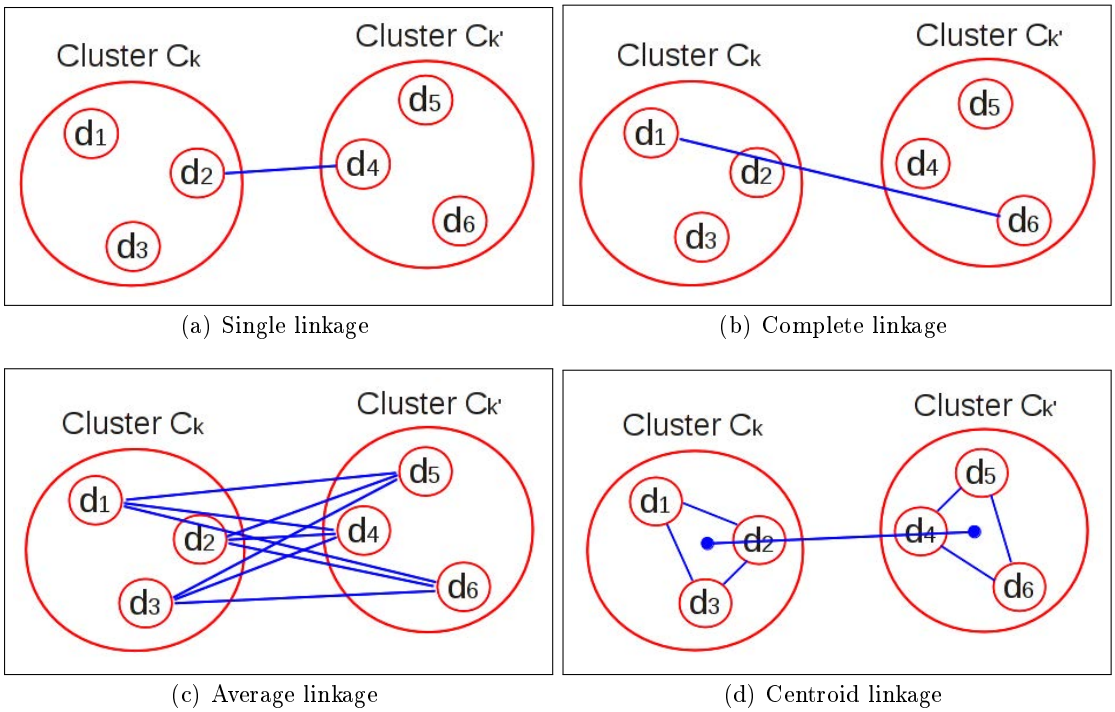
Les mesures de similarité ont déjà été mentionnées dans le chapitre précédent. Dans cette partie nous décrivons les différents critères d'agrégation. Nous allons décrire ensuite les différentes techniques de découpage qui vont servir à obtenir un nombre de clusters optimal en utilisant la hiérarchie de clusters générés par la `AHC`.

#### 3.2.1.1 Critères d'agrégation

Les critères d'agrégation calculent la distance entre deux clusters. Dans la littérature il existe un grand nombre de critères d'agrégation avec des caractéristiques particulières et ce en fonction de chaque critère, les clusters obtenus par la `AHC` peuvent être différents. Il est donc très important de bien choisir le bon critère d'agrégation. Voici ci-après quatre critères classiques d'agrégation les plus connus :

**Lien simple (single linkage)** La distance entre le cluster $C_k$ et le cluster $C_{k'}$ est la plus petite distance entre un élément de $C_k$ et un élément de $C_{k'}$ :

$$\delta^*(C_k, C_{k'}) = \min_{d_r \in C_k, d_s \in C_{k'}} \delta(d_r, d_s)$$




(a) Single linkage (b) Complete linkage

(c) Average linkage (d) Centroid linkage

FIGURE 3.2 – Representation graphique des critères d'agrégation : (a) Single linkage, (b) Complete linkage, (c) Average linkage, (d) Centroid linkage

Ce critère d'agrégation est très simple à calculer. Par contre, il génère parfois une hiérarchie de qualité moyenne avec un petit nombre de grands clusters ("effet de chaîne").

**Lien complet (complete linkage)** La distance entre le cluster $C_k$ et le cluster $C_{k'}$ est la plus grande distance entre un élément de $C_k$ et un élément de $C_{k'}$ :

$$\delta^*(C_k, C_{k'}) = \max_{d_r \in C_k, d_s \in C_{k'}} \delta(d_r, d_s)$$

De même que pour le critère "single", le critère "complete" est très simple à calculer. Par contre, ce critère génère parfois une hiérarchie de qualité moyenne avec un grand nombre de petits clusters.

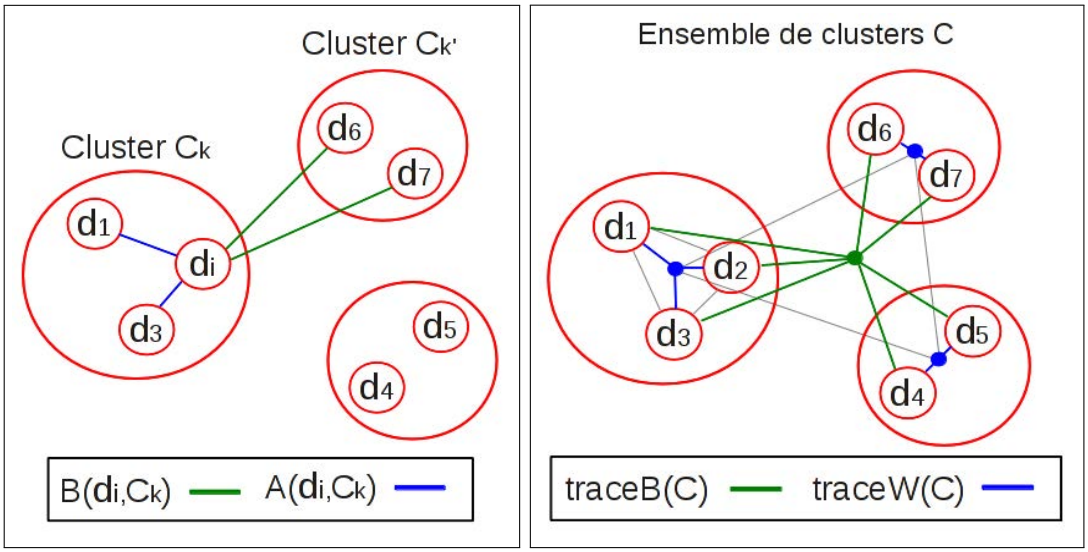
**Lien moyen (average linkage)** La distance entre le cluster $C_k$ et le cluster $C_{k'}$ est la moyenne de la distance de chaque élément de $C_k$ avec chaque élément de $C_{k'}$ :

$$\delta^*(C_k, C_{k'}) = \frac{\sum_{d_r \in C_k} \sum_{d_s \in C_{k'}} \delta(d_r, d_s)}{n_{C_k} \times n_{C_{k'}}}$$

Le critère d'agrégation "average" est plus coûteuse en complexité que les critères "single" et "complete", mais "average" donne un bon compromis entre les deux critères précédents pour générer une hiérarchie de bonne qualité.

**Lien Centroïde (centroid linkage)** Soit $\mu_{C_k}$ le centre de gravité du cluster $C_k$ et $\mu_{C_{k'}}$ le centre de gravité du cluster $C_{k'}$, la distance $\delta^*(C_k, C_{k'})$ est la distance entre les barycentres :

$$\delta^*(C_k, C_{k'}) = \delta(\mu_{C_k}, \mu_{C_{k'}})$$




(a) Découpage "Kaufman et Rousseaux" (b) Découpage "Calinski et Harabasz"

FIGURE 3.3 – Représentation graphique des techniques de découpage : (a) "Kaufman et Rousseaux" avec le calcul de la confidence du document $d_i$ du cluster $C_k$, (b) "Calinski et Harabasz" avec le calcul de la dispersion de l'ensemble de clusters $C$

De même que pour le critère "average", le critère "centroid" donne aussi un bon compromis pour générer une hiérarchie de bonne qualité. Par contre, le critère "centroid" est seulement adapté à certains descripteurs. Par exemple, si le descripteur est de coordonnées GPS, la valeur moyenne de ce descripteur n'est pas évidente à calculer.

Dans la figure 3.2 nous montrons une représentation graphique des quatre critères classiques d'agrégation.

**Effet de chaîne** L'effet de chaîne est un comportement souvent trouvé dans le critère "single" qui consiste à créer des hiérarchies déséquilibrées (la génération d'un petit nombre de grands clusters pour le critère "single"). Cet effet a l'inconvénient de rendre plus difficile le bon endroit où on doit couper la hiérarchie de clusters.

#### 3.2.1.2 Techniques de découpage

Après avoir généré la hiérarchie des clusters, on doit couper horizontalement cette hiérarchie pour obtenir une partition des clusters qui sont les clusters restant sous la coupe.

Le problème c'est de choisir le bon endroit où on doit couper la hiérarchie pour obtenir le bon nombre de clusters. Dans la littérature le choix du nombre de clusters est un problème sans réponse unique. Des auteurs utilisent des méthodes statistiques, des approches selon le nombre de données, la densité des clusters, etc. Nous citons trois techniques classiques de découpage :

**Découpage Traditionnelle (`Traditional`)** Le découpage `Traditional` [39] est une technique classique qui coupe la hiérarchie de clusters à l'endroit au moment où on a le plus grand écart (où les branches de la hiérarchie sont longues) en sachant que chaque étape de l'algorithme de la `AHC` corresponde à chaque niveau de la hiérarchie généré. Soit $\delta^{*(i)}$ la distance entre les deux clusters qui ont fusionné au niveau $i$ et $\delta^{*(i+1)}$ celle où ils

ont fusionné au niveau $i+1$ alors :

$$\Delta(i, i+1) = \delta^{*(i)} - \delta^{*(i+1)}$$

et on choisit de couper la hiérarchie de clusters au niveau $i'$, tel que :

$$i' = \underset{i}{\operatorname{argmax}}\, \Delta(i, i+1)$$

Cette technique de découpage est très simple à calculer, mais elle n'assure pas toujours un bon choix du nombre de clusters, spécialement lorsqu'on travaille avec un grand nombre de données ; cette technique fait des coupures le plus près de la racine.

**Découpage de Kaufman et Rousseeuw** La technique de découpage de Kaufman et Rousseeuw mentionnée dans [27] consiste à calculer la valeur de confidence des clusters générés dans chaque niveau $i$ de la hiérarchie des clusters, afin de choisir de couper la hiérarchie au niveau $i'$ qui obtient la plus grande valeur de confidence, exprimée ainsi :

$$i' = \underset{i}{\operatorname{argmax}} \frac{\sum_{C_k^{(i)} \in C^{(i)}} LVRC(C_k^{(i)})}{n_{C^{(i)}}}$$

où $C^{(i)}$ sont les clusters générés au niveau $i$ et $LVRC(C_k)$ est une valeur de confidence du cluster $C_k$ généré au niveau $i$.

La valeur de confidence $LVRC(C_k)$ du cluster $C_k$ est calculée ainsi :

$$LVRC(C_k) = \frac{1}{n_{C_k}} \sum_{d_i^{C_k} \in C_k} lvrc(d_i^{C_k})$$

où $lvrc(d_i^{C_k})$ est la valeur de confidence du document $d_i$ dans le cluster $C_k$.

La valeur de confidence $lrvc(d_i^{C_k})$ du document $d_i$ est calculée ainsi :

$$lvrc(d_i^{C_k}) = \frac{B(d_i^{C_k}) - A(d_i^{C_k})}{\max(A(d_i^{C_k}), B(d_i^{C_k}))}$$

où $A(d_i^{C_k})$ est la distance moyenne du document $d_i$ par rapport à tout les autres documents du même cluster $C_k$ et $B(d_i^{C_k})$ est la distance moyenne du document $d_i$ du cluster $C_k$ par rapport aux documents du cluster le plus proche $C'_k$ au document $d_i$. Dans la figure 3.3(a) nous montrons une représentation visuelle des valeurs $A(d_i^{C_k})$ (en bleu) et $B(d_i^{C_k})$ (en vert) pour le calcul de la confidence du document $d_i$ dans le cluster $C_k$.

Contrairement au découpage `Traditional`, le découpage de Kaufman et Rousseeuw est plus coûteux, mais il peut obtenir des meilleurs résultats que celui `Traditional`.

**Découpage de Calinski et Harabasz** La technique de découpage Calinski et Harabasz mentionnée dans [70] consiste à calculer la dispersion des clusters générés dans chaque niveau $i$ de la hiérarchie des clusters afin de choisir de couper la hiérarchie au niveau $i'$ qui obtient la plus grande valeur de dispersion, exprimée ainsi :

$$i' = \underset{i}{\operatorname{argmin}}\, GVRC(C^{(i)})$$

où $GVRC(C^i)$ est la valeur de dispersion pour l'ensemble de clusters $C$ générés au niveau $i$ de la hiérarchie.

TABLE 3.1 – Comparaisons des méthodes classiques de clustering : `K-Means`, `K-Medoids` et `AHC`. $K$ est le nombre de clusters, $I$ est le nombre d'itérations et $n_D$ est le nombre de documents dans la collection $D$

| Méthode | Clustering Plat | | Clustering Hiérarchique |
|---|---|---|---|
| | `K-Means` | `K-Medoids` | `AHC` |
| Algorithme | classique | PAM | classique |
| Complexité | $O(K \times n_D \times I)$ | $O(K \times (n_D - K)^2 \times I)$ | $O(n_D^2)$ |
| Implémentation | facile | compliqué | moyen |
| Initialisation du nombre de clusters | requis | requis | pas requis |
| Résultat dépend de initialisation | oui | oui | no |
| Limitation de descripteurs | oui (seulement numérique) | no | no |

La valeur de dispersion $GVRC(C)$ pour l'ensemble de clusters $C$ est calculé ainsi :

$$GVRC(C) = \frac{traceB(C) * (n_D - n_C)}{traceW(C) * (n_C - 1)}$$

où $traceB(C)$ est la dispersion globale et $traceW(C)$ la dispersion locale de l'ensemble de clusters $C$. Dans la figure 3.3(b) nous montrons une représentation visuelle des valeurs $traceW$ (en bleu) et $traceB$ (en vert) pour le calcul de la dispersion dans l'ensemble de clusters $C$.

La valeur de la dispersion locale $traceW(C)$ pour l'ensemble de clusters $C$ est calculée ainsi :

$$traceW(C) = \sum_{C_k \in C} ( \sum_{d_i^{C_k} \in C_k} \| d_i^{\vec{C}_k} - \mu_{\vec{C}_k} \|^2 )$$

où $d_i^{\vec{C}_k}$ est le vecteur du document $d_i$ dans le cluster $C_k$ et $\mu_{\vec{C}_k}$ est le vecteur moyen des documents dans le cluster $C_k$

La valeur de la dispersion globale $traceB(C)$ pour l'ensemble de clusters $C$ est calculée ainsi :

$$traceB(C) = \sum_{C_k \in C} \| \mu_{\vec{C}_k} - \mu_{\vec{C}} \|^2 - traceW(C)$$

où $\mu_{\vec{C}}$ est le vecteur du barycentre de tous les clusters dans l'ensemble de clusters $C$.

Le découpage de "Calinski et Harabasz" est beaucoup plus coûteux que celui de "Kaufman et Rousseeuw" et `Traditional`, mais il peut obtenir de meilleurs résultats que les techniques précédentes de découpage .

## 3.3 Discussions

Dans cette section nous comparons trois méthodes classiques de clustering : `K-Means` et `K-Medoids` qui sont deux méthodes de clustering plat et `AHC` qui est une méthode de clustering hiérarchique. Dans le tableau 3.1 nous résumons les caractéristiques les plus importantes de chaque méthode de clustering.

Un inconvénient général des méthodes classiques de clustering est le fait que on doit indiquer le nombre de clusters en sachant que dans la littérature le bon choix du nombre de clusters est un problème sans réponse unique. Mais, la différence entre les

deux types de clustering, est que dans le clustering plat on a besoin d'indiquer au départ ce nombre de clusters contrairement à la `AHC` qui ne nécessite pas de le connaître pour lancer l'algorithme.

Les méthodes de clustering plat sont généralement des méthodes qui cherchent à optimiser le résultat final, contrairement au clustering hiérarchique qui utilise une optimisation locale. Par contre, ces dernières ont une étape initialisation ce qui fait dépendre la qualité du résultat final de celle-ci. De plus, la sensibilité à l'initialisation devient plus grande quand le nombre de données est important. Entre les deux méthodes de clustering plat, l'algorithme PAM de `K-Medoids` est plus robuste que celui classique de `K-Means` en présence de bruits.

L'avantage de la méthode `K-Means` est d'être moins coûteuse en complexité par rapport aux méthodes `K-Medoids` et `AHC`, ce qui permet à `K-Means` d'être plus efficient dans le traitement de grands ensemble de données et d'utiliser un nombre d'images beaucoup plus grand dans un système de recherche en "ligne".

L'algorithme `K-Means` calcule des centroïdes ce qui n'est pas adapté pour certains descripteurs, tandis que l'algorithme `K-Medoids` (qui calcule de medoïdes) et la `AHC` qui peut utiliser certains critères d'agrégations comme "simple", "average", etc. utilisent des méthodes plus adaptées à la plupart des descripteurs.

# Chapitre 4

# Diversité pour la recherche d'information

Les SRI's traditionnels ont tendance à retourner des documents de plus en plus pertinents. Cependant, quand un utilisateur pose une requête, ce qui l'intéresse c'est d'avoir des documents qui soient certes tous pertinents, mais aussi qui soient les plus différents les uns des autres. Les SRI's traditionnels supposent que la pertinence d'un document est indépendante de celle des autres documents. Cependant, pour une requête donnée, l'utilisation d'un document peut dépendre de ceux que l'utilisateur a déjà vus [72], car s'il en a déjà vu un d'un certain type, il pourra être plus intéressé par un document d'un autre type.

La plupart des méthodes de diversité sont exécutées après la requête : une recherche traditionnelle est effectuée, une liste ordonnée de résultats est obtenue, puis dans une phase de post-traitement, les résultats sont réordonnés afin de produire la diversité. Dans ce processus un point important à prendre en compte est d'atténuer les deux composants : la pertinence et la diversité où l'amélioration d'un composant entraîne généralement une dégradation de l'autre. Par exemple, incrémenter seulement la diversité peut entraîner à la perte d'images pertinentes, tandis qu'incrémenter seulement la pertinence peut entraîner à la génération des images similaires. Il existe aussi des méthodes [62] qui intègrent le processus de diversification lors de la phase de recherche. Par exemple, en créant un partitionnement de la base d'images.

Dans la littérature, il existe plusieurs approches liées à la diversité appliquées dans différents contextes. Dans [73] les auteurs étudient la diversification des résultats dans le contexte de la recherche Web. Ils proposent un nouveau schéma d'ordonnancement appelé "Affinity Ranking" qui consiste à réordonner les résultats de recherche en optimisant deux mesures : la diversité et la richesse de l'information. Dans [53] les auteurs reconnaissent la nécessité de la diversification des résultats pour la recherche d'images. Ils proposent une méthode de réordonnancement basée sur l'analyse de la richesse des thèmes pour enrichir la couverture de différents thèmes dans les résultats de recherche. De plus, dans [13] les auteurs proposent une méthode qui optimise de manière conjointe la diversité et la pertinence des résultats pour la recherche d'images en utilisant des techniques inspirées par des algorithmes de programmation dynamique.

Dans [74] les auteurs étudient la diversification dans le contexte des systèmes de recommandation. Ils proposent une méthode désignée à équilibrer et diversifier des listes de recommandation personnalisées pour répondre aux mieux l'intérêt de différents utilisateurs. Bien que leur système est préjudiciable en précision moyenne, ils démontrent expérimentalement que les utilisateurs préfèrent des résultats plus diversifiés.

Dans [64] la diversification est étudiée dans le contexte de base de données structurées avec des applications d'achat en ligne. Dans leur travail, les documents sont représen-

tés comme un ensemble de caractéristiques et le but est de sélectionner un ensemble de documents qui soient les plus diverses possibles selon un ordre lexicographique de caractéristiques. Comme les préférences lexicographiques sont connues à l'avance et directement utilisés dans la formulation du problème, elle constitue une forme de diversification explicite. Cependant, dans leur travail la notion de pertinence est supprimée ce qui lui rend moins applicable dans la recherche Web où les documents diffèrent grandement dans leur capacité à satisfaire les besoins des utilisateurs.

Dans ce chapitre, nous décrivons différentes approches classiques de diversité, plus spécialement, l'approche de celle par partitionnement et par optimisation. Puis, nous comparons quelques méthodes classiques de diversité de l'état de l'art.

## 4.1 Diversité par partitionnement

Une manière de diversifier les résultats est d'utiliser une méthode de partitionnement (clustering) qui permet de regrouper une liste de documents en générant plusieurs clusters. On émet alors l'hypothèse que les clusters retrouvés correspondent aux sous-thèmes de la requête ce qui n'est pas forcement le cas. Lorsqu'un utilisateur fait une requête, le SRI ne connaît pas le nombre de sous-thèmes correspondant, or de nombreux algorithmes de clustering demandent de connaître le nombre de clusters à générer. Une des difficultés des méthodes de diversité par partitionnement est donc de déterminer le nombre de clusters qu'il faut choisir pour retrouver le maximum de sous-thèmes dans les premiers résultats.

Une stratégie pour obtenir les sous-thèmes c'est de faire un **clustering de toute la collection de documents** (clustering hors-ligne). Malheureusement, cette approche est difficile à envisager quand le nombre de documents est trop élevé. De plus, que se passe-t-il quand de nouveaux documents se présentent ? Faut-il relancer cette stratégie ? Un autre problème est celui de connaître le niveau de granularité du clustering que l'on doit choisir pour une requête donnée. En effet, la granularité de la diversité d'une requête est différent d'une requête à l'autre.

**Clustering "post-traitement"** Une autre stratégie pour générer les sous-thèmes est de faire un clustering des résultats pour chaque requête. Cette stratégie est adaptée à être utilisable "en ligne", et ne nécessite pas de lourds calculs à faire. De plus, elle donne souvent une meilleure qualité des clusters que celle précédente, car elle est adaptée à la requête.

**Algorithme** Pour utiliser du clustering dans la phase de "post-traitement", la plupart des algorithmes réalisent les étapes suivantes :

1. D'abord, les documents sont retrouvés et ordonnés par similarité à la requête ;
2. Ensuite, on fait un clustering des $N$ premiers documents retrouvés (où on obtient $K$ clusters) ;
3. Enfin, les résultats sont réordonnés en utilisant les clusters générés pour améliorer la diversité dans les $n$ premiers résultats. On veut que les $n$ premiers documents représentent des documents en provenance d'autant de clusters différents que possible.

Cependant, utiliser une méthode de clustering afin d'obtenir une liste de résultats diversifiés (étape 3) n'est pas évident à faire, car il y a un certain nombre de difficultés qui doivent être prises en compte pour diversifier de manière optimal les résultats. Ces difficultés sont :

**Choix de l'ordre entre les clusters** Le clustering ne fournit pas d'ordre sur les clusters, on doit se demander quel cluster doit on choisir en premier pour diversifier de manière optimal les résultats ? Une manière simple est de trier les clusters en fonction du rang des documents (voir par exemple [56]) et de choisir en premier le cluster qui contient l'image de rang 1, puis en deuxième le cluster qui contient l'image restante de rang le plus fort et ainsi de suite. Cependant, lorsque beaucoup d'images sont très pertinentes, cet ordre n'est pas toujours très significatif.

Dans la littérature d'autres variantes existent pour trier les clusters. Dans [3] les clusters sont triés en fonction du nombre d'utilisateurs qui ont annoté les documents. Dans [63] les auteurs calculent d'abord un représentant synthétique visuelle (SRI) pour chaque cluster, puis les clusters sont triés en fonction de la distance entre le SRI du cluster et le vecteur SRI moyen sur tous les clusters. Dans cette thèse nous proposons de trier les clusters en fonction du nombre de documents dans chacun desdits clusters. Nous décrivons ces méthodes dans la section 5.1.5 (voir aussi les expérimentations dans les sections 7.2, 8.1.1 et 8.2.1).

**Choix de l'ordre entre les documents** Le clustering ne fournit pas non plus d'ordre sur les documents et ne précise pas celui à choisir en premier pour chaque cluster pour diversifier les documents. Un choix classique est de trier les documents en fonction de leur rang (voir par exemple [56] )et de prendre en premier le document avec le rang le plus fort. Dans la littérature aussi d'autres variantes existent pour ordonner les documents. Dans [3] les auteurs prennent d'abord le document le plus similaire au représentant du cluster. Dans [10] les documents sont triés selon le score fourni par les forêts d'arbres de décisions.

**Autres limitations** Il faut remarquer aussi que l'utilisation des méthodes de clustering (étape 2) rencontre un certain nombre de difficultés qui peuvent influencer les scores de diversité. Par exemple, comment choisir le bon nombre de clusters afin d'augmenter la diversité ? Comment choisir le représentant du cluster ? ; Que faire quand un document appartient à plusieurs sous-thèmes ?, etc.

Dans la littérature, différentes familles de clustering ont été utilisées pour augmenter la diversité. Par exemple, dans [63], les auteurs utilisent une méthode classique de clustering hiérarchique par un descripteur visuel pour améliorer la diversité. Cette méthode permet d'obtenir de meilleurs résultats en diversité par rapport à une approche gloutonne par un descripteur textuel. Cependant, l'utilisation d'une étape de réordonnancement par la similarité à la requête ("reranking") par le texte, permet d'augmenter encore plus la diversité.

Dans [56], les auteurs utilisent aussi une méthode classique de clustering hiérarchique par le visuel. Dans ces expériences, on voit que l'utilisation d'un filtre de visages (élimination des images qui montrent des visages en premier plan) offre de meilleurs résultats en diversité qu'avec l'utilisation du clustering hiérarchique par le visuel. Cependant, la combinaison de ces stratégies, en utilisant d'abord un filtre de visage, et puis un clustering hiérarchique par le visuel permet d'augmenter encore plus la diversité.

Dans [11], les auteurs utilisent l'algorithme BIRCH qui est une variante du clustering hiérarchique plus adaptée à des grands ensembles de données. Puis, les auteurs améliorent la hiérarchie générée par BIRCH en enlevant les clusters isolés ou fusionnés à d'autres clusters de cette hiérarchie. Enfin, dans l'étape de réordonnancement par similarité à la requête, les auteurs donnent la priorité d'une part aux clusters qui ont plus d'images et, d'autre part à l'intérieur de chacun à l'image la plus proche au centroïde. Dans cette article, l'utilisation du clustering hiérarchique avec du visuel donne des meilleurs résultats en diversité par rapport au texte, mais aussi par rapport à la combinaison du texte et

du visuel. De plus, l'utilisation d'une étape de "pré-filtre" avant de lancer le clustering hiérarchique, permet d'augmenter encore plus la diversité. Cette étape de "pré-filtre" consiste à enlever les images non pertinentes du contexte de la diversité sociale (par exemple, des images qui montrent des visages en premier plane, des images qui ont une quantité faible de visites, etc.).

Dans [22] un clustering hiérarchique est aussi effectué, mais sur des vecteurs construits à partir de concepts visuels ; dans [48, 17] les auteurs utilisent une méthode `K-Means` de clustering plat ; [3] utilise une variante du `K-Means` (Kmeans++) ; et [7] utilise une méthode de clustering plat basée en médoïdes (`K-Medoids`).

La plupart de ces travaux sont contemporains à notre travail. Cela montre que notre idée d'utiliser du clustering sur la diversité, a déjà été utilisée par d'autres chercheurs. Cependant, l'apport de cette thèse est d'avoir cherché à obtenir des résultats qui ne soient pas dépendants d'un seul benchmark. Pour cela, nous avons travailler sur trois benchmarks assez différents en nombre de requêtes, en nombre d'images, en type de descripteurs... et provenant de sources différentes afin d'obtenir des comportements les plus généraux possibles.

## 4.2 Diversité par optimisation

Un des problèmes souvent rencontrés par les méthodes de diversité est qu'en augmentant la diversité, la pertinence des résultats a tendance à diminuer [74, 60]. C'est pourquoi certains travaux [13] proposent d'effectuer une optimisation conjointe avec l'objectif d'obtenir un résultat qui soit pertinent et divers à la fois. Parmi les méthodes de diversité par optimisation nous avons les méthodes de programmation dynamique, les algorithmes probabilistes, les algorithmes gloutons, etc. Dans cette section nous décrivons quelques algorithmes gloutons classiques.

**Diversité par algorithmes gloutons** Un algorithme glouton est un algorithme d'optimisation qui, à chaque étape, fait un choix optimal local dans l'espoir d'obtenir un résultat optimal global. Contrairement aux méthodes de clustering où l'on doit spécifier certaines difficultés comme le choix du nombre de clusters ou le choix de l'ordre entre les clusters et les documents, les algorithmes gloutons n'ont pas besoin de spécifier ces paramètres, car cette méthode ne regroupe pas les documents. Cependant, un des inconvénients de cet algorithme est son initialisation qui dépend fortement du choix du premier document, car un mauvais choix peut détériorer le résultat final.

Dans la littérature différents algorithmes gloutons existent, citons par exemple, l'algorithme MMR [8], la dissimilarité moyenne [13], l'algorithme `Maxmin`, etc.

Avant de décrire quelques algorithmes gloutons, il nous faut d'abord définir les notations suivantes :

— Un ensemble de documents $D = (d_1, \cdots, d_i, \cdots, d_{n_D})$.

— Un ensemble des $i-1$ premiers documents sélectionnés $Z = (d_1, \cdots, d_{i-1})$.

### 4.2.1 Maximal Marginal Relevance MMR

Dans l'algorithme MMR (Maximal Marginal Relevance) cité dans [8] la pertinence et la diversité sont calculées de manière indépendante, puis reliées par une combinaison linéaire.

**Algorithme** Cet algorithme peut être exprimé ainsi :

1. initialiser l'ensemble $Z$ en choisissant un premier document

2. pour chaque document $d_r \in D \backslash Z$ (non sélectionné) calculer $value_{MMR}(d_r)$ exprimé ainsi :

$$value_{MMR}(d_r) = \alpha \times sim_1(d_r, q) - (1 - \alpha) \times \max_{d_z < Z} sim_2(d_r, d_z)$$

où $q$ est la requête, $sim_1(d_z, q)$ est une similarité classique qui calcule la pertinence, $sim_2(d_z, d_j)$ est une similarité entre documents qui calcule la redondance et $\alpha$ est un paramètre qui contrôle l'importance de la pertinence et la diversité.

3. sélectionner dans $Z$ le document $d_r$ qui possède la plus grande valeur $value_{MMR}(d_r)$
4. réitérer $(n_D - 1)$ fois les étapes (2) et (3) jusqu'à ce que tous les documents soient dans l'ensemble $Z$.

A l'étape 2 de l'algorithme un document $d_r$ a une forte valeur $value_{MMR}$ si ce document est pertinent à la requête et s'il possède une faible similarité en relation aux documents déjà sélectionnés. Si $\alpha = 1$, alors c'est la similarité classique entre la requête et chaque document qui est effectuée. Si $\alpha = 0$, alors la similarité produit une diversité maximale.

### 4.2.2 Algorithme `Maxmin`

`Maxmin` est un algorithme glouton classique très utilisé dans la littérature qui consiste à maximiser la distance minimale d'un document donné par rapport aux documents les plus dissimilaires déjà sélectionnés.

**Algorithme** l'algorithme `Maxmin` peut être résumé ainsi :

1. initialiser l'ensemble $Z$ en choisissant un premier document
2. pour chaque document $d_r \in D \backslash Z$ (non sélectionné) calculer $value_{MAXMIN}(d_r)$ en choisissant la distance minimale entre le document $d_r$ et chaque document de l'ensemble $Z$

$$value_{MAXMIN}(d_r) = \min_{d_z \in Z} \delta(d_r, d_z)$$

3. sélectionner dans $Z$ le document $d_r$ qui a la plus grande valeur $value_{MAXMIN}(d_r)$
4. réitérer $(n_D - 1)$ fois les étapes (2) et (3) jusqu'à ce que tous les documents soient dans l'ensemble $Z$.

Dans l'algorithme on passe dans la boucle $n_D - 1$ fois et à chaque itération on fait $n_{D \backslash Z} \times n_Z$ comparaisons. Donc, la complexité de l'algorithme `Maxmin` est $O(n_D \times n_{D \backslash Z} \times n_Z)$ qui semblerait être très coûteuse. Cependant, comme à l'étape 2 de l'algorithme on n'a pas besoin de recalculer les distances à chaque itération, sa complexité pourrait se réduire à $O(n_{D \backslash Z} \times n_Z)$. Une amélioration ce cet algorithme est proposée dans [47] avec une complexité de $O(n_D)$ dans les meilleurs des cas.

Dans [15, 2, 28] les auteurs mentionnent plusieurs applications de `Maxmin` : l'optimisation basée dans la simulation, le contrôle de la pollution, l'ingénierie génétique, l'amélioration de plantes, l'élimination de la redondance du résultat final.

Dans la littérature, plusieurs travaux [13, 19, 1] proposent d'utiliser des algorithmes d'optimisation pour diversifier les images. Dans [26] un algorithme glouton "Min-max" (Minimisation de la similarité maximale) est utilisé ; [45] utilise l'algorithme "Max-avg" (Maximisation de la distance moyenne). Dans [10, 69], les auteurs utilisent un algorithme de diversification qui optimise de manière conjointe la pertinence et la diversité.

## 4.3 Autres types de diversité

Il existe une manière simple de produire de la diversité en permutant aléatoirement les résultats. Cependant, comment choisir le bon nombre de documents à permuter sans que l'on soit pénalisé par les documents non-pertinents ? Certains travaux proposent des autres approches comme l'utilisation de la date de prise de vue de l'image ou bien sa géolocalisation (voir par exemple [35, 43]). Ce type d'informations semblent permettre une certaine diversité, mais n'est pas toujours disponible.

### 4.3.1 Par fusion de résultats

Des travaux [1, 60] proposent de fusionner plusieurs listes ordonnées de résultats. Dans chaque liste, l'élément ayant le rang le plus faible est sélectionné, puis le second élément de chaque liste est sélectionné, et ainsi de suite, mais en évitant les doublons. Les listes ordonnées peuvent provenir de recherches effectuées sur des modalités différentes (texte, image, texte-image...) ou de méthodes différentes (clustering, permutation...). Dans [42] une fusion de trois méthodes de clustering est utilisée : un clustering hiérarchique `AHC`, un clustering spectral et un clustering "Metis" en utilisant des descripteurs visuels. Cependant, cette fusion n'a pas donné des résultats très significatifs par rapport à l'utilisation directe de la `AHC`. La diversité par "méta-moteur" utilise le même principe en lançant la même requête sur plusieurs moteurs de recherche, puis en alternant le résultat de chaque moteur. Cette solution est coûteuse, mais elle peut résoudre le problème de la redondance ou de la similarité des documents. Cependant, on risque de retomber sur la classe majoritaire si les moteurs ne sont pas assez différents.

### 4.3.2 Par reformulation de la requête

Une autre alternative pour produire de la diversité c'est que l'utilisateur puisse reformuler sa requête en rajoutant des informations supplémentaires (quelques mots-clés) afin de retourner des documents plus précises à l'utilisateur. Le fait de choisir les bons mots pour étendre la requête pourrait ne pas être évident pour l'utilisateur, car parfois il ne sait pas comment exprimer explicitement son besoin d'information en émettant des mots ambigüs qui pourraient contenir plusieurs interprétations. Dans ce cas, l'utilisateur risque de récupérer des documents qui ne sont pas en rapport avec la requête et donc de faire baisser la précision. De plus, reformuler la requête ne garantit pas de résoudre le problème de la redondance de documents.

### 4.3.3 Par pseudo-relevance feedback

Le retour de pertinence a pour but de prendre en compte le point de vue des utilisateurs sur les documents présentés par le système de recherche. Les systèmes exploitant ce concept permettent de récupérer les jugements de l'utilisateur et les exploitent pour améliorer les résultats de recherche. Un des algorithmes classiques de retour de pertinence est l'algorithme de Rocchio [50]. En utilisant l'ensemble de documents pertinents et non pertinentes sélectionnés par l'utilisateur, l'algorithme affine la requête initiale de l'utilisateur en donnant plus de poids aux caractéristiques des exemples positifs et en pénalisant les caractéristiques des exemples négatifs.

En exploitant ce concept pour améliorer la diversité, dans [6] les auteurs construisent d'abord une vérité terrain automatique en faisant l'hypothèse de ce que, pour chaque requête, les premières $N_p$ images sont des exemples positifs et les dernières $N_n$ images sont des exemples négatifs ($N_p$ et $N_n$ sont des paramètres à calculer). Puis, pour diversifier les images, les auteur utilisent d'abord une méthode de clustering agglomeratif hiérarchique

TABLE 4.1 – Comparaisons des méthodes classiques : `K-Means`, `K-Medoids`, `AHC` et `Maxmin`. $K$ est le nombre de clusters, $I$ est le nombre d'itérations et $n_D$ est le nombre de documents dans la collection $D$

| | Clustering | | | Algorithme |
|---|---|---|---|---|
| | Plat | | Hiérarchique | Glouton |
| Méthode | `K-Means` | `K-Medoids` | `AHC` | `Maxmin` |
| Complexité | $O(K \times n_D \times I)$ | $O(K \times (n_D - K)^2 \times I)$ | $O(n_D^2)$ | $O(n_D^2)$ |
| Implémentation | facile | compliqué | moyen | facile |
| Choix du nombre de cluster | oui | oui | oui | no |
| Initialisation du nombre de clusters | requis | requis | pas requis | pas requis |
| Résultat dépend de la initialisation | oui | oui | no | oui |
| Limitation de descripteurs | oui (seulement numérique) | no | no | no |

et puis ils prennent seulement les clusters pertinents en faisant l'hypothèse de qu'un cluster est non-pertinent s'il contient plus d'exemples négatifs que de positifs.

## 4.4 Comparaison des modèles de diversité

Dans la littérature de la recherche d'information, la plupart des chercheurs utilisent deux grandes approches pour augmenter la diversité : une méthode de diversité par clustering (en utilisant par exemple des méthodes de clustering plat, des méthodes de clustering hiérarchique, etc.) ; et une méthode de diversité par optimisation (en utilisant par exemple des algorithmes gloutons, etc.).

Dans le tableau 4.1 nous résumons les caractéristiques les plus importantes de quelques méthodes classiques pour produire de la diversité. Dans ce tableau on voit que la méthode `K-Means` semblerait être moins coûteuse en complexité par rapport aux méthodes `K-Medoids`, `AHC` et `Maxmin`, ce qui pourrait permettre à cette méthode d'être utilisable dans un système de recherche en "ligne". Cependant, il faut remarquer que les méthodes dans ce tableau sont des méthodes de base et dans la littérature, il existe d'autres versions plus rapides que ces algorithmes.

On voit aussi que les méthodes de clustering plat et `Maxmin` ont une étape d'initialisation ce qui fait que la qualité du résultat final peut dépendre de cette initialisation. La méthode `K-Means` a besoin du calcul de centroïdes qui n'est pas adapté pour certains descripteurs tandis que les méthodes `AHC`, `K-Medoids` et `Maxmin` peuvent être utilisables avec une plus grande variété de descripteurs.

Dans la littérature ils existent des travaux qui ont comparé différentes méthodes de clustering pour augmenter la diversité. Dans [71] quatre méthodes de clustering sont comparées : un clustering hiérarchique (`AHC`), un clustering plat (`K-Means`), un clustering de densité (MeanShift) qui n'a pas besoin de définir le nombre de clusters et le clustering "Lingo" qui est basé sur la décomposition de valeurs singulières très utilisées dans la recherche d'information par le texte. Dans [22] un clustering hiérarchique est aussi comparé avec un clustering plat (`K-Means`). Dans ces articles le clustering hiérarchique donne de meilleurs résultats en diversité. Ces travaux sont proches de notre travail parce qu'il compare des méthodes clustering classique pour augmenter de la diversité, mais ils diffèrent, car ils n'utilisent pas une arborescence de concepts, ni différents descripteurs, et ne font pas une comparaison plus générale avec d'autres méthodes de diversité sans clustering.

TABLE 4.2 – Comparaison des scores P@20, CR@20 moyens des résultats des quelques chercheurs qui ont participé au benchmark MediaEval 2014

| Équipe (Méthode) | P@20 | CR@20 |
|---|---|---|
| Baseline | 0.807 | 0.343 |
| PRa-MM (pre-filter+AHC) [11] | **0.851** | 0.469 |
| SocSens (joint relevance-diversity) [69] | 0.815 | **0.475** |
| CEALIST (pre-filter+Kmeans) [17] | 0.793 | 0.456 |
| TUW (pre-filter+mergingClustering) [42] | 0.769 | 0.450 |

**Campagnes d'évaluation** Différents benchmarks publiques existent pour évaluer la diversité dans la recherche d'images. Nous citons par exemple la campagne CLEF Cross Language Recherche d'images - ImageCLEF [1] et la campagne MediaEval Benchmarking Initiative for Multimedia Evaluation [2].

Dans ImageCLEF, la tâche de la diversité est apparue en 2008 et 2009. Afin de nous comparer avec des autres méthodes de l'état de l'art, nous utilisons l'ensemble de données de la tâche de diversité du benchmark ImageCLEF 2008 qui contient 39 requêtes. Dans la section 6.2 page 68 nous décrivons plus en détail cet ensemble de données.

Dans MediaEval la tâche de la diversité est apparue en 2013, puis en 2014 et actuellement en 2015. Dans cette tâche les photos ont été extraites du Moteur de Recherche Flickr [3] en utilisant des requêtes qui représentent divers emplacements dans le monde entier, par exemple, des musées, des sites archéologiques, des cathédrales, des routes, des ponts, etc.

Nous avons participé à la tâche de diversité du benchmark MediaEval 2013 [24] qui a utilisé un ensemble de données appelé Div400 [25] et qui contient 396 requêtes avec jusqu'à 150 photos par requête. Dans la section 6.3 page 71 nous décrivons plus en détail cet ensemble de données.

La tâche de diversité du benchmark MediaEval 2014 a utilisée un ensemble de données appelé Div150Cred [23] et qui contient moins de requêtes (153 requêtes) que l'ensemble Div400, mais avec une quantité plus grande de photos par requête (jusqu'à 300 photos par requête). Dans le tableau 4.2 nous montrons les résultats de quelques chercheurs qui ont participé à la campagne Multimedia MediaEval 2014. Dans les résultats de cette campagne, on voit que la plupart des chercheurs ont bien réussi à obtenir des meilleurs résultats que la Baseline (provenant des images de Flickr). Certains chercheurs ont utilisé des méthodes de diversité avec des prétraitements afin d'éviter d'être trop pénalisés par la diminution de la pertinence. Par exemple, en enlevant les images non-pertinentes (pre-filtering) qui ne correspondent pas au contexte de la diversité sociale (des images qui ont des visages en premier plane, des images floues, etc.), mais aussi en réordonnant les images par sa similarité à la requête (re-ranking) afin d'augmenter encore plus la pertinence. Cependant, ces prétraitements sont trop spécifiques au contexte du benchark utilisé ce qui pourrait donner des autres résultats sur un autre benchmark.

Dans ce tableau on voit aussi que le clustering hiérarchique (version améliorée de la `AHC`) utilisé dans [11] obtient de meilleurs résultats en diversité par rapport à une méthode de clustering plat `K-Means` utilisée dans [17], mais aussi à la fusion de différentes méthodes de clustering utilisées dans [42]. On voit aussi que la méthode de diversité par optimisation utilisée dans [69] obtient une diversité plus intéressante, mais avec une pénalisation de la pertinence. Il faut remarquer qu'on ne peut pas vraiment conclure sur les différentes méthodes utilisées lors de cette campagne, car certains participants

1. http ://www.imageclef.org/
2. http ://www.multimediaeval.org/
3. https ://www.flickr.com/

ajoutent des prétraitements et d'autres non, les expériences ne sont donc pas toujours exactement comparables.

# Deuxième partie

# Propositions

# Chapitre 5

# Cadre théorique et expérimental

Pour résoudre le problème de la diversité, nous choisissons d'étudier les méthodes de clustering, plus spécialement, celle hiérarchique afin de prendre en compte la nature intrinsèquement hiérarchique de la diversité. Cependant, utiliser du clustering n'est pas évident à faire, car il sert à regrouper des images et non pas à faire de la diversité.

Pour cela, dans ce chapitre nous définissons un **cadre théorique** qui nous permet d'utiliser le clustering pour la diversité. Ce cadre est résumé dans une chaîne de traitement qui est composé de deux parties principales : une première partie dont l'objectif est d'augmenter la pertinence, une deuxième partie dont l'objectif est d'augmenter la diversité.

Ce cadre nous permet d'étudier nos différentes propositions, mais aussi il nous permet d'étudier deux cas d'étude de la diversité : le cas idéal où tous les documents retrouvés sont pertinents et le cas réel où parmi les documents retrouvés un certain nombre est non-pertinents.

L'utilisation du clustering sur la diversité n'est pas évident à faire, car il y a un certain nombre de difficultés à prendre en compte. Par exemple, comment réordonner les résultats après le clustering ? ce dernier ne fournit pas d'ordre sur les clusters ni sur les images, comment déterminer celui des clusters et des images pour améliorer la diversité des résultats ? Dans ce chapitre, nous décrivons des approches classiques et quelques solutions proposées qui vont répondre aux difficultés rencontrées.

De même, l'utilisation d'une arborescence de concepts comme descripteur de l'image, dans les méthodes de clustering, n'est pas évidente à faire, car nous n'avons pas trouvé dans la littérature une mesure qui calcule la similarité dans le cas où les images sont associées à plusieurs concepts de l'arborescence. Pour cela, dans ce chapitre, nous proposons une méthode généralisée qui répond à cette difficulté.

Dans ce chapitre, nous définissons aussi un **cadre expérimental** qui nous permettra de savoir si nos méthodes proposées ont permis d'augmenter la diversité. Pour éviter que les résultats obtenus ne soient pas dépendants d'un benchmark, nous avons volontairement choisi de travailler sur trois benchmarks assez différents en nombre de requêtes, en nombre d'images, en type de descripteurs... et en provenant de sources différentes ce qui nous permet d'obtenir des comportements comparables et plus généraux. Parmi les trois benchmarks, nous en avons construit un qui nous permet d'étudier l'intérêt d'utiliser l'arborescence de concepts sur la diversité et nous en utilisons deux autres publics qui nous permettent de nous comparer avec d'autres méthodes de diversité de l'état de l'art.

Pour mesurer la pertinence et la diversité nous utilisons les métriques classiques d'évaluation de précision et de cluster rappel sur les $n$ premières images. Nous utilisons principalement ces métriques, car elles sont très utilisées dans la plupart des benchmarks publics.

Dans la section 5.1 nous décrivons d'abord le cadre théorique où nous parlons de

notre positionnement, notre chaîne de traitement proposée ainsi que la description et les limitations trouvées à chaque étape. Puis, la section 5.2 nous décrivons un cadre expérimental où nous citons les benchmarks utilisés, les différents cas d'étude de la diversité et les limitations des métriques d'évaluation utilisées. Enfin, dans la section 5.3 nous résumons les points importants du cadre théorique et expérimental.

## 5.1 Cadre théorique

Afin d'étudier le problème de la diversité, dans cette section nous définissons un cadre général étudié qui nous permettra de nous concentrer sur un aspect de la diversité. Puis, nous décrivons une chaîne de traitement que nous avons développée et qui nous permettra d'étudier l'utilisation du clustering pour augmenter la diversité tout en conservant une pertinence acceptable.

### 5.1.1 Cadre général étudié (positionnement)

Le problème de la diversité est très vaste et apparaît dans plusieurs domaines. Par exemple, dans la recherche d'information, dans les systèmes de recommandation, etc. Dans cette thèse, nous nous plaçons dans un **cadre général** basé sur la recherche d'images en ligne. La définition d'un cadre général nous permettra de nous concentrer sur un aspect spécifique de la diversité sans être perturbé par des autres facteurs spécifiques aux autres domaines de la diversité.

Dans ce cadre étudié, un utilisateur effectue une requête, le SRI lui renvoie alors un ensemble d'images ordonnées. Ces résultats sont de plus en plus pertinents, mais, dans la plupart des cas, les résultats similaires ont tendance à se regrouper entre eux. L'utilisateur peut être intéressé à retrouver des documents qui soient certes tous pertinents par rapport à sa requête, mais aussi qui soient différents les uns des autres.

Dans ce cas, le problème de la diversité peut être vu comme un problème de réordonnancement. Dans la littérature, il en existe diverses méthodes qui permettent de diversifier les images. **Notre solution étudiée c'est le clustering** ce qui n'est pas évident à faire, car il sert à regrouper des images et non pas à faire de la diversité. Donc, il nous faut proposer un cadre théorique qui nous permette d'utiliser le clustering pour la diversité.

### 5.1.2 Chaîne de traitement

Nous proposons un cadre théorique qui est résumé dans une **chaîne de traitement** que nous avons développée avec l'objectif d'augmenter la diversité tout en conservant une pertinence acceptable. Cette chaîne de traitement est divisée en quatre étapes :

1. Réordonner des images pour améliorer la pertinence.
2. Regrouper les résultats en utilisant une méthode de clustering.
3. Trier les clusters obtenus sur la base d'un critère de priorité, puis trier les images qui sont à l'intérieur de chaque cluster.
4. Enfin, réordonner les résultats en alternant les images de différents clusters.

Remarquons qu'à l'étape 2, la plupart des méthodes de clustering ne prennent pas en compte le rang de l'image obtenue à l'étape 1. Cependant, à l'étape 3, quand nous ordonnons les clusters et les images dans chaque cluster, nous utilisons le rang obtenu à l'étape 1.

Dans les prochaines parties nous allons expliquer les différentes composantes de notre chaîne de traitement et nous expliquerons pourquoi nous en avons besoin dans cet ordre.

### 5.1.3 Étape 1 : Amélioration de la pertinence par réordonnancement des images

Nous savons que dans le cadre étudié nous pouvons trouver le problème de la diversité et le problème de la pertinence. Cependant, comme notre objectif principal est d'augmenter la diversité, dans cette étape de notre chaîne de traitement nous utiliserons seulement des réordonnancements classiques des images par similarité à la requête pour augmenter la pertinence. De plus, ceux-ci n'utilisent pas de lourds calculs ce qui rend ces méthodes adaptées dans une application de recherche d'images en ligne.

Le réordonnancement classique consiste simplement à calculer les scores entre le descripteur de la requête et les descripteurs de chaque image, puis à réordonner les images par son score obtenu. Nous supposons que nous disposons de plusieurs ressources (texte, visuel, GPS, etc.) et plus particulièrement de l'arborescence de concepts de Xilopix. Pour calculer le score entre la requête et chaque image, nous utilisons des mesures classiques de similarité liées au type de descripteur : pour des descripteurs visuels nous utilisons la distance "Euclidean" (voir section 2.3.3 page 23), pour le texte nous utilisons la similarité "Cosinus" (voir équation 2.3 page 20), pour le GPS nous utilisons la similarité "Haversine" (voir section 2.4.2 page 24), etc. Dans cette étape nous avons plutôt utilisé chaque ressource de manière indépendante, alors qu'il aurait été intéressant de combiner plusieurs ressources.

### 5.1.4 Étape 2 : Regroupement des images par clustering

Pour faire un regroupement des images par clustering, les méthodes de clustering se servent généralement d'une **mesure de similarité** ou de distance entre deux images liées au type descripteur à utiliser. Par exemple, si on dispose des histogrammes visuels on peut utiliser une distance "Euclidean", si on dispose du texte on peut utiliser une similarité "Cosinus", etc. L'idéal c'est de pouvoir utiliser une mesure de similarité et un descripteur (visuel, textuel, etc.) qui puissent aider la méthode de clustering à faire des regroupements les plus similaires aux sous-thèmes de la vérité terrain.

#### 5.1.4.1 Mesures de similarité

Pour calculer la similarité entre deux images, plusieurs travaux ont utilisé différentes mesures de similarité ou des distances classiques comme celle "Euclidean" pour les descripteurs visuels, la similarité "Cosinus" pour le texte ou la similarité "Haversine" pour le GPS. Dans le cas d'une arborescence de concepts, une image est associée à un ensemble de chemins dans cette arborescence. Par exemple, dans la figure 1.4 page 7 on voit que l'image est associée à deux chemins : le chemin du concept "Italy" et le chemin du concept "lighthouse".

Si chaque image est associée à **un seul chemin**, nous pouvons utiliser deux mesures de similarité classique : la similarité de Wu-Palmer (voir section 2.5.1.3 page 27) ou la similarité de Lin (voir section 2.5.1.3 page 27).

**Méthode de généralisation entre deux ensembles de chemins** Dans le cas où chaque image est associée à **un ensemble de chemins**, nous n'avons pas trouvé dans la littérature des mesures de similarité. Pour cette raison, nous proposons une méthode de généralisation entre deux ensembles de chemins. Nous faisons cependant l'hypothèse que, pour un univers donné, chaque image est décrite tout au plus par un seul concept. De plus, pour avoir un modèle identique dans le cas des descripteurs visuels (où la distance euclidienne est utilisée), et dans le cas des descripteurs basés sur l'arborescence de concepts, nous souhaitons obtenir une dissimilarité plutôt qu'une similarité. C'est

pourquoi au lieu d'utiliser $sim_{WP}$, nous utilisons $\delta_{WP} = 1 - sim_{WP}$, et de même au lieu de $sim_{LIN}$, nous utilisons $\delta_{LIN} = 1 - sim_{LIN}$.

La dissimilarité $\delta_G$ entre l'ensemble de chemins $A_{i_1}$ de l'image $i_1$ et l'ensemble de chemins $A_{i_2}$ de l'image $i_2$ est définie par la formule suivante :

$$\delta_G(A_{i_1}, A_{i_2}) = \frac{1}{|J(i_1) \cup J(i_2)|} \sum_{j \in J(i_1) \cup J(i_2)} \delta(A^j_{i_1}, A^j_{i_2}) \tag{5.1}$$

où $J(i_1)$ (resp. $J(i_2)$ ) est l'ensemble des univers de l'image $i_1$ (resp. $i_2$) et $\delta(A^j_{i_1}, A^j_{i_2})$ est une dissimilarité entre le chemin de l'image $i_1$ et le chemin de l'image $i_2$ dans l'univers $j$, dans ce travail, nous utilisons soit $\delta_{WP}(A^j_{i_1}, A^j_{i_2})$ soit $\delta_{LIN}(A^j_{i_1}, A^j_{i_2})$.

Par exemple, si nous reprenons l'image $i_1$ de la figure 1.4 page 7, cette image est associée aux concepts "Italy" dans l'univers 1 ("Travel") et "lighthouse" dans l'univers 2 ("Concept"), c'est-à-dire : $A_{i_1} = (A^1_{i_1}, A^2_{i_1})$. Prenons une autre image $i_2$ associée aux concepts "Spain" dans l'univers 1 ("Travel") et "truck" dans l'univers 3 ("Transport"), c'est-à-dire : $A_{i_2} = (A^1_{i_2}, A^3_{i_2})$. Pour ces deux ensembles des chemins, le calcul de la dissimilarité généralisée en utilisant Wu-Palmer est :

$$\begin{aligned} \delta_G(A_{i_1}, A_{i_2}) &= \tfrac{1}{3}(\delta(A^1_{i_1}, A^1_{i_2}) + \delta(A^2_{i_1}, A^2_{i_2}) + \delta(A^3_{i_1}, A^3_{i_2})) \\ &= \tfrac{1}{3}(1 - sim_{WP}(A^1_{i_1}, A^1_{i_2}) + 1 + 1) = 1 - \tfrac{1}{3}\tfrac{2\times 2}{3+3} = 0.778 \end{aligned}$$

Cette mesure suppose qu'une image décrite par un seul mot est plus proche d'une image décrite par un seul mot du même univers, que d'une image décrite par un mot du même univers et d'autres mots en provenance d'autres univers. Par exemple, une image décrite par "avocat" sera plus proche d'une image décrite par "magistrat" que d'une image décrite par "avocat" et "robe", car les magistrats et les avocats appartiennent au même univers "Justice", tandis que "robe" appartient à un autre univers.

#### 5.1.4.2 Choix du nombre de clusters

La plupart des méthodes classiques de clustering ont besoin de savoir le nombre de clusters à utiliser pour regrouper les images. Cependant, le problème du choix du nombre de clusters est un problème qui n'a pas de réponse unique. Dans cette section nous citons deux techniques de découpage de base totalement utilisables dans la plupart des méthodes de clustering classiques.

**Découpage Fixe** Un choix simple est de choisir un nombre fixe de clusters, méthode que nous appelons `Fixe`($n$) où $n$ est le nombre de clusters à obtenir.

**Découpage Adapté** Nous pouvons également adapter le nombre de clusters pour chaque requête si nous connaissons le nombre de sous-thèmes attendus pour chaque requête (méthode que nous appelons `Adapt`). Dans la réalité, on ne connaît pas le nombre de sous-thèmes pour une requête donnée, mais cette méthode `Adapt` est intéressante comme élément de comparaison. Pour connaître le nombre de sous-thèmes, nous utilisons la vérité terrain.

### 5.1.5 Étape 3 : Ordonnancement des clusters

Une fois que nous avons regroupé les images, nous obtenons une liste non-ordonnée de clusters. Nous devons alors déterminer la priorité des clusters, c'est-à-dire dans quel ordre ils vont être utilisés. Lorsqu'une requête est effectuée, les résultats sont souvent ordonnés en fonction de leur similarité à la requête [22].

**Priorité de clusters selon le rang des images** Une manière classique est de prendre en premier le cluster qui contient l'image de rang 1, puis en deuxième celui qui contient celle restante de rang le plus fort et ainsi de suite (priorité que nous appelons `Rank`). Cependant, lorsque beaucoup d'images sont très pertinentes, cet ordre n'est pas toujours très significatif.

**Priorités de clusters selon la taille des clusters** Afin de trouver un ordre de priorité des clusters, nous proposons de prendre en compte le nombre d'images que contient chaque cluster. Si les clusters sont ordonnés du plus grand nombre d'images au plus petit nombre d'images (priorité que nous appelons `Decreasing`), les premiers résultats seront des images que l'utilisateur s'attend à trouver pour sa requête ; dans le cas contraire (priorité que nous appelons `Increasing`), les premiers résultats montreront en premier des images originales. Par exemple, si nous ordonnons les clusters de manière décroissante (priorité `Decreasing`), nous obtenons pour la requête "phare" d'abord des images de phares maritimes, puis les images de phares de voiture. Inversement, si nous ordonnons les clusters de manière croissante (priorité `Increasing`), nous obtenons d'abord les images de phares de voiture, puis les phares maritimes.

### 5.1.6 Étape 4 : Réordonnnacement des images par cluster

Enfin, pour produire une liste d'images diversifiées à partir des clusters générés (obtenu par la méthode de clustering) et ordonnés (obtenu par les techniques de priorité), nous utilisons un réordonnancement plat qui est une méthode classique pour réordonner les images (méthode que nous appelons `FlatO`). Dans cette méthode, les images sont réordonnées en prenant dans l'ordre des clusters une image de chacun d'entre eux.

Par exemple, à partir de la requête textuelle "jaguar" le système retourne une liste d'images qui sont certes toutes pertinentes, mais pas diversifiées. Puis, la méthode de clustering sépare ces images en deux clusters : des images qui sont liées au thème "jaguar animal" et celles liées à celui de "jaguar voiture". Enfin, le résultat final, en utilisant la méthode classique `FlatO`, réordonnera les images en alternant les images du sous-thème "jaguar animal" et celles de celui "jaguar voiture".

## 5.2 Cadre expérimental

Après avoir défini un cadre théorique, dans cette section nous abordons la définition d'un cadre expérimental qui nous permettra de tester et d'étudier les méthodes de diversité dans le but de savoir s'ils ont permis de résoudre le problème de la diversité.

Pour rendre ce cadre expérimental adapté à notre cadre théorique, nous avons pris en compte trois aspects importants : le cas d'étude de la diversité, l'utilisation de plusieurs benchmarks et l'utilisation de plusieurs descripteurs, ce qui nous permet d'obtenir des comportements les plus généraux possibles.

Dans cette section nous montrons les benchmarks utilisées dans ce cadre expérimental, puis nous montrons les méthodes de référence de réordonnancement de base et enfin nous montrons les métriques d'évaluations à utiliser.

### 5.2.1 Benchmarks

Dans notre cadre expérimental nous utilisons trois benchmarks : XiloDiv, ImageCLEF 2008 et MediaEval 2013. Ces benchmarks sont décrits plus en détail dans la section 6.1 page 64 pour XiloDiv, la section 6.2 page 68 pour ImageCLEF 2008 et la section 6.3 page 71 pour MediaEval 2013 (voir chapitre 6).

Les benchmarks ImageClef 2008 et MediaEval 2013 sont des benchmarks publics utilisés lors de conférences internationales et qui nous permettent de faire une comparaison avec les autres méthodes de diversité. Ces benchmarks possèdent une grande quantité de requêtes qui nous donnent des scores moyens très robustes aux cas particuliers.

Le benchmark MediaEval 2013 possède plusieurs types de descripteurs qui permettent d'étudier l'influence de ceux-ci sur la diversité. Cependant, nous n'avons pas trouvé un benchmark public qui possède une arborescence de concepts. Pour cette raison nous avons construit le benchmark XiloDiv avec la collaboration d'une entreprise qui dispose de cette arborescence et qui nous permet d'étudier l'influence de cette ressource sur la diversité.

L'utilisation de plusieurs benchmarks offre la possibilité d'étudier la diversité sur différents types de baselines. Les benchmarks MediaEval 2013 et XiloDiv utilisent une baseline provenant de grands bases de données avec beaucoup d'images pertinentes. Par contre, le benchmark ImageClef 2008 utilise une baseline provenant d'un corpus de 20 000 images avec peu d'images pertinentes.

Nous avons généré un descripteur en commun (histogramme de couleur) sur les trois benchmarks ce qui nous permet de comparer les résultats obtenus sur les trois benchmarks et obtenir ainsi des résultats plus généraux.

### 5.2.2 Méthodes de référence de réordonnancement

Dans cette section nous allons décrire différentes méthodes de référence de réordonnancement pour faire des points de comparaisons de base avec nos méthodes proposées.

#### 5.2.2.1 Cas réel et cas idéal

Lorsqu'on fait une requête, les résultats offerts par un moteur de recherche contiennent des images pertinentes et des images non-pertinentes. La précision des résultats de la requête dépend de divers facteurs, notamment :

- la difficulté de la requête : pour certains requêtes, retrouver les documents pertinents est plus facile que d'autres ;
- la qualité des annotations des images : annotations manuelles, automatiques, tags ;
- l'application utilisée : dans certaines applications, comme par exemple les systèmes de recommandation, tous les documents retrouvés sont en rapport avec la requête, alors que l'utilisateur est seulement intéressé par diversifier les résultats et non par augmenter la pertinence.

Puisque la pertinence des résultats dépend de nombreux facteurs, nous avons choisi d'étudier le problème de la diversité dans deux cas différents. Premièrement, dans le cas où les requêtes sont effectués sur un moteur de recherche d'images classique, l'ensemble des résultats obtenus contient des images pertinentes et des images non pertinentes, nous appelons ce cas le **cas réel**.

Deuxièmement, pour étudier le problème de la diversité sans être biaisé par le problème de la pertinence, nous allons prendre les mêmes résultats que précédemment pour les mêmes requêtes, mais en gardant seulement les documents pertinents (dans l'ordre des résultats de la requête). C'est le **cas idéal** (sans images non-pertinentes).

Dans cette thèse, les résultats ordonnés qui nous servent de point de départ pour lancer notre chaîne de traitement sont appelés *baseline* dans le cas réel et *idealBaseline* dans le cas idéal.

#### 5.2.2.2 Diversité aléatoire (Random)

Lorsqu'on effectue une requête, généralement, les documents résultats similaires ont tendance à se regrouper. Une manière très simple de produire une certaine diversité est de faire $n$ permutations aléatoires de ces résultats. Nous proposons de comparer les ordonnancements obtenus par nos méthodes avec les ordonnancements obtenus par une méthode de diversité par permutation aléatoire des images résultats. Nous avons générés 10 permutations aléatoires pour les benchmark XiloDiv et MediaEval 2013, et les auteurs de [60] ont généré 9 permutations pour le benchmark ImageCLEF 2008. La moyenne des scores de ces 9 ordonnancements aléatoires nous donne le score de la méthode que nous appelons **Random**.

### 5.2.3 Métriques d'évaluation

Dans cette thèse nous étudions le problème de la diversité dans le cadre de la recherche d'information qui peut présenter aussi le problème de la pertinence. Pour mesurer la diversité et la pertinence dans notre cadre expérimental, nous utilisons les métriques d'évaluation de cluster rappel (CR@n) et de précision (P@n) sur les $n$ premières images (décrites dans la section 2.6 page 28). Nous choisissons ces mesures, car elles sont des mesures classiques très utilisés dans la plupart des benchmarks publics.

#### 5.2.3.1 Scores maximum théorique (Pmax et CRmax)

Si on étudie précisément les scores de précision et de cluster rappel qui vont être abordées ci-après, on notera que certains cas particuliers font que les scores, normalement sont compris entre 0 et 1, ne peuvent jamais atteindre 1. Cela vient du fait que ces mesures s'intéressent à un nombre $n$ données de résultats.

En ce qui concerne la *précision*, nous appelons **Pmax** la précision maximale théoriquement possible. Généralement, la plupart des requêtes peuvent atteindre la valeur 1, mais si le nombre de documents pertinents est inférieur au nombre de documents considérés, alors la valeur de Pmax ne sera pas 1.

Par exemple, pour une requête donnée, si on évalue ses 10 premières images résultats et en sachant que dans la vérité terrain de la requête nous avons 2 images pertinentes, alors la valeur Pmax va être égal à 0.2. Donc, pour cette requête on ne peut pas obtenir une valeur de précision P@10 supérieur à 0.2.

En ce qui concerne le *cluster rappel*, nous appelons **CRmax** la valeur maximale de cluster rappel théoriquement possible. Si le nombre de sous-thèmes d'une requête est plus grand que celui des documents considérés, alors le CRmax ne sera pas 1.

Par exemple, pour une requête donnée, si on évalue ses 10 premières images résultats en sachant que dans la vérité terrain de la requête ses images pertinentes ont été regroupées en 12 groupes, alors la valeur maximale de cluster rappel (CRmax) est 0.83. Donc, pour cette requête on ne peut pas obtenir une valeur de cluster rappel CR@10 supérieur à 0.83.

#### 5.2.3.2 Influence du nombre d'images à évaluer

Dans la comparaison des scores CR@n entre différentes méthodes de diversité, le choix de la valeur $n$ (nombre d'images à évaluer) peut être très important afin de mieux voir la différence en diversité entre ces méthodes. Pour cela, il serait intéressant de voir, pour chaque valeur CR@n, la différence en diversité entre une méthode sans diversité (baseline) et la valeur CRmax qui est le score maximum de cluster rappel théoriquement possible. Comme on est dans une application d'un moteur de recherche d'images, l'utilisateur veut rapidement trouver les images qui l'intéresse, il ne veut pas parcourir 100 ou 200

images. Donc avoir un système qui est bon en diversité en CR@100 ou CR@200 n'est pas intéressant, ce qu'on veut c'est un CR@n qui soit bon pour des valeurs $n$ petites. En général, on choisit CR@20 parce que cela correspond au nombre d'images que l'on peut afficher sur un écran, mais il peut être intéressant d'étudier d'autres valeurs de CR@n pour étudier le comportement des méthodes pour d'autres valeurs pour voir par exemple si il n'y a pas de cas particulier intéressant. Pour cette raison, dans cette étude, nous choisissons des valeurs $n$ entre 5 et 50.

Dans la figure 5.1 nous montrons les scores CR@n moyens de la baseline et des scores CRmax en utilisant différentes valeurs $n$ d'images à évaluer dans le cas idéal et réel sur les benchmarks (a) (b) MediaEval 2013, (c) (d) ImageClef 2008 et (e) XiloDiv. Dans ces figures on voit que les courbes CRmax augmentent jusqu'à attendre 1 quand $n$ est supérieur au nombre moyen de sous-thèmes. On voit bien que lorsque $n$ est inférieur au nombre de sous-thèmes de la vérité terrain, le CRmax n'atteint pas 1. En ce qui concerne les courbes de la baseline, il est à noter qu'elles augmentent progressivement. Afin de choisir le bon nombre $n$ d'images à évaluer, nous expliquons d'abord le choix des valeurs extrêmes (CR@5 et CR@50), puis nous choisissons un bon compromis entre les deux cas.

Si on choisit CR@5, qui utilise un nombre $n$ d'images inférieur au nombre moyen des sous-thèmes, on voit que, dans la plupart des cas, les scores de CRmax et la baseline sont très proches, car probablement il suffit à la baseline de trouver quelques sous-thèmes pour être proche du CRmax (des sous-thèmes qui sont faciles à retrouver) ce qui peut rendre ce choix pas très discriminant si on veut comparer des autres méthodes de diversité.

Si on choisit CR@50, qui utilise un nombre $n$ d'images supérieur au nombre moyen des sous-thèmes, on voit que, dans le cas idéal, les scores de CRmax et la baseline sont très proches, car probablement le fait d'évaluer beaucoup d'images facilite à la baseline de trouver beaucoup de sous-thèmes ce qui peut rendre aussi ce choix pas discriminant au moment de comparer des autres méthodes de diversité. Dans le cas réel on n'obtient pas ce comportement à causes des images non-pertinentes qui existent dans ses baseline.

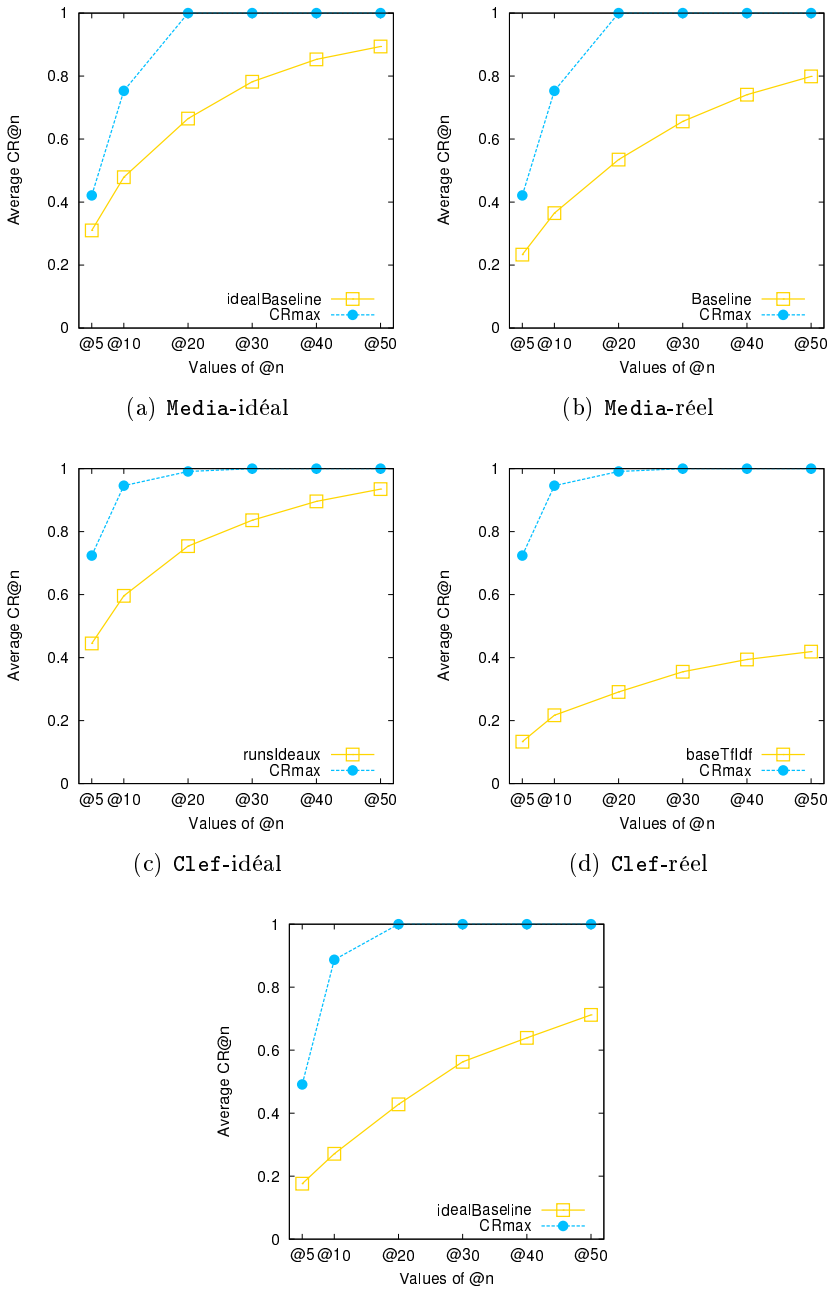
Donc, de manière générale, le choix de CR@20 semble être un bon compromis entre les deux cas précédents, car c'est un choix minimal pour attendre les plus de sous-thèmes (CRmax = 1) et pour utiliser le moins nombre d'images à évaluer ce qui nous permettra de mieux voir la différence en CR entre deux méthodes de diversité. De plus, les valeurs CR@10 et CR@20 correspond à un scénario d'usage typique où les valeurs $n$ sont, en particulier, le nombre d'images qui correspondent à la fenêtre de résultats d'un moteur de recherche et que l'utilisateur consulte en priorité par rapport aux nombreux résultats. Ces valeurs nous servent aussi pour se comparer aux autres méthodes de l'état de l'art. Donc, pour les figures importantes nous utiliserons les scores CR@10 et CR@20.

## 5.3 Résumé

Comme le problème de la diversité est très vaste et se retrouve dans plusieurs domaines, nous nous plaçons dans un cadre général basé sur la recherche d'images en ligne afin de nous concentrer sur un aspect spécifique de la diversité.

Pour résoudre le problème de la diversité, nous choisissons les méthodes de clustering ce qui n'est pas évident à faire, car il sert à regrouper des images et non pas à faire de la diversité. Pour utiliser le clustering sur la diversité, nous proposons un cadre théorique qui se résume dans une chaîne de traitement que nous avons développée et qui est composée de deux parties principales : une première partie dont l'objectif est d'augmenter la pertinence, la deuxième dont l'objectif est d'augmenter la diversité.

Dans cette chaîne nous proposons quelques solutions aux difficultés rencontrées pour utiliser du clustering sur la diversité.




(a) `Media`-idéal (b) `Media`-réel

(c) `Clef`-idéal (d) `Clef`-réel

(e) `Xilo`-idéal

FIGURE 5.1 – Comparaison des scores de diversité. Les scores CR@n moyens représentent la baseline (en jaune) et la valeur CRmax (en bleu) en utilisant différentes valeurs $n$ d'images à évaluer (en abscisse) dans un cas idéal et réel sur les benchmarks (a) (b) `Media` (sous-ensemble `testset`), (c) (d) `Clef` et (e) `Xilo`

— Pour choisir le **nombre de clusters optimal**, nous allons comparer deux techniques de découpage : une technique de découpage `Fixe` de $n$ clusters et une technique de découpage idéal (`Adapt`) qui prend un nombre de clusters égal au nombre de sous-thèmes de la vérité terrain.
— Pour déterminer l'**ordre des clusters**, nous allons comparer une technique classique de priorité par le rang (`Rank`) avec deux techniques de priorité que nous proposons (`Increasing` et `Decreasing`) qui trient les clusters selon leur taille (en

fonction du nombre d'images).
- Pour déterminer l'**ordre des images** dans le cluster, nous allons utiliser une technique classique de tri par le rang.
- Pour **réordonner les résultats** après le clustering, nous allons utiliser une méthode classique (`FlatO`) qui alterne les images de chaque cluster.

Un inconvénient apparaît dans l'utilisation d'une arborescence de concepts comme descripteur dans la recherche d'images car dans la littérature, nous n'avons pas trouvé une mesure qui calcule la similarité dans le cas d'association d'images à plusieurs concepts de l'arborescence. Pour résoudre ce problème, dans cette chaîne de traitement, nous proposons une méthode généralisée qui répond à cette difficulté.

Ce cadre nous permet aussi d'étudier deux cas d'étude de la diversité : le cas idéal où tous les documents retrouvés sont pertinents et le cas réel où parmi les documents retrouvés un certain nombre est non-pertinents.

Nous définissons aussi un **cadre expérimental** qui nous permet de savoir si les méthodes proposées ont permis d'augmenter la diversité. Pour éviter que les résultats obtenus ne soient pas dépendants d'un benchmark, nous avons volontairement choisi de travailler sur trois benchmarks assez différents en nombre de requêtes, d'images, en types de descripteurs... et en provenant de sources différentes ce qui nous permet d'obtenir des comportements comparables et plus généraux. Parmi les trois benchmarks, nous avons construit un benchmark qui permet d'étudier l'intérêt d'utiliser l'arborescence de concepts sur la diversité et nous utilisons deux benchmarks publics qui nous permet de comparer avec des autres méthodes de diversité de l'état de l'art.

Pour mesurer la diversité et la pertinence nous utilisons les mesures de cluster rappel et de précision qui sont des métriques d'évaluation très utilisés dans la plupart des benchmarks publics liés à la diversité d'images.

# Chapitre 6

# Contextes expérimentaux

Dans ce chapitre nous montrons les résultats obtenus des contextes expérimentaux dans chaque benchmark. Dans notre cadre expérimental nous utilisons trois benchmarks : deux benchmarks publics ImageClef 2008 et MediaEval 2008 et un benchmark XiloDiv que nous avons construit.

Dans le benchmark XiloDiv nous disposons d'une arborescence de concepts qui est pour nous une ressource très intéressante pour augmenter la diversité. Nous proposons d'utiliser cette ressource comme descripteur afin de l'utiliser dans la `AHC` et pour diversifier les images. Dans nos expérimentations nous comparons l'intérêt de cette proposition avec l'utilisation directe de cette arborescence sur la diversité. Nous comparons aussi cette proposition avec l'utilisation d'un autre descripteur (histogramme de couleur) dans la `AHC`.

Dans le benchmark MediaEval 2013 nous disposons de plusieurs types de descripteurs (du texte, du visuel, des coordonnées GPS, etc.). Dans ces expérimentations, nous utilisons les étapes de notre chaîne de traitement pour comparer l'utilisation de différents descripteurs pour augmenter la pertinence (étape 1 de notre chaîne de traitement) et puis pour augmenter la diversité (étapes 2, 3 et 4 de notre chaîne de traitement).

Contrairement aux benchmarks XiloDiv et MediaEval qui ont une baseline provenant des moteurs de recherche avec un grand nombre d'images pertinentes, le benchmark ImageClef 2008 a une baseline provenant d'un corpus de 20 000 images qui contient peu d'images pertinentes. Cette particularité nous permet d'étudier la diversité avec un autre type de baseline dans un cas réel et dans un cas idéal.

Les benchmarks publics ImageClef 2008 et MediaEval 2013 ont permis de nous comparer avec d'autres méthodes de diversité de l'état de l'art.

Dans la partie 6.1, 6.2 et 6.3 nous présentons les caractéristiques des benchmarks utilisés : nombre d'images, de requêtes, les descripteurs disponibles, les scores des baselines, etc., ainsi que l'étude des scores obtenus sur les différents descripteurs de chaque benchmark afin de trouver les descripteurs qui donnent globalement un bon compromis en pertinence et en diversité et de détecter les descripteurs qui donnent de mauvais résultats pour éviter de les prendre par la suite de nos expérimentations. Dans la partie 6.4 nous comparons nos méthodes proposées avec d'autres modèles de l'état de l'art lors de campagnes internationales. Enfin, dans la partie 6.5 nous concluons et montrons les avantages et les inconvénients d'utiliser chaque benchmark, ainsi que les paramètres choisis de nos méthodes proposées dans chaque benchmark pour les utiliser dans les autres chapitres.

```
<photos title="animaux">
  <photo data_taken="2011-04-12" id="146067" rank="0"
  latitude="45.7021408081054700" longitude="13.7136735916137700"
  tags="Bec, Plumes, Parois, Plumages " url_b="jfge-2515Sony"
  title="Cormoran accroché à la falaise " username="7564" />
...
</photos>
```

FIGURE 6.1 – Description sémantique du contenu d'une image disponible sur le benchmark `Xilo`

TABLE 6.1 – Les 21 requêtes du benchmark `Xilo` (*subthemes* est le nombre de sous-thèmes pour chaque requête dans la vérité terrain)

| Requête | subthemes | Requête | subthemes | Requête | subthemes |
|---|---|---|---|---|---|
| ancien | 11 | fond | 12 | île | 11 |
| animaux | 12 | commonwealth | 12 | marie | 11 |
| aquatique | 10 | composition | 16 | mer | 11 |
| blanc | 12 | corde | 7 | rouge | 16 |
| bleu | 12 | fer | 6 | tradition | 12 |
| bois | 10 | fête | 6 | transport | 12 |
| ciel | 9 | grande | 11 | voiture | 9 |

## 6.1 Contexte XiloDiv (`Xilo`)

Nous avons construit le benchmark `Xilo` [4] en partenariat avec l'entreprise Xilopix. Les images et leurs descripteurs textuels sont mis à disposition par l'entreprise. Ce benchmark est donc le reflet d'une application réaliste.

Nous avons construit ce benchmark parce que nous n'avons pas trouvé un benchmark de référence qui possède une arborescence de concepts.

### 6.1.1 Descripteurs disponibles

Le texte associé aux images consiste généralement en quelques mots-clés choisis manuellement, ainsi toutes les images retrouvées sont en rapport avec les termes de la requête. Dans la figure 6.1 nous montrons un exemple des métadonnées d'une image et qui contient : un titre, des mots-clés ("tags"), parfois des coordonnées GPS ("latitude", "longitude"), la date de l'image et l'utilisateur qui a pris l'image.

Cette entreprise a construit manuellement une arborescence de concepts où chaque image est associée manuellement à une ou plusieurs concepts de l'arborescence de concepts de Xilopix. Cette arborescence est composée en plusieurs univers (faune, flore, voyage, personne, etc.) où chaque univers a une structure d'arbre. Par exemple, dans la figure 1.4 page 7 nous avons une image qui est associée à deux concepts cette arborescence : le concept *Italie* ("Italy") qui appartient à l'univers *Voyage* ("travel") ; et le concept *sémaphore* ("lighthouse") qui appartient à l'univers *Concept* ("concept"). Cette arborescence utilise 13 univers différents et la profondeur moyenne de l'arborescence de concept est de 6.54. En moyenne sur les images du corpus, il y a 1.75 concepts associés à une image (maximum 3 concepts).

De plus, nous avons extrait de chaque image un histogramme de couleurs dans l'espace HSV (voir l'annexe B page 197).


FIGURE 6.2 – Interface de groupement d'images par rapport à la requête "fleur"

### 6.1.2 Construction d'une vérité terrain

La vérité terrain de ce benchmark a été construite en partenariat avec les documentalistes de l'entreprise selon le protocole suivant :

— Les documentalistes ont choisi 21 requêtes qui offrent une grande diversité parmi les requêtes les plus demandées par les utilisateurs dans les logs du moteur de recherche ;

— Pour chacune des 21 requêtes, les 100 premiers résultats retournés ont été récupérés. Ces résultats sont tous pertinents, car les images sont annotées à la main. Par contre, les résultats ne sont pas diversifiés ;

— Après avoir obtenu les 21 requêtes, les documentalistes ont groupé les images résultats afin de construire la vérité terrain. Pour faciliter cette tâche, nous avons développé une interface qui est montrée dans l'image 6.2. Par exemple, pour la requête "voiture", les documentalistes ont analysé et groupé les images résultats de cette requête en 9 sous-thèmes (voiture dans la ville, voiture de foire, voiture ancienne, voiture neuve, voiture crachée, voiture artistique, volant de voiture, voiture d'enfants, voiture du train). Le tableau 6.1 donne la liste des 21 requêtes.

Pour les 21 requêtes, il y a en moyenne 10.9 sous-thèmes (minimum 6 sous-thèmes, maximum 16 sous-thèmes) par requête.

### 6.1.3 Résultats obtenus

Le tableau 6.2 présente les principaux résultats obtenus sur ce benchmark. La baseline correspond aux résultats obtenus par le moteur de recherche de l'entreprise (sans diversification). La pertinence est proche de 1 parce que les images ont été annotées manuellement. Cependant, le CR@20 de la baseline est seulement de 0.428, ce qui signifie que les résultats similaires ont tendances à se regrouper. En effet, les images étant annotées avec des mots-clés, beaucoup d'images ont les mêmes indexations, et sont donc ordonnées par rapport à leur date d'indexation. Or généralement, les images indexées au

TABLE 6.2 – Synthèse des principaux résultats sur le benchmark `Xilo` (21 requêtes). Le temps est le temps moyen de réordonnancement par requête (en seconde)

| Méthode | CR@20 moyen | Temps |
|---|---|---|
| Baseline | 0.428 (ref) | - |
| CRmax | 1.000 (+134%) | - |
| Random | 0.673 (+57%) | - |
| XiloTree `Increasing` | 0.793 (+85%) | - |
| XiloTree `Decreasing` | 0.800 (+87%) | - |
| XiloTree `Rank` | 0.751 (+75%) | - |
| `AHC`(HSV) `Rank fixed20` | 0.733 (+71%) | 1.20 |
| `AHC`(Tree) `Rank fixed20` | 0.851 (+99%) | 0.90 |

même moment ont des mots-clés similaires. À partir de ce résultat ordonné, nous allons maintenant essayer d'améliorer la diversité.

En permutant aléatoirement les résultats de chaque requête (méthode que nous appelons Random), le CR@20 augmente à 0.673.

Pour améliorer la diversité, nous lançons une `AHC` pour chacun des résultats ordonnés des 21 requêtes du benchmark `Xilo` et pour chaque descripteur disponible. Dans la `AHC` nous utilisons les paramètres suivants : pour le descripteur de couleur HSV nous utilisons le critère 'agrégation "Average" et la distance "Euclidean" ; et pour l'arborescence de concepts (abrégé en "Tree") nous utilisons le critère "RootFusion" décrite dans la section 7.1.1 page 97 et une dissimilarité généralisée décrite dans la section 5.1.4.1 page 55.

La figure 6.3 montre les scores de CR@20 moyens de la `AHC` avec l'arborescence de concepts (figures 6.3(a) et 6.3(b)) et avec la couleur (figure 6.3(c)) sur le benchmark `Xilo`.

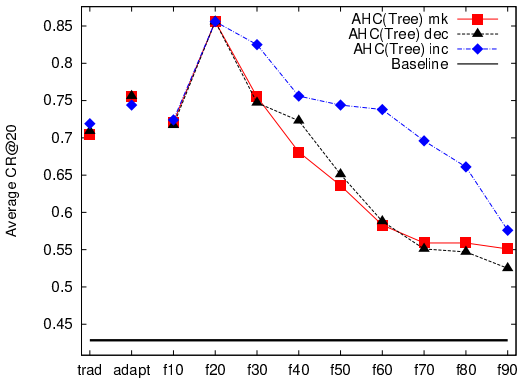
Pour utiliser `AHC` sur la diversité nous utilisons différents paramètres comme les techniques de découpage : `Adapt`, `Traditional` et `Fixe` (décrites dans la section 5.1.4.2), les techniques de priorité de clusters : `Rank`, `Increasing` et `Decreasing` (décrites dans la section 5.1.5) et un réordonnancement d'images classique. Pour l'instant nous cherchons seulement à étudier les caractéristiques propres du benchmark (comme le nombre d'images, les descripteurs, etc.), et non pas à émettre des conclusions sur ces paramètres, mais nous étudions plus en détail ces paramètres dans le chapitre 7.

#### 6.1.3.1 Comparaison des mesures de similarité Wu-Palmer et Lin

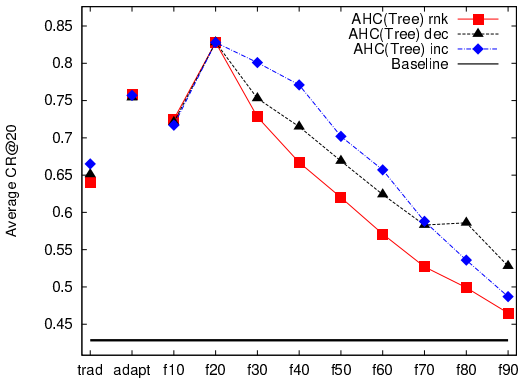
La comparaison de la figure 6.3(a) avec la figure 6.3(b) obtenue à partir de la dissimilarité généralisée utilisant la dissimilarité de Lin, montre que globalement elle obtient des résultats assez similaires en terme de comportement, mais légèrement inférieurs à ceux obtenus à l'aide de Wu-Palmer.

#### 6.1.3.2 Comparaison d'un descripteur visuel et une arborescence de concepts

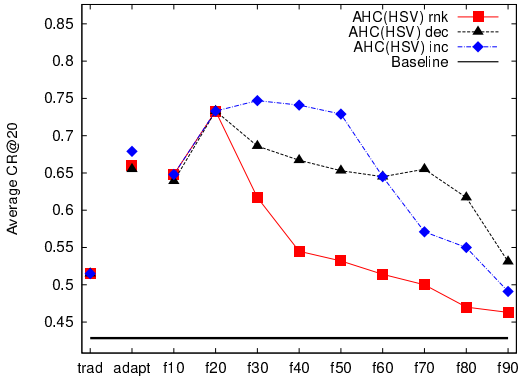
Si nous comparons les figures 6.3(a) et 6.3(c), nous remarquons que les méthodes basées sur la couleur donnent des résultats nettement inférieurs à celles basées sur l'arborescence. Cependant, les courbes obtenues en utilisant la couleur ont des comportements assez similaires à ceux obtenus à partir de l'arborescence de concept qui est donc une source d'information riche qui offre la diversité, tandis que la couleur n'est pas très adaptée pour ce qui est de trouver des groupes sémantiques. Néanmoins, si on compare




(a) `Xilo` Arborescence Wu-Palmer




(b) `Xilo` Arborescence Lin




(c) `Xilo` Couleur

FIGURE 6.3 – Comparaison des CR@20 moyens de la `AHC` avec (a) l'arborescence de concepts et la similarité Wu-Palmer, (b) l'arborescence de concepts et la similarité de Lin, et (c) les descripteurs de couleurs, en utilisant comme paramètres les techniques de priorité `Increasing` (en bleu), `Decreasing` (en noir) et `Rank` (en rouge), différents techniques de découpage (en abscisse) et un réordonnancement classique sur le benchmark `Xilo`

les courbes de couleurs par rapport à la baseline sans diversité, qui obtient un CR@20 de 0.428, on remarque que la couleur améliore nettement les résultats.

#### 6.1.3.3 Diversité sans clustering ("XiloTree")

Finalement, pour mesurer l'intérêt de la `AHC` nous créons une méthode qui utilise l'arborescence de concepts de manière directe (sans clustering) où les clusters sont construits à partir des feuilles de cette arborescence (méthode que nous appelons "XiloTree"). Pour cette méthode, nous ne pouvons prendre en compte que le premier concept qui annote chaque image.

Dans le tableau 6.2 on voit que "XiloTree" permet d'augmenter le CR@20 jusqu'à 0.8. Ces résultats sont supérieurs aux résultats utilisant la `AHC` sur les descripteurs de couleurs. Mais, nous voyons l'intérêt d'utiliser une `AHC`, car les résultats obtenus avec les méthodes `HierO` et `FlatO` sur l'arborescence de concepts sont nettement supérieurs à ceux de la méthode "XiloTree" ou des autres méthodes.

```
<top>
<num>Number: 5</num>
<title>animals swimming</title>
<cluster>animal</cluster>
<narr>animal</narr>
<image>3739.jpg</image>
<image>4968.jpg</image>
<image>30823.jpg</image>
</top>
```

FIGURE 6.4 – Description sémantique du contenu de la requête disponible sur le benchmark `Clef`

```
<DOC>
<DOCNO>annotations/00/60.eng</DOCNO>
<TITLE>Palma</TITLE>
<DESCRIPTION>two lane street with large shops on the right and smaller
shops on the left; people are waliking on the sidewalk, some are crossing
the street; cars are parked along the left side of the street as well;
</DESCRIPTION>
<NOTES>The mian shopping street in Paraguay;</NOTES>
<LOCATION>Asuncion, Paraguay</LOCATION>
<DATE>March 2002</DATE>
<IMAGE>images/00/60.jpg</IMAGE>
<THUMBNAIL>thumbnails/00/60.jpg</THUMBNAIL>
</DOC>
```

FIGURE 6.5 – Description sémantique du contenu de l'image disponible sur le benchmark `Clef`

## 6.2 Contexte ImageCLEF 2008 (`Clef`)

Le benchmark ImageCLEF 2008 Photo Retrieval [4] a été utilisé lors d'une campagne internationale et il nous sert pour faire la comparaison avec les méthodes de diversité des autres chercheurs. Ce benchmark contient 20k images et 39 requêtes.

### 6.2.1 Descripteurs disponibles

Ce benchmark ne contient pas une arborescence de concepts, mais nous avons extrait de chaque image un histogramme de couleurs dans l'espace HSV.

Dans ce benchmark chaque requête est composée de textes et d'images, ainsi que d'un thème imposé de diversité. Les figures 6.4 et 6.5 montrent des exemples de la description sémantique du contenu de la requête et d'une image. Pour chaque requête, la vérité terrain indique les images pertinentes ainsi que les sous-thèmes de diversité auxquels ils appartiennent. Par exemple, la requête de la figure 6.4 demande de retourner des animaux qui font l'action de nager (animals swimming) ; le thème de diversité pour cette requête est "animal", c'est-à-dire qu'on cherche à obtenir que dans les premiers résultats chaque image soit d'un type d'animal diffèrent. Dans chaque image de la requête, nous avons comme information textuelle (voir figure 6.5) un titre, une description, des notes, le lieu et la date de l'image.

Dans la vérité terrain il y a en moyenne 7.9 sous-thèmes (minimum 2 sous-thèmes, maximum 23 sous-thèmes, écart-type de 5.0) et 62 images pertinentes (minimum 18 images, maximum 185, écart-type de 33.8) par requête.

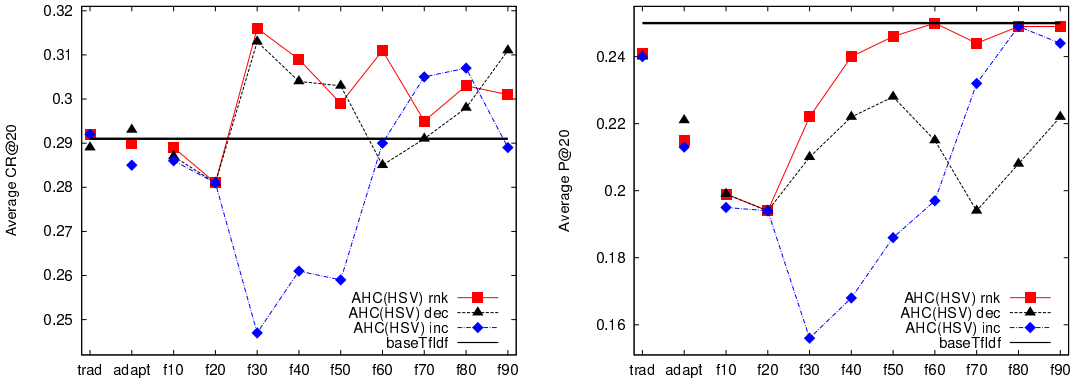



FIGURE 6.6 – Comparaison des CR@20 et P@20 moyens de la `AHC` avec le descripteur HSV en utilisant comme paramètres les techniques de priorité `Increasing` (en bleu), `Decreasing` (en noir) et `Rank` (en rouge), différents techniques de découpage (en abscisse) et un réordonnancement classique dans un cas réel sur le benchmark `Clef`

### 6.2.2 Baseline

Contrairement au benchmark `Xilo` où on disposait des résultats ordonnés du moteur de recherche de Xilopix et au benchmark `Media` où on dispose des résultats ordonnés du moteur de recherche de Flickr, dans ce benchmark, les requêtes sont données, mais pas les résultats ordonnés de ces requêtes. Nous avons donc récupéré les résultats ordonnés d'autres chercheurs (run non-diversifié du "LIP6" [60]).

De plus, pour nous placer dans les mêmes conditions que d'autres expérimentations que nous avons réalisées sur d'autres benchmarks, nous avons construit une baseline composée, pour chaque requête, des 100 premières images résultats du run non-diversifié "TfIdf" du "LIP6". Cette baseline contient des images pertinentes et non-pertinentes et elle représente un **cas réel**. Nous appelons cette baseline *baseTfIdf*.

### 6.2.3 Runs ideaux

Pour étudier la diversité sans être biaisée par la pertinence, nous avons récupéré les résultats ordonnés fournis dans [60] appelés "runs idéaux". Un "run" est l'ensemble des 39 résultats ordonnés. Ces runs idéaux ont été construits à l'aide de la vérité terrain. Pour chaque requête, seules les images pertinentes pour la requête sont considérées (run idéal numéro 1). Puis une permutation aléatoire de ces résultats pertinents est réitérée 9 fois pour chaque requête afin d'obtenir en tout 10 runs idéaux. Ces runs idéaux représentent donc un **cas idéal**.

### 6.2.4 Résultats obtenus

Contrairement au benchmark `Xilo` où sa baseline représente un cas idéal avec une pertinence proche de 1 (car ses images ont été annotées manuellement), dans le benchmark `Clef` nous disposons de deux cas à étudier, un cas réel avec la baseline "baseTfIdf" et un cas idéal avec les "runs idéaux".

**Résultats obtenus dans un cas réel** Le tableau 6.3 présente les principaux résultats obtenus sur ce benchmark dans un **cas réel**. La baseline "baseTfIdf" a une faible pertinence (P@20 = 0.25) car le résultat de cette baseline a été obtenu à partir de requêtes effectuées sur un corpus de seulement 20000 images, où il y a peu d'images pertinentes et où il est donc difficile de retrouver beaucoup d'images pertinentes dès

TABLE 6.3 – Synthèse des principaux résultats sur le benchmark `Clef` (39 requêtes) dans un cas réel

| Méthode | P@20 moyen | CR@20 moyen |
|---|---|---|
| baseTfIdf | 0.250 (ref) | 0.291 (ref) |
| CRmax | 0.997 (+299%) | 0.991 (+241%) |
| Random | 0.166 (-34%) | 0.263 (-10%) |
| `AHC`(HSV) `Rank fixed20` | 0.194 (-22%) | 0.281 (-3%) |

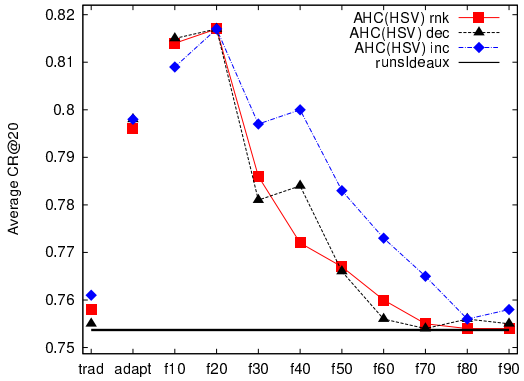



FIGURE 6.7 – Comparaison des CR@20 moyens de la `AHC` avec le descripteur HSV en utilisant comme paramètres les techniques de priorité `Increasing` (en bleu), `Decreasing` (en noir) et `Rank` (en rouge), différents techniques de découpage (en abscisse) et un réordonnancement classique dans un cas idéal sur le benchmark `Clef`

les premiers résultats. De plus, sa diversité est aussi faible (CR@20 = 0.291) car si on a moins d'images pertinentes on aurait moins de probabilité de ce que ces images pertinentes représentent différents sous-thèmes de la vérité terrain.

Contrairement au benchmark `Xilo` où nous arrivons à augmenter la diversité avec une méthode aléatoire, dans le cas réel du benchmark `Clef` nous n'arrivons pas à l'améliorer en utilisant la même méthode sur "baseTfIdf" car cette baseline a une faible quantité d'images pertinentes.

Afin de calculer la `AHC` avec le descripteur HSV nous lançons cette méthode pour chacun des 100 premières images résultats de "baseTfIdf" et nous utilisons le critère d'agrégation "Average" et la distance "Euclidean". Dans la figure 6.6 nous comparons les scores de nos méthodes proposées. Les scores CR@20 et P@20 moyens représentent la `AHC` avec le descripteur HSV dans un cas réel sur le benchmark `Clef`.

Pour utiliser `AHC` sur la diversité nous utilisons différents paramètres comme les techniques de découpage : `Adapt`, `Traditional` et `Fixe` (décrites dans la section 5.1.4.2), les techniques de priorité de clusters : `Rank`, `Increasing` et `Decreasing` (décrites dans la section 5.1.5) et un réordonnancement d'images classiques. Pour rappel, ces paramètres seront étudiés plus en détail dans le chapitre 7 car dans cette section nous cherchons seulement à étudier les caractéristiques propres du benchmark (le nombre d'images, les descripteurs, etc.).

Dans ces figures on peut voir que notre méthode proposée n'arrive pas à augmenter la diversité de la baseline qui a la particularité d'avoir une pertinence assez faible. De plus, ces méthodes montrent des comportements assez instables, contrairement aux résultats obtenus sur le benchmark `Xilo`.

TABLE 6.4 – Synthèse des principaux résultats sur le benchmark Clef dans un cas idéal. CR@20 moyen : moyenne sur 10 runs idéaux de la moyenne des CR@20 de 39 requêtes

| Méthode | CR@20 moyen |
|---|---|
| Runs ideaux | 0.754 (ref) |
| CRmax | 0.991 (+31%) |
| Random | 0.774 (+3%) |
| AHC(HSV) Rank fixed20 | 0.817 (+8%) |

**Résultats obtenus dans un cas idéal** Le tableau 6.4 présente les principaux résultats obtenus sur ce benchmark dans un **cas idéal**. Pour les 10 runs idéaux, la précision à 20 documents (P@20) est proche de 1 (car tous les documents sont pertinents), mais le CR@20 varie de 0.683 à 0.807 avec une moyenne de 0.754.

En diversifiant de manière aléatoire ces runs idéaux on améliore la diversité seulement un +3% car une permutation aléatoire a été réalisée sur la plupart de ces runs idéaux.

Afin de calculer la AHC avec le descripteur HSV nous lançons cette méthode pour chacun des 100 premières images résultats des 10 runs idéaux. Nous utilisons les mêmes paramètres de l'expérimentation précédente. Dans la figure 6.7 nous comparons les scores de nos méthodes proposées. Les scores de CR@20 moyens représentent la AHC avec le descripteur HSV dans un cas idéal sur le benchmark Clef. Dans cette figure on peut voir que nos méthodes Flat0 et Hier0 améliorent la diversité de +9% par rapport au score moyen des 10 runs idéaux. On peut voir aussi que ces méthodes montrent des comportements similaires à ceux obtenus sur le benchmark Xilo qui a aussi une baseline avec une précision proche de 1.

## 6.3 Contexte MediaEval 2013 (Media)

Le benchmark MediaEval 2013 [24] a été utilisé lors d'une campagne internationale. Ce benchmark contient 392 requêtes divisées en 2 ensembles : un d'apprentissage (devset) avec 50 requêtes ; et l'autre de test (testset) avec 342 requêtes. L'ensemble d'apprentissage nous sert à étudier les paramètres de certains descripteurs utilisés dans ce benchmark. Les requêtes sont des endroits autour du monde (par exemple, la Tour Eiffel à Paris, la statue de la Liberté à New York ou le Big Ben à l'Angleterre). Chacun de ces ensembles contient deux sous-ensembles : le premier appelé keywords (abrégé par la suite en "k") où les requêtes effectuées par Flickr l'ont été à partir seulement du nom de l'endroit, le deuxième appelé keywordsGPS (abrégé par la suite en "kGPS") où les requêtes effectuées par Flickr l'ont été à partir du nom de l'endroit et des coordonnées GPS.

Pour chaque requête nous avons un ensemble d'images récupérées du moteur de recherche Flickr (150 premières images) et des informations supplémentaires comme : le nom de l'endroit, ses coordonnées GPS, lien à Wikipedia, une photo représentative (récupéré de Wikipedia) et des descripteurs visuels et textuels.

Pour chaque image nous avons comme information textuelle des métadonnées (provenant de Flickr). Nous montrons dans la figure 6.8 un exemple de la métadonnée d'une image où nous avons comme information : une description, un titre, des mots-clés ("tags"), la date de l'image, parfois ses coordonnées GPS, le nombre de visites, le nom de l'utilisateur, son rang dans la liste des images résultats de la requête.

Dans ce benchmark nous disposons de deux types de vérité terrain : une première dont les annotations ont été faites par des experts et une deuxième dont les annotations ont été faites par des personnes non experts.

Dans la vérité terrain fait par des **experts** (392 requêtes) il y a en moyenne : sur

```
<photos monument="Basilica of St Mary of Health Venice">
  <photo date_taken="2003-10-09 15:19:30" description="The Basilica di
  Santa Maria della Salute (Basilica of St Mary of Health/Salvation),
  commonly known simply as the Salute, is a famous church in Venice,
  placed scenically at a narrow finger of land which lies between the
  Grand Canal and the Bacino di San Marco on the lagoon, visible as one
  enters the Piazza San Marco from the water. (Wikipedia)"
  id="2323210481" latitude="0" license="3" longitude="0" nbComments="50"
  rank="1" tags="europe italy venice basilicadisantamariadellasalute
  basilicaofstmaryofhealthsalvation grandcanal sunrise church basilica
  travel sony f717" title="Basilica of St Mary of Health/Salvation,
  Venice" username="Christopher Chan" views="6824"
  url_b="http://static.flickr.com/2410/2323210481_31da0f0311_b.jpg" />
...
</photos>
```

FIGURE 6.8 – Description sémantique des métadonnées d'une image dans le benchmark `Media`

`devset` 11.6 sous-thèmes (minimum 3 sous-thèmes, maximum 19 sous-thèmes, écart-type de 4.0) et 75.8 images pertinentes (minimum 3 images, maximum 142, écart-type de 43.7) par requête ; et sur `testset` 13.6 sous-thèmes (minimum 1 sous-thèmes, maximum 20 sous-thèmes, écart-type de 4.6) et 75.7 images pertinentes (minimum 1 images, maximum 150, écart-type de 43.7) par requête.

La vérité terrain fait par des personnes **non-experts** est de type Crowd-Sourcing [21]. Cela signifie que dans cette vérité terrain la tâche de l'annotation d'images a été faite par un grand nombre de personnes non-experts sous la forme d'un appel ouvert à la contribution. Pour cela, la plate-forme de Crowd-Sourcing CrowdFlower [1] a été utilisé et les personnes qui ont participé à cette tâche ont utilisé les mêmes conditions que pour les annotations faites par les experts. Dans cette vérité terrain (49 requêtes) il y a en moyenne 4.7 sous-thèmes (minimum 1 sous thème, maximum 16 sous-thèmes, écart-type de 3.2) et 85.9 images pertinentes (minimum 15 images, maximum 143, écart-type de 38.0) par requête.

Le but d'avoir deux types de vérité terrain c'est d'explorer les différences entre les annotations faites par des experts et par des personnes non-experts. Dans [24] les auteurs décrivent plus en détail la procédure utilisée pour la construction de ces vérités terrain.

Dans les tableaux A.2 et A.3 (voir annexe A.1 page 173) nous montrons une information plus générale sur le contenu de chaque sous-ensemble de données du benchmark `Media`. Nous allons appeler `GT0` le sous-ensemble qui utilise les mêmes requêtes que `GT1,2,3`, mais avec une évaluation établie par des experts.

Dans ce benchmark la **baseline** correspond aux résultats obtenus par Flickr. Ces images peuvent être pertinentes et non-pertinentes, donc les baselines dans ce benchmark représentent un **cas réel**. Dans les tableaux A.4 et A.5 (voir annexe A.1 page 176) nous montrons les scores de la baseline dans chaque sous-ensemble des données.

### 6.3.1 Descripteurs disponibles

Un avantage de ce benchmark est qu'il dispose de plusieurs types de descripteurs : textuels, visuels et des coordonnées GPS.

1. http ://crowdflower.com/

TABLE 6.5 – Benchmark Media : Descripteurs visuels disponibles

| Nom | Description | Dimension |
|---|---|---|
| CN [61] | maps colors to 11 universal color names | 11 |
| HOG [34] | global histogram of oriented gradients | 81 |
| CM [54] | global color moments on HSV color space | 9 |
| LBP [41] | global locally binary patterns on gray scale | 16 |
| CSD [36] | global color structure descriptor | 64 |
| GLRLM [57] | global statistics on gray level run length matrix | 44 |
| CN3x3 | CN descriptor with spatial pyramid representation | 99 |
| CM3x3 | CM descriptor with spatial pyramid representation | 81 |
| LBP3x3 | LBP descriptor with spatial pyramid representation | 144 |
| GLRLM3x3 | GLRLM descriptor with spatial pyramid representation | 396 |

TABLE 6.6 – Benchmark Media : Descripteurs textuels disponibles

| Nom | Description |
|---|---|
| Proba [44] | Modèle probabiliste |
| TfIdf [51] | Pondération TfIdf |
| Social [46] | Pondération Social TfIdf |

#### 6.3.1.1 Descripteurs Visuels

Dans ce benchmark, les organisateurs ont fourni plusieurs descripteurs visuels listés dans le tableau 6.5 page 73.

De plus, nous avons extrait de chaque image un histogramme des couleurs dans l'espace HSV (décrite dans l'annexe B page 197) avec l'idée principale de faire un point de comparaison sur plusieurs benchmarks.

#### 6.3.1.2 Descripteurs Textuels

Outre des descripteurs visuels, les organisateurs ont fourni des fichiers qui vont servir à la génération de 3 types de descripteurs textuels listés dans le tableau 6.6 page 73.

Ces fichiers contiennent comme information : des valeurs "Idf" de chaque terme pour générer le descripteur TfIdf (abrégé par la suite en "TfIdf") ; le nombre total d'occurrences de chaque terme pour générer un descripteur Probabiliste (abrégé par la suite en "Proba") et le nombre total d'utilisateurs qui utilisent chaque terme pour générer le descripteur Social (abrégé par la suite en "Social"). Ces descripteurs sont décrites dans la section 2.2.2 page 19.

**Génération des descripteurs textuels** Pour expliquer cette procédure, nous avons dessiné la figure 6.9 qui montre les différentes étapes suivies pour générer le descripteur TfIdf. Dans cette figure on voit que pour une image donnée nous prenons d'abord les champs des métadonnées. Ensuite, nous exécutons une procédure classique de tokenization ("tokenizer") afin de séparer le texte en plusieurs termes (t1, t2, t3, ...). Puis, nous calculons le poids de chaque terme obtenu en utilisant les formules décrites dans la section 2.2.2 page 19 et les fichiers fournis par Media. Dans la figure on voit que pour calculer le poids de chaque terme nous utilisons la formule "TfIdf" et le fichier qui contient les valeurs "Idf". Finalement, chaque association "terme-poids" serait le descripteur final de l'image.

La procédure de tokenization que nous utilisons est la suivante :

— Transformer les accents (par exemple : la lettre "à" en "a") ;

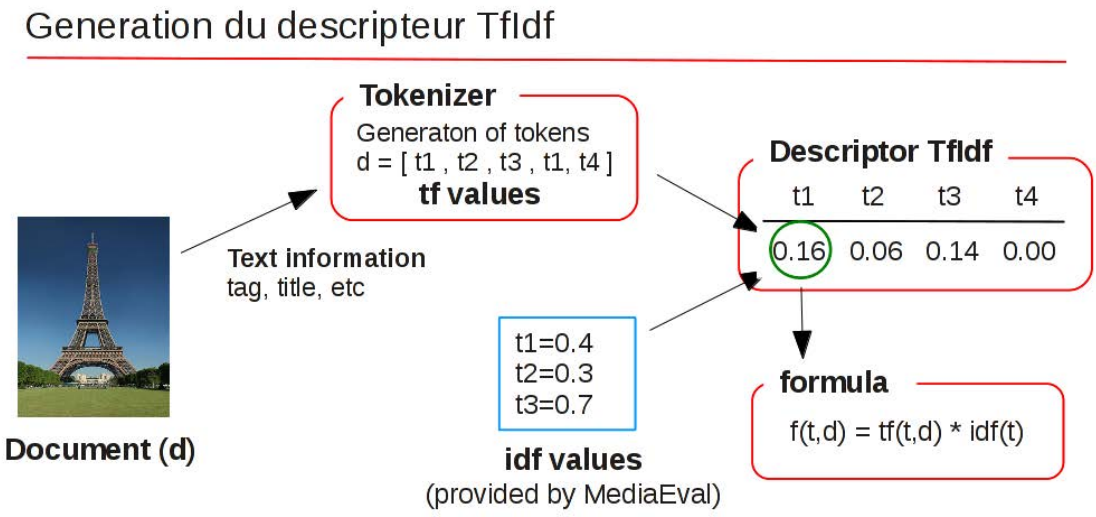



FIGURE 6.9 – Exemple de la procédure de la génération du descripteur TfIdf

TABLE 6.7 – Benchmark `Media` : Statistiques en relation aux images sans coordonnées GPS

| | `devk` | `devkGPS` | `testk` | `testkGPS` |
|---|---|---|---|---|
| nb des requêtes sans GPS | 0 of 25 | 0 of 25 | 0 of 132 | 0 of 210 |
| nb des images sans GPS | 1252 of 2281 | 2 of 2837 | 7861 of 13220 | 0 of 24607 |
| moyenne des images sans GPS par requête | 25.04 | 0.04 | 59.55 | 0 |

— Transformer les lettres de majuscule en minuscule ;
— Enlever les caractères spéciaux (par exemple : "=" "+" "-" "." "," ";" etc.) ;
— Couper le texte où il y a des espaces pour obtenir les termes (par exemple : "voiture" "rouge") ;
— Enlever les termes d'un seul caractère ;
— Enlever les termes qui appartiennent à la liste de "stopwords".

#### 6.3.1.3 Coordonnées GPS

Dans les métadonnées des images nous pouvons trouver des coordonnées GPS, mais pas toutes les images en sont fournies.

Pour connaître la totalité d'images avec GPS nous avons réalisé le tableau 6.7 qui montre le nombre de requêtes sans GPS, le nombre d'images qui n'ont pas du GPS et le nombre moyen des images qui n'ont pas du GPS. Dans ce tableau nous voyons que dans les sous-ensembles `devkGPS` et `testkGPS` toutes les images résultats ont du GPS. Par contre, dans les sous-ensembles `devk` et `testk` environ 60% d'images n'ont pas du GPS.

**Attribution automatique du GPS** Pour les images qui n'ont pas de GPS nous proposons une méthode appelée **gpsv** qui attribue les coordonnées GPS de l'image la plus proche visuellement (distance visuelle) entre celles de la même requête. La figure 6.10 montre un exemple de cette méthode avec la requête "Tour Eiffel" où l'on voit que l'image avec GPS la plus proche visuellement à celle sans GPS est encadrée en rouge.

Cette stratégie semble intéressante parce que, pour deux images qui se ressemblent

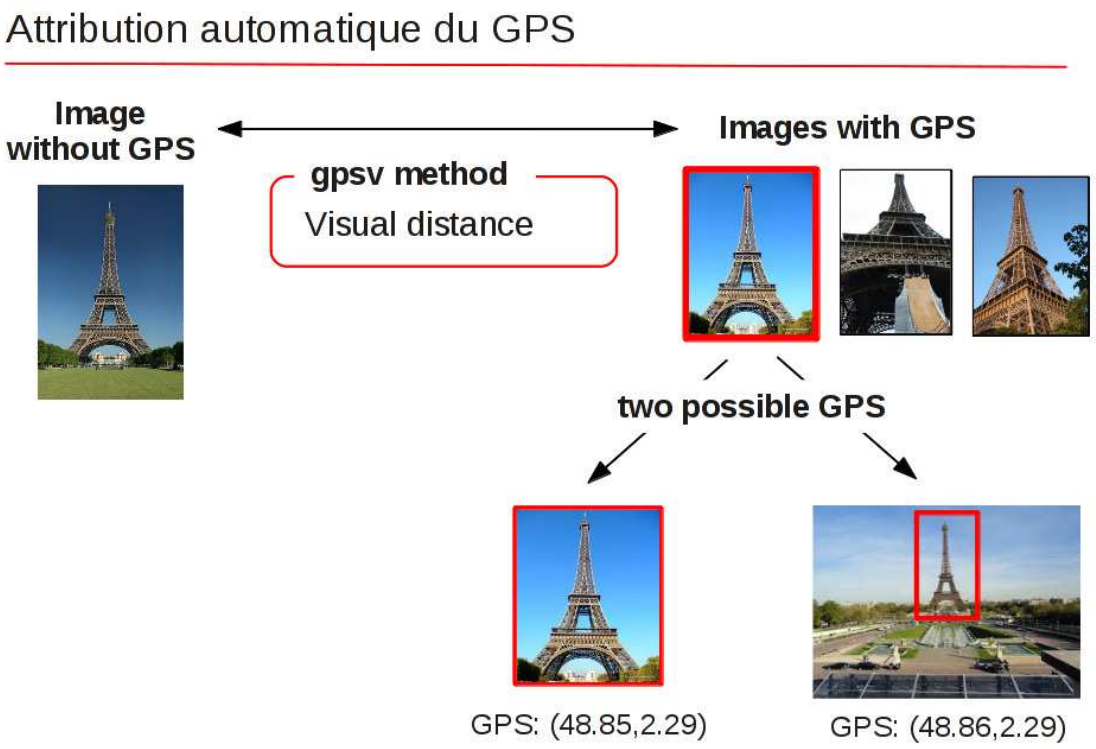



FIGURE 6.10 – Exemple de la méthode proposée **gpsv** qui attribue de manière automatique du GPS dans la requête "Tour Eiffel"

visuellement, on peut dire qu'elles peuvent être prises au même endroit et donc avoir des coordonnées GPS assez proches. Par contre, comme le montre la figure, on peut avoir aussi le cas où l'image avec GPS a été prise en utilisant un zoom depuis un endroit plus éloigné et donc avec des coordonnées GPS différents.

#### 6.3.1.4 Arborescence de concepts

Le benchmark `Media` ne possède pas une arborescence de concepts. Cependant, nous utilisons l'univers "Voyage" de l'arborescence de concepts développé par l'entreprise Xilopix (ressource externe au benchmark) afin de mieux exploiter la granularité géographique entre les images. L'univers "Voyage" contient plusieurs niveaux de profondeur : continents, pays, régions, départements et lieux. Chaque concept de cet univers est fourni avec son nom et de ses coordonnées GPS.

Dans cette partie, nous indexons d'abord de manière automatique les images du corpus `Media` dans l'arborescence de concepts "Voyage". Puis, avec cette indexation nous générons les descripteurs qui proviennent de cette arborescence (descripteur que nous appelons "Tree").

**Indexation et génération du descripteur Tree** Pour générer le descripteur "Tree", nous proposons d'abord des méthodes d'indexation des images du corpus dans chaque concept de l'univers "Voyage". Dans la figue 6.11 nous montrons les deux cas possibles d'indexation que nous utilisons :

1. Si **l'image à indexer a du GPS**, nous proposons une méthode que nous appelons **tree-gps** qui calcule la distance entre le GPS de l'image (*GPS(image)* dans l'exemple) et le GPS de chaque concept (*GPS(concept)* dans l'exemple).

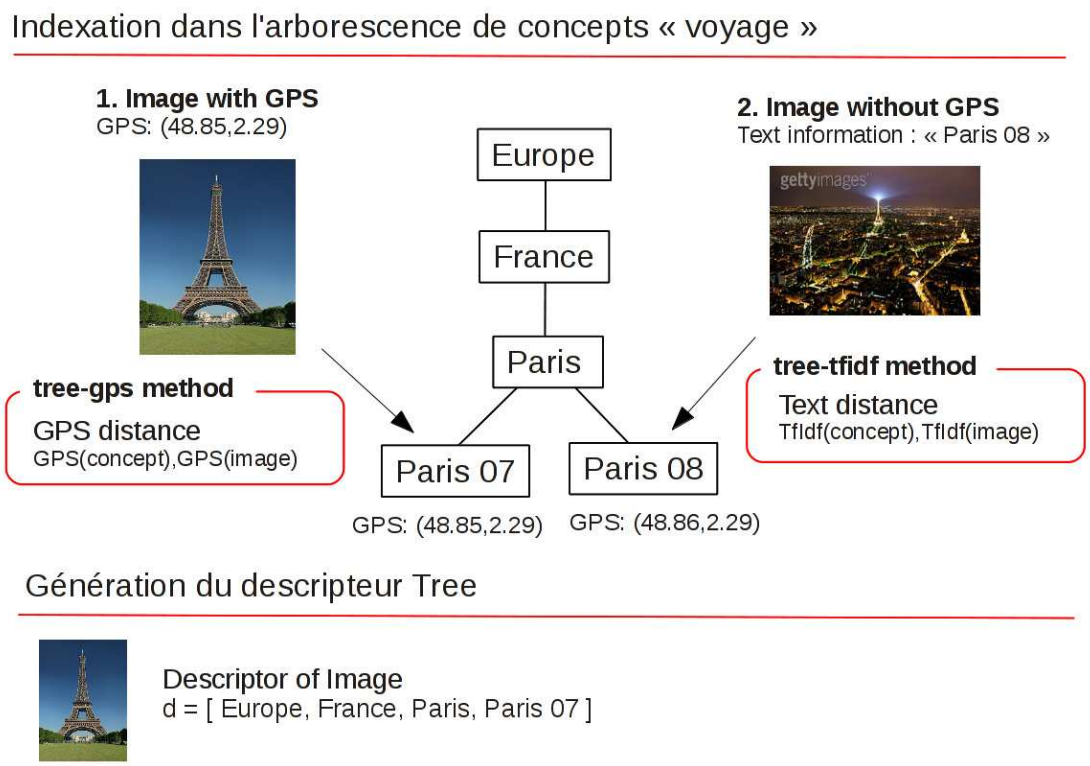



FIGURE 6.11 – Exemple des méthodes proposées d'indexation automatique d'une image dans l'arborescence de concepts "voyage". Pour indexer les images, la méthode proposée **tree-gps** utilise des coordonnées GPS et la méthode proposée **tree-tfidf** utilise des informations textuels

2. Si **l'image à indexer n'a pas du GPS**, nous proposons deux méthodes qui utilisent des informations textuelles pour calculer la distance entre le descripteur textuel de l'image (*TfIdf(image)* dans l'exemple) et le descripteur textuel de chaque concept (*TfIdf(concept)* dans l'exemple). Les descripteurs textuels utilisés sont : le descripteur "TfIdf" (méthode que nous appelons **tree-tfidf**) et le descripteur "Proba" (méthode que nous appelons **tree-proba**).

Dans les deux cas, l'image est associée au concept qui obtient la distance la plus proche. Puis, pour générer le descripteur "Tree" de l'image, nous prenons son concept associé et ses nœuds parents.

Nous avons vu déjà que dans `Xilo` utiliser l'arborescence de concepts comme descripteur semble être très intéressant pour augmenter la diversité, car sur un benchmark réaliste, ce type de descripteur obtient une meilleure diversité par rapport à un descripteur de couleur.

La méthode **tree-gps** proposée semble être intéressante pour indexer les images parce que les coordonnées GPS sont des informations très fiables et précises pour indexer correctement les images dans l'univers. Par contre, dans ce corpus il y a des images qui n'ont pas forcement des coordonnées GPS. Pour résoudre ce problème, nous proposons deux méthodes **tree-tfidf** et **tree-proba** qui utilisent le texte pour indexer les images, mais on sait que le texte peut contenir des fausses informations de l'image, donc il n'est pas très adapté pour indexer correctement les images.

### 6.3.2 Étude des paramètres

Dans cette section nous allons étudier les différents paramètres qui utilisent nos méthodes proposées sur l'ensemble d'apprentissage (`devset`), comme par exemple les différents paramètres existant dans les descripteurs textuels ou ceux les plus adaptés pour augmenter la pertinence. Dans ces expérimentations, nous allons utiliser aussi l'ensemble de test (`testset`) pour confirmer les résultats obtenus dans `devset`. Il faut remarquer que dans `devset` nous avons seulement 50 requêtes, quantité qui paraît un peu insuffisante pour étudier le comportement des paramètres en comparaison aux 342 requêtes qui possèdent `testset`.

Dans ces expérimentations nous allons étudier les paramètres de l'utilisation directe de la `AHC` afin d'augmenter la diversité. Mais comme dans ce benchmark la baseline représente un cas réel, les images non-pertinentes peuvent diminuer la qualité des clusters générés par la `AHC` et donc diminuer aussi sa diversité. Dans cette section nous allons aussi étudier les paramètres des méthodes de réordonnancement classique par similarité à la requête afin d'augmenter la pertinence. Cette étude va nous servir par la suite pour faire une étude plus approfondie de l'influence des images non-pertinentes sur la diversité.

Nous allons donc diviser cette section en deux parties : une première partie où nous allons étudier les paramètres des méthodes de réordonnancement pour augmenter la pertinence ; une deuxième partie où nous allons analyser les paramètres de notre méthode proposée pour augmenter la diversité.

Contrairement aux benchmarks précédents où nous avons évalué les 20 premières images (CR@20, P@20), dans le benchmark `Media` nous évaluons seulement les 10 premières images (CR@10, P@10) car c'est la mesure d'évaluation officielle de cette campagne internationale que nous avons participé.

#### 6.3.2.1 Étude de la pertinence

Dans cette partie nous allons étudier les paramètres des méthodes de réordonnancement classique par similarité à la requête (voir section 5.1.3 page 55) avec l'objectif d'augmenter la pertinence.

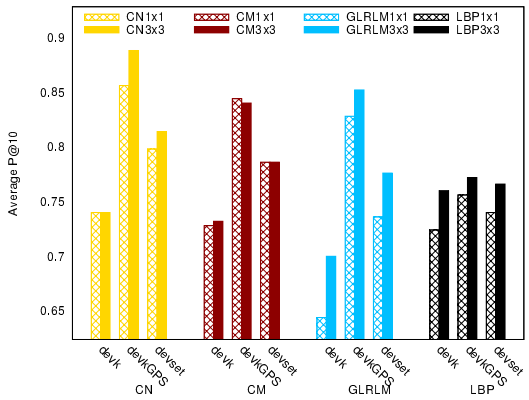
Ces méthodes, abrégées par la suite en "Rerank", calculent la similarité entre la requête et chaque image résultat de la baseline et puis réordonnent les images selon leur valeur de similarité. Si deux images ont la même valeur de similarité, on prend en priorité l'image avec le rang de la baseline le plus fort.

Comme nous souhaitons augmenter la pertinence, nous allons utiliser comme métrique d'évaluation la **précision** dans les 10 premières images (P@10).

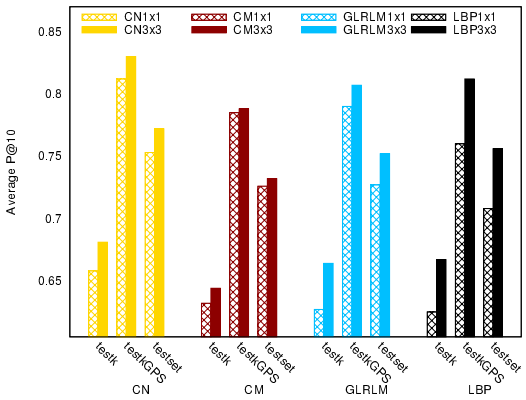
Nous allons diviser ces expérimentations selon le type de descripteur à utiliser :

**6.3.2.1.1 Descripteurs visuels** Dans cette partie nous allons comparer les méthodes de réordonnancement avec différents descripteurs visuels comme paramètre (décrites dans le tableau 6.5). Nous allons utiliser la distance "Euclidean" comme mesure de similarité entre l'image de la requête et les images résultats de cette requête.

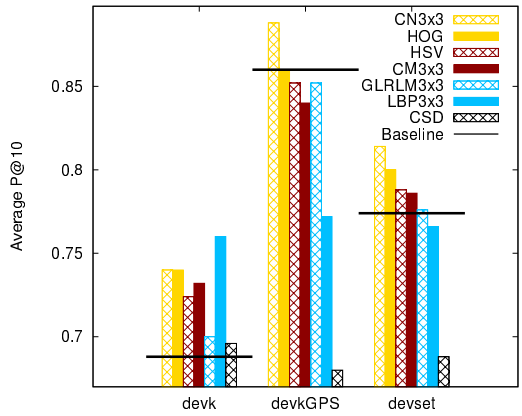
**Stratégie de diviser l'image en plusieurs régions ("3x3")** Avant de comparer les descripteurs visuels, nous allons étudier la stratégie de diviser l'image en plusieurs régions "3x3" par rapport à une stratégie sans découpage de l'image "1x1". Dans les figures 6.12(a) et 6.12(b) nous montrons les scores de précision P@10 moyens des deux stratégies avec les descripteurs visuels CN, GLRLM, LBP et CM dans un cas réel sur le benchmark `Media` sous-ensemble `devset` (figure 6.12(a)) et sous-ensemble `testset` (figure 6.12(b)). Dans ces figures nous pouvons voir que la stratégie de diviser l'image en



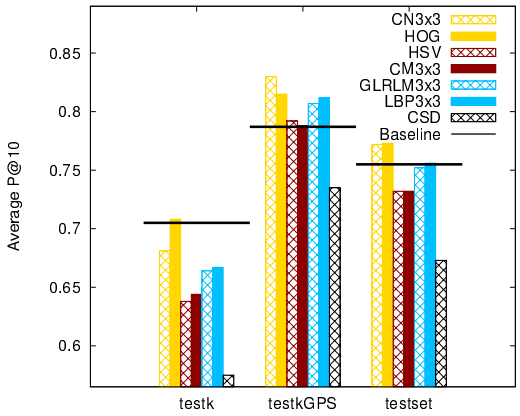

(a) Rerank(visual) Strategy 1x1 vs 3x3 (`Media-devset`)




(b) Rerank(visual) Strategy 1x1 vs 3x3 (`Media-testset`)




(c) Rerank(visual) (`Media-devset`)




(d) Rerank(visual) (`Media-testset`)

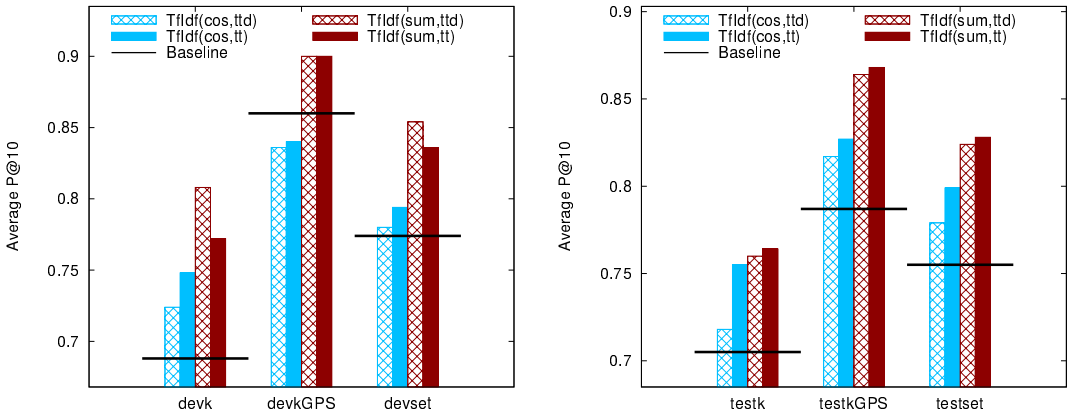
FIGURE 6.12 – En haut nous comparons la stratégie de diviser l'image en régions de "3x3" vs la stratégie sans découpage de l'image "1x1". En bas nous comparons les meilleurs descripteurs visuels (ils sont triés dans l'ordre des résultats sur `devset`). Les scores P@10 moyens représentent le réordonnancement classique (Rerank) avec les descripteurs visuels dans un cas réel sur le benchmark `Media` (a) (c) sous-ensemble `devset` et (b) (d) sous-ensemble `testset`

régions "3x3" donne des meilleurs résultats que la stratégie sans découpage "1x1". Donc cette stratégie semble très intéressante pour augmenter la pertinence.

**Autres descripteurs visuels** Finalement, nous allons comparer les descripteurs visuels manquantes (HSV, HOG, CSD) avec les descripteurs qui utilisent la stratégie "3x3". Dans les figures 6.12(c) et 6.12(d) nous montrons les scores de précision moyen de ces descripteurs visuels dans un cas réel sur le benchmark `Media` sous-ensemble `devset` (figure 6.12(c)) et sous-ensemble `testset` (figure 6.12(d)). Dans ces figures nous pouvons voir que le descripteur de couleur CN3x3 semble obtenir de manière globale les meilleurs résultats en pertinence.

**6.3.2.1.2 Descripteurs textuels** Dans cette partie nous allons comparer les méthodes de réordonnancement avec trois descripteurs textuels. De plus, nous allons étudier différents paramètres qui participent à la génération de ces descripteurs (voir section 6.3.1.2).

Dans ces expérimentations nous allons utiliser deux combinaisons : la combinaison des champs "tag","title" (tt) qui possède des mots très représentatifs à l'image ; et la combinaison des champs "tag","title","description" (ttd) qui possède, en outre des mots qui donnent des informations complémentaires de l'image.




(a) Rerank(TfIdf) (`Media-devset`) (b) Rerank(TfIdf) (`Media-testset`)

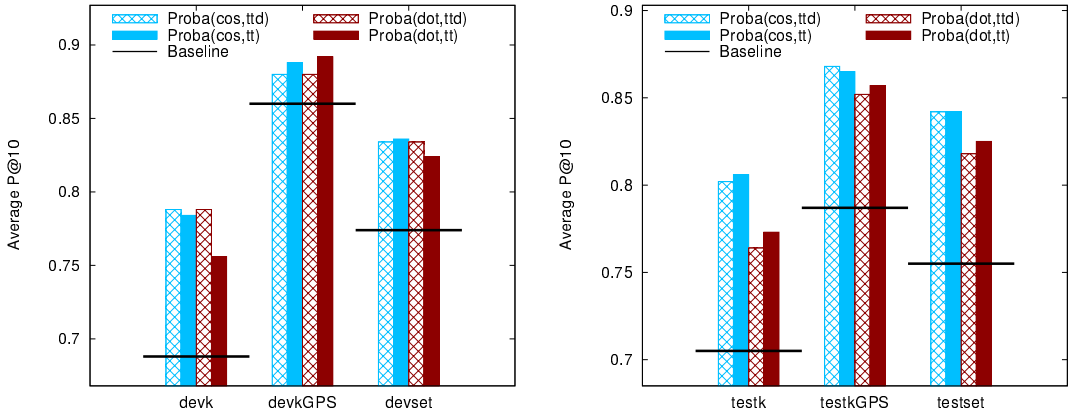
FIGURE 6.13 – Comparaison des descripteurs TfIdf(cos) vs TfIdf(sum). Les scores P@10 moyens représentent le Rerank avec le descripteur "TfIdf" en utilisant les mesures de similarité "cos" (en bleu) et "sum" (en rouge) avec différents champs de métadonnées (tt, ttd) dans un cas réel sur le benchmark `Media` (a) sous-ensemble `devset` et (b) sous-ensemble `testset`

**Descripteur "TfIdf"** Pour utiliser ce descripteur avec les méthodes de réordonnancement on doit choisir entre deux mesures de similarité : la similarité Cosinus (abrégé par "cos") et une similarité "sum" qui est décrite dans la formule 2.1 page 19.

Dans la figure 6.13 nous comparons les scores P@10 moyens du réordonnancement (Rerank) avec le descripteur "TfIdf" en utilisant différents mesures de similarité (cos et sum), différents champs de métadonnées (tt, ttd) dans un cas réel sur le benchmark `Media` (a) sous-ensemble `devset` et (b) sous-ensemble `testset`. Dans cette figure on peut voir que la similarité "sum" obtient globalement une meilleure valeur de pertinence que la similarité "cos". Donc, par la suite nous pouvons prendre la similarité "sum" pour "TfIdf".

**Descripteur "Proba"** Comme dans l'expérimentation précédente, nous étudions le descripteur "Proba" pour deux mesures de similarité : la similarité "cos" et la similarité décrite dans la formule 2.6 page 20 (abrégée en "dot") (voir figure 6.14). Nous avons également réalisé quelques expériences avec le lissage de Dirichlet, mais nous n'avons pas étudié toutes les valeurs possibles du paramètre $\mu$. Dans l'ensemble `devset` (voir figure 6.14(a)) on voit que les deux mesures de similarité obtiennent des résultats similaires. Par contre, dans l'ensemble `testset` qui contient beaucoup plus de requêtes (voir figure 6.14(b)) on voit que la similarité "cos" obtient des meilleurs résultats en pertinence qu'avec "dot". Donc, par la suite nous allons prendre comme paramètre la similarité "cos" pour "Proba".

**Choix des champs de métadonnées** Finalement, nous allons comparer les deux combinaisons des champs de métadonnées. Pour cela nous allons prendre comme descripteurs textuels : le descripteur "Social", le descripteur "TfIdf" avec la similarité "sum" et




(a) Rerank(Proba) (`Media-devset`) (b) Rerank(Proba) (`Media-testset`)

FIGURE 6.14 – Comparaison des descripteurs Proba(cos) vs Proba(dot). Les scores P@10 moyens représentent le Rerank avec le descripteur "Proba" en utilisant les mesures de similarité "cos" (en bleu) et "dot" (en rouge) avec différents champs de métadonnées (tt, ttd) dans un cas réel sur le benchmark `Media` (a) sous-ensemble `devset` et (b) sous-ensemble `testset`

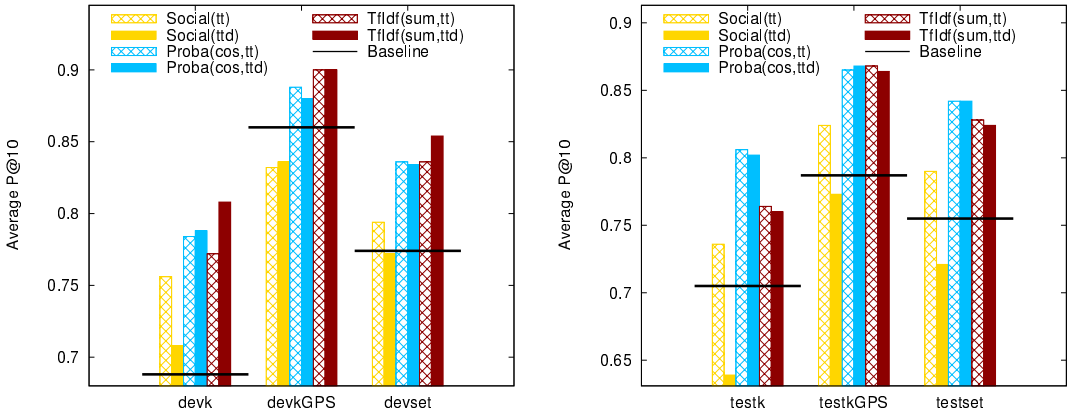
le descripteur "Proba" avec la similarité "cos".

Dans la figure 6.15 nous montrons les scores P@10 moyens du réordonnancement avec les descripteurs textuels en utilisant les champs de métadonnées (tt, ttd) dans un cas réel sur le benchmark `Media` (a) sous-ensemble `devset` et (b) sous-ensemble `testset`. Dans cette figure nous pouvons voir que les descripteurs "TfIdf" et "Proba" donnent des meilleurs résultats en pertinence qu'avec "Social". On peut voir aussi que les champs "tt" donnent des résultats presque similaires qu'avec les champs "ttd". Par contre, l'utilisation des champs "tt" est plus rapide que les champs "ttd" parce que "tt" utilise beaucoup moins de termes. Par la suite, nous allons prendre comme paramètres les champs "tt" pour améliorer la pertinence.

**6.3.2.1.3 Coordonnées GPS** Dans cette partie nous avons deux stratégies possibles pour faire du réordonnancement avec du GPS : une méthode (**gps**) qui va prendre seulement les images qui ont du GPS ; et une méthode (**gpsv**) qui d'abord va attribuer automatiquement du GPS aux images sans GPS (voir section 6.3.1.3) et puis réordonner avec les nouvelles images qui ont du GPS. Pour la méthode *gpsv* nous avons pris comme paramètre la distance "Euclidean" et le descripteur visuel CN3x3 car avec ces paramètres on obtient de meilleurs résultats en pertinence.

Pour étudier l'intérêt de la méthode **gpsv**, nous allons utiliser les sous-ensembles `devk` et `testk` car dans ces sous-ensembles nous avons des images avec et sans GPS. La mesure de similarité que nous allons utiliser est la distance "Haversine" (voir section 2.4.2 page 24).

Dans la figure 6.16(a) nous comparons les scores P@10 moyens du réordonnancement avec les deux stratégies **gps** et **gpsv** dans un cas réel sur le benchmark `Media` (sous-ensemble `devk` et sous-ensemble `testk`). Dans cette figure nous pouvons voir que la stratégie *gps* est légèrement meilleur que la stratégie *gpsv* pour augmenter la pertinence. Donc, il n'a pas trop l'intérêt d'utiliser la stratégie *gpsv* qui attribue automatique du GPS sur ce benchmark. Par la suite, nous allons prendre seulement la méthode *gps*.




(a) Rerank(text) (`Media-devset`) (b) Rerank(text) (`Media-testset`)

FIGURE 6.15 – Comparaison des descripteurs textuels "Social" vs "Proba(cos)" vs "TfIdf(sum)". Les scores P@10 moyens représentent le Rerank avec ces descripteurs en utilisant différents champs de métadonnées (tt, ttd) dans un cas réel sur le benchmark `Media` (a) sous-ensemble `devset` et (b) sous-ensemble `testset`

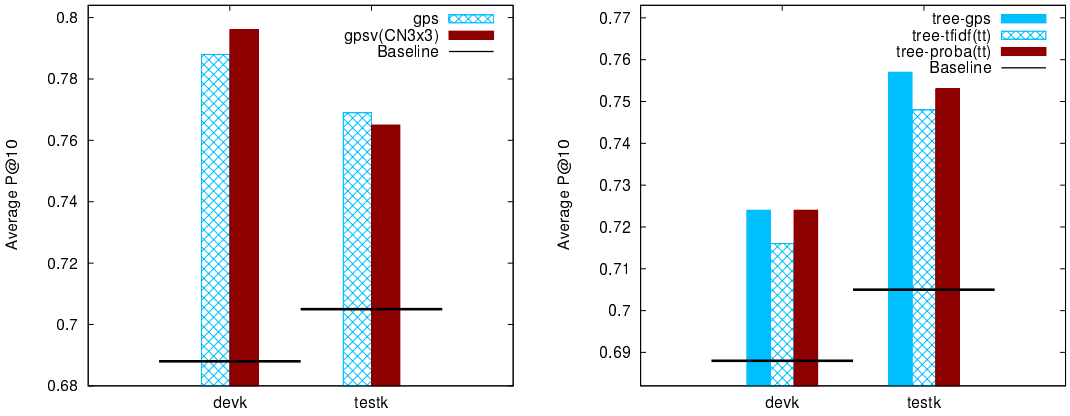
**6.3.2.1.4 Descripteur Tree** Dans cette partie nous allons comparer trois stratégies possibles pour faire du réordonnancement. La différence entre ces stratégies c'est la façon dont le descripteur "Tree" a été généré. Ces trois méthodes de génération du descripteur Tree sont décrites dans la section 6.3.1.4. Pour la méthode **tree-tfidf** nous avons utilisé comme paramètre les champs "tt" et la similarité "sum". Pour celle **tree-proba** nous avons utilisé comme paramètre les champs "tt" et la similarité "cos".

Pour étudier l'intérêt des méthodes **tree-tfidf** et **tree-proba** qui utilisent du texte pour générer le descripteur Tree (dans le cas où l'image n'a pas du GPS), nous allons utiliser les sous-ensembles `devk` et `testk` car dans ces sous-ensembles nous avons des images avec et sans GPS. La mesure de similarité que nous allons utiliser est la similarité "Wu-Palmer" (voir section 2.5.1.3).

Dans la figure 6.16(b) nous comparons les scores P@10 moyens du réordonnancement avec les trois méthodes **tree-gps**, **tree-tfidf** et **tree-proba** dans un cas réel sur le benchmark `Media` sous-ensemble `devk` et sous-ensemble `testk`. Dans cette figure nous pouvons voir qu'il n'y pas une grande différence entre les trois méthodes. Les méthodes **tree-gps** et **tree-proba** donnent légèrement les meilleurs résultats. Par la suite, nous allons prendre seulement la méthode **tree-gps**.

**6.3.2.1.5 Choix des descripteurs** Pour faire un résumé des choix faits jusqu'à maintenant, nous réalisons le tableau 6.8 qui montre les paramètres du réordonnancement classique (Rerank) avec différents types de descripteurs qui obtiennent de bons résultats en pertinence. Par la suite, sauf indication contraire, nous prendrons les paramètres indiqués dans ce tableau.

Une fois étudiés les petits paramètres de chaque descripteur, nous allons comparer ces descripteurs pour savoir lequel nous offre une meilleure valeur de pertinence. Dans le tableau 6.9 nous comparons les scores de précision P@10 et CR@10 moyens de la méthode de réordonnancement classique avec les descripteurs mentionnés dans le tableau 6.8. Le tableau montre que la stratégie de faire un réordonnancement classique arrive bien à augmenter la pertinence initiale de la baseline. Les descripteurs textuels donnent globalement des meilleurs résultats en pertinence qu'avec les autres descripteurs. Par la suite, nous prendrons le descripteur textuel TfIdf comme paramètre optimal pour augmenter




(a) Rerank(GPS) (`Media-devk/testk`)

(b) Rerank(Tree) (`Media-devk/testk`)

FIGURE 6.16 – A gauche nous comparons les stratégies *gps* vs *gpsv* pour générer le descripteur GPS. A droite nous comparons les stratégies *tree-gps* vs *tree-tfidf* vs *tree-proba* pour générer le descripteur Tree. Les scores P@10 moyens représentent le Rerank avec ces stratégies dans un cas réel sur le benchmark `Media` sous-ensemble `devk` et sous-ensemble `testk`

TABLE 6.8 – Liste des paramètres du Rerank avec différents descripteurs qui ont donné de bons résultats en pertinence sur le benchmark `Media`. Par la suite, sauf indication contraire, nous prendrons les paramètres indiqués dans ce tableau

| Descriptor | | Metadata | Similarity |
|---|---|---|---|
| Text | TfIdf | tag, title | sum |
| | Proba | tag, title | Cosinus |
| Visual | CN3x3 | - | Euclidean |
| | HOG | - | Euclidean |
| | HSV | - | Euclidean |
| GPS | | GPS | Haversine |
| Tree | | GPS | Wu-Palmer |

la pertinence.

#### 6.3.2.2 Étude de la diversité

Dans cette partie nous allons étudier les paramètres de notre méthode proposée afin de trouver les meilleurs paramètres qui puissent augmenter la diversité. Comme dans cette partie nous souhaitons augmenter la diversité, nous allons utiliser comme métrique d'évaluation le **cluster rappel** dans les 10 premières images (CR@10). De plus, nous allons utiliser la **précision** dans les 10 premières images (P@10) comme information complémentaire.

Notre méthode proposée est divisée en deux étapes : une première étape où l'on regroupe les images avec une méthode de clustering hiérarchique (`AHC`) ; et une deuxième étape où l'on réordonne les images à partir des clusters obtenus. À la première étape nous avons comme paramètres les mesures de similarité et les critères d'agrégation. Pour le critère d'agrégation nous allons prendre "Average" car diverses études montrent avoir obtenu de bons résultats avec cette méthode. Le choix de la mesure de similarité dépend du type de descripteur. À la deuxième étape nous avons comme paramètres le choix du nombre de clusters et les techniques de priorité.

TABLE 6.9 – Comparaison des scores P@10 et CR@10 moyens du Rerank avec les descripteurs décrites dans le tableau 6.8 sur le benchmark Media. Entre parenthèse nous avons le gain en % comparé avec la baseline

| | | keywords | | keywordsGPS | | all | |
|---|---|---|---|---|---|---|---|
| | | RÉSULTATS SUR devset | | | | | |
| Méthode | | P@10 | CR@10 | P@10 | CR@10 | P@10 | CR@10 |
| Baseline | | 0.688(ref) | 0.464(ref) | 0.860(ref) | 0.412(ref) | 0.774(ref) | 0.438(ref) |
| Rerank | TfIdf | 0.772(+12) | 0.493(+6) | **0.900(+5)** | 0.429(+4) | **0.836(+8)** | 0.461(+5) |
| | Proba | 0.784(+14) | 0.512(+10) | 0.888(+3) | 0.404(-2) | 0.836(+8) | 0.458(+5) |
| | CN3x3 | 0.740(+8) | 0.404(-13) | 0.888(+3) | 0.335(-19) | 0.814(+5) | 0.370(-16) |
| | HOG | 0.740(+8) | 0.418(-10) | 0.860(+0) | 0.390(-5) | 0.800(+3) | 0.404(-8) |
| | HSV | 0.724(+5) | 0.405(-13) | 0.852(-1) | 0.375(-9) | 0.788(+2) | 0.390(-11) |
| | GPS | **0.788(+15)** | 0.474(+2) | 0.860(+0) | 0.379(-8) | 0.824(+6) | 0.426(-3) |
| | Tree | 0.724(+5) | 0.464(+0) | 0.860(+0) | 0.412(+0) | 0.792(+2) | 0.438(+0) |
| | | RÉSULTATS SUR testset | | | | | |
| Méthode | | P@10 | CR@10 | P@10 | CR@10 | P@10 | CR@10 |
| Baseline | | 0.705(ref) | 0.399(ref) | 0.787(ref) | 0.344(ref) | 0.755(ref) | 0.365(ref) |
| Rerank | TfIdf | 0.764(+8) | 0.403(+1) | **0.868(+10)** | 0.373(+8) | 0.828(+10) | 0.385(+5) |
| | Proba | **0.806(+14)** | 0.433(+9) | 0.865(+10) | 0.376(+9) | **0.842(+12)** | 0.398(+9) |
| | CN3x3 | 0.681(-3) | 0.328(-18) | 0.830(+5) | 0.306(-11) | 0.772(+2) | 0.315(-14) |
| | HOG | 0.708(+0) | 0.358(-10) | 0.815(+4) | 0.331(-4) | 0.773(+2) | 0.341(-7) |
| | HSV | 0.638(-10) | 0.328(-18) | 0.792(+1) | 0.318(-8) | 0.732(-3) | 0.322(-12) |
| | GPS | 0.769(+9) | 0.405(+2) | 0.804(+2) | 0.352(+2) | 0.790(+5) | 0.373(+2) |
| | Tree | 0.757(+7) | 0.422(+6) | 0.788(+0) | 0.344(+0) | 0.776(+3) | 0.374(+2) |

Nous ne pouvons pas tester tous les descripteurs beaucoup trop nombreux avec tous les paramètres possibles. Nous souhaitons donc faire une première sélection pour éliminer les descripteurs qui ne sont pas performants en diversité, pour cela, nous allons étudier les performances de ces descripteurs avec les paramètres qui lors de nos nombreux tests nous ont donné les résultats les plus cohérents (`Flat0`, `Rank`, `Fixe`10, CR@10)

**6.3.2.2.1 Descripteurs visuels** Dans cette partie nous allons comparer les différents descripteurs visuels avec la `AHC`. Pour ces descripteurs nous allons utiliser la similarité "Euclidean" comme mesure de similarité.

Le tableau 6.10 montre les scores P@10, CR@10 moyens de la `AHC` avec les descripteurs visuels en utilisant comme paramètre la priorité `Rank` et le découpage `fixed10` sur le benchmark `Media` sous-ensemble `devset` et `testset`. Dans ce tableau on peut voir que globalement le descripteur CN3x3 obtient de bons résultats en diversité (CR@10). En sachant que dans l'étude de la pertinence de la section 6.3.2.1 ce descripteur a donné aussi de bons résultats, par la suite nous prenons comme paramètre optimal ce descripteur.

Comme dans l'étude de la pertinence où les descripteurs LBP3x3 et GLRLM3x3 obtiennent de mauvais résultats en pertinence, dans cette étude on voit aussi que ces descripteurs obtiennent de mauvais résultats en diversité. Donc, par la suite de nos chapitres on évitera d'utiliser ces types de descripteurs visuels.

**6.3.2.2.2 Descripteurs textuels** Dans cette partie nous allons comparer les différents descripteurs textuels avec la `AHC` et utiliser la similarité "cos" comme mesure.

Comme dans l'expérimentation précédente, nous prenons deux combinaisons : des champs "tt" (tag, title) et des champs "ttd" (tag, title, description). Pour cela, nous allons prendre les mêmes descripteurs textuels : "Social", "Proba" et "TfIdf". Le tableau 6.11 montre les scores P@10, CR@10 moyens de la `AHC` avec les trois descripteurs textuels en utilisant différents champs de métadonnées (tt,ttd), une priorité par `Rank` et le découpage `fixed10` sur le benchmark `Media` sous-ensembles `devset` et `testset`. Dans ce tableau on voit que les champs "ttd" donnent globalement des meilleures valeurs en diversité qu'avec les champs "tt". Dans le tableau 6.11 on voit aussi que les descripteurs "TfIdf" et "Social" sont les plus adaptés pour augmenter la diversité. Mais, nous avons

TABLE 6.10 – Comparaison des scores P@10 et CR@10 moyens de la `AHC` avec les descripteurs visuels en utilisant une priorité par `Rank` et le découpage `fixed10` dans un cas réel sur le benchmark `Media` (ils sont triés dans l'ordre de CR@10 moyens sur `devset`). Entre parenthèse nous avons le gain en % comparé avec la baseline

| | | `keywords` | | `keywordsGPS` | | `all` | |
|---|---|---|---|---|---|---|---|
| | | RESULTATS SUR `devset` | | | | | |
| Méthode | | P@10 | CR@10 | P@10 | CR@10 | P@10 | CR@10 |
| Baseline | | 0.688(ref) | 0.464(ref) | 0.860(ref) | 0.412(ref) | 0.774(ref) | 0.438(ref) |
| AHC | CN3x3 | 0.652(-5) | **0.506(+9)** | 0.780(-9) | 0.444(+8) | 0.716(-7) | **0.475(+8)** |
| | CM | 0.692(+1) | 0.474(+2) | 0.800(-7) | **0.454(+10)** | 0.746(-4) | 0.464(+6) |
| | HOG | 0.716(+4) | 0.493(+6) | 0.780(-9) | 0.428(+4) | 0.748(-3) | 0.461(+5) |
| | CN | 0.632(-8) | 0.453(-2) | 0.808(-6) | **0.455(+10)** | 0.720(-7) | 0.454(+4) |
| | CSD | 0.652(-5) | 0.464(+0) | 0.784(-9) | 0.439(+7) | 0.718(-7) | 0.451(+3) |
| | HSV | 0.656(-5) | 0.442(-5) | 0.800(-7) | 0.451(+9) | 0.728(-6) | 0.447(+2) |
| | LBP | 0.648(-6) | 0.453(-2) | 0.736(-14) | 0.426(+3) | 0.692(-11) | 0.439(+0) |
| | CM3x3 | 0.740(+8) | 0.475(+2) | 0.776(-10) | 0.401(-3) | 0.758(-2) | 0.438(+0) |
| | GLRLM | 0.644(-6) | 0.443(-5) | 0.796(-7) | 0.406(-1) | 0.720(-7) | 0.424(-3) |
| | LBP3x3 | 0.684(-1) | 0.451(-3) | 0.728(-15) | 0.383(-7) | 0.706(-9) | 0.417(-5) |
| | GLRLM3x3 | 0.708(+3) | 0.423(-9) | 0.816(-5) | 0.408(-1) | 0.762(-2) | 0.415(-5) |
| | | RESULTATS SUR `testset` | | | | | |
| Méthode | | P@10 | CR@10 | P@10 | CR@10 | P@10 | CR@10 |
| Baseline | | 0.705(ref) | 0.399(ref) | 0.787(ref) | 0.344(ref) | 0.755(ref) | 0.365(ref) |
| AHC | CN3x3 | 0.632(-10) | 0.407(+2) | 0.759(-4) | **0.415(+21)** | 0.710(-6) | **0.412(+13)** |
| | CM | 0.641(-9) | 0.393(-2) | 0.770(-2) | 0.393(+14) | 0.720(-5) | 0.393(+8) |
| | HOG | 0.612(-13) | 0.376(-6) | 0.741(-6) | 0.384(+12) | 0.692(-8) | 0.381(+4) |
| | CN | 0.642(-9) | 0.410(+3) | 0.754(-4) | 0.412(+20) | 0.711(-6) | **0.411(+13)** |
| | CSD | 0.603(-14) | 0.391(-2) | 0.754(-4) | 0.396(+15) | 0.696(-8) | 0.394(+8) |
| | HSV | 0.639(-9) | 0.412(+3) | 0.754(-4) | 0.408(+19) | 0.710(-6) | 0.410(+12) |
| | LBP | 0.631(-10) | **0.414(+4)** | 0.727(-8) | 0.378(+10) | 0.690(-9) | 0.392(+7) |
| | CM3x3 | 0.670(-5) | 0.401(+1) | 0.793(+1) | 0.381(+11) | 0.746(-1) | 0.389(+7) |
| | GLRLM | 0.634(-10) | 0.398(-0) | 0.759(-4) | 0.389(+13) | 0.711(-6) | 0.392(+7) |
| | LBP3x3 | 0.648(-8) | 0.409(+3) | 0.760(-3) | 0.383(+11) | 0.717(-5) | 0.393(+8) |
| | GLRLM3x3 | 0.663(-6) | 0.381(-5) | 0.784(-0) | 0.353(+3) | 0.737(-2) | 0.364(-0) |

vu que "Social" n'a pas donné de bons résultats dans l'étude de la pertinence. Donc, nous prenons le descripteur TfIdf et les champs "ttd" comme paramètres optimaux.

**6.3.2.2.3 Choix des descripteurs** Pour faire un résumé des choix faits, voir le tableau 6.12 qui montre les paramètres de la `AHC` avec différents descripteurs qui ont fourni de bons scores en diversité en utilisant la priorité par `Rank` et un découpage fixe à 10 clusters (`fixed10`). Par la suite, sauf indication contraire, nous prendrons les paramètres indiqués dans ce tableau.

Une fois étudiée les paramètres des descripteurs, nous allons comparer ces descripteurs pour savoir le comportement de ces descripteurs sur la diversité. Dans le tableau 6.13 nous comparons les scores P@10 et CR@10 moyens de la `AHC` avec les meilleurs descripteurs en utilisant une priorité par `Rank` et le découpage `fixed10`. De manière générale, on voit que le fait d'utiliser la `AHC` nous permet globalement d'augmenter les résultats en diversité. De plus, les descripteurs visuels ou l'arborescence de concepts "Voyage" donnent des meilleurs résultats en diversité par rapport aux autres descripteurs. Par la suite, nous prendrons le descripteur visuel CN3x3 comme paramètre optimal pour augmenter la diversité avec la `AHC`.

### 6.3.2.3 Discussion

Dans cette partie nous allons discuter sur les résultats obtenus dans l'étude des paramètres de la pertinence et de la diversité.

**6.3.2.3.1 Choix des descripteurs** En général, nous avons vu que si les descripteurs textuels étaient plus adaptés pour améliorer la pertinence, ceux visuels l'étaient

TABLE 6.11 – Comparaison des scores P@10 et CR@10 moyens de la AHC avec les descripteurs textuels en utilisant différents champs des métadonnées (tt,ttd), la priorité par Rankt le découpage fixed10 dans un cas réel sur le benchmark Media. Entre parenthèse nous avons le gain en % comparé avec la baseline

| | | keywords | | keywordsGPS | | all | |
|---|---|---|---|---|---|---|---|
| | | RÉSULTATS SUR devset | | | | | |
| Méthode | | P@10 | CR@10 | P@10 | CR@10 | P@10 | CR@10 |
| Baseline | | 0.688(ref) | 0.464(ref) | 0.860(ref) | 0.412(ref) | 0.774(ref) | 0.438(ref) |
| AHC | TfIdf(tt) | 0.644(-6) | 0.443(-5) | 0.776(-10) | 0.416(+1) | 0.710(-8) | 0.429(-2) |
| | TfIdf(ttd) | 0.652(-5) | 0.473(+2) | 0.804(-7) | **0.426(+3)** | 0.728(-6) | **0.450(+3)** |
| | Social(tt) | 0.664(-3) | **0.480(+3)** | 0.776(-10) | 0.405(-2) | 0.720(-7) | 0.443(+1) |
| | Social(ttd) | 0.704(+2) | **0.479(+3)** | 0.804(-7) | 0.422(+2) | 0.754(-3) | **0.450(+3)** |
| | Proba(tt) | 0.632(-8) | 0.474(+2) | 0.756(-12) | 0.414(+0) | 0.694(-10) | 0.444(+1) |
| | Proba(ttd) | 0.640(-7) | 0.477(+3) | 0.756(-12) | 0.421(+2) | 0.698(-10) | **0.449(+3)** |
| | | RÉSULTATS SUR testset | | | | | |
| Méthode | | P@10 | CR@10 | P@10 | CR@10 | P@10 | CR@10 |
| Baseline | | 0.705(ref) | 0.399(ref) | 0.787(ref) | 0.344(ref) | 0.755(ref) | 0.365(ref) |
| AHC | TfIdf(tt) | 0.589(-16) | 0.391(-2) | 0.742(-6) | 0.389(+13) | 0.683(-10) | 0.390(+7) |
| | TfIdf(ttd) | 0.612(-13) | 0.406(+2) | 0.755(-4) | **0.403(+17)** | 0.700(-7) | **0.405(+11)** |
| | Social(tt) | 0.599(-15) | 0.383(-4) | 0.748(-5) | 0.387(+13) | 0.690(-9) | 0.386(+6) |
| | Social(ttd) | 0.655(-7) | **0.411(+3)** | 0.765(-3) | 0.397(+15) | 0.723(-4) | 0.402(+10) |
| | Proba(tt) | 0.549(-22) | 0.355(-11) | 0.717(-9) | 0.389(+13) | 0.652(-14) | 0.376(+3) |
| | Proba(ttd) | 0.572(-19) | 0.385(-4) | 0.722(-8) | 0.393(+14) | 0.664(-12) | 0.390(+7) |

TABLE 6.12 – Liste de paramètres de la AHC avec différentes descripteurs qui ont donné de bons résultats en diversité en utilisant une priorité par Rank et le découpage fixe à 10 clusters (fixed10) sur le benchmark Media. Par la suite, sauf indication contraire, nous prendrons les paramètres indiqués dans ce tableau

| Descriptor | | Metadata | Similarity | Linkage |
|---|---|---|---|---|
| Text | TfIdf | tag, title, description | Cosinus | Average |
| | Social | tag, title, description | Cosinus | Average |
| Visual | HSV | - | Euclidean | Average |
| | CN3x3 | - | Euclidean | Average |
| GPS | | GPS | Haversine | Average |
| Tree | | GPS | Wu-Palmer | RootFusion |

davantage pour améliorer la diversité. Le texte donne des informations plus utiles pour retrouver plus d'images pertinentes que par une ressemblance visuelle. Par contre, pour améliorer la diversité, il est plus intéressant d'utiliser du visuel, car dans ce benchmark les experts ont regroupé les images pertinentes par un aspect visuel.

Par rapport aux descripteurs visuels, nous avons vu que CN3x3 et HSV ont donné de bons résultats en pertinence et en diversité. Ces descripteurs appartiennent à la même famille, couleur, mais sont quand même différents. Le descripteur HSV, décrit dans l'annexe B page 197, utilise une palette subjective de 31 couleurs, tandis que le descripteur CN3x3 [61] utilise une stratégie d'apprentissage de 11 couleurs.

Nous avons vu aussi que les descripteurs GLRLM3x3 ou LBP3x3 ont donné de mauvais résultats en pertinence et en diversité. Donc, par la suite nous n'allons pas les reprendre.

Le descripteur HOG obtient de bons résultats en pertinence, mais il est moins bon en diversité. Nous préférons en choisir un qui obtient constamment des résultats supérieurs à la moyenne (comme CN3x3 ou HSV), plutôt que de prendre un descripteur qui obtient de bons résultats dans certains cas (comme HOG).

Par rapport aux descripteurs textuels, nous avons vu que TfIdf donne un bon compromis en pertinence et en diversité. Comme dans HOG, le descripteur Social a donné de bons résultats en diversité, mais de mauvais résultats en pertinence.

TABLE 6.13 – Comparaison des scores P@10 et CR@10 moyens de la `AHC` avec les descripteurs décrites dans le tableau 6.12, en utilisant une priorité par `Rank` et le découpage `fixed10` sur le benchmark `Media`. Entre parenthèse nous avons le gain en % comparé avec la baseline

| | | `keywords` | | `keywordsGPS` | | `all` | |
|---|---|---|---|---|---|---|---|
| | | RÉSULTATS SUR `devset` | | | | | |
| Méthode | | P@10 | CR@10 | P@10 | CR@10 | P@10 | CR@10 |
| Baseline | | 0.688(ref) | 0.464(ref) | 0.860(ref) | 0.412(ref) | 0.774(ref) | 0.438(ref) |
| AHC | TfIdf(ttd) | 0.652(-5) | 0.473(+2) | 0.804(-7) | 0.426(+3) | 0.728(-6) | 0.450(+3) |
| | Social(ttd) | 0.704(+2) | 0.479(+3) | 0.804(-7) | 0.422(+2) | 0.754(-3) | 0.450(+3) |
| | CN3x3 | 0.652(-5) | **0.506(+9)** | 0.780(-9) | 0.444(+8) | 0.716(-7) | 0.475(+8) |
| | HSV | 0.656(-5) | 0.442(-5) | 0.800(-7) | 0.451(+9) | 0.728(-6) | 0.447(+2) |
| | GPS | 0.676(-2) | 0.475(+2) | 0.792(-8) | 0.382(-7) | 0.734(-5) | 0.428(-2) |
| | Tree | 0.704(+2) | 0.488(+5) | 0.840(-2) | **0.468(+14)** | 0.772(-0) | **0.478(+9)** |
| | | RÉSULTATS SUR `testset` | | | | | |
| Méthode | | P@10 | CR@10 | P@10 | CR@10 | P@10 | CR@10 |
| Baseline | | 0.705(ref) | 0.399(ref) | 0.787(ref) | 0.344(ref) | 0.755(ref) | 0.365(ref) |
| AHC | TfIdf(ttd) | 0.612(-13) | 0.406(+2) | 0.755(-4) | 0.403(+17) | 0.700(-7) | 0.405(+11) |
| | Social(ttd) | 0.655(-7) | **0.411(+3)** | 0.765(-3) | 0.397(+15) | 0.723(-4) | 0.402(+10) |
| | CN3x3 | 0.632(-10) | 0.407(+2) | 0.759(-4) | **0.415(+21)** | 0.710(-6) | **0.412(+13)** |
| | HSV | 0.639(-9) | **0.412(+3)** | 0.754(-4) | 0.408(+19) | 0.710(-6) | 0.410(+12) |
| | GPS | 0.609(-14) | 0.376(-6) | 0.757(-4) | 0.390(+13) | 0.700(-7) | 0.385(+5) |
| | Tree | 0.579(-18) | 0.378(-5) | 0.766(-3) | 0.395(+15) | 0.694(-8) | 0.388(+6) |

**6.3.2.3.2 Choix des champs des métadonnées** Dans le choix des champs des métadonnées nous avons constaté que la combinaison "tt" (tag, title) était la plus adaptée pour augmenter la pertinence et "ttd" (tag, title, descriptor) pour augmenter la diversité.

Par rapport à la pertinence, on voit que "tt" donne des meilleurs résultats, parce que les mots choisis dans la requête vont être retrouvés plus précisément dans "tt" (qui sont des mots importants) et pas avec "ttd" qui contient des informations complémentaires à l'image qui peuvent faire également ressortir des images qui n'ont rien à voir avec la requête (rajout du bruit). Par exemple, si la requête est "fleur blanche" :

— img1. rank="1", tt="fleur blanche", d="printemps".
— img2. rank="2", tt="fleur blanche", d="Italie".
— img3. rank="3", tt="transport", d="fleur, voiture blanche".

Donc, si on utilise un réordonnancement textuel avec les champs "tt" on obtiendra seulement les images "img1" et "img2". Par contre, si on utilise les champs "ttd" on obtiendra aussi l'image "img3" qui est une image non-pertinente à la requête.

Par rapport à la diversité, on voit que "ttd" donne des meilleurs résultats, car si on n'utilise pas les champs description, plusieurs images vont se retrouver avec quasiment les mêmes mots et donc ils vont se regrouper alors qu'elles sont peut-être différentes. Par exemple, si la requête est "Tour Eiffel" :

— img1. rank="1", tt="tour Eiffel", d="porte clef".
— img2. rank="2", tt="tour Eiffel", d="porte clef".
— img3. rank="3", tt="tour Eiffel", d="belle photo".

Donc, si on utilise la `AHC` textuel avec "tt", on obtiendra un seul cluster (cluster "tour Eiffel") et les images résultats seront : img1, img2, img3. Par contre, si on utilise les champs "ttd", on obtiendra deux clusters (cluster "porte clef" et cluster "belle photo") et les images résultats seront un peu plus diversifiées : img1, img3, img2.

## 6.4 Comparaison avec l'état de l'art lors des campagnes d'évaluation

Après avoir étudié nos méthodes proposées avec plusieurs paramètres, dans la section qui suit nous allons comparer nos méthodes proposées avec des autres méthodes de

TABLE 6.14 – Comparaison avec des autres méthodes de diversité présentées dans un cas réel sur `Clef`. CR@20 moyen : moyenne des CR@20 de 39 requêtes

| Méthode | P@20 moyen | CR@20 moyen |
|---|---|---|
| baseTfIdf | 0.250 (ref) | 0.291 (ref) |
| CRmax | 0.997 (+299%) | 0.991 (+241%) |
| Random | 0.166 (-34%) | 0.263 (-10%) |
| DIVALEA [59] | 0.215 (-14%) | 0.298 (+2%) |
| DIVVISU [59] | 0.215 (-14%) | 0.305 (+5%) |
| VISKMEANS [35] | 0.232 (-7%) | 0.306 (+5%) |
| `AHC`(HSV) `Rank fixed20` | 0.194 (-22%) | 0.281 (-3%) |

diversité de l'état de l'art. Pour cela, nous utilisons deux benchmarks publics présents lors de campagnes internationales.

### 6.4.1 Campagne ImageCLEF 2008 (benchmark `Clef`)

Dans ce benchmark nous nous comparons avec des autres méthodes de l'état de l'art qui utilisent seulement des descripteurs visuels et qui sont : la DIVALEA [59] qui permute aléatoirement les 40 premiers résultats, la VISKMEANS [35] qui est une méthode de partitionnement des données et la DIVVISU [59] basée sur un partitionnement de l'espace visuel.

Nous comparons ces méthodes sur deux cas d'étude de la diversité, un cas réel et un cas idéal.

**Comparaison des résultats dans un cas réel** Le tableau 6.14 compare nos résultats avec ceux de l'état de l'art dans un cas réel. La baseline "baseTfIdf" est très faible en pertinence (P@20 = 0.25) car le résultat de cette baseline a été obtenu à partir de requêtes effectuées sur un corpus de seulement 20000 images, où il y a peu d'images pertinentes et où il est donc difficile d'en retrouver beaucoup dès les premiers résultats, ce qui donne en conséquence une faible diversité (CR@20 = 0.291).

Par rapport aux méthodes aléatoires, on voit que dans un cas réel, il semble qu'il vaille mieux choisir un nombre réduit d'images (40 premières images sur DIVALEA) pour les diversifier aléatoirement que de choisir un ensemble plus grand, 150 premières images sur Random, car si on en prend plus on risque d'être trop pénalisé par les images non-pertinentes du cas réel.

Nous remarquons aussi que dans un cas réel notre méthode proposée qui utilise un clustering hiérarchique obtient de moins bons résultats en diversité par rapport aux deux méthodes du clustering par partitionnement, de -8% par rapport à VISKMEANS et DIVVISU.

**Comparaison des résultats dans un cas idéal** Le tableau 6.15 compare nos résultats avec ceux de l'état de l'art dans un cas idéal. Le CR@20 moyen sur les 10 runs idéaux (voir section 6.2.3) est de 0.754, ce qui est déjà assez élevé, car une permutation aléatoire a été réalisée sur ces runs, il est donc difficile d'augmenter la diversité sur ces 10 runs.

Dans les méthodes aléatoires on voit que contrairement au cas réel, dans un cas idéal où on est plus pénalisé par des images non-pertinentes il vaut mieux choisir un ensemble plus grand d'images (150 premières images pour Random) pour diversifier aléatoirement les images que d'en choisir un nombre réduit (40 premières images pour DIVALEA) car

TABLE 6.15 – Comparaison avec des autres méthodes de diversité présentées dans un cas idéal sur `Clef`. CR@20 moyen : moyenne sur 10 runs idéaux de la moyenne des CR@20 de 39 requêtes

| Méthode | CR@20 moyen |
|---|---|
| Runs ideaux | 0.754 (ref) |
| CRmax | 0.991 (+31%) |
| Random | 0.774 (+3%) |
| DIVALEA [59] | 0.764 (+2%) |
| DIVVISU [59] | 0.787 (+5%) |
| VISKMEANS [35] | 0.767 (+2%) |
| `AHC`(HSV) `Rank fixed20` | 0.817 (+8%) |

si on en prend plus on aurait plus de probabilité de trouver des images appartenant à des sous-thèmes qui ne sont pas facilement retrouvés dans les premières images.

Contrairement au cas réel, dans le cas idéal on voit que notre méthode proposée obtient de meilleurs résultats en diversité que les deux méthodes de clustering par partitionnement, de +5% par rapport à VISKMEANS et de +7% par rapport à DIVVISU.

**Discussion** Il faut remarquer qu'on ne peut pas vraiment conclure sur les différentes méthodes lors de cette campagne, car ces expériences ne sont pas toujours exactement comparables où les descripteurs visuels utilisés et le nombre de clusters sont différents.

Les résultats obtenus dans le cas idéal montrent l'intérêt d'utiliser un clustering hiérarchique par rapport aux méthodes de clustering par partitionnement (`K-Means`) et par rapport aux différents types de diversité aléatoire pour augmenter la diversité. Par contre, dans un cas réel où nous avons la particularité d'avoir un baseline trop faible en pertinence (P@20 = 0.25), notre méthode proposée obtient de mauvais résultats en diversité (avec des scores qui sont autour de la baseline) ce qui n'a pas permis d'obtenir les mêmes résultats que dans le cas idéal.

### 6.4.2 Campagne MediaEval 2013 (benchmark `Media`)

Cette campagne internationale [24] s'est déroulée en 2013 et a offert un total de douze tâches. Nous avons participé à celle appelée "Retrieving Diverse Social Images" à laquelle 11 chercheurs ont participé. Ils ont utilisé différentes méthodes de diversité (des méthodes de clustering, représentation des graphes, algorithmes gloutonnes, etc.).

Les organisateurs ont fourni les requêtes du `devset` (50 requêtes) avec leur vérité terrain (4 mois à l'avance), nous avons alors essayé d'optimiser les paramètres sur `devset`. Puis les organisateurs nous ont fourni les requêtes du `testset` (342 requêtes) (2 semaines à l'avance), nous avons alors soumis des runs en prenant les paramètres optimaux obtenues sur `devset`.

L'inconvénient que nous avons subi pendant l'étape d'optimisation des paramètres, est que le nombre de requêtes disponibles était un peu insuffisant pour obtenir des comportements stables. Par contre, un avantage de ce benchmark a été la quantité de descripteurs disponibles (du texte, du visuel) ce qui nous a permis d'étudier l'influence de chaque descripteur sur la pertinence et la diversité. Par exemple, nous avons remarqué qu'augmenter la pertinence par le texte, puis faire un clustering visuel semblait être une bonne stratégie pour augmenter la diversité. Nous aurions aimé faire aussi des expérimentations en combinant les descripteurs visuels et textuels avec une fusion tardive pour savoir si cette stratégie permet aussi d'obtenir de bons résultats en diversité.

TABLE 6.16 – Runs : paramètres (en haut), résultats obtenus sur `devset` (au milieu) et résultats obtenus sur `testset` (en bas). Entre parenthèses, le gain en % comparé avec la baseline

| | `keywords` | | | `keywordsGPS` | | |
|---|---|---|---|---|---|---|
| | PARAMÈTRES DES RUNS | | | | | |
| | Rerank | AHC | Priority | Rerank | AHC | Priority |
| run1_vis | LBP3x3 | CSD | dec,fixed20 | Baseline | CN3x3 | dec,fixed35 |
| run2_txt | TfIdf(ttd) | Social(ttd) | rnk,fixed25 | TfIdf(tt) | Proba(tt) | rnk,fixed35 |
| run3_txtvis | TfIdf(ttd) | CSD | rnk,fixed25 | TfIdf(tt) | CSD | dec,fixed20 |
| run5_all | TfIdf(ttd) | Tree(ttd) | rnk,fixed15 | Tree | CSD | dec,fixed30 |
| run1_final | Baseline | CN3x3 | rnk,fixed10 | Baseline | CN3x3 | rnk,fixed10 |
| run2_final | TfIdf(tt) | - | - | TfIdf(tt) | - | - |
| run3_final | TfIdf(tt) | CN3x3 | rnk,fixed10 | TfIdf(tt) | CN3x3 | rnk,fixed10 |

#### 6.4.2.1 Runs soumis sur `Media`

Selon les résultats obtenus sur `devset`, dans [2] nous avons choisi de prendre les **paramètres optimaux** de chaque sous-ensemble (`keywords` et `keywordsGPS`). Puis, nous avons soumis 4 types de runs qui utilisent différents types de descripteurs :

**run1_vis** Run avec seulement des descripteurs visuels ;

**run2_txt** Run avec seulement des descripteurs textuels ;

**run3_txtvis** Run avec l'utilisation des descripteurs textuels, visuels, GPS ;

**run5_all** Run avec n'importe quel type de descripteur (par exemple une arborescence de concepts).

Dans cette thèse, nous faisons un autre choix qui est d'utiliser les paramètres choisis dans la partie 6.3.2 afin d'obtenir un **bon compromis** de pertinence et diversité sur `devset`. Avec ce type de choix nous ajoutons 3 nouveaux runs appelés :

**run1_final** Run avec les paramètres qui donnent un bon compromis en pertinence ;

**run2_final** Run avec les paramètres qui donnent un bon compromis en diversité ;

**run3_final** Run avec les paramètres qui donnent un bon compromis en pertinence et diversité.

Le tableau 6.17 résume les paramètres de chaque run et les scores obtenus sur `devset` et `testset` (vérité terrain fait par des experts), tandis que le tableau 6.18 compare les scores obtenus sur le Crowd-Sourcing.

Si on compare les runs des deux types de choix, on voit que celui des paramètres optimaux donne de meilleurs résultats sur `devset`. Par contre, le choix des paramètres qui ont un bon compromis pertinence-diversité donne des meilleurs résultats sur `testset` composé d'un ensemble avec beaucoup plus de requêtes que sur `devset`.

Donc, le fait de choisir des paramètres optimaux n'est pas vraiment une bonne idée, car avec ce choix on risque d'hyper-optimiser les paramètres et de tomber sur des cas particuliers (fausse optimisation) qui vont donner sûrement de bons résultats sur `devset` mais qui n'assurent pas de bons résultats sur un autre ensemble (`testset`).

Par contre, le fait de choisir des paramètres qui donnent un bon compromis de pertinence et de diversité évite de tomber sur des cas particuliers. Peut-être que ces paramètres ne vont pas donner les meilleurs résultats sur `devset`, mais on aurait plus de probabilité à obtenir de bons résultats sur un autre ensemble (`testset`).

Dans les tableaux on voit que faire du réordonnancement par le texte (run2_final) permet d'augmenter la pertinence. Et on voit aussi que le fait d'augmenter la pertinence par le texte, puis faire du clustering visuel (run3_final) permet d'obtenir des meilleurs résultats en diversité par rapport à l'utilisation directe de la `AHC` (run1_final).

Dans les tableaux on voit bien que le comportement sur `keywords` et `keywordsGPS` n'est pas le même, or dans la partie 6.3.2 nous avons étudié les paramètres sur l'ensemble

TABLE 6.17 – Runs : paramètres (en haut), résultats obtenus sur `devset` (au milieu) et résultats obtenus sur `testset` (en bas). Entre parenthèses, le gain en % comparé avec la baseline

| | `keywords` | | `keywordsGPS` | | `all` | |
|---|---|---|---|---|---|---|
| | RÉSULTATS SUR `devset` | | | | | |
| nombre de requêtes | 25 | | 25 | | 50 | |
| | P@10 | CR@10 | P@10 | CR@10 | P@10 | CR@10 |
| Baseline | 0.688(ref) | 0.464(ref) | 0.860(ref) | 0.412(ref) | 0.774(ref) | 0.438(ref) |
| CRmax | 0.952(+38) | 0.868(+87) | 1.000(+16) | 0.794(+93) | 0.976(+26) | 0.831(+90) |
| Random | 0.692(+1) | 0.470(+1) | 0.808(-6) | 0.439(+7) | 0.750(-3) | 0.455(+4) |
| run1_vis | 0.696(+1) | 0.543(+17) | 0.868(+1) | 0.498(+21) | 0.782(+1) | 0.520(+19) |
| run2_txt | 0.788(+15) | **0.586(+26)** | **0.928(+8)** | 0.493(+20) | **0.858(+11)** | 0.540(+23) |
| run3_txtvis | **0.812(+18)** | **0.584(+26)** | 0.844(-2) | **0.509(+24)** | 0.828(+7) | **0.547(+25)** |
| run5_all | 0.760(+10) | 0.560(+21) | 0.808(-6) | 0.483(+17) | 0.784(+1) | 0.521(+19) |
| run1_final | 0.652(-5) | 0.506(+9) | 0.780(-9) | 0.444(+8) | 0.716(-7) | 0.475(+8) |
| run2_final | 0.772(+12) | 0.493(+6) | 0.900(+5) | 0.429(+4) | 0.836(+8) | 0.461(+5) |
| run3_final | 0.700(+2) | 0.515(+11) | 0.808(-6) | 0.462(+12) | 0.754(-3) | 0.489(+12) |
| | RÉSULTATS SUR `testset` | | | | | |
| nombre de requêtes | 210 | | 132 | | 342 | |
| | P@10 | CR@10 | P@10 | CR@10 | P@10 | CR@10 |
| Baseline | 0.705(ref) | 0.399(ref) | 0.787(ref) | 0.344(ref) | 0.755(ref) | 0.365(ref) |
| CRmax | 0.965(+37) | 0.813(+104) | 0.990(+26) | 0.715(+108) | 0.980(+30) | 0.753(+106) |
| Random | 0.581(-18) | 0.353(-12) | 0.757(-4) | 0.375(+9) | 0.689(-9) | 0.367(+1) |
| run1_vis | 0.630(-11) | 0.400(+0) | 0.774(-2) | 0.370(+8) | 0.718(-5) | 0.382(+5) |
| run2_txt | 0.745(+6) | 0.412(+3) | 0.844(+7) | 0.404(+17) | 0.806(+7) | 0.407(+12) |
| run3_txtvis | 0.718(+2) | **0.417(+5)** | 0.823(+5) | 0.426(+24) | 0.782(+4) | 0.423(+16) |
| run5_all | 0.705(+0) | 0.388(-3) | 0.766(-3) | 0.378(+10) | 0.742(-2) | 0.382(+5) |
| run1_final | 0.632(-10) | 0.407(+2) | 0.759(-4) | 0.415(+21) | 0.710(-6) | 0.412(+13) |
| run2_final | **0.764(+8)** | 0.403(+1) | **0.868(+10)** | 0.373(+8) | **0.828(+10)** | 0.385(+5) |
| run3_final | 0.640(-9) | 0.412(+3) | 0.799(+2) | **0.438(+27)** | 0.737(-2) | **0.428(+17)** |

TABLE 6.18 – Scores obtenus pour la vérité terrain de Crowd-Sourcing (GT1, GT2, GT3) et pour la vérité terrain des experts (GT0). Les scores représentent la moyenne d'un sous-ensemble de 49 requêtes de `testset`. *nb* est le nombre de requêtes entre les 49 requêtes qui ont un CR@10=1. Entre parenthèses, nous avons le gain en % comparé avec la baseline

| | VÉRITÉ TERRAIN CROWD-SOURCING | | | | | | |
|---|---|---|---|---|---|---|---|
| | GT1,2,3 | GT1 | | GT2 | | GT3 | |
| | P@10 | CR@10 | *nb* | CR@10 | *nb* | CR@10 | *nb* |
| Baseline | 0.798 (ref) | 0.754 (ref) | 29 | 0.667 (ref) | 20 | 0.572 (ref) | 15 |
| run1_vis | 0.694(+2) | 0.786(+4) | 27 | 0.754(+13) | 20 | 0.645(+13) | 14 |
| run2_txt | 0.757(+11) | 0.836(+11) | 31 | 0.756(+13) | 16 | 0.645(+13) | 14 |
| run3_txtvis | 0.749(+10) | **0.886(+18)** | 33 | **0.792(+19)** | 21 | **0.687(+20)** | 16 |
| run5_all | 0.708(+4) | 0.828(+10) | 29 | 0.768(+15) | 20 | 0.643(+12) | 15 |
| run1_final | 0.663(-3) | 0.849(+13) | 29 | 0.777(+16) | 20 | 0.670(+17) | 15 |
| run2_final | **0.818(+20)** | 0.824(+9) | 29 | 0.733(+10) | 18 | 0.650(+14) | 15 |
| run3_final | 0.696(+2) | 0.833(+10) | 29 | 0.782(+17) | 20 | 0.676(+18) | 15 |
| | VÉRITÉ TERRAIN DES EXPERTS GT0 | | | | | | |
| | P@10 | CR@10 | | | | | *nb* |
| Baseline | 0.798 (ref) | 0.335 (ref) | | | | | 0 |
| run1_vis | 0.806(+1) | 0.367(+10) | | | | | 0 |
| run2_txt | 0.851(+7) | 0.408(+22) | | | | | 1 |
| run3_txtvis | 0.841(+5) | 0.415(+24) | | | | | 0 |
| run5_all | 0.794(-1) | 0.377(+13) | | | | | 0 |
| run1_final | 0.782(-2) | 0.398(+19) | | | | | 0 |
| run2_final | **0.896(+12)** | 0.369(+10) | | | | | 0 |
| run3_final | 0.839(+5) | **0.446(+33)** | | | | | 0 |

composé des deux sous-ensembles, ce qui fait que cela favorise l'ensemble `keywordsGPS` (qui contient plus de requêtes sur `testset`). Peut-être si nous avions choisi de prendre

TABLE 6.19 – Comparaison des scores P@10, CR@10 moyens des résultats des autres participants qui ont participé au benchmark `Media` sous-ensemble `testset`

| Équipe (Méthode) | Run | `testset` P@10 | `testset` CR@10 |
|---|---|---|---|
| Baseline | - | 0.755 (ref) | 0.365 (ref) |
| SOTON-WAIS (Minmax) [26] | run3_txtvis | 0.816 (+8%) | **0.440 (+21%)** |
| SocSens (Greedy) [10] | run1_vis | 0.733 (-3%) | 0.429 (+18%) |
| UPMC (Rerank+AHC) | run3_final | 0.737 (-2%) | 0.428 (+17%) |
| UPMC (Rerank+AHC) [2] | run3_txtvis | 0.783 (+4%) | 0.423 (+16%) |
| CEA (Maxavg) [45] | run2_txt | 0.769 (+2%) | 0.424 (+16%) |
| BMEMTM (face/AHC) [56] | run1_vis | 0.739 (-2%) | 0.408 (+12%) |
| MUCKE (kNN/Kmeans++) [3] | run2_txt | 0.724 (-4%) | 0.389 (+7%) |

les paramètres qui ont un bon compromis de pertinence et de diversité sur chaque sous-ensemble indépendamment les scores seraient meilleurs.

#### 6.4.2.2 Comparaison avec les autres participants

Dans le tableau 6.19 nous montrons les meilleurs scores P@10, CR@10 moyens de plusieurs participants qui ont utilisé différentes méthodes pour augmenter la diversité sur le benchmark `Media` sous-ensemble `testset` où on voit que notre méthode proposée (notre meilleure run) est positionné à la quatrième place d'un total de 39 runs soumis.

On ne peut pas vraiment conclure sur les différentes méthodes utilisées lors de la campagne, car certains participants ont ajouté des prétraitements (comme dans SOTON-WAIS) et d'autres non, les expériences ne sont donc pas toujours exactement comparables.

Par rapport aux meilleurs résultats obtenus, on voit que des méthodes qui n'ont pas utilisé du clustering ont obtenu des résultats assez similaires en diversité par rapport à notre méthode de clustering hiérarchique, avec un gain d'environ 18% par rapport à la Baseline. Il semblerait que les experts ont été rigoureux au moment de regrouper des images, il en résulte une tâche très difficile pour augmenter la diversité. Mais, on a vu aussi que la plupart des participants ont bien réussi à obtenir des meilleurs résultats que la Baseline.

On peut noter que des travaux qui ont utilisé du clustering par partitionnement ont obtenu de moins bons résultats en diversité que notre méthode de clustering hiérarchique.

## 6.5 Bilan et discussion

Dans cette section nous allons comparer d'abord les avantages et inconvénients des benchmarks utilisés dans ce cadre expérimental. Dans le tableau 6.20 nous résumons les caractéristiques de chaque benchmark et nous comparons quelques caractéristiques importantes.

— **Provenance du benchmark.** Contrairement au benchmark `Xilo` que nous avons construit avec la collaboration d'une entreprise, les benchmark `Clef` et `Media` sont des benchmarks publics utilisés lors de campagnes internationales ce qui nous a permis de nous placer dans les mêmes conditions et de nous comparer aux autres méthodes de diversité de l'état de l'art.

— **Nombre de requêtes.** Contrairement au benchmark `Xilo` où nous disposons seulement de 21 requêtes, dans `Media` nous disposons d'un nombre de requêtes beaucoup plus grand (396 requêtes). Cet avantage nous permet d'avoir des scores

TABLE 6.20 – Comparaison des caractéristiques sur les benchmarks `Xilo`, `Clef` et `Media`. En bas nous montrons aussi les paramètres (descripteur "Desc", similarité "Sim", agrégation "Link") choisis pour la `AHC` et pour Rerank. Les notations utilisés pour les mesures de similarité sont : Euclidean "Euc", Haversine "Hav", Wu-Palmer généralisée "WuG" ; et pour les critères d'agrégation : Average "Avg", RootFusion "Root"

| Benchmark | `Xilo` | | `Clef` | | `Media` | |
|---|---|---|---|---|---|---|
| CARACTÉRISTIQUES DES BENCHMARKS | | | | | | |
| Provenance | Construit | | Campagne Internationale | | Campagne Internationale | |
| Baseline | Moteur Xilopix | | Prend tout le corpus d'images | | Moteur Flickr | |
| Nb requêtes | 21 | | 39 | | 392 | |
| Nb moyen sous-thèmes | 10.9 | | 7.9 | | 13.31 | |
| Descripteurs choisis | HSV,Tree | | HSV | | HSV, CN3x3, TfIdf, GPS | |
| Type de descripteur | PARAMÈTRES CHOISIS POUR `AHC` | | | | | |
| | Desc | Sim,Link | Desc | Sim,Link | Desc | Sim,Link |
| Visuel | HSV | Euc,Avg | HSV | Euc,Avg | HSV,CN3x3 | Euc,Avg |
| Text | - | - | - | - | TfIdf(ttd) | Cos,Avg |
| GPS | - | - | - | - | GPS | Hav,Avg |
| Tree | Tree | WupG,Root | - | - | - | - |
| Type de descripteur | PARAMÈTRES CHOISIS POUR Rerank | | | | | |
| | Desc | Sim | Desc | Sim | Desc | Sim |
| Visuel | - | - | - | - | CN3x3 | Euc |
| Text | - | - | - | - | TfIdf(tt) | Sum |
| GPS | - | - | - | - | GPS | Hav |
| Tree | - | - | - | - | - | - |

TABLE 6.21 – Scores de la baseline sur les trois benchmarks `Xilo`, `Clef` et `Media`. Pour `Clef` nous avons pris la baseline "baseTfIdf" et pour `Media` nous avons pris la baseline du sous-ensemble `testset`

| Benchmark | `Xilo` | | `Clef` | | `Media` | |
|---|---|---|---|---|---|---|
| | P@n | CR@n | P@n | CR@n | P@n | CR@n |
| Scores moyens : | | | | | | |
| n=10 | 1.000 | 0.271 | 0.277 | 0.217 | 0.755 | 0.365 |
| n=20 | 1.000 | 0.428 | 0.250 | 0.291 | 0.727 | 0.535 |

moyens de ces requêtes qui soient très robustes aux cas particuliers (des requêtes avec des valeurs abruptes).

— **Descripteurs disponibles.** Contrairement aux autres benchmarks, dans `Media` nous disposons d'une grande variété de types de descripteurs, ce qui nous permet d'étudier l'influence de chaque descripteur sur la diversité. Par contre, dans `Xilo` nous disposons d'une arborescence de concepts qui est une source d'information très riche et intéressant pour augmenter la diversité.

Pour comparer la baseline de chaque benchmark, dans le tableau 6.21 nous montrons les scores CR@n et P@n moyens de chaque baseline. Pour `Clef` nous avons pris la baseline "baseTfIdf" et pour `Media` nous avons pris la baseline du sous-ensemble `testset`. Dans ce tableau nous voyons que la baseline du benchmark `Clef` a une pertinence plus faible par rapport aux autres baselines, ce qui donne en conséquence des résultats aussi faibles en diversité. Les baselines `Xilo` et `Media` ont une forte pertinence, car les résultats de ces baselines ont été obtenues à partir des requêtes effectuées sur de grandes bases d'images qui contiennent beaucoup d'images pertinentes, et donc pour lesquels il est facile de retrouver dès les premiers résultats beaucoup d'images pertinentes, tandis que la baseline `Clef` a été obtenue à partir de requêtes effectuées sur un corpus de seulement 20000

images, où il y a peu d'images pertinentes, et où il est donc difficile de retrouver beaucoup d'images pertinentes dès les premiers résultats.

Puis, nous avons étudié les scores obtenus de nos méthodes proposées sur différents descripteurs de chaque benchmark. Nous avons vu que dans `Xilo` nos méthodes basées sur l'arborescence de concepts ont donné des meilleurs résultats par rapport à la couleur. Et dans `Media` nos méthodes basées sur la couleur ont donné des meilleurs résultats par rapport au texte et aux coordonnées GPS. Il semblerait donc qu'utiliser nos méthodes proposées avec une approche de couleur ou sémantique (arborescence de concepts) semble être très intéressant pour augmenter la diversité, malgré la différence entre ces approches.

Nous avons aussi étudié les scores de nos méthodes proposées sur différents types de baseline. Nous avons vu que dans `Clef` où sa baseline a une faible pertinence, nos méthodes proposées obtiennent de mauvais résultats en diversité. Par contre, dans `Media` et `Xilo` où sa baseline a une forte pertinence, nos méthodes donnent de bons résultats en diversité. De plus, dans `Media` nous avons vu que le fait d'augmenter la pertinence initiale de la baseline (par le texte), nous a permis d'augmenter encore plus la diversité (par un clustering visuel), par rapport à l'utilisation directe de la `AHC`. Il semblerait donc que la pertinence initiale de la baseline influe sur la diversité de nos méthodes proposées.

Il faut remarquer que dans le benchmark `Media` on n'a pas étudié les paramètres dans le cas idéal.

Finalement, dans ce cadre expérimental nous nous comparons avec des autres méthodes de diversité de l'état de l'art dans les benchmarks `Clef` et `Media`. Par contre, il faut remarquer qu'on ne peut pas vraiment conclure sur les différentes méthodes lors de ces campagnes, car ces expériences ne sont pas toujours comparables (par exemple avec l'ajout des prétraitements pour certains chercheurs ou l'utilisation de descripteurs différents, etc.). D'où l'importance de définir un cadre expérimental avec plusieurs benchmarks avec un descripteur en commun pour comparer les résultats obtenus sur les trois benchmarks et avec plusieurs descripteurs pour étudier l'influence des descripteurs sur la diversité. La comparaison de ces méthodes de diversité serait étudiée plus en détail dans les chapitres suivants.

Dans le tableau 6.20 nous montrons un résumé des paramètres choisis de notre méthode proposée (pour Rerank et `AHC`) sur chaque benchmark et qui ont donné globalement un bon compromis en pertinence et diversité. Par la suite des prochains chapitres, sauf indication contraire, nous prendrons les paramètres indiqués dans ce tableau.

# Chapitre 7

# Étude de la diversité par clustering hiérarchique

Dans ce chapitre, nous allons étudier l'utilisation de la méthode de clustering agglomératif hiérarchique (`AHC`) pour la diversité.

La diversité est un problème que nous pensons hiérarchique, car pour une requête donnée, les résultats peuvent être regroupés par thème, puis par sous-thème et ainsi de suite. Nous pensons aussi que la `AHC` est une méthode très intéressante pour faire de la diversité, car elle génère une hiérarchie de clusters qui peuvent bien correspondre aux différents niveaux de granularité de la diversité.

Une difficulté dans la `AHC`(en général dans les méthodes de clustering) est le choix du nombre de clusters que dans la littérature ce choix est un problème sans réponse unique. Cependant, la `AHC` peut offrir une alternative à ce problème, car la hiérarchie de clusters générés par cette méthode permet à l'utilisateur de changer facilement le niveau de granularité de diversité sans avoir besoin de relancer à nouveau la `AHC` (comme dans `K-Means` par exemple où il faut relancer à nouveau la méthode pour un nombre de clusters différents).

Une limitation de la `AHC` est sa complexité de type quadratique ($O(n^2)$). Est-ce qu'utiliser la `AHC` dans une application de recherche d'images "en ligne" est adapté ?

Une autre limitation de cette méthode est qu'elle a besoin de plusieurs paramètres comme par exemple le type de descripteur, le choix du nombre de clusters ou l'ordre de clusters à utiliser. De plus, nous n'avons pas trouvé dans la littérature des travaux qui ont fait une étude approfondie sur ces paramètres. Le fait d'avoir plusieurs paramètres peut rendre difficile la tâche de trouver un bon choix des paramètres pour augmenter la diversité, car ils peuvent être spécifiques à un seul cas particulier. D'ici l'importance d'étudier cette méthode sur plusieurs benchmarks et sur plusieurs descripteurs pour obtenir des comportements qui soient les plus généralisées possibles. Il faut remarquer que les descripteurs (et ses paramètres) que nous utilisons sont détaillés dans le tableau 6.20 page 92. Pour le benchmark `Media` nous utilisons seulement le sous-ensemble `testset`, car ce sous-ensemble est plus robuste que `devset` en termes de quantité de requêtes.

Pour construire la hiérarchie de clusters, la `AHC` prend comme paramètres une liste d'images résultats (baseline) et le type de descripteur à utiliser. Les baselines du benchmark `Xilo` et `Media` ont une forte pertinence, car les résultats de ces baselines ont été obtenus à partir de requêtes effectuées sur de grandes bases d'images dont beaucoup sont pertinentes et donc pour lesquelles il est facile de retrouver dès les premiers résultats beaucoup d'images pertinentes, tandis que la baseline du `Clef` a été obtenue à partir de requêtes effectuées sur un corpus de seulement 20 000 images, où il y a peu d'images pertinentes et où il est donc difficile de retrouver beaucoup d'images pertinentes dès les premiers résultats. Ces différents types de baseline nous permet de faire une première

étude sur les conséquences d'utiliser différents types de pertinence sur la AHC.

Un cas souhaitable pour augmenter la diversité avec la AHC est qu'il puisse générer une hiérarchie de clusters qui soient les plus similaires aux sous-thèmes de la vérité terrain. Pour cela, le bon choix du type de descripteur peut être très important, car il peut influencer sur la qualité de cette hiérarchie. Par exemple, si on utilise la AHC avec la couleur, sa hiérarchie de clusters ne va pas être la même qu'avec une hiérarchie généré par le texte. Donc, d'ici l'intérêt d'étudier la AHC sur plusieurs descripteurs.

Dans le benchmark Xilo nous disposons d'une arborescence de concepts. Dans la littérature la plupart des chercheurs utilisent souvent les feuilles de cette arborescence afin de diversifier les images. Dans cette thèse, nous proposons d'utiliser cette ressource comme descripteur dans la AHC pour profiter de la structure hiérarchique de cette ressource et pouvoir ainsi obtenir des meilleurs résultats en diversité.

Dans la partie 7.1, nous expliquons les particularités de l'utilisation de la AHC pour la diversité, ainsi que les méthodes que nous proposons pour l'utiliser. Dans la partie 7.2, nous étudions l'influence de l'ordre des clusters sur la diversité en comparant une technique de priorité classique par le rang avec deux techniques que nous proposons lesquelles qui trient les clusters par leur taille. Dans la partie 7.3, nous étudions l'influence du nombre de clusters sur la diversité en comparant une technique de découpage classique, une qui prend un nombre fixe de clusters et une qui prend un nombre de clusters égal au nombre des sous-thèmes de la vérité terrain. Finalement, dans la partie 7.4, nous comparons un réordonnancement classique avec un réordonnancement que nous proposons et qui tire profit de la hiérarchie de clusters généré par la AHC.

## 7.1 Adaptation de la AHC pour la diversité

La diversité est un problème que nous pensons hiérarchique, car pour une requête donnée, les résultats peuvent être regroupés par thème, puis par sous-thème et ainsi de suite. Pour cette raison, nous pensons aussi que la AHC [31] (décrite dans la section 3.2.1 page 34) pourrait être une méthode très intéressante pour faire de la diversité, car elle génère une hiérarchie de clusters d'images qui peut bien correspondre aux différents niveaux de granularité de la diversité. Ainsi, cette méthode pourrait permettre à l'utilisateur de parcourir en profondeur la hiérarchie en fonction du niveau de granularité qu'il souhaite obtenir sans avoir besoin de relancer la méthode.

Pour utiliser la AHC sur la diversité, nous générons d'abord une hiérarchie de clusters avec la AHC, puis nous coupons de manière horizontale la hiérarchie pour obtenir des clusters non ordonnés. Enfin, nous trions les clusters obtenus et nous réordonnons les images en alternant les images de chaque cluster. Dans cette section nous expliquons les paramètres particuliers de la AHC sur la diversité, ainsi que les méthodes que nous proposons pour mieux adapter cette méthode à la diversité.

### 7.1.1 Critères d'agrégation

Pour générer la hiérarchie de clusters, la AHC a besoin des mesures de similarité pour calculer celle entre deux images et des critères d'agrégation pour calculer la similarité entre deux clusters. Les mesures classiques de similarité qui vont servir sont déjà décrites dans le chapitre 2.

Pour calculer la similarité entre deux clusters, il existe de nombreux critères d'agrégation. Citons par exemple le lien simple, le lien complet, le lien moyen, etc. Le critère centroïde est un critère d'agrégation classique qui calcule la distance entre deux clusters en calculant la distance entre les barycentres des clusters, mais ce critère n'est pas adapté pour certains descripteurs comme les coordonnées GPS où son vecteur moyen ne

correspondrait pas à la réalité. Ces critères classiques d'agrégation sont détaillés dans la section 3.2.1.1 page 35.

**Critère d'agrégation proposé "RootFusion"** Dans le cas où les images sont décrites par un ensemble de chemins dans une arborescence, les critères classiques ne sont pas adaptés à ce type de descripteur, c'est pourquoi nous proposons une nouvelle méthode appelée "RootFusion" adaptée à l'arborescence de concepts. En nous inspirant du critère centroïde, la méthode proposée consiste à calculer d'abord un représentant pour chaque cluster, puis calculer la dissimilarité $\delta_G$ entre les deux représentants. Le représentant est composé par les nœuds communs des images qui sont dans le cluster. À l'initialisation de la AHC, chaque cluster est initialisé par une seule image : le représentant du cluster $Ci_1$ est donc l'ensemble de chemins $A_{i_1}$ de l'image $i_1$. Ensuite, le système calcule la dissimilarité $\delta_G$ entre tous les couples de clusters et agrège les deux clusters dont les représentants sont les plus similaires. Les deux représentants sont alors agrégés en gardant leurs nœuds communs afin de former le représentant du nouveau cluster. Par exemple, pour l'arborescence de la figure 1.4 page 7, l'image $i_1$ est associé aux concepts : "Italy" et "lighthouse". Si nous considérons une autre image $i_3$ associée aux concepts : "Spain", "semaphore" et "car", alors le nouveau centroïd obtenu par la méthode "RootFusion" est l'ensemble de chemins : $\{(a_{0,0}, a_{1,1}, a_{2,1}), (a_{0,0}, a_{1,2}, a_{2,3})\}$. Ce critère est intéressant, car le nouveau représentant de chaque cluster est facile à calculer et ne nécessite que peu de temps de calcul, il est donc adapté pour une utilisation en ligne.

### 7.1.2 Techniques de découpage

Une fois que la hiérarchie de clusters ou dendrogramme a été créé par la AHC, nous devons déterminer à quel niveau couper horizontalement le dendrogramme afin d'obtenir des clusters adaptés à notre problème de diversité. Le problème de la coupure du dendrogramme est un problème qui n'a pas de réponse unique. Dans la section 5.1.4.2 page 56 nous avons décrit deux techniques classiques de découpage : le découpage d'un nombre Fixe de $n$ clusters et un Adapt qui choisit un nombre de clusters égal au nombre de sous-thèmes de la vérité terrain. De plus, dans [39] les auteurs proposent une technique de découpage souvent utilisée dans la AHC qui consiste à calculer, à chaque itération de la AHC, la différence $\Delta(n, n+1)$ entre la valeur de la distance d'agrégation à l'étape $n$ et celle à l'étape $n+1$, et à couper le dendrogramme après l'étape $n$ qui a la plus grande différence (méthode que nous appelons Traditional).

### 7.1.3 Réordonnancement hiérarchique

Une fois les clusters non ordonnées obtenus, nous trions d'abord les clusters et puis nous réordonnons les images. Pour trier les clusters nous utilisons des techniques de priorité de clusters décrites dans la section 5.1.5 page 56.

Pour réordonner les images, nous utilisons un réordonnancement classique (Flat0) où les images sont réordonnées en alternant une image de chaque cluster. Afin d'utiliser les différents niveaux de granularité fournis par la AHC, nous proposons une nouvelle méthode de réordonnancement des images que nous appelons Hier0 et nous la comparons avec le réordonnancement Flat0.

Dans le cas du réordonnancement hiérarchique (Hier0), les images sont réordonnées en prenant dans l'ordre des clusters et en alternant les sous-clusters, une image de chaque cluster. Les sous-clusters sont obtenus en appliquant une deuxième fois la même technique de coupure de la hiérarchie. Par exemple, pour la technique de coupure Fixe(10), la hiérarchie est coupée une première fois afin d'obtenir 10 clusters, puis la hiérarchie est coupée une deuxième fois afin d'obtenir 20 clusters. La méthode Hier0 Fixe(10) n'est

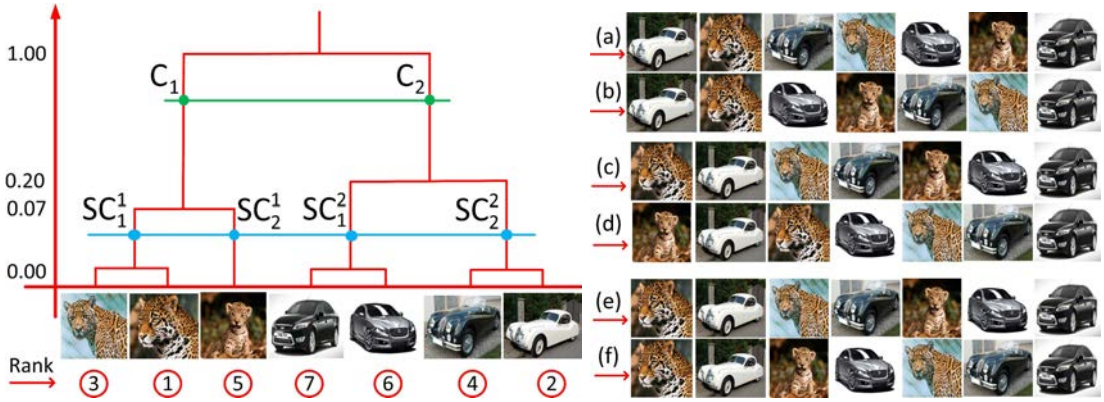


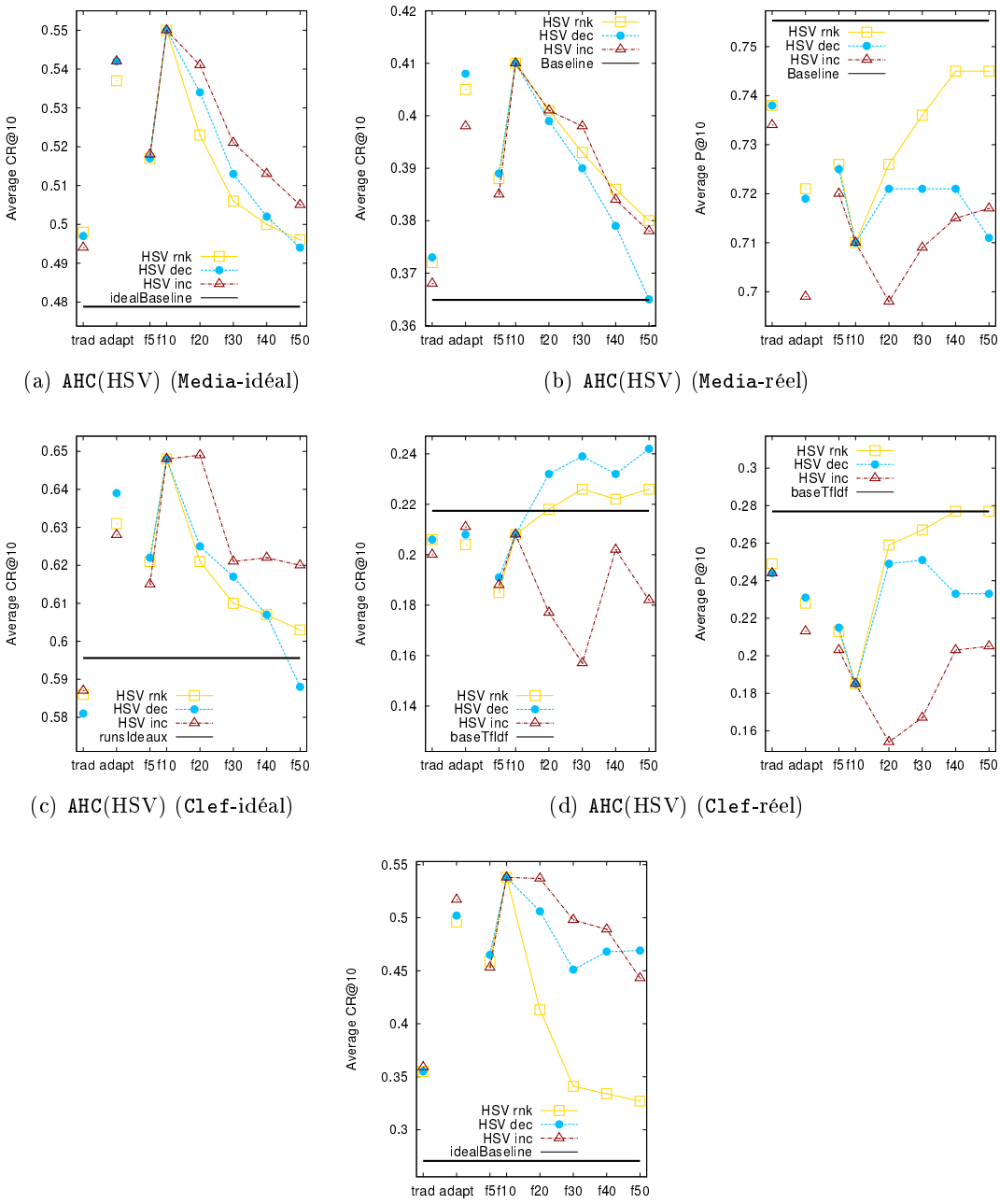

FIGURE 7.1 – Dendrogramme de la `AHC` et résultats de réordonnancement. Les images du cluster 1 sont liées au thème "jaguar animal" (et au sous-thème "bébé jaguar" et "jaguar adulte") et les images du cluster 2 sont liées au thème "jaguar voiture" (et au sous-thème "voiture actuelle" et "voiture ancienne"). Les résultats fournis par le réordonnancement `FlatO` avec les priorités des clusters (a) `Decreasing`, (c) `Increasing` ou (e) `Rank` sont moins diversifiés que les résultats fournis par le réordonnancement `HierO` avec les priorités (b) `Decreasing`, (d) `Increasing` ou (f) `Rank`

pas équivalente à `FlatO Fixe`(10) (10 clusters) ni à `FlatO Fixe`(20) (20 clusters), car dans le cas de `HierO Fixe`(10) les 10 clusters sont triés une première fois selon le critère de priorité des clusters choisis, puis lors de la deuxième coupure, les sous-clusters sont retriés mais seulement à l'intérieur des premiers clusters.

Pour produire le résultat final, les images sont triées dans chaque cluster selon leur rang, puis on ajoute au résultat final la première image de chaque cluster, puis au deuxième passage, on choisit la seconde image dans un sous-cluster différent que celui de la première image. Par exemple, en utilisant le réordonnancement hiérarchique, le résultat est une alternance d'images entre les sous-thèmes du thème "jaguar animal" et les sous-thèmes du thème "jaguar voiture" (voir résultats (b), (d) et (f) dans la figure). Si on compare les images réordonnées, nous remarquons que les quatre premières images des résultats (b), (d) et (f) appartiennent bien à quatre sous-thèmes différents, contrairement aux résultats (a), (c) et (e) où les quatre premières images contiennent seulement deux sous-thèmes. Il y a donc bien une plus grande diversité des sous-thèmes, avec le réordonnancement hiérarchique qu'avec celui plat.

## 7.2 Influence de l'ordre des clusters

Après avoir mentionné les différentes méthodes pour utiliser la `AHC` sur la diversité, nous allons étudier l'influence de ces méthodes. Dans cette section, nous étudions l'influence de l'ordre des clusters sur la diversité. Pour cela, nous comparons trois techniques de priorité : la première, une priorité classique (que nous appelons `Rank`) qui trie les clusters selon le rang des images (méthode abrégé en "rnk"). Et puis, deux techniques de priorité proposées (que nous appelons `Increasing` et `Decreasing`) qui trient les clusters selon leur taille où `Increasing` trie les cluster du plus petit nombre d'images au plus grand nombre d'images (méthode abrégé en "inc") et `Decreasing` dans le cas contraire (méthode abrégé en "dec").




(a) AHC(HSV) (Media-idéal) (b) AHC(HSV) (Media-réel)

(c) AHC(HSV) (Clef-idéal) (d) AHC(HSV) (Clef-réel)

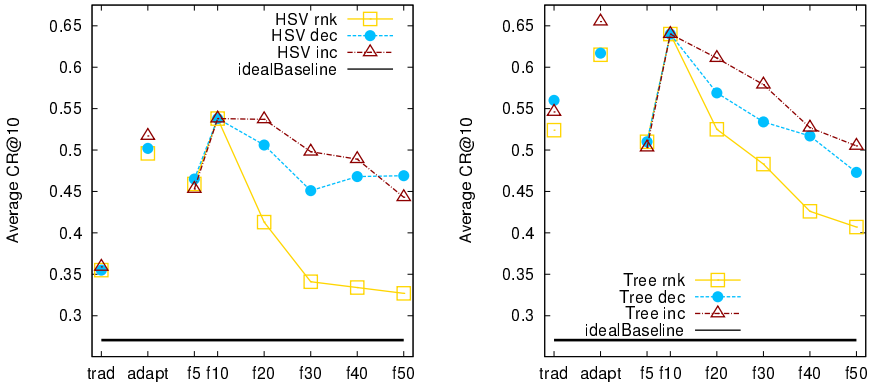
(e) AHC(HSV) (Xilo-idéal)

FIGURE 7.2 – Comparaison des techniques de priorité sur plusieurs benchmarks. Les scores CR@10 et P@10 moyens représentent la AHC avec le descripteur HSV en utilisant différentes techniques de découpage (en abscisse) dans un cas idéal et réel sur les benchmarks (a) (b) Media (sous-ensemble testset), (c) (d) Clef et (e) Xilo

### 7.2.1 Comparaison Increasing, Decreasing et Rank

Dans cette partie nous comparons les trois techniques de priorité sur plusieurs benchmarks et sur plusieurs descripteurs. Dans ces expériences, nous utilisons différentes techniques de découpage et un réordonnancement classique.

**Comparaison des techniques de priorité sur plusieurs benchmarks** Dans la figure 7.2 nous comparons les trois techniques de priorité sur plusieurs benchmarks. Les




(a) `AHC`(HSV) (`Xilo`-idéal) (b) `AHC`(Tree) (`Xilo`-idéal)

FIGURE 7.3 – Comparaison des techniques de priorité avec les descripteurs HSV et Tree. Les scores CR@10 et P@10 moyens représentent la `AHC` avec les descripteurs (a) HSV et (b) Tree en utilisant différentes techniques de découpage (en abscisse) dans un cas idéal sur le benchmark `Xilo`

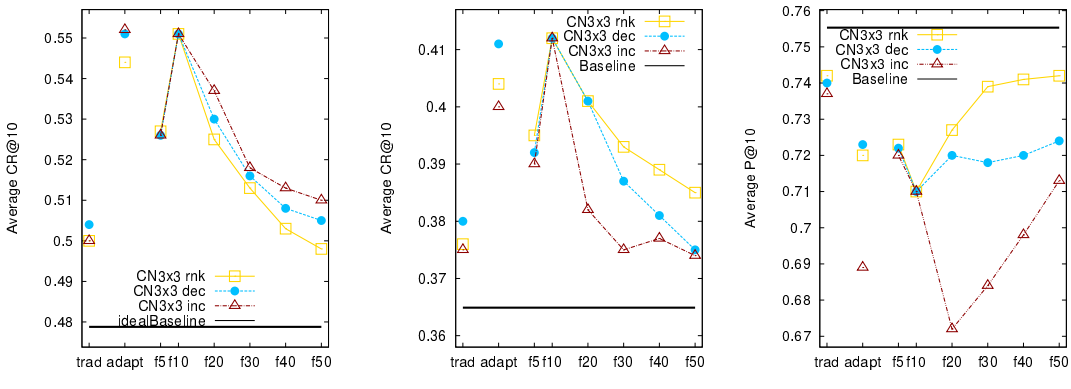
scores CR@10 et P@10 moyens représentent la `AHC` avec le descripteur HSV en utilisant différentes techniques de découpage dans un cas idéal et réel.

Si nous analysons les scores CR@10 dans un **cas idéal** (voir les figures 7.2(a), 7.2(c) et 7.2(e)) on peut voir que la technique de priorité `Increasing` (en rouge) obtient généralement des meilleurs résultats, mais dans un **cas réel** (voir les figures 7.2(b) et 7.2(d)) cette technique obtient parfois de mauvais résultats en CR@10 et P@10. En effet, dans un cas idéal, le fait de donner plus d'importance aux petits clusters et les placer en premier dans la liste de résultats semble être une bonne stratégie pour augmenter la diversité parce que les petits clusters peuvent représenter des images originales. Par contre, dans un cas réel, les petits clusters peuvent être des images originales, mais aussi des images non-pertinentes. Donc, si on donne plus d'importance aux petits clusters dans un cas réel, on risque d'être pénalisé en diversité et en pertinence par les images non-pertinentes.

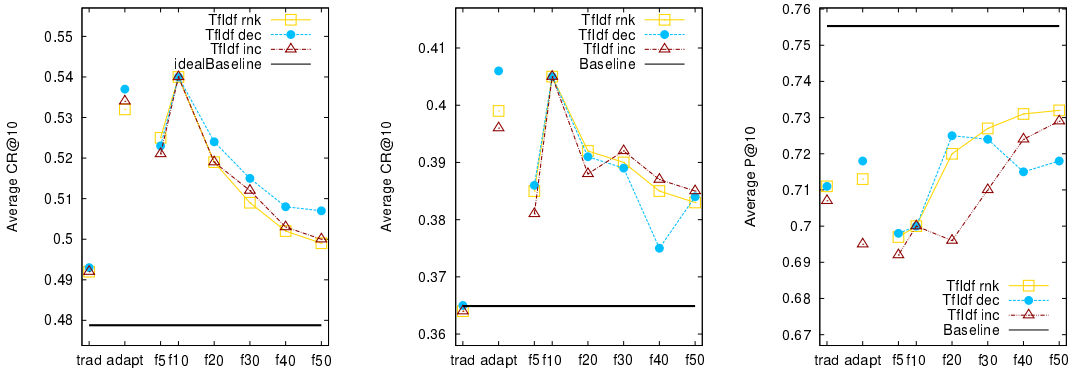
On voit aussi dans ces figures que les courbes de la priorité `Decreasing` (en bleu) et `Rank` (en jaune) sont assez proches (obtiennent des résultats similaires) dans la plupart des cas. Il semblerait que lorsqu'on trie les images par le rang de la baseline, on suit un comportement similaire à la priorité `Decreasing` car généralement cette baseline est ordonnée d'abord par des images que l'utilisateur s'attend à retrouver (grands clusters), puis par d'autres originales (petits clusters) et finalement par celles non-pertinentes.

**Comparaison des techniques de priorité sur plusieurs descripteurs** Dans les figures 7.3 et 7.4 nous comparons les trois techniques de priorité sur plusieurs descripteurs. Les scores CR@10 et P@10 moyens représentent la `AHC` avec le descripteur Tree sur le benchmark `Xilo` (figure 7.3) et les descripteurs CN3x3, TfIdf, GPS sur le benchmark `Media` (figure 7.4) en utilisant différentes techniques de découpage dans le cas idéal et réel.

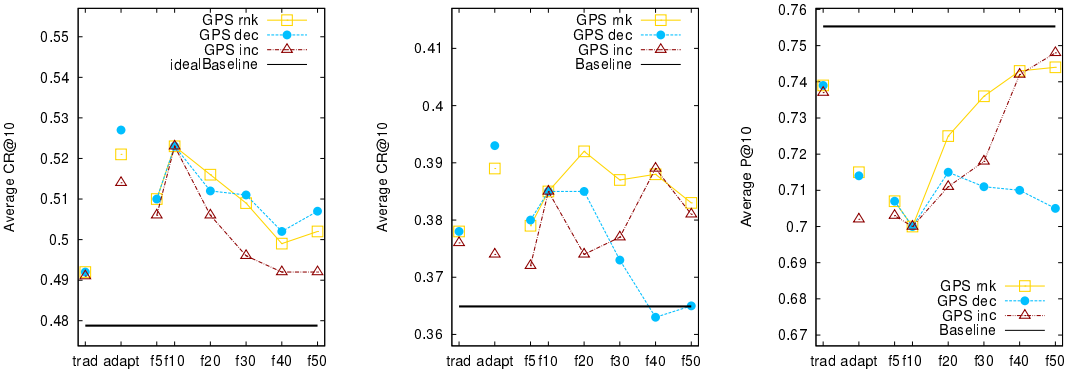
Dans ces figures on voit que dans le **cas idéal** la priorité `Increasing` obtient aussi de bons résultats en diversité avec le descripteur de couleur CN3x3 (voir la figure 7.4(a)) qui est un descripteur de la même famille que HSV, mais aussi avec le descripteur Tree (voir la figure 7.3(b)) totalement différent de ceux de couleurs, que "Wu-Palmer" est une mesure de similarité très différente de la distance "Euclidean", et que "RootFusion" est un critère d'agrégation très différente de "Average". Par contre, `Increasing` obtient de moins bons résultats avec le descripteur TfIdf et le GPS.




(a) `AHC`(CN3x3) (`Media`-idéal) (b) `AHC`(CN3x3) (`Media`-réel)




(c) `AHC`(TfIdf) (`Media`-idéal) (d) `AHC`(TfIdf) (`Media`-réel)




(e) `AHC`(GPS) (`Media`-idéal) (f) `AHC`(GPS) (`Media`-réel)

FIGURE 7.4 – Comparaison des techniques de priorité sur plusieurs descripteurs. Les scores CR@10 et P@10 moyens représentent la `AHC` avec les descripteurs (a) (b) CN3x3, (c) (d) TfIdf et (e) (f) GPS en utilisant différentes techniques de découpage (en abscisse) dans un cas idéal et réel sur le benchmark `Media` (sous-ensemble `testset`)

En sachant que le choix du type de descripteur peut influencer sur la qualité de la hiérarchie de clusters ; par exemple, une hiérarchie générée par la couleur ne va pas être la même qu'avec une hiérarchie générée par texte, il semblerait que le fait d'utiliser une similarité par la couleur ou par une arborescence de concepts aident à la `AHC` à obtenir une hiérarchie de clusters de bonne qualité et qui permet à la priorité `Increasing` à obtenir de bons résultats en diversité dans un cas idéal.

Dans le **cas réel** on obtient les mêmes résultats que le comportement précédent où

la priorité `Increasing` obtient de mauvais résultats à cause des images non-pertinentes.

**Discussion** De manière générale, on voit que dans un **cas idéal** l'intérêt de donner plus d'importance aux petits clusters (priorité `Increasing`) permet d'augmenter la diversité (car ces petits clusters peuvent être des images originales), mais en utilisant seulement les descripteurs de couleur (CN3x3 ou HSV) ou une arborescence de concepts, car ils semblent générer une meilleure qualité de hiérarchie de clusters qu'avec le texte ou les coordonnées GPS.

Par contre, dans un **cas réel** cette priorité est moins bonne qu'avec `Decreasing` ou `Rank` pour augmenter la diversité, car dans un cas réel `Increasing` est fortement pénalisé par des images non-pertinentes qui peuvent se trouver dans les petits clusters générés par la `AHC`.

De surcroît, on voit que les descripteurs de couleur et l'arborescence de concepts donnent de bons résultats en diversité, de moins bons en pertinence, tandis que le texte et le GPS donnent de moins bons résultats en diversité.

Donc, si on se trouve dans un cas idéal et que nous disposons des descripteurs donnant de bons résultats en diversité, il vaut mieux ordonner les clusters par une priorité `Increasing` pour augmenter la diversité. Par contre, si on utilise des descripteurs moins bons en diversité ou on est dans un cas réel, il semble mieux ordonner les clusters par une priorité classique.

### 7.2.2 Point de convergence des résultats

Dans les figures précédentes (voir les figures 7.2, 7.3 et 7.4) on voit que les trois techniques de priorité obtiennent les mêmes scores (**convergence exacte**) avec un découpage `Fixe` à 10 clusters (abrégé en "f10") pour CR@10 et P@10. Voir aussi les figures A.1 et A.2 en annexe A.2 page 178 pour les courbes à CR@20.

La raison de cette convergence est que, même si le tri des clusters est différent (ordre croissant, décroissant ou par le rang), le nombre de clusters étant identique au nombre d'images évaluées, on prend exactement une image par cluster. Or pour chaque cluster, on choisit en premier l'image qui a le plus fort rang dans le cluster quelle que soit la méthode de tri des clusters. Donc dans les $m$ premières images, on prend les mêmes $m$ images (même si leur ordre est différent). Les scores sont donc identiques avec les 3 méthodes de tri des clusters. Donc, on peut conclure que pour CR@10, CR@20, CR@30 (pour CR@m en général), les 3 techniques de priorité vont converger toujours de manière exacte avec un découpage `Fixe` à $m$ clusters. En conséquence, dans ce point de convergence, on peut utiliser n'importe quelle technique de priorité, car ils vont obtenir toujours les mêmes résultats.

### 7.2.3 Discussion

De manière générale, nous avons vu que les descripteurs de couleur et l'arborescence de concepts donnent de bons résultats en diversité tandis que le texte et le GPS donnent de moins bons résultats en diversité.

Dans la comparaison des techniques de priorité nous avons vu que si on se trouve dans un **cas idéal** et que nous disposons des descripteurs donnant de bons résultats en diversité, il vaut mieux ordonner les clusters par une priorité `Increasing` pour augmenter la diversité. Par contre, si on utilise des descripteurs moins bons en diversité ou on est dans un cas réel, il semble mieux ordonner les clusters par une priorité classique.

Nous avons vu aussi que pour CR@m, les 3 techniques de priorité convergent toujours de manière exacte avec un découpage `Fixe` à $m$ clusters ce qui rend à ce point de convergence indépendant de la technique de priorité.

## 7.3 Influence du nombre de clusters

Maintenant, nous étudions le choix du nombre de clusters qui peut nous donner de bons résultats en diversité. Pour cela nous comparons trois techniques de découpage : La première, une technique de découpage (que nous appelons `Traditional`) qui coupe la hiérarchie de clusters où on a le plus grand écart entre la différence $\Delta(n, n+1)$ de la distance d'agrégation à l'étape $n$ et celle à l'étape $n+1$ à chaque itération de la `AHC` (abrégé en "trad"). La deuxième, une technique de découpage (que nous appelons `Fixe`) qui coupe la hiérarchie de clusters de manière à obtenir un nombre fixe de $n$ clusters (abrégé en "fn"). La troisième, une technique de découpage (que nous appelons `Adapt`) qui semblerait faire un choix idéal, car il prend un nombre de clusters égal au nombre de sous-thèmes de la vérité terrain.

Il faut rappeler que le découpage `Adapt` n'est pas applicable dans la vie réelle car on ne connaît pas le nombre de sous-thèmes pour une requête donnée. Donc, ce découpage nous sert seulement comme point de comparaison par rapport aux autres techniques de découpage.

### 7.3.1 Comparaison `Adapt`, `Traditional` et `Fixe`

Nous utilisons les mêmes figures de la section précédente pour comparer les trois techniques de découpage sur plusieurs benchmarks (voir figure 7.2) et sur plusieurs descripteurs (voir figures 7.3 et 7.4).

De manière générale, on voit que les découpages `Adapt` et `Fixe` donnent de meilleurs résultats en diversité que le découpage `Traditional`. Par contre, on voit aussi que le découpage `Traditional` donne pratiquement de meilleurs résultats en pertinence. Il semblerait que le découpage `Traditional` génère moins des clusters que les autres techniques de découpage. Par exemple, si on utilise 1 cluster la génération du résultat final serait dépendant seulement de l'ordre des images dans chaque cluster (ordonnancement des images par son rang) ce qui génère un résultat final avec une pertinence similaire à celle de la baseline. Donc, le fait de générer moins de clusters pourrait expliquer les bons résultats en pertinence obtenus par le découpage `Traditional`.

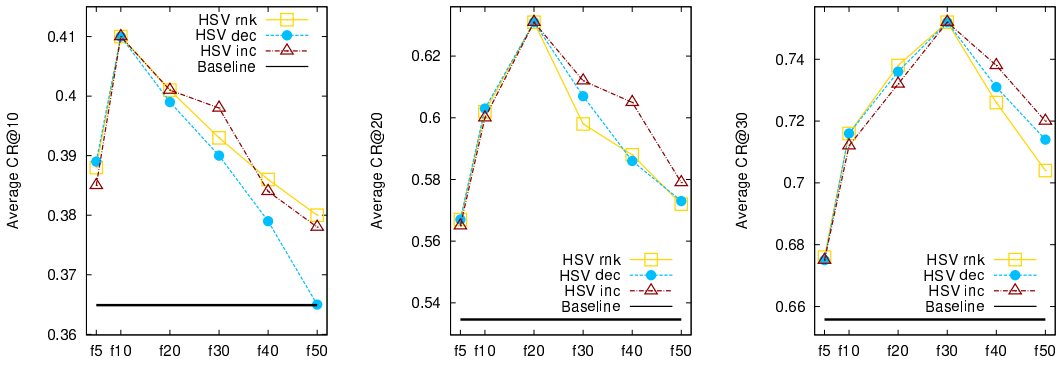
Le plus étonnant c'est de voir que le découpage `Fixe` donne en général des meilleurs résultats en diversité que le fait de choisir le vrai nombre des sous-thèmes de la vérité terrain (découpage `Adapt`) car le découpage `Adapt` semble faire un choix idéal. Peut-être la `AHC` ne génère pas exactement les mêmes clusters que les sous-thèmes de la vérité terrain ce qui peut fausser le choix idéal du nombre de clusters. Donc, on peut dire que de même si on connaît le vrai nombre de sous-thèmes dans la vie réel (c'est qui est déjà improbable), c'est toujours mieux d'utiliser un découpage `Fixe` de clusters pour augmenter la diversité.

Par contre, un inconvénient du découpage `Fixe` c'est de choisir le bon nombre de clusters qui va nous donner la valeur maximale en diversité, car si on ne choisit pas le bon nombre de clusters son score risque d'être inférieur aux autres techniques de découpage. Donc, d'ici l'importance d'étudier plus en détail cette technique de découpage.

### 7.3.2 Valeur maximale pour CR@10

Si on analyse le découpage `Fixe` dans la plupart des figures de la section précédente (voir les figures 7.2, 7.3 et 7.4) on voit un comportement intéressant pour CR@10 où les scores augmentent de "f5" jusqu'à "f10", puis ils diminuent de "f10" jusqu'à "f50" avec une valeur maximale obtenue avec le découpage "f10".

De manière générale, il semblerait que le fait de choisir un nombre $n$ de cluster égal au nombre $m$ d'images à évaluer est un bon compromis entre le deux cas précédents




(a) `AHC`(HSV) (`Media`-réel) (b) `AHC`(HSV) (`Media`-réel) (c) `AHC`(HSV) (`Media`-réel)

FIGURE 7.5 – Comparaison de la valeur maximale pour CR@m sur un benchmark. Les scores CR@m moyens représentent la `AHC` avec le descripteur HSV en utilisant différents découpages `Fixe` "fn" (en abscisse) et différentes techniques de priorité dans un cas réel sur le benchmark `Media` (sous-ensemble `testset`)

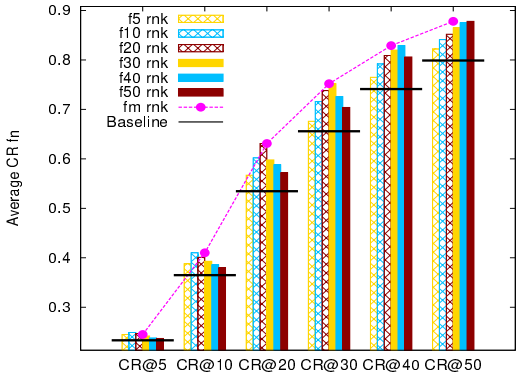



FIGURE 7.6 – Comparaison des scores CR@m moyens de la `AHC` avec le descripteur HSV en utilisant différents découpages `Fixe` "fn". Dans ces figures nous utilisons aussi une priorité classique `Rank` dans un cas réel sur les benchmarks `Media` (sous-ensemble `testset`)

pour obtenir la valeur maximale pour CR@10. De plus, le point de cette valeur maximale devient aussi la convergence exacte pour les trois techniques de priorité (voir la section 7.2.2), ce qui permet à ce choix d'être indépendant de la technique de priorité.

Nous avons réalisé de très nombreuses expériences sur les différents benchmarks avec les différents descripteurs et en faisant varier tous les paramètres et nous avons trouvé une exception à ce comportement dans le cas réel du benchmark `Clef` (voir figure 7.2(d)) où nous avons une baseline faible en pertinence (P@10 = 0.28). De plus, il semblerait aussi que la `AHC` est sensible à ce type de pertinence initiale, car il obtient des scores aussi faibles en diversité (des scores qui sont autour de la baseline).

On voit aussi que ce comportement est plus évident quand on utilise des bons descripteurs en diversité (comme la couleur ou l'arborescence de concepts) et il est moins évident avec des descripteurs moins bons en diversité (comme le texte ou le GPS). Donc, il semblerait que ce comportement soit sensible au type de descripteur.

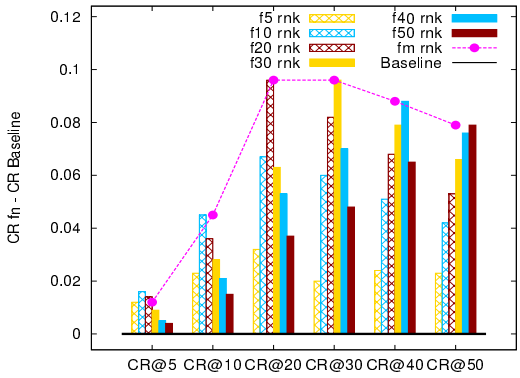



FIGURE 7.7 – Comparaison de la différence entre les scores CR@m moyens de la baseline et les scores CR@m moyens de la `AHC` avec le descripteur HSV en utilisant différents découpages `Fixe` "fn" et une priorité classique `Rank` dans un cas réel sur les benchmarks `Media` (sous-ensemble `testset`)

### 7.3.3 Valeur maximale pour CR@m

Nous avons vu que la valeur maximale pour CR@10 est obtenue avec le découpage `Fixe` "f10". Est-ce que ce comportement va être similaire pour CR@20 ou pour CR@30 ?

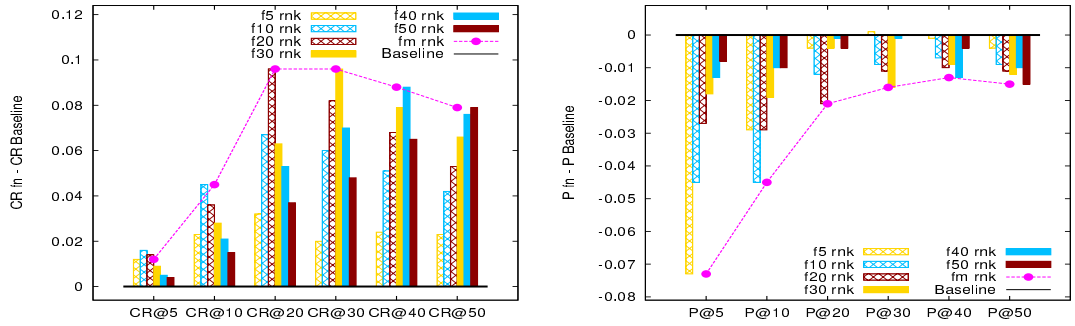
Pour commencer cette étude, dans la figure 7.5 nous comparons la valeur maximale pour CR@10, CR@20 et CR@30. Les scores CR@m moyens représentent la `AHC` avec le descripteur HSV en utilisant différents découpages `Fixe` "fn" dans un cas réel sur le benchmark `Media`. Dans cette figure, on voit que le bon choix pour obtenir la valeur maximale de CR@m est de toujours choisir un nombre de clusters égal au nombre d'images à évaluer.

Pour résumer le comportement la valeur maximale pour CR@m dans une seule figure, nous montrons la figure 7.6 qui compare la valeur maximale pour plusieurs valeurs de CR@m (en abscisse) en utilisant les mêmes paramètres de la figure précédente. Dans cette figure nous ajoutons un découpage `Fixe` appelé "fm" où le choix du nombre de clusters varie selon le nombre $m$ d'images à évaluer. Par exemple, si on évalue les 20 premières images (CR@20), le découpage "fm" serait en réalité "f20". Ce découpage nous servira comme guide pour valider le comportement initial obtenu pour CR@m. Dans cette figure on voit que la valeur maximale pour CR@m est obtenue avec le découpage `Fixe` "fm" ce qui confirme le résultat du comportement initial. On voit aussi une grande différence entre l'ordre de grandeur des scores de CR@5 et CR@50 ce qui ne permet pas trop visualiser la différence entre les scores des techniques de découpages `Fixe`.

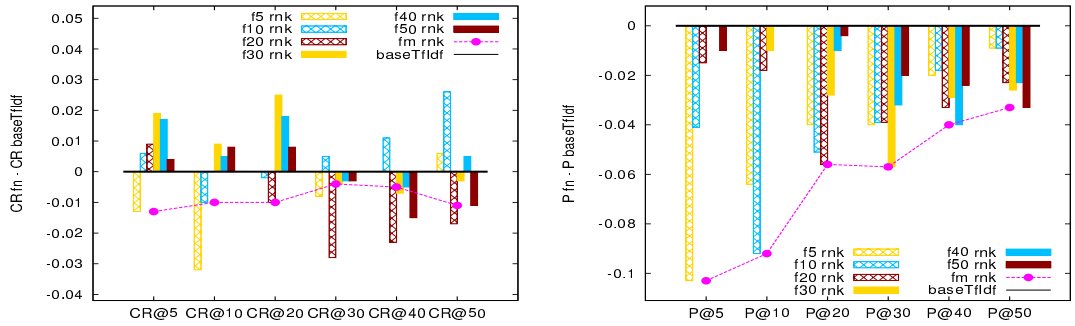
Pour montrer autrement la figure précédente, dans la figure 7.7 nous comparons la différence entre les scores CR@m moyens de la baseline et les scores CR@m moyens de la `AHC` en utilisant toujours les mêmes paramètres. Dans cette figure on peut mieux voir la différence entre les scores de différentes techniques de découpages `Fixe`. Donc, par la suite nous prendrons ce type de figure pour faire l'étude de la valeur maximale sur plusieurs benchmarks et sur plusieurs descripteurs.

Nous utilisons comme paramètres une priorité classique (`Rank`) et un réordonnancement classique.

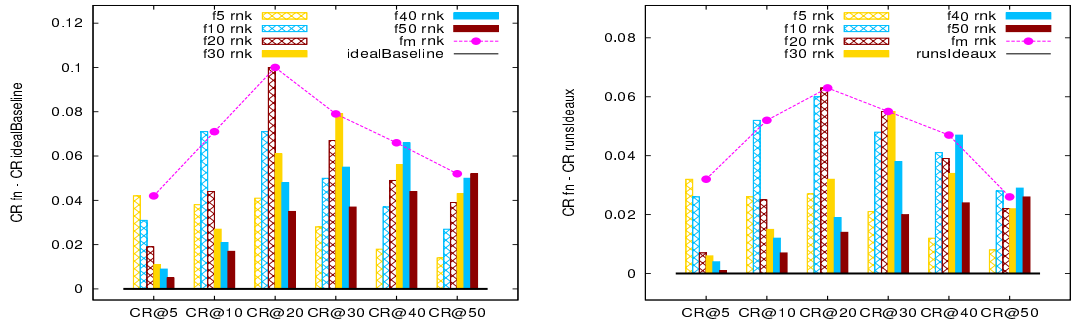
**Valeur maximale pour CR@m sur plusieurs benchmarks** Dans la figure 7.8 nous comparons la différence entre les scores CR@m et P@m moyens de la baseline et les scores CR@m et P@m moyens de la `AHC` avec le descripteur HSV en utilisant différents découpages `Fixe` "fn", dans un cas idéal et réel. Par exemple, dans la figure 7.8(c) on




(a) `AHC`(HSV) (`Media`-réel)

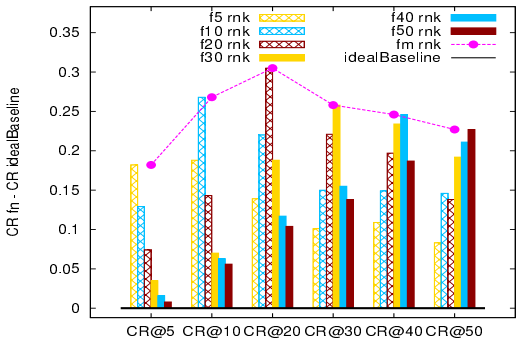



(b) `AHC`(HSV) (`Clef`-réel)



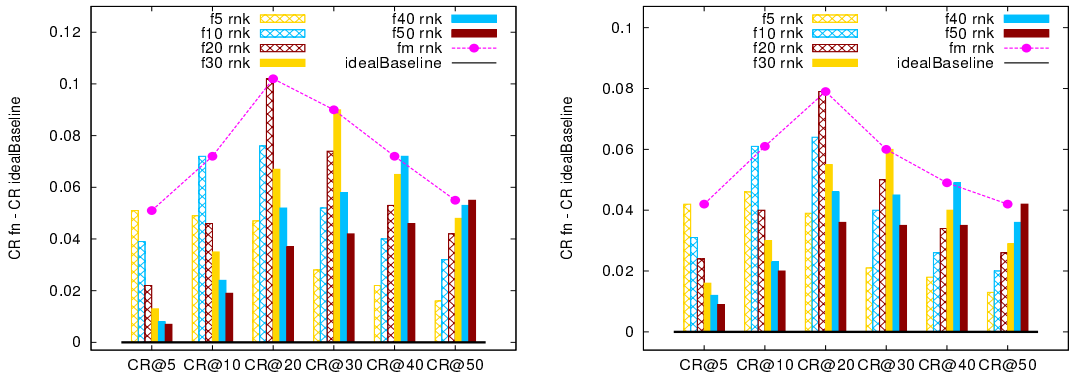

(c) `AHC`(HSV) (`Media`-idéal)

(d) `AHC`(HSV) (`Clef`-idéal)

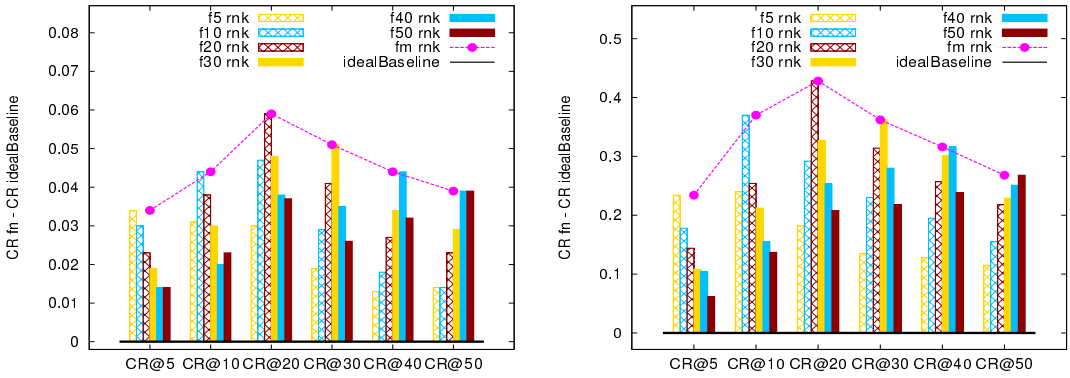



(e) `AHC`(HSV) (`Xilo`-idéal)

FIGURE 7.8 – Comparaison de la différence entre les scores CR@m et P@m moyens de la baseline et les scores CR@m et P@m moyens de la `AHC` avec HSV en utilisant différents découpages `Fixe` "fn" et une priorité `Rank` dans un cas réel et idéal sur les benchmarks (a) (c) `Media` (sous-ensemble `testset`), (b) (d) `Clef` et (e) `Xilo`




(a) `AHC`(CN3x3) (`Media`-idéal) (b) `AHC`(TfIdf) (`Media`-idéal)




(c) `AHC`(GPS) (`Media`-idéal) (d) `AHC`(Tree) (`Xilo`-idéal)

FIGURE 7.9 – Comparaison de la différence entre les scores CR@m moyens de la baseline et les scores CR@m moyens de la `AHC` avec les descripteurs (a) CN3x3, (b) TfIdf et (c) GPS sur le benchmark `Media` (sous-ensemble `testset`) et le descripteur (d) Tree sur le benchmark `Xilo` en utilisant différents découpages `Fixe` "fn" et une priorité classique `Rank` dans un cas idéal

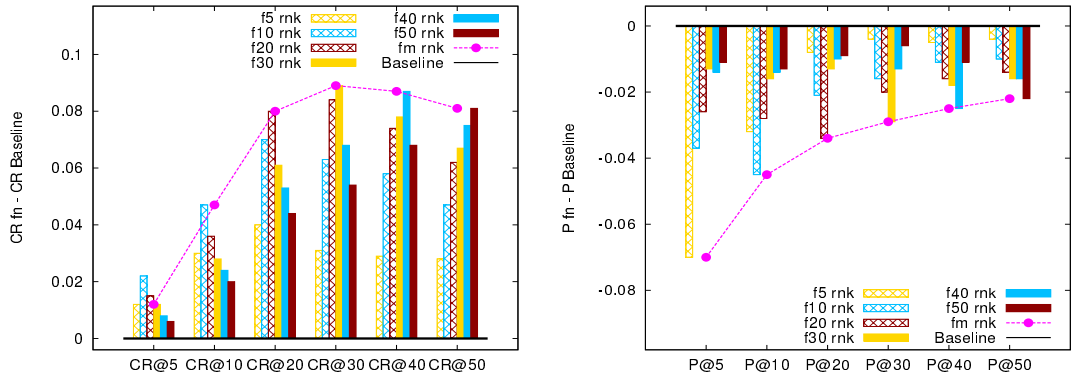
voit que pour CR@5 la plus grande différence entre la baseline et chaque découpage `Fixe` "fn" est obtenue avec le découpage "f5", donc la valeur maximale est obtenue avec ce découpage.

Dans la plupart des figures (voir figures 7.8(c), 7.8(a), 7.8(e) et 7.8(d)), on voit que la valeur maximale pour CR@m est obtenue avec le découpage `Fixe` "fm" (en rose) ce qui confirme le résultat du comportement initial.

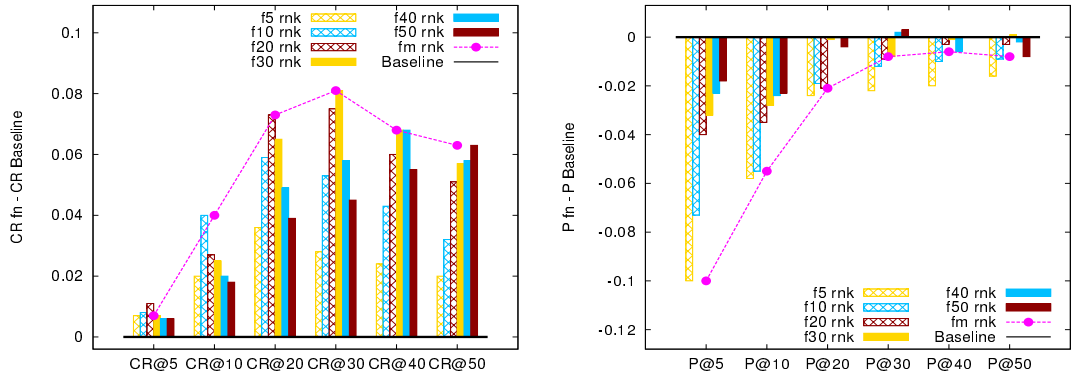
L'exception à ce comportement est toujours dans le cas réel du benchmark `Clef` (voir figure 7.8(b)) où on voit que le comportement de la valeur maximale pour CR@m est totalement instable. En sachant que la baseline "baseTfIdf" a une pertinence très faible, il semblerait que ce comportement est sensible à la pertinence initiale.

**Valeur maximale pour CR@m sur plusieurs descripteurs** Les figures 7.9 et 7.10 comparent la différence entre les scores CR@m et P@m moyens de la baseline et les scores CR@m et P@m moyens de la `AHC` avec les descripteurs CN3x3, TfIdf et GPS sur le benchmark `Media` et le descripteur Tree sur le benchmark `Xilo` en utilisant différents découpages `Fixe` "fn", dans un cas idéal (voir figure 7.9) et réel (voir figure 7.10).

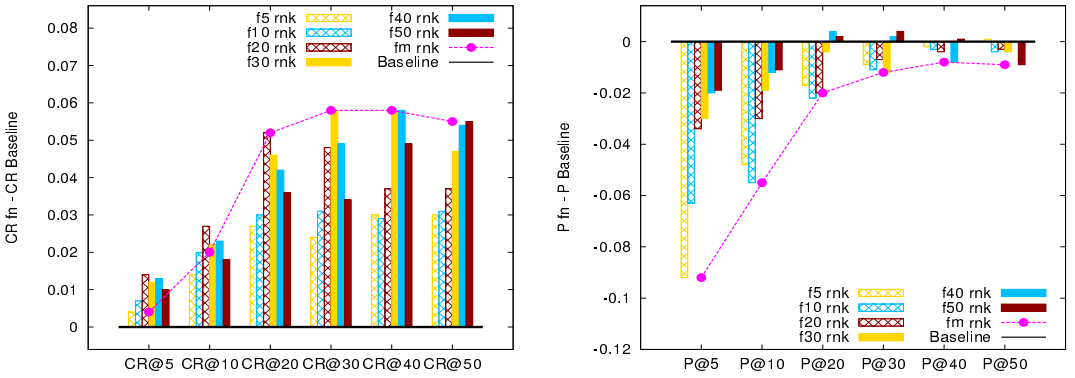
Dans ces figures, on voit que dans la plupart des cas, on obtient aussi le même comportement de la valeur maximale pour CR@m.




(a) `AHC`(CN3x3) (`Media`-réel)




(b) `AHC`(TfIdf) (`Media`-réel)




(c) `AHC`(GPS) (`Media`-réel)

FIGURE 7.10 – Comparaison de la différence entre les scores CR@m et P@m moyens de la baseline et les scores CR@m et P@m moyens de la `AHC` avec les descripteurs (a) CN3x3, (b) TfIdf et (c) GPS sur le benchmark `Media` (sous-ensemble `testset`) en utilisant différents découpages `Fixe` "fn" et une priorité classique `Rank` dans un cas réel

**Discussion** De manière générale, on voit que dans la plupart des cas, la valeur maximale pour CR@m est obtenue avec le découpage `Fixe` "fm" qui utilise un nombre de clusters égal au nombre $m$ d'images à évaluer. Par contre, ce comportement est fortement sensible quand la pertinente initiale est trop faible.

Donc, si nous disposons d'une baseline avec un bon score en pertinence (spécialement si on est dans un cas idéal), il vaut mieux choisir un nombre fixe de clusters égal au nombre d'images à évaluer pour obtenir la valeur maximale en diversité. Par contre, si

notre baseline est trop faible en pertinence, il semble mieux utiliser une autre méthode de diversité qui soit moins sensible à la pertinence initiale.

## 7.4 Réordonnancement Flat0 et Hier0

Finalement, nous étudions deux techniques de réordonnancement : La première, une technique classique que nous appelons Flat0 qui réordonne les images en alternant une image de chaque cluster. La deuxième, une technique proposée Hier0 qui diversifie les images de chaque cluster (en faisant une coupure plus profonde dans la hiérarchie de clusters) avant de faire le réordonnancement classique avec l'objectif d'utiliser les différents niveaux de granularité fournis par la AHC. Par exemple, si on utilise Flat0 avec un découpage "f10", on utilise 10 clusters pour réordonner les images. Par contre, si on utilise Hier0 avec le même découpage, on diversifie d'abord les 10 clusters, en faisant une coupure plus profonde pour en obtenir 20 puis on utilise ces 10 clusters diversifies pour réordonner les images.

L'intérêt de Hier0 c'est de s'assurer au deuxième passage de choisir les images les plus diversifiées possibles par rapport aux images déjà choisies au premier passage. Donc, on peut voir l'intérêt de cette méthode seulement à partir du deuxième passage. Dans ces expériences, nous utilisons différentes techniques de découpage et une priorité classique (Rank).

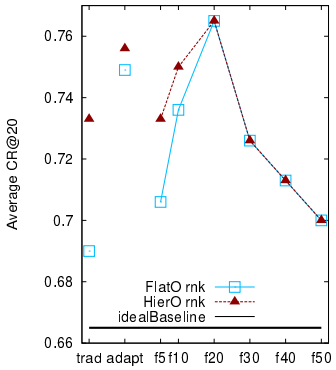
**Comparaison des techniques de réordonnancement sur plusieurs benchmarks** Dans les figures 7.11 et 7.12 nous comparons les techniques de réordonnancement sur plusieurs benchmarks. Les scores CR@20, P@20, CR@30 et P@30 moyens représentent la AHC avec le descripteur HSV en utilisant différentes techniques de découpage dans un cas idéal et réel.

Dans ces figures on voit que quand le nombre $n$ de clusters est supérieur ou égal au nombre $m$ d'images à évaluer, les deux techniques de réordonnancent ont les mêmes scores en CR@m et P@m. Ce comportement semble normal parce que, pour chaque cluster, on choisit au premier passage l'image qui a le plus fort rang dans le cluster quelle que soit la technique de réordonnancement. Donc, si on choisit $n$ clusters on prend les mêmes $n$ images au premier passage quel que soit le type de réordonnancement

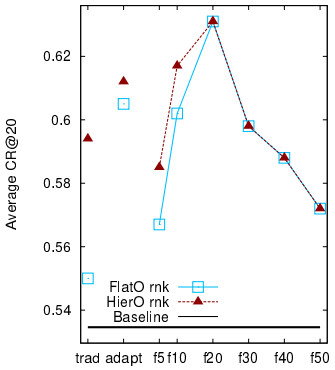
Par contre, quand le nombre $n$ de clusters est inférieur au nombre $m$ d'images à évaluer, on voit que Hier0 donne des meilleurs résultats qu'un réordonnancement classique. De plus, on obtient ce même résultat avec les découpages Traditional et Adapt ce qui pourrait déduire que ces types de découpage utilisent en moyenne un nombre de clusters inférieur au nombre d'images à évaluer. Dans ce cas, le fait de diversifier chaque cluster avant d'alterner les images (approche hiérarchique) semblerait être une bonne stratégie pour augmenter encore plus la diversité par rapport à une approche plate.

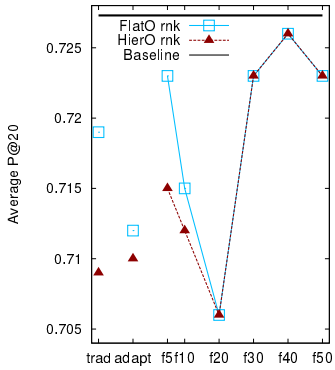
De même si notre réordonnancement proposée (qui est plus sophistiqué) semble être plus intéressant dans certains cas, en général, pour obtenir la valeur maximale pour CR@m il semble être suffisant de choisir un réordonnancement classique, car dans ce point les deux méthodes obtiennent les mêmes scores. Nous avons fait des expériences avec CR@10, CR@40 et CR@50 (voir figures A.3, A.4 et A.5 dans l'annexe A.2 page 180) et on voit aussi que Hier0 permet d'augmenter encore plus la diversité dans certains cas, mais pour obtenir la valeur maximale de diversité il semble suffisant de faire toujours un réordonnancement classique.

Comme nous utilisons un descripteur de couleur qui donne en général de bons résultats en diversité, ce n'est pas la peine de continuer cette étude sur des autres descripteurs qui sont moins performants.

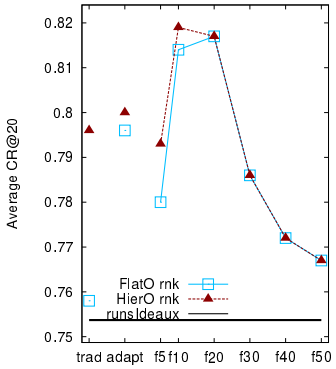



(a) `AHC`(HSV) (`Media`-idéal)

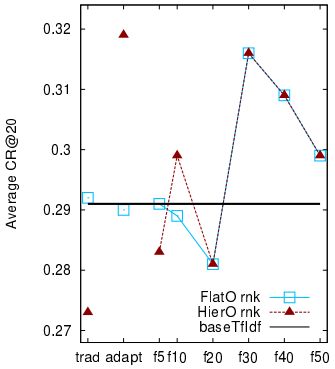


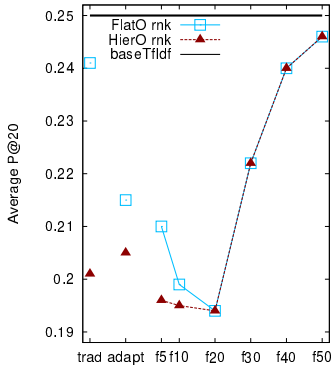




(b) `AHC`(HSV) (`Media`-réel)

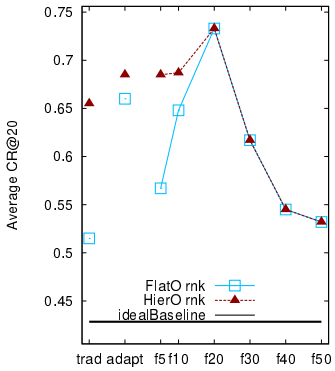



(c) `AHC`(HSV) (`Clef`-idéal)




(d) `AHC`(HSV) (`Clef`-réel)

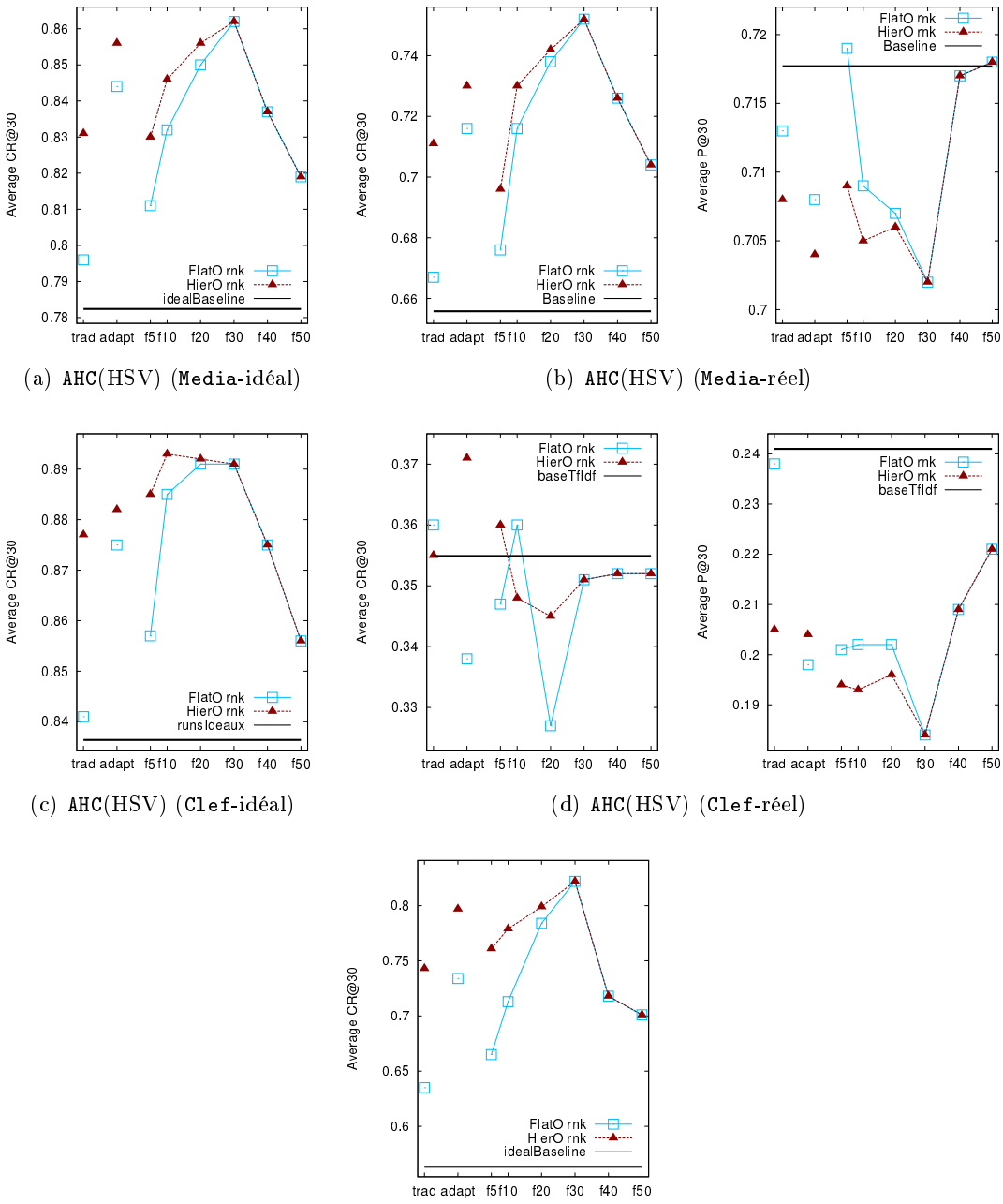



(e) `AHC`(HSV) (`Xilo`-idéal)

FIGURE 7.11 – Comparaison des techniques de réordonnancement sur plusieurs benchmarks pour CR@20. Les scores CR@20 et P@20 moyens représentent la `AHC` avec le descripteur HSV en utilisant différentes techniques de découpage (en abscisse) et une priorité classique `Rank` dans un cas idéal et réel sur les benchmarks (a) (b) `Media` (sous-ensemble `testset`), (c) (d) `Clef` et (e) `Xilo`

## 7.5 Bilan et discussion

La `AHC` est très intéressante pour faire de la diversité, car sa hiérarchie des clusters peut bien correspondre aux différents niveaux de granularité de la diversité ce qui pourrait permettre à l'utilisateur de changer facilement du niveau de granularité sans avoir besoin de relancer à nouveau la `AHC` (comme dans `K-Means` par exemple où il faut relancer la méthode pour un nombre de clusters différent).




(a) `AHC`(HSV) (`Media`-idéal) (b) `AHC`(HSV) (`Media`-réel)

(c) `AHC`(HSV) (`Clef`-idéal) (d) `AHC`(HSV) (`Clef`-réel)

(e) `AHC`(HSV) (`Xilo`-idéal)

FIGURE 7.12 – Comparaison des techniques de réordonnancement sur plusieurs benchmarks pour CR@30. Les scores CR@30 et P@30 moyens représentent la `AHC` avec le descripteur HSV en utilisant différentes techniques de découpage (en abscisse) et une priorité classique `Rank` dans un cas idéal et réel sur les benchmarks (a) (b) `Media` (sous-ensemble `testset`), (c) (d) `Clef` et (e) `Xilo`

Pour générer la `AHC` on a besoin de divers paramètres comme : le type de descripteur, les mesures de similarité et les critères d'agrégation et pour utiliser cette méthode sur la diversité on a besoin d'autres paramètres comme le nombre des clusters, les techniques de priorité et les types de réordonnancement. De plus, dans la littérature nous n'avons pas trouvée des travaux qui ont fait une étude approfondie sur ces paramètres. Dans ce chapitre nous avons étudié les paramètres de la `AHC` sur plusieurs benchmarks et sur

plusieurs descripteurs pour obtenir des comportements qui soient les plus généralisés possibles. De plus, nous proposons différentes approches pour certains types de paramètres classiques.

Un cas souhaitable pour augmenter la diversité avec la `AHC` est qu'il puisse générer une hiérarchie de clusters qui soit le plus similaire aux sous-thèmes de la vérité terrain. Pour cela, la qualité du descripteur est très importante, car elle peut influencer sur la qualité de cette hiérarchie.

Notre proposition d'utiliser l'arborescence de concepts comme descripteur dans la `AHC` semble être très intéressant, car elle donne bons résultats en diversité. De plus, ce descripteur a des comportements très similaires à la couleur qui obtient une meilleure diversité par rapport au texte ou le GPS, en sachant que l'arborescence de concepts est un descripteur très différent des descripteurs de couleurs, que la similarité "Euclidean" est une mesure de similarité très différente de "Wu-Palmer" et que "Average" est un critère d'agrégation très différent de "RootFusion". En sachant que la vérité terrain des benchmarks a été construite par des spécialistes qui semblent faire un regroupement des images plus par une approche de couleur ou sémantique (arborescence de concepts) que par celle textuelle ou par le GPS .

Pour utiliser la `AHC` sur la diversité nous avons d'autres paramètres comme le choix du nombre de clusters, son ordre ou la technique de réordonnancement.

La plupart des auteurs utilisent un **ordre des clusters** classique par le rang. Cependant, lorsque beaucoup d'images sont très pertinentes, le rang n'est pas toujours très significatif. Nous proposons une nouvelle approche qui donne plus d'importance à la taille des clusters qu'au rang des images, car nous pensons qu'un petit cluster par exemple peut représenter des images originales que l'utilisateur ne s'attend pas à retrouver pour sa requête. Par contre, nous avons vu que le fait de donner plus d'importance aux petits clusters (priorité `Increasing` proposée) semble être intéressante seulement dans un cas idéal et avec la condition d'utiliser des bons descripteurs (comme la couleur ou l'arborescence de concepts). Dans le cas contraire, il semble toujours mieux d'ordonner les clusters par une approche classique.

De même, le plus souvent c'est d'utiliser un **réordonnancement des images** classique en alternant les images de chaque cluster. Afin d'utiliser les différents niveaux de granularité fournis par la `AHC`, nous proposons un réordonnancement hiérarchique `HierO` qui diversifie chaque cluster avant de réordonner les images. Nous avons vu que cette proposition permet d'augmenter encore plus la diversité dans certains cas, mais, en général, il semble suffisant de faire un réordonnancement classique pour obtenir la valeur maximale en diversité.

Dans la littérature le bon choix du **nombre de clusters** est un problème sans réponse unique, des auteurs proposent différentes approches (selon le nombre total de documents, selon la densité des clusters, etc.). Nous avons comparé différents découpages avec celui qui semble faire le choix idéal, car ce dernier prend le vrai nombre de sous-thèmes dans la vérité terrain. Nous avons vu que de même, si on connaît le vrai nombre de sous-thèmes dans la vie réel (ce qui est déjà improbable), il vaut mieux choisir un nombre `Fixe` des clusters pour augmenter la diversité avec la `AHC`.

Après avoir étudié le découpage `Fixe` sur plusieurs valeurs de CR@m, on peut dire que le bon choix pour obtenir la valeur maximale en diversité est de choisir un nombre fixe de clusters égal au nombre d'images à évaluer. De plus, le point de cette valeur maximale devient aussi la convergence exacte pour les trois techniques de priorité et les deux techniques de réordonnancement, ce qui rend ce choix pas dépendant de ces paramètres.

Nous proposons donc une autre approche au problème du choix du nombre de clusters qui est de choisir un nombre fixe des clusters égal au nombre d'images à afficher en

sachant que dans la vie réelle un utilisateur est plus attentif aux 10 ou 20 premières images. Par contre, cette proposition est limitée seulement à la `AHC` qui est un clustering de type hiérarchique. Donc, par la suite nous souhaiterons continuer cette étude sur des autres méthodes de clustering classiques (comme `K-Means` par exemple) pour rendre cette proposition le plus généralisé possible.

Après avoir réalisé de nombreuses expériences sur les différents benchmarks avec les différents descripteurs et en faisant varier divers paramètres, nous avons trouvé une exception aux comportements obtenus dans le cas réel du benchmark `Clef` qui a la particularité d'avoir une baseline avec beaucoup d'images non-pertinentes. Dans cette exception, les comportements deviennent instables avec des scores aussi faibles en diversité (au niveau de la baseline) qui sont peut-être dus à cause des images non-pertinentes.

Nous avons vu que la complexité de l'algorithme de `AHC` est de type quadratique. Or, nous sommes dans une application de recherche d'images "en ligne", ce qui fait qu'on ne peut pas la lancer avec un grand nombre d'images si on veut obtenir un clustering en un temps raisonnable. Cependant, dans notre cas, nous savons que les résultats pertinents se trouvent dans les premiers résultats, il n'est donc pas nécessaire de lancer la `AHC` sur beaucoup de résultats, il suffit de la lancer sur les premiers résultats, ce qui nous permet d'utiliser la `AHC` en un temps raisonnable.

Même si la complexité de la `AHC` dans le pire de cas est quadratique, ils existent plusieurs stratégies pour réduire ce temps de calcul. Nous pouvons utiliser un critère centroïde qui semble être plus rapide, spécialement quand on utilise un grand nombre d'images. Nous pouvons aussi calculer à l'avance les distances entre les images (étape offline) ou générer la hiérarchie de clusters jusqu'à $n$ clusters. Nous pouvons diminuer aussi le nombre d'itérations de la `AHC` pour générer la hiérarchie de clusters en fusionnant les distances entre clusters qui sont inférieurs à un seuil déterminé.

# Chapitre 8

# Étude de la diversité par clustering plat

Pour résoudre le problème de la diversité nous avons utilisé une méthode de clustering hiérarchique, car nous pensons que la diversité est aussi un problème de type hiérarchique. Cependant, dans la littérature de recherche d'images il existe d'autres méthodes de clustering comme les méthodes de clustering plat très utilisées par les chercheurs pour résoudre le problème de la diversité.

Dans ce chapitre nous allons étudier deux méthodes classiques du clustering plat, celle de `K-Means` qui est une des les plus populaires et utilisées dans l'état de l'art et celle de `K-Medoids` qui est moins utilisée dans la littérature, mais qui a certains avantages par rapport à `K-Means`.

Les méthodes de clustering plat sont généralement des méthodes qui cherchent à optimiser le résultat final, contrairement au clustering hiérarchique qui utilise une optimisation locale. Par contre, la qualité du résultat final du clustering plat peut varier selon l'initialisation des clusters ce qui n'assure pas toujours une réponse unique et qui peut fausser le bon choix de certains paramètres pour augmenter la diversité. Pour éviter cela, nous lançons ces méthodes plusieurs fois pour éviter de tomber sur des scores abrupts ou particuliers.

Un avantage de `K-Means` est son algorithme qui est moins coûteux en complexité que `K-Medoids` ce qui permettrait d'utiliser plus facilement `K-Means` dans une application de recherche d'images "en ligne". Par contre, `K-Means` a besoin du calcul des centroïdes (le calcul d'un vecteur moyen pour représenter le cluster). Cette caractéristique peut limiter le choix du descripteur, car ce calcul n'est pas très adapté pour certains descripteurs comme les coordonnées GPS où son vecteur moyen ne correspondrait pas à la réalité. Pour cela, nous étudions `K-Medoids` qui prend des medoïdes (le choix d'un vecteur d'une image pour représenter le cluster) ce qui fait que cette méthode soit adaptée à n'importe quel type de descripteur et nous permet en conséquence d'étudier le clustering plat sur plusieurs types de descripteurs.

En sachant que l'arborescence de concepts (disponible dans le benchmark `Xilo`) est un descripteur qui a donné de bons résultats en diversité avec la `AHC`, est-ce que le fait d'utiliser ce descripteur avec un clustering plat nous donnera aussi de bons résultats en diversité ? Est-ce que ce descripteur est dépendant du clustering hiérarchique pour augmenter cette diversité ?

Le plan de notre chapitre est le suivant : Dans la partie 8.1 nous étudions l'influence de divers paramètres comme le nombre de clusters et leur ordre avec la méthode `K-Means`. Dans la partie 8.2 nous étudions l'influence des mêmes paramètres avec la méthode `K-Medoids`. Enfin, dans la partie 8.3 nous concluons et montrons les avantages et inconvénients des deux méthodes de clustering plat.

## 8.1 Étude de la diversité par K-Means

L'algorithme K-Means (décrite dans la section 3.1.1 page 32) est une des méthodes de clustering le plus populaire et utilisées dans l'état de l'art. Un avantage de cette méthode est sa complexité qui est moins coûteuse que la complexité de la AHC du clustering hiérarchique ce qui pourrait faciliter à cette méthode d'être utilisable dans une application de recherche d'images "en ligne".

Dans l'algorithme de K-Means nous utilisons une initialisation aléatoire de $n$ clusters qui consiste à prendre $n$ images de manière aléatoire. Comme la qualité de son résultat final peut varier selon cette initialisation, nous lançons cette méthode 10 fois pour chaque jeu de paramètres et puis nous prenons son score moyen comme résultat final, afin d'éviter ainsi de tomber sur des scores particuliers.

Par contre, l'algorithme K-Means a besoin du calcul des centroïdes qui n'est pas très adapté pour certains descripteurs. Pour cette raison dans ces expérimentations nous utilisons certains descripteurs comme les histogrammes de couleur qui sont très adaptés au calcul des centroïdes.

Dans cette section nous allons étudier l'influence de divers paramètres comme l'ordre et le nombre de clusters sur plusieurs benchmarks. Et puis, nous étudions l'influence de l'initialisation aléatoire de clusters avec K-Means afin de savoir si avec elle K-Means risque de tomber sur des scores abrupts.

### 8.1.1 Influence de l'ordre des clusters

De même que pour les résultats obtenus avec la AHC (voir section 7.2 page 98), dans cette section nous comparons les mêmes 3 techniques de priorité ou ordre des clusters avec K-Means. La première, une priorité classique (que nous appelons Rank) qui trie les clusters selon le rang des images. Et puis, les deux techniques de priorité proposées (que nous appelons Increasing et Decreasing) qui donnent plus d'importance à la taille des clusters.

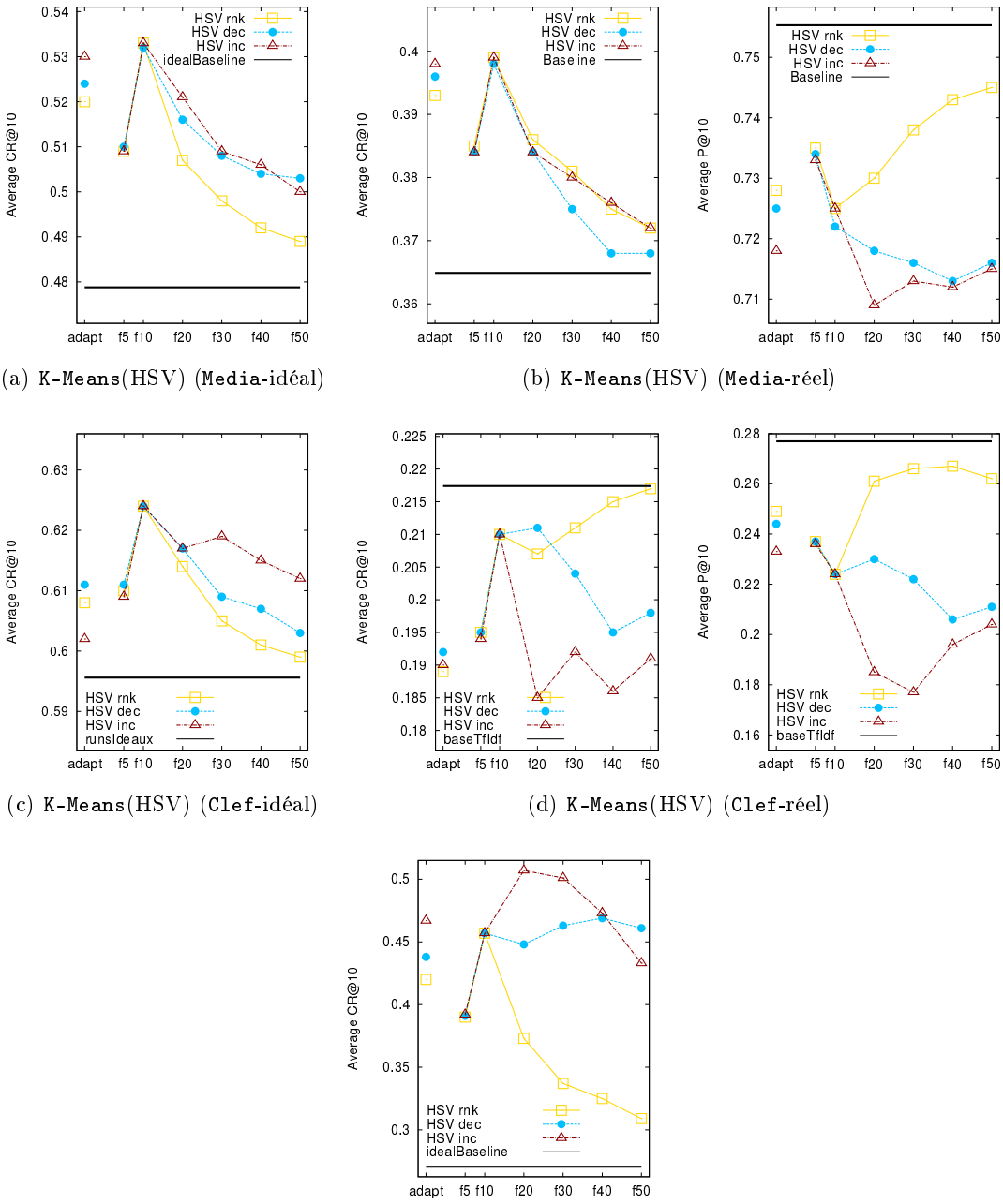
#### 8.1.1.1 Comparaison Increasing, Decreasing et Rank

Dans cette partie nous comparons les trois techniques de priorité sur plusieurs benchmarks et sur plusieurs descripteurs visuels avec l'utilisation de différentes techniques de découpage.

**Comparaison des techniques de priorité sur plusieurs benchmarks** Dans la figure 8.1 nous comparons les trois techniques de priorité sur plusieurs benchmarks. Les scores CR@10 et P@10 moyens représentent la méthode K-Means avec le descripteur HSV en utilisant différentes techniques de découpage dans un cas idéal et réel.

De même que pour la AHC, dans cette figure on voit aussi que la priorité Increasing obtient globalement de bons résultats en diversité, car cette priorité donne plus d'importance aux petits clusters qui peuvent représenter des images originales et qui peuvent en conséquence augmenter la diversité. Par contre, dans la figure 8.1(d) on voit l'exception à ce comportement où la priorité Increasing obtient de moins bons résultats que Rank et Decreasing peut-être à cause de la grande quantité d'images non-pertinentes qui existent dans cette baseline et qui peuvent se trouver dans les petits clusters.

**Comparaison des techniques de priorité sur un autre descripteur visuel** Dans la figure 8.2 nous comparons les trois techniques de priorité sur un autre descripteur visuel (CN3x3). Les scores CR@10 et P@10 moyens représentent la méthode K-Means




(a) K-Means(HSV) (Media-idéal)

(b) K-Means(HSV) (Media-réel)

(c) K-Means(HSV) (Clef-idéal)

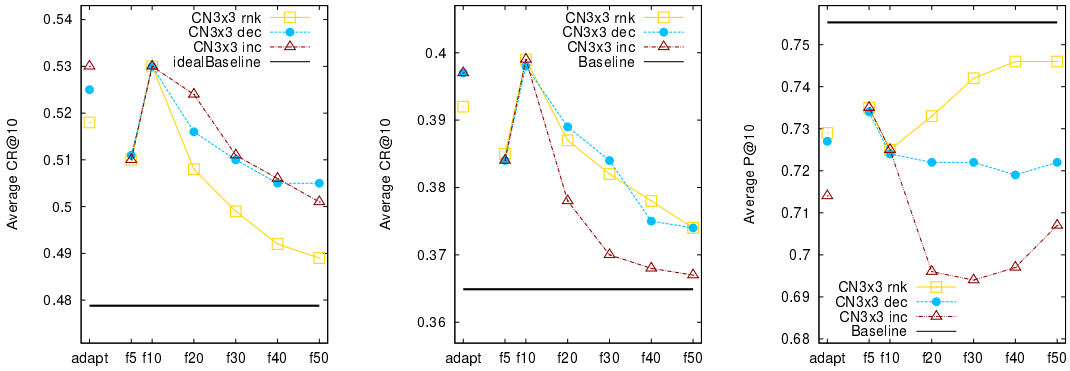
(d) K-Means(HSV) (Clef-réel)

(e) K-Means(HSV) (Xilo-idéal)

FIGURE 8.1 – Comparaison des techniques de priorité sur plusieurs benchmarks. Les scores CR@10 et P@10 moyens représentent la méthode K-Means avec le descripteur HSV en utilisant différentes techniques de découpage (en abscisse) dans un cas idéal et réel sur les benchmarks (a) (b) Media (sous-ensemble testset), (c) (d) Clef et (e) Xilo

avec le descripteur CN3x3 en utilisant différentes techniques de découpage dans le cas idéal et réel sur le benchmark Media.

Dans cette figure on voit aussi que Increasing obtient le même comportement dans un cas idéal avec un descripteur visuel différent (CN3x3) qui est un descripteur de la même famille que HSV et qui donne aussi de bons résultats en diversité. Par contre, dans un cas réel cette priorité donne de moins bons résultats que Rank et Decreasing peut-être à cause de la présence des images non-pertinentes.




(a) `K-Means`(CN3x3) (`Media`-idéal)

(b) `K-Means`(CN3x3) (`Media`-réel)

FIGURE 8.2 – Comparaison des techniques de priorité sur le descripteur CN3x3. Les scores CR@10 et P@10 moyens représentent la méthode `K-Means` avec le descripteur CN3x3 en utilisant différentes techniques de découpage (en abscisse) dans un cas idéal et réel sur le benchmark `Media` (sous-ensemble `testset`)

**Discussion** De manière générale on peut voir que les trois techniques de priorité obtiennent des comportements similaires avec les deux méthodes de clustering `K-Means` et la `AHC` ce qui semblerait confirmer ce comportement dans deux approches différentes de clustering . Nous avons vu que dans un cas idéal l'intérêt de donner plus d'importance aux petits clusters (priorité `Increasing`) permet d'augmenter la diversité, mais dans un cas réel cette priorité est parfois moins bonne que `Rank` ou `Decreasing` à cause des images non-pertinentes. Donc, si on se trouve dans un cas idéal et nous disposons des descripteurs de bonne qualité en diversité (qui donnent de bons résultats en diversité) il semble mieux d'ordonner les clusters par une priorité `Increasing` pour augmenter la diversité. Dans le cas contraire, il semble toujours mieux ordonner les clusters par une priorité classique par `Rank`.

#### 8.1.1.2 Point de convergence des résultats

Dans les figures précédentes, nous voyons que les scores des trois techniques de priorité convergent lorsqu'on utilise un découpage `Fixe` à 10 clusters pour CR@10 et P@10. Mais, contrairement à la `AHC` où la convergence est obligatoirement vers exactement la même valeur, dans le cas du `K-Means` cette convergence n'est pas exacte, c'est-à-dire que la valeur vers laquelle convergent ces techniques, n'est pas forcément exactement la même. La raison de cette convergence est la même que dans le point de convergence trouvé dans la `AHC` (voir section 7.2.2 page 102) où même si le tri des clusters est différent (ordre croissant, décroissant ou par le rang), le nombre de clusters étant identique au nombre d'images évaluées, on prend exactement une image par cluster. Or pour chaque cluster, on choisit en premier l'image qui a le plus fort rang dans le cluster quelle que soit la méthode de tri des clusters. Donc dans les $n$ premières images, on prend les mêmes $n$ images (même si leur ordre est différent). Malgré la variation du résultat final dans le clustering plat (à cause de l'initialisation aléatoire), les scores sont très proches avec les 3 méthodes de tri des clusters.

Donc, on peut dire que pour CR@10, CR@20, CR@30 (CR@n en général) les trois techniques de priorité convergent toujours (de manière inexacte) avec un découpage `Fixe` à $n$ clusters. En conséquence, dans ce point de convergence, on pourrait utiliser n'importe quelle technique de priorité, car dans ce point les trois techniques vont donner des scores

similaires.

### 8.1.2 Influence du nombre de clusters

De même que pour les expérimentations faites avec la AHC (voir section 7.3 page 103), dans cette section nous comparons deux techniques de découpage. La première, une technique de découpage Fixe de $n$ clusters (abrégé par "fn"). La deuxième, une technique de découpage Adapt qui choisit un nombre de clusters égal au nombre de sous-thèmes de la vérité terrain. Puis, nous étudions plus en détail le découpage Fixe sur plusieurs valeurs de CR, car ce découpage semblerait donner un comportement intéressant pour choisir le bon nombre de clusters pour augmenter la diversité.

Il faut remarquer aussi que dans ces expérimentations nous n'utilisons pas le découpage Traditional car il est adapté seulement au clustering hiérarchique.

#### 8.1.2.1 Comparaison Adapt et Fixe

Nous utilisons les mêmes figures de la section précédente pour comparer les deux techniques de découpage sur plusieurs benchmarks (voir figure 8.1) et sur un autre descripteur visuel (voir figure 8.2).

De même que pour la AHC, dans ces figures on voit aussi que le découpage Fixe donne en général des meilleurs résultats en diversité que le découpage Adapt qui est censé faire un choix idéal, car il prend le nombre de sous-thèmes de la vérité terrain. Peut-être K-Means ne génère pas exactement les mêmes clusters que les sous-thèmes de la vérité terrain ce qui peut fausser le choix idéal du nombre de clusters et qui pourrait empêcher à Adapt d'obtenir les meilleurs résultats en diversité.

Par contre, dans le découpage Fixe nous avons toujours l'inconvénient de choisir le bon nombre de clusters qui nous donnera la valeur maximale en diversité.

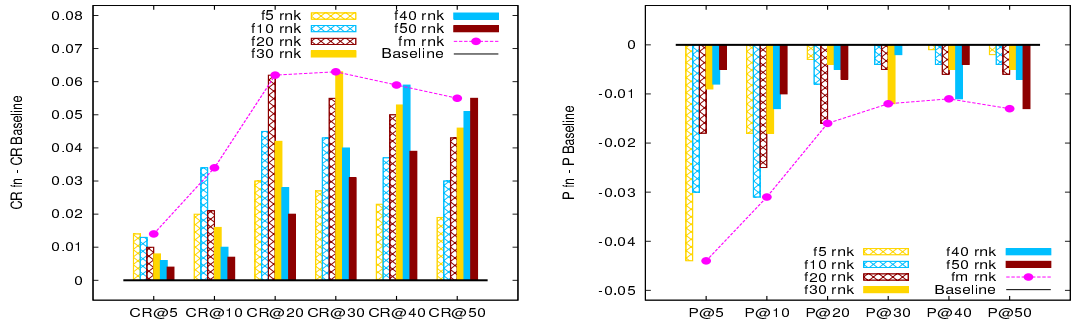
#### 8.1.2.2 Valeur maximale pour CR@m

De même que pour la AHC (voir section 7.3.3 page 105), dans les figures de K-Means on peut voir aussi le même comportement pour CR@10 où les scores du découpage Fixe augmentent de "f5" jusqu'à "f10", puis ils diminuent de "f10" jusqu'à "f50" avec une valeur maximale obtenue avec le découpage "f10".

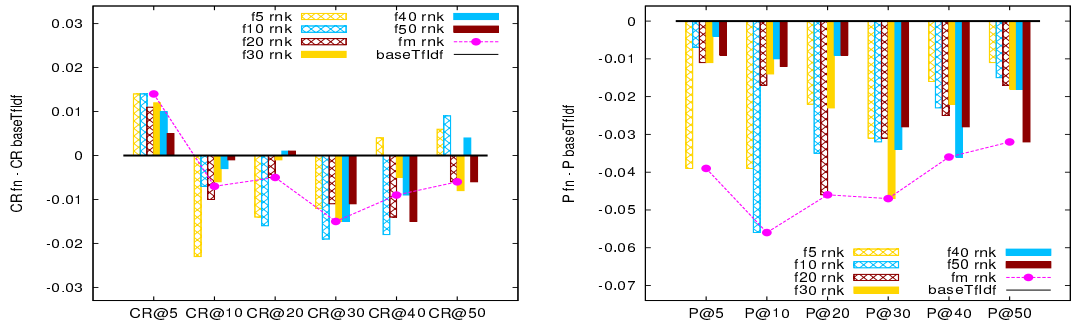
En sachant que dans les expérimentations faites avec la AHC la valeur maximale pour CR@m est obtenue avec un découpage Fixe à $m$ clusters, est-ce qu'on arrive à retrouver le même comportement de la valeur maximale avec K-Means ? Pour valider cette hypothèse, nous ajoutons un découpage Fixe "fm" où le choix du nombre de clusters varie selon le nombre $m$ d'images à évaluer (CR@m, P@m). Ce découpage nous servira comme guide pour valider le comportement de la valeur maximale sur K-Means.

**Valeur maximale pour CR@m sur plusieurs benchmarks** Dans la figure 8.3 nous comparons la différence entre les scores CR@m et P@m moyens de la baseline et les scores CR@m et P@m moyens de K-Means avec le descripteur HSV en utilisant différents découpages Fixe "fn", dans un cas idéal et réel.

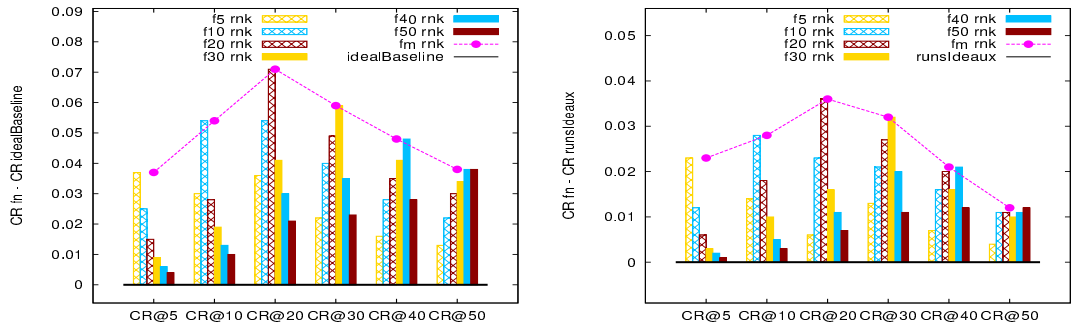
De même que pour la AHC, dans ces figures on voit qu'en général, la valeur maximale pour CR@m est obtenue avec le découpage Fixe "fm" ce qui confirme l'hypothèse. Nous avons trouvé une seule exception à ce comportement dans le cas réel du benchmark Clef (voir figure 8.3(b)) où on voit que le comportement de la valeur maximale est instable peut-être à cause de la faible pertinence de cette baseline.




(a) `K-Means`(HSV) (`Media`-réel)

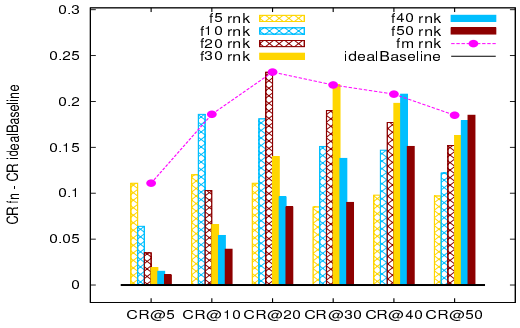



(b) `K-Means`(HSV) (`Clef`-réel)



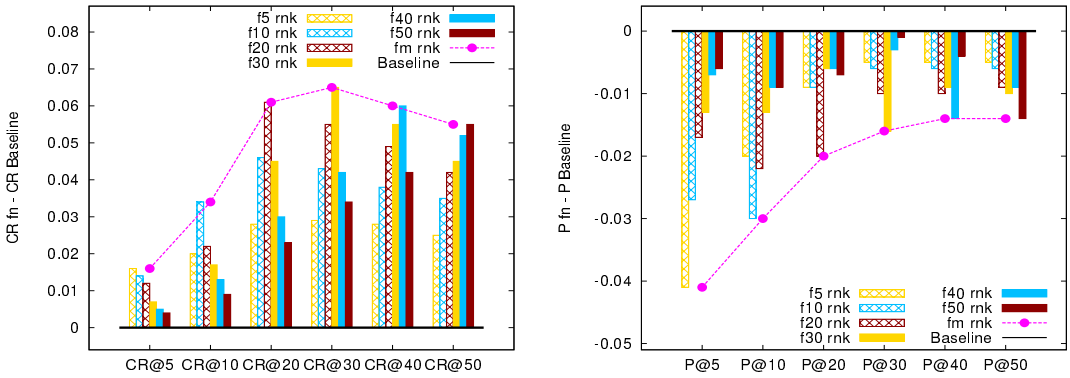

(c) `K-Means`(HSV) (`Media`-idéal)

(d) `K-Means`(HSV) (`Clef`-idéal)

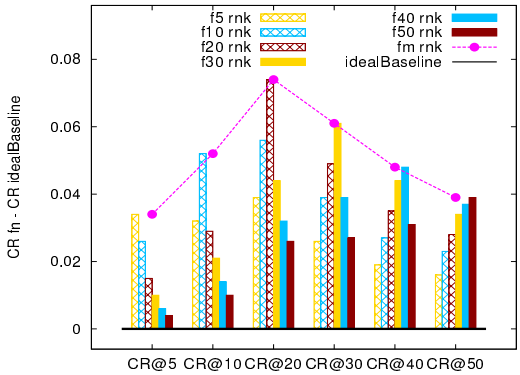



(e) `K-Means`(HSV) (`Xilo`-idéal)

FIGURE 8.3 – Valeur maximale pour CR@m sur plusieurs benchmarks. Les scores représentent la différence entre les scores CR@m, P@m moyens de la baseline et les scores CR@m, P@m moyens de la `K-Means` avec HSV en utilisant différents découpages `Fixe` "fn" et une priorité `Rank` sur les benchmarks (a) (c) `Media-testset`, (b) (d) `Clef` et (e) `Xilo`




(a) `K-Means`(CN3x3) (`Media`-réel)




(b) `K-Means`(CN3x3) (`Media`-idéal)

FIGURE 8.4 – Étude de la valeur maximale pour CR@m sur le descripteur de couleur CN3x3. Les scores représentent la différence entre les scores CR@m moyens de la référence et les scores CR@m moyens de la `K-Means` avec le descripteur CN3x3 dans un cas (a) réel et (b) idéal sur le benchmark `Media` (sous-ensemble `testset`) en utilisant différents découpages `Fixe` "fn" et une priorité classique `Rank`

**Valeur maximale pour CR@m sur un autre descripteur visuel** Dans la figure 8.4 nous comparons la différence entre les scores CR@m et P@m moyens de la baseline et les scores CR@m et P@m moyens de `K-Means` avec le descripteur CN3x3 en utilisant différents découpages `Fixe` "fn", dans un cas idéal et réel sur le benchmark `Media`.

Dans cette figure on peut voir aussi le même comportement de la valeur maximale pour CR@m en utilisant un descripteur visuel de la même famille que HSV et qui donne aussi de bons résultats en diversité.

**Discussion** De manière générale on peut voir des comportements similaires avec les deux méthodes de clustering `K-Means` et la `AHC` ce qui semblerait confirmer ces comportements dans deux approches différentes de clustering.

Premièrement, nous avons vu que le découpage `Adapt` qui est censé faire un choix idéal nous a donné de moins bons résultats en diversité qu'un découpage `Fixe` peut-être, car `K-Means` ne génère pas exactement les mêmes clusters que les sous-thèmes de la vérité terrain ce qui pourrait fausser le choix idéal du nombre de clusters pour augmenter la diversité.

Puis, nous avons vu que dans `K-Means` la valeur maximale de diversité pour CR@m est obtenue avec un découpage `Fixe` de $m$ clusters en utilisant des descripteurs de bonne

TABLE 8.1 – Étude des scores P@10, CR@10 moyens de `K-Means` avec les descripteurs HSV et CN3x3 en utilisant la priorité par `Rank` et le découpage `Fixe` à 10 clusters dans un cas réel sur les benchmarks `Clef` et `Media` (sous-ensemble `testset`). Pour chaque descripteur nous avons exécuté 10 fois cette méthode

| Méthode | | `Clef` (baseTfIdf) | | `Media` (Baseline) | |
|---|---|---|---|---|---|
| | | P@10 | CR@10 | P@10 | CR@10 |
| `K-Means`(HSV) | min | 0.205 | 0.195 | 0.720 | 0.394 |
| | max | 0.236 | 0.221 | 0.728 | 0.404 |
| | mean | **0.221** | **0.210** | **0.724** | **0.399** |
| | std | 0.010 | 0.007 | 0.002 | 0.003 |
| `K-Means`(CN3x3) | min | - | - | 0.717 | 0.390 |
| | max | - | - | 0.731 | 0.404 |
| | mean | **-** | **-** | **0.725** | **0.399** |
| | std | - | - | 0.005 | 0.003 |

TABLE 8.2 – Étude des scores P@10, CR@10 moyens de `K-Means` avec les descripteurs HSV et CN3x3 en utilisant la priorité par `Rank` et le découpage `Fixe` à 10 clusters dans un cas idéal sur les benchmarks `Xilo`, `Clef` et `Media` (sous-ensemble `testset`). Pour chaque descripteur nous avons exécuté 10 fois cette méthode

| Méthode | | `Xilo` (idealBaseline) | | `Clef` (runsIdeaux) | | `Media` (idealBaseline) | |
|---|---|---|---|---|---|---|---|
| | | P@10 | CR@10 | P@10 | CR@10 | P@10 | CR@10 |
| `K-Means`(HSV) | min | 1.000 | 0.421 | 0.993 | 0.619 | 0.980 | 0.526 |
| | max | 1.000 | 0.496 | 0.994 | 0.630 | 0.980 | 0.544 |
| | mean | **1.000** | **0.457** | **0.993** | **0.624** | **0.980** | **0.533** |
| | std | 0.000 | 0.022 | 0.000 | 0.003 | 0.000 | 0.005 |
| `K-Means`(CN3x3) | min | - | - | - | - | 0.980 | 0.521 |
| | max | - | - | - | - | 0.980 | 0.537 |
| | mean | **-** | **-** | **-** | **-** | **0.980** | **0.530** |
| | std | - | - | - | - | 0.000 | 0.004 |

qualité en diversité comme les descripteurs visuels HSV et CN3x3. Donc, il semblerait que le bon choix du nombre de clusters pour `K-Means` est de choisir un nombre fixe de clusters égal au nombre d'images à évaluer pour obtenir la valeur maximale en diversité.

Nous avons réalisé de très nombreuses expériences sur les différents benchmarks avec les différents descripteurs et en faisant varier divers paramètres, nous avons trouvé une exception au comportement de la valeur maximale en diversité dans le cas réel du `Clef`. Dans cette exception, le comportement de la valeur maximale en diversité devient instable peut-être à cause de la faible pertinence de cette baseline.

### 8.1.3 Influence de l'initialisation des clusters

Finalement, dans cette section nous allons étudier l'influence de l'initialisation aléatoire des clusters avec la méthode `K-Means`. Pour cela, dans les tableaux 8.1 et 8.2 nous montrons les scores CR@10 et P@10 moyens d'utiliser `K-Means` 10 fois avec les descripteurs HSV et CN3x3 en utilisant la priorité par `Rank` et un découpage `Fixe` à 10 clusters dans les cas réel (voir tableau 8.1) et idéal (voir tableau 8.2) sur les trois benchmarks.

Dans ces tableaux on peut voir qu'exécuter plusieurs fois la méthode `K-Means` avec le même jeu de paramètres, génère des scores moyens similaires avec des écarts types

moyens, malgré l'initialisation aléatoire de clusters qui ne garantie pas toujours la même qualité de clusters. Peut-être l'ensemble de données est déjà regroupé et donc la méthode K-Means arriverait à converger plus facilement et obtenir ainsi des scores proches. Ce résultat nous permet d'utiliser une seule fois cette méthode sans avoir trop le risque de tomber sur des scores abrupts qui puissent fausser les comportements obtenus.

### 8.1.4 Discussion

Pour utiliser la méthode K-Means sur la diversité, nous avons besoin de divers paramètres comme le nombre de clusters à utiliser et indiquer l'ordre de clusters pour générer le résultat final. Dans cette section nous avons étudié l'influence de ces paramètres sur la diversité et nous avons vu principalement que les comportements obtenus avec K-Means sont similaires à ceux obtenues avec la AHC, en sachant qu'il utilise une approche, par partitionnement plat, très différent de la méthode AHC, par partitionnement hiérarchique, pour générer les clusters et d'autre part un critère d'agrégation différent (centroïde) à celui de la AHC (average).

Par rapport à **l'ordre de clusters**, il semblerait qu'utiliser une priorité qui donne plus d'importance aux petits clusters (des clusters qui peuvent avoir des images originales) permet d'augmenter la diversité dans un cas idéal. Dans le cas contraire, il vaut toujours mieux d'ordonner les clusters par une priorité classique par Rank, car elle est moins pénalisé par les images non-pertinentes.

Par rapport au **nombre de clusters**, il semblerait que le bon choix du nombre de clusters (qui est un problème sans réponse unique) est de choisir toujours un nombre fixe des $n$ clusters égal au nombre de $n$ images à évaluer. De plus, ce point de la valeur maximale en diversité devient aussi le point de convergence entre les trois techniques de priorité, ce qui permet que le bon choix du nombre de clusters ne soit pas dépendant de la technique de priorité.

Enfin, nous avons vu qu'exécuter plusieurs fois la méthode K-Means avec le même jeu de paramètres, génère des scores moyens assez proches, malgré l'initialisation aléatoire des clusters ce qui nous permet de confirmer certains comportements sans risquer de tomber sur des cas particuliers (sur une fausse valeur maximale de diversité par exemple).

## 8.2 Étude de la diversité par K-Medoids

Dans la section précédente nous avons étudié K-Means qui est une des méthodes classiques les plus utilisées par les chercheurs. Par contre, cette méthode n'est pas adaptée à certains descripteurs, car elle a besoin du calcul des centroïdes. Afin d'étudier le clustering plat sur plusieurs descripteurs, nous étudions une autre méthode de clustering plat appelé K-Medoids, qui contrairement à K-Means, calcule des medoïdes (le choix du vecteur d'une image pour représenter le cluster) ce qui le rend plus adapté à la plupart de descripteurs.

Dans l'algorithme de K-Medoids nous utilisons aussi une initialisation aléatoire de clusters qui peut faire varier son résultat final. De même que pour K-Means, nous lançons cette méthode 10 fois pour chaque jeu de paramètres pour éviter ainsi de tomber sur un score particulier.

Cependant, l'algorithme de K-Medoids est plus coûteux en complexité que celui de K-Means, ce qui pourrait limiter à K-Medoids d'être utilisable dans une application de recherche d'images "en ligne", mais cette méthode nous servira pour étudier l'influence d'utiliser le clustering plat avec différents descripteurs sur la diversité.

Dans cette section, nous allons étudier d'abord l'influence de divers paramètres comme le nombre de clusters et leur ordre sur la diversité. Puis, nous étudions l'in-

fluence de l'initialisation aléatoire de clusters sur `K-Medoids` afin de savoir si on risque de tomber sur des scores abrupts.

### 8.2.1 Influence de l'ordre des clusters

De même que pour `K-Means` (voir section 8.1.1), dans `K-Medoids` nous étudions aussi les mêmes trois techniques de priorité : d'abord par `Rank` qui trie les clusters selon le rang des images et deux autres `Increasing` et `Decreasing` qui trient les clusters selon la taille des clusters.

#### 8.2.1.1 Comparaison `Increasing`, `Decreasing` et `Rank`

Dans cette partie nous comparons les trois techniques de priorité sur plusieurs benchmarks et sur plusieurs descripteurs. Dans ces expériences, nous utilisons différentes techniques de découpages.

**Comparaison des techniques de priorité sur plusieurs benchmarks** Dans la figure 8.5 nous comparons les trois techniques de priorité sur plusieurs benchmarks. Les scores CR@10 et P@10 moyens représentent la méthode `K-Medoids` avec le descripteur HSV en utilisant différentes techniques de découpage dans un cas idéal et réel.

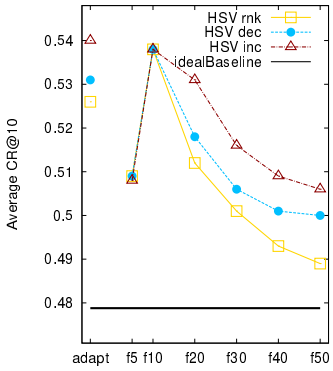
De même que pour `K-Means` et la `AHC`, dans `K-Medoids` on voit aussi que la priorité `Increasing` obtient de bons résultats en diversité, car elle donne plus d'importance aux petits clusters qui peuvent représenter des images originales. Par contre, dans la figure 8.5(d) on voit toujours l'exception à ce comportement où `Increasing` obtient de moins bons résultats en diversité que `Rank` et `Decreasing` (peut-être à cause des images non-pertinentes).

**Comparaison des techniques de priorité sur plusieurs descripteurs** Dans les figures 8.6 et 8.7 nous comparons les trois techniques de priorité sur plusieurs descripteurs. Les scores CR@10 et P@10 moyens représentent la méthode `K-Medoids` avec les descripteurs HSV et Tree sur le benchmark `Xilo` (figure 8.6) et les descripteurs CN3x3, TfIdf et GPS sur le benchmark `Media` (figure 8.7) en utilisant différentes techniques de découpage dans le cas idéal et réel.

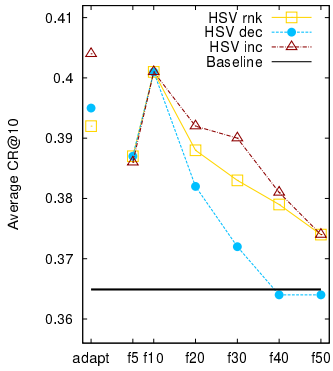
Dans ces figures on peut voir que lorsqu'on utilise des descripteurs de bonne qualité en diversité (comme CN3x3 ou Tree) la priorité `Increasing` continue à donner de bons résultats en diversité en sachant que Tree est un descripteur totalement différent des descripteurs des couleurs (HSV et CN3x3) et que "Wu-Palmer" est une mesure de similarité différente de "Euclidean". Par contre, on voit aussi que cette priorité devient moins intéressante que `Rank` ou `Decreasing` quand on utilise des descripteurs moins bons en diversité (comme le texte ou le GPS).

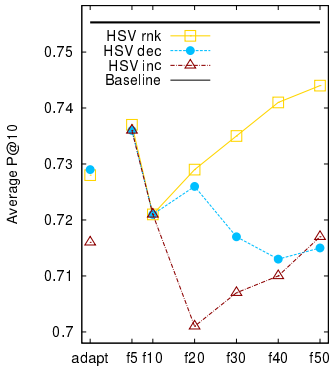
**Discussion** De manière générale on peut voir que les trois techniques de priorité obtiennent toujours des comportements similaires avec deux méthodes de clustering plat et une méthode de clustering hiérarchique qui utilisent les mêmes critères d'agrégation. Donc, le comportement des techniques de priorité ne semblerait pas dépendre du résultat fourni par les méthodes de clustering.

Nous avons vu que le fait de donner plus d'importance aux petits clusters (priorité `Increasing`) semble être très intéressant pour augmenter la diversité dans un cas idéal et avec l'utilisation des descripteurs de bonne qualité en diversité. Dans le cas contraire, cette priorité devient moins intéressante que `Rank` et `Decreasing` à cause des images non-pertinentes et des mauvais descripteurs en diversité qui peuvent générer des clusters qui se rassemblent le moins aux sous-thèmes de la vérité terrain.

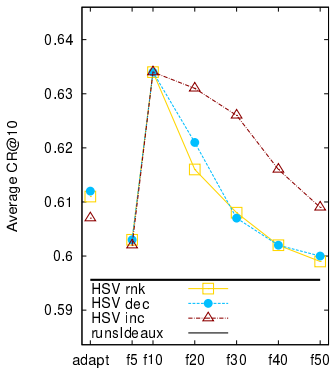



(a) `K-Medoids`(HSV) (`Media`-idéal)

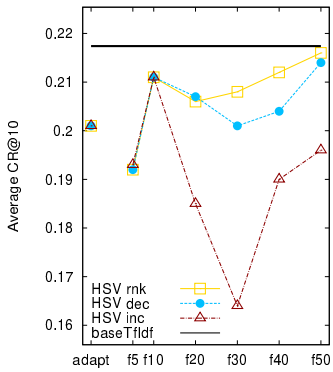


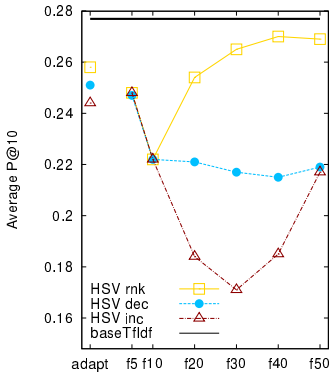




(b) `K-Medoids`(HSV) (`Media`-réel)

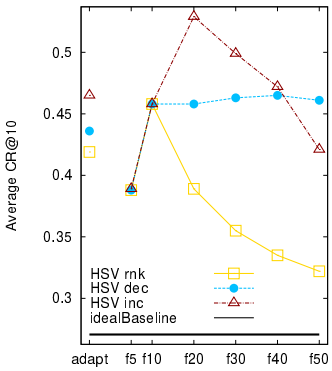



(c) `K-Medoids`(HSV) (`Clef`-idéal)



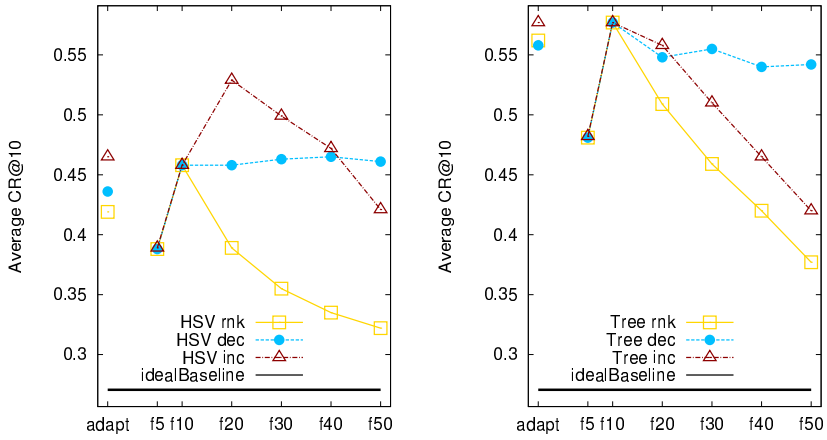

(d) `K-Medoids`(HSV) (`Clef`-réel)




(e) `K-Medoids`(HSV) (`Xilo`-idéal)

FIGURE 8.5 – Comparaison des techniques de priorité sur plusieurs benchmarks. Les scores CR@10 et P@10 moyens représentent `K-Medoids` avec le descripteur HSV en utilisant différentes techniques de découpage (en abscisse) dans un cas idéal et réel sur les benchmarks (a) (b) `Media` (sous-ensemble `testset`), (c) (d) `Clef` et (e) `Xilo`

Donc, si on se trouve dans un cas idéal et si nous disposons des descripteurs de bonne qualité en diversité, il semble mieux ordonner les clusters par une priorité `Increasing` pour augmenter la diversité. Dans le cas contraire, il semble toujours mieux d'ordonner les clusters par une priorité classique.




(a) `K-Medoids`(HSV) (`Xilo`-idéal) (b) `K-Medoids`(Tree) (`Xilo`-idéal)

FIGURE 8.6 – Comparaison des techniques de priorité avec les descripteurs HSV et Tree. Les scores CR@10 et P@10 moyens représentent `K-Medoids` avec les descripteurs (a) HSV et (b) Tree en utilisant différentes techniques de découpage (en abscisse) dans un cas idéal sur le benchmark `Xilo`

#### 8.2.1.2 Point de convergence des résultats

De même que pour `K-Means`, les figures de `K-Medoids` montrent aussi que les trois techniques de priorité ont un point de convergence inexacte (avec des scores similaires) dans le découpage `Fixe` à 10 clusters pour CR@10 et P@10 (voir l'explication de cette convergence dans la section 8.1.1.2).

Donc, on peut dire que pour CR@10, CR@20, CR@30 (CR@n en général) les trois techniques de priorité vont converger toujours un découpage `Fixe` à $n$ clusters, malgré l'utilisation de différents descripteurs. En conséquence, dans ce point de convergence, on peut utiliser n'importe quelle technique de priorité, car dans ce point les trois techniques vont donner toujours des scores similaires.

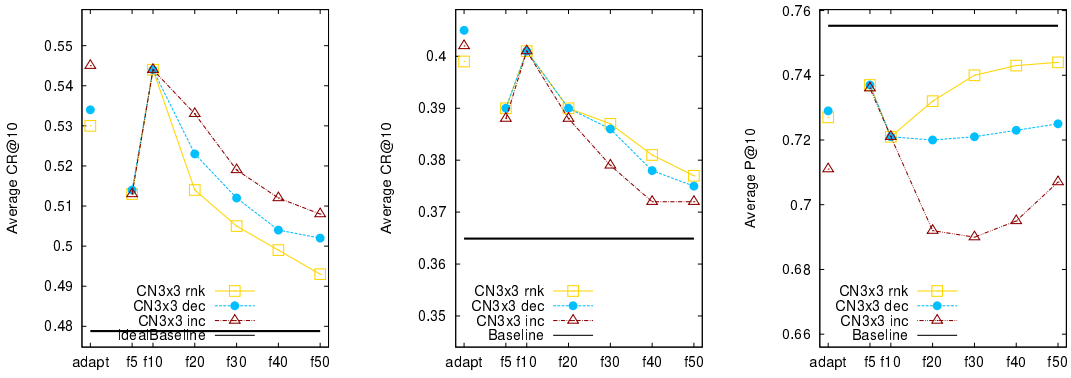
### 8.2.2 Influence du nombre de clusters

De même que pour `K-Means` (voir section 8.1.2), dans `K-Medoids` nous comparons les mêmes techniques de découpage afin d'étudier l'influence du nombre de clusters sur la diversité. Ces techniques de découpages sont deux : un découpage `Adapt` qui choisit un nombre de clusters égal au nombre de sous-thèmes de la vérité terrain et un découpage `Fixe` de $n$ clusters (abrégé par "fn"). Puis, nous étudions le découpage `Fixe` sur plusieurs valeurs de CR@m car ce découpage semble donner le bon choix du nombre de clusters sur la diversité.

#### 8.2.2.1 Comparaison `Adapt` et `Fixe`

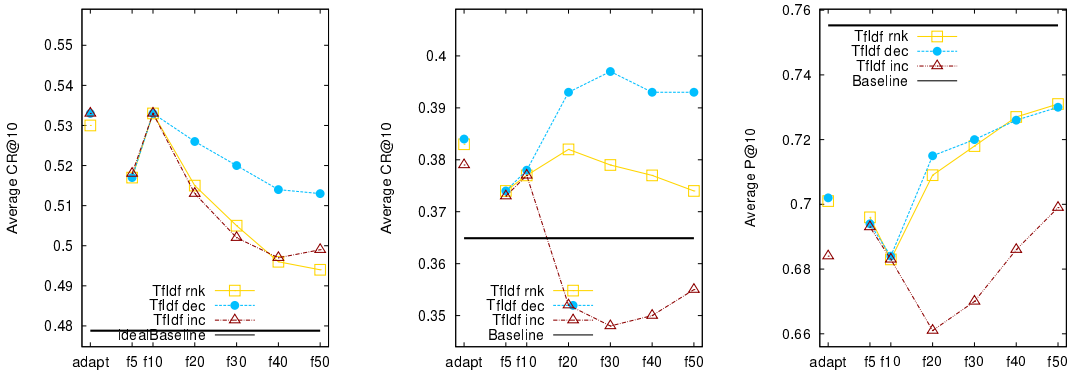
Dans cette partie nous utilisons les mêmes figures que celles utilisées dans la section précédente pour comparer les deux techniques de découpage sur plusieurs benchmarks (voir figure 8.5) et sur plusieurs descripteurs (voir figure 8.7).

De même que pour `K-Means` et la `AHC`, dans `K-Medoids` on voit aussi que le découpage `Fixe` donne en général des résultats similaires ou meilleurs en diversité que le découpage `Adapt` qui est censé faire un choix idéal, car il prend le nombre de sous-thèmes de la vérité terrain. Peut-être que `K-Medoids` ne génère pas exactement les mêmes clusters que les sous-thèmes de la vérité terrain ce qui peut fausser le choix idéal du nombre de



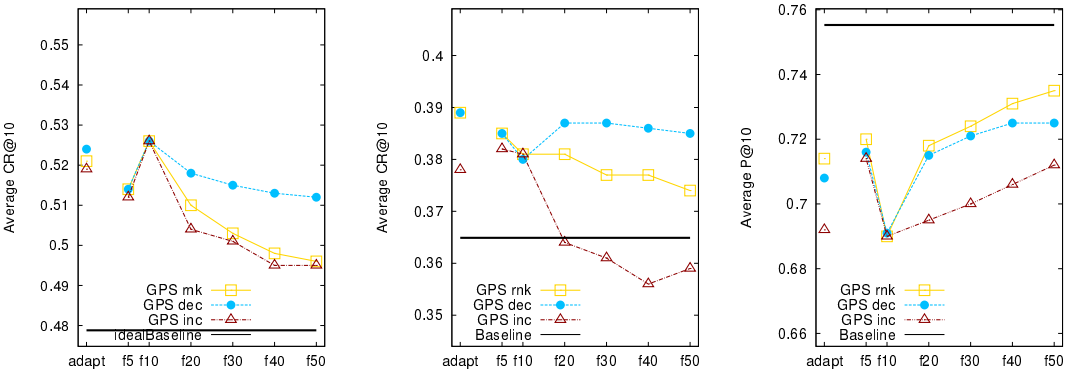

(a) `K-Medoids`(CN3x3) (`Media`-idéal)

(b) `K-Medoids`(CN3x3) (`Media`-réel)




(c) `K-Medoids`(TfIdf) (`Media`-idéal)

(d) `K-Medoids`(TfIdf) (`Media`-réel)

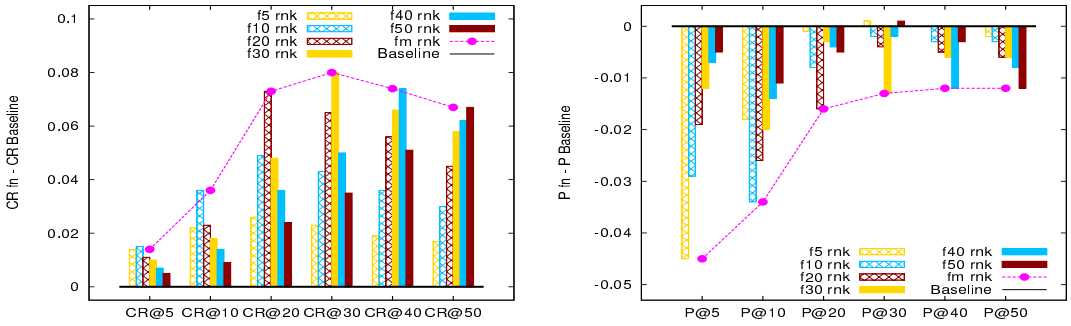



(e) `K-Medoids`(GPS) (`Media`-idéal)

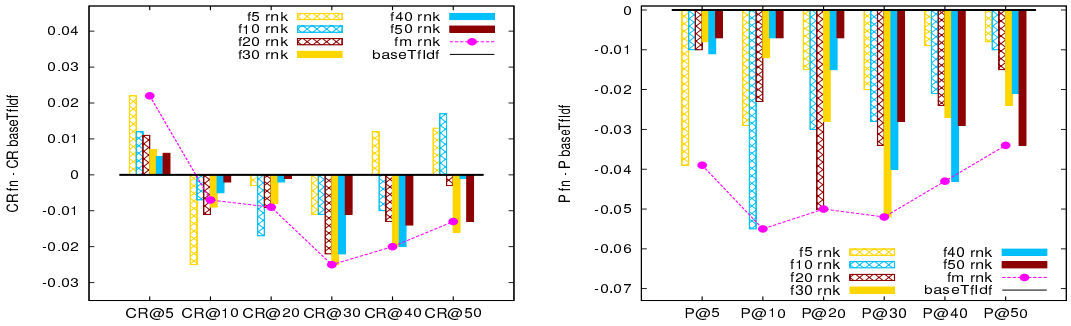
(f) `K-Medoids`(GPS) (`Media`-réel)

FIGURE 8.7 – Comparaison des techniques de priorité sur plusieurs descripteurs. Les scores CR@10 et P@10 moyens représentent la `K-Medoids` avec les descripteurs (a) (b) CN3x3, (c) (d) TfIdf et (e) (f) GPS en utilisant différentes techniques de découpage (en abscisse) dans un cas idéal et réel sur le benchmark `Media` (sous-ensemble `testset`)

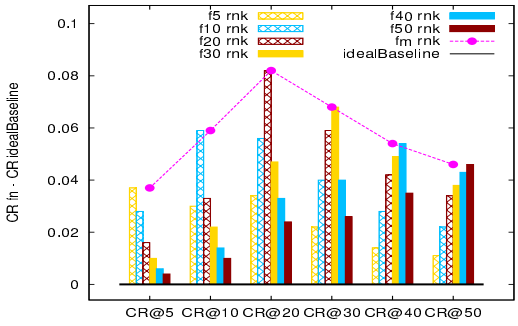
clusters. Par contre, on a toujours l'inconvénient pour le découpage `Fixe` de choisir le bon nombre de clusters qui pourrait nous donner la valeur maximale en diversité.



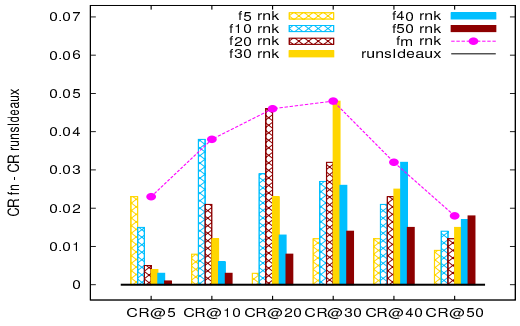

(a) `K-Medoids`(HSV) (`Media`-réel)



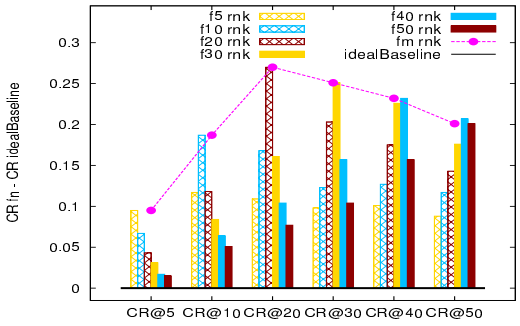

(b) `K-Medoids`(HSV) (`Clef`-réel)




(c) `K-Medoids`(HSV) (`Media`-idéal)

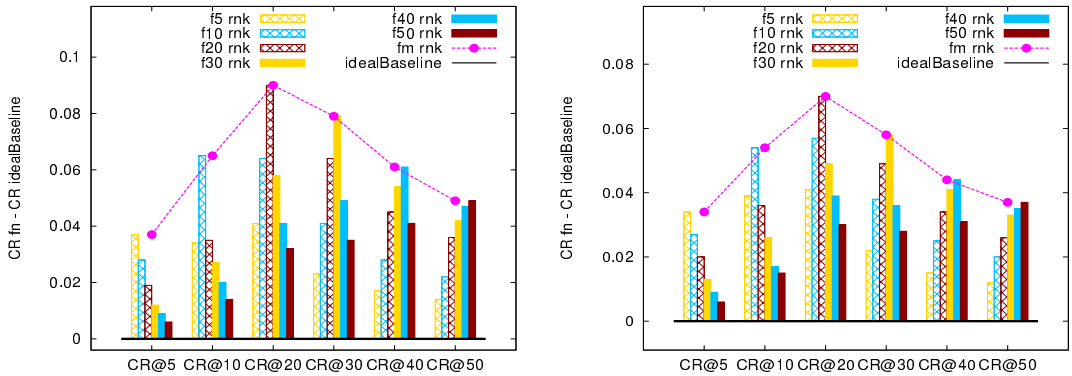



(d) `K-Medoids`(HSV) (`Clef`-idéal)




(e) `K-Medoids`(HSV) (`Xilo`-idéal)

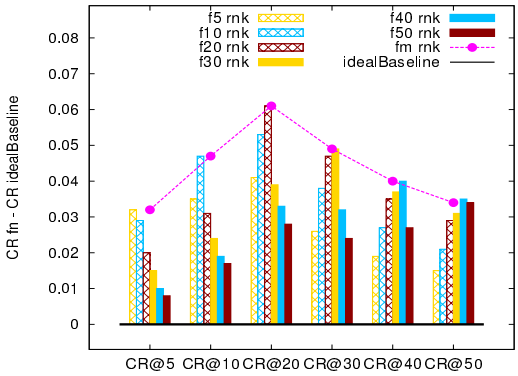
FIGURE 8.8 – Valeur maximale pour CR@m sur plusieurs benchmarks. Les scores représentent la différence entre les scores CR@m, P@m moyens de la baseline et les scores CR@m, P@m moyens de la `K-Medoids` avec HSV en utilisant différents découpages `Fixe` "fn" et une priorité `Rank` sur les benchmarks (a) (c) `Media-testset`, (b) (d) `Clef` et (e) `Xilo`



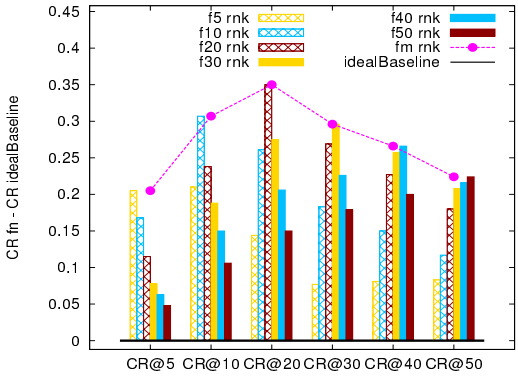

(a) K-Medoids(CN3x3) (Media-idéal)




(b) K-Medoids(TfIdf) (Media-idéal)




(c) K-Medoids(GPS) (Media-idéal)




(d) K-Medoids(Tree) (Xilo-idéal)

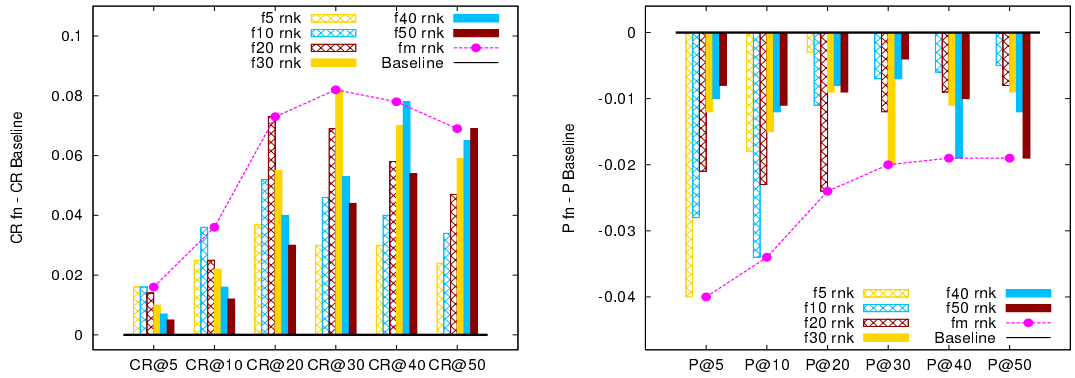
FIGURE 8.9 – Comparaison de la différence entre les scores CR@m moyens de la baseline et les scores CR@m moyens de la K-Medoids avec les descripteurs (a) CN3x3, (b) TfIdf et (c) GPS sur le benchmark Media (sous-ensemble testset) et le descripteur (d) Tree sur le benchmark Xilo en utilisant différents découpages Fixe "fn" et une priorité classique Rank dans un cas idéal

#### 8.2.2.2 Valeur maximale pour CR@m

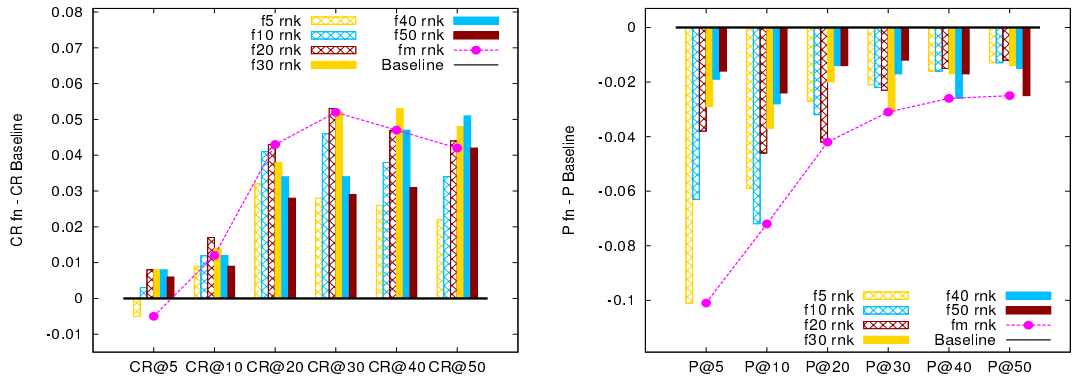
De même que pour K-Means et la AHC, les figures de K-Medoids montrent aussi que pour CR@10 les scores du découpage Fixe augmentent de "f5" jusqu'à "f10", puis ils diminuent de "f10" jusqu'à "f50" avec une valeur maximale obtenue avec le découpage "f10". En sachant que dans K-Means et dans la AHC, la valeur maximale pour CR@m est obtenue avec un découpage Fixe à $m$ clusters, est-ce qu'on peut arriver à obtenir le même comportement de la valeur maximale avec K-Medoids ? Pour valider cette hypothèse nous ajoutons un découpage Fixe "fm" où le choix du nombre de clusters varie selon le nombre $m$ d'images à évaluer. Ce découpage nous servira comme guide pour valider le comportement initial de la valeur maximale.

**Valeur maximale pour CR@m sur plusieurs benchmarks** Dans la figure 8.8 nous comparons la différence entre les scores CR@m et P@m moyens de la baseline et les scores CR@m et P@m moyens de K-Medoids avec le descripteur HSV en utilisant différents découpages Fixe "fn", dans un cas idéal et réel.

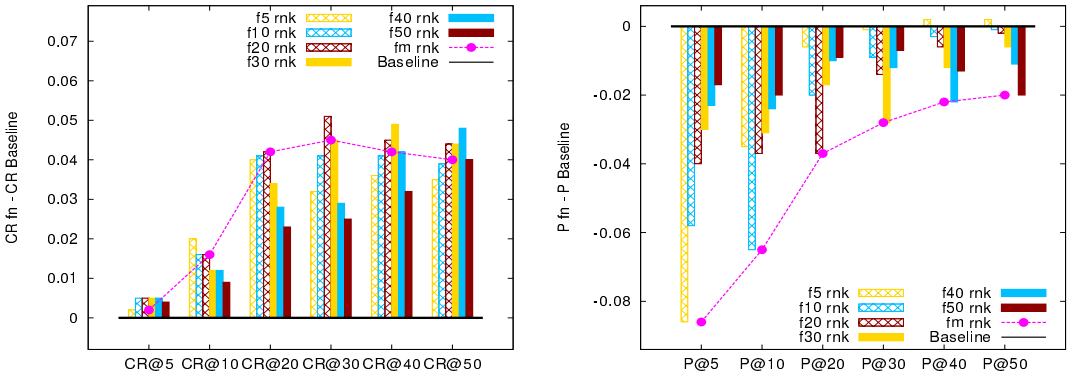
De même que pour K-Means et la AHC, dans K-Medoids on voit qu'en général, la valeur maximale pour CR@m est obtenue aussi avec le découpage Fixe "fm" ce qui confirme notre hypothèse avec cette méthode. Par contre, nous avons trouvé une seule exception




(a) `K-Medoids`(CN3x3) (`Media`-réel)




(b) `K-Medoids`(TfIdf) (`Media`-réel)




(c) `K-Medoids`(GPS) (`Media`-réel)

FIGURE 8.10 – Comparaison de la différence entre les scores CR@m et P@m moyens de la baseline et les scores CR@m et P@m moyens de la `K-Medoids` avec les descripteurs (a) CN3x3, (b) TfIdf et (c) GPS sur le benchmark `Media` (sous-ensemble `testset`) et le descripteur (d) Tree sur le benchmark `Xilo` en utilisant différents découpages `Fixe` "fn" et une priorité classique `Rank` dans un cas réel

à ce comportement qui est toujours obtenu dans le cas réel du benchmark `Clef` (voir figure 8.8(b)) où on voit que le comportement de la valeur maximale est très instable peut-être à cause de la faible pertinence de ce type de baseline.

**Valeur maximale pour CR@m sur plusieurs descripteurs** Les figures 8.9 et 8.10 comparent la différence entre les scores CR@m et P@m moyens de la baseline et

les scores CR@m et P@m moyens de K-Medoids avec les descripteurs CN3x3, TfIdf et GPS sur le benchmark Media et le descripteur Tree sur le benchmark Xilo en utilisant différents découpages Fixe "fn", dans un cas idéal (figure 8.9) et réel (figure 8.10).

Dans ces figures on peut voir que le comportement de la valeur maximale devient moins stable quand on est dans un cas réel et quand on utilise de moins bons descripteurs en diversité (comme le texte ou GPS) où "fm" pourrait ne pas devenir le bon choix du nombre de clusters sur certaines valeurs de CR@m.

**Discussion** De manière générale nous avons vu que le comportement de la valeur maximale pour CR@m est obtenu sur deux méthodes de clustering plat et une méthode de clustering hiérarchique. Il semblerait donc que ce comportement peut être généralisé sur des autres méthodes de clustering en sachant que le bon choix du nombre de clusters est toujours un problème sans réponse unique.

Nous avons vu, en général, que le bon choix du nombre de clusters pour obtenir la valeur maximale en diversité est de choisir un nombre Fixe de $m$ clusters égal au nombre d'images à évaluer (CR@m). Par contre, si on est dans un cas réel et on utilise de moins bons descripteurs en diversité (comme le texte et le GPS) ce choix peut devenir faux sur certaines valeurs de CR@m.

Après avoir réalisé de nombreuses expériences sur les différents benchmarks avec les différents descripteurs et en faisant varier divers paramètres, nous avons trouvé une exception du comportement de la valeur maximale en diversité dans le cas réel du benchmark Clef où ce comportement devient très instable peut être à cause de sa baseline qui contient beaucoup d'images non-pertinentes.

### 8.2.3 Influence de l'initialisation des clusters

En sachant que les méthodes de clustering plat peuvent varier à cause de l'initialisation aléatoire des clusters, dans cette section nous étudions l'influence de ce type d'initialisation sur la méthode K-Medoids avec les paramètres de la valeur maximale pour CR@10 afin de savoir si on risque de tomber sur des scores abrupts qui peuvent fausser le score de la valeur maximale. Dans les tableaux 8.3 et 8.4 nous montrons les scores CR@10 et P@10 moyens de K-Medoids exécute 10 fois avec les descripteurs HSV, CN3x3, TfIdf, GPS et Tree en utilisant la priorité par Rank et le découpage Fixe à 10 clusters dans les cas réel (voir tableau 8.3) et idéal (voir tableau 8.4) sur les trois benchmarks.

De même que pour K-Means, les tableaux de K-Medoids montrent aussi qu'exécuter plusieurs fois cette méthode avec le même jeu de paramètres, génère des scores moyens proches avec des écarts types moyens, malgré l'initialisation aléatoire de clusters qui pourrait faire varier le résultat final. Peut-être que l'ensemble de données est déjà regroupé ce qui permet à K-Medoids de converger rapidement et obtenir ainsi des scores proches.

### 8.2.4 Discussion

Dans cette section nous avons étudié la méthode K-Medoids qui nous a permis d'étudier le clustering plat sur plusieurs descripteurs (car K-Means n'est pas adapté pour certains types de descripteurs) et de confirmer certains résultats obtenus avec K-Means sur plusieurs benchmarks. Par contre, elle est plus coûteuse en complexité que K-Means ce qui pourrait lui empêcher d'être utilisable dans une application de recherche d'images "en ligne".

De même que pour K-Means et la AHC, dans K-Medoids nous avons étudié l'influence de divers paramètres comme le nombre et l'ordre de clusters.

Par rapport à **l'ordre de clusters**, il semblerait qu'utiliser une priorité qui donne plus d'importance aux petits clusters permet d'augmenter la diversité dans un cas idéal et en utilisant des descripteurs de bonne qualité en diversité. Dans le cas contraire, il vaut toujours mieux ordonner les clusters par une priorité classique par `Rank` moins pénalisé par les images non-pertinentes.

Par rapport au **nombre de clusters**, il semblerait que le bon choix, problème qui n'a pas qu'une solution, est de prendre un nombre fixe de clusters égal au nombre d'images à évaluer. De plus, la valeur maximale en diversité devient aussi le point de convergence entre les trois techniques de priorité, ce qui permet le bon choix ne soit pas dépendant de la technique de priorité. Par contre, si on est dans un cas réel et on utilise des descripteurs moins bons en diversité (comme le texte et le GPS) il peut devenir faux sur certaines valeurs de CR.

Nous avons vu aussi que l'arborescence de concepts a donné de bons résultats en diversité avec des comportements très similaires aux descripteurs de couleur HSV et CN3x3, en sachant que l'arborescence de concepts est un descripteur très différent de descripteurs de couleurs et que "Wu-Palmer" est une mesure de similarité différente de "Euclidean".

Enfin, nous avons vu qu'exécuter plusieurs fois la méthode `K-Medoids` avec le même jeu de paramètres, génère aussi des scores moyens assez proches, malgré l'initialisation aléatoire des clusters ce qui nous permet d'utiliser celle-ci une fois sans risquer de tomber sur un cas particulier (sur une fausse valeur maximale de diversité par exemple).

## 8.3 Bilan et discussion

Dans ce chapitre nous avons étudié deux méthodes de clustering plat : la méthode `K-Means` qui est très populaire et utilisée dans l'état de l'art et celle `K-Medoids` qui est moins connue, mais qui a certains avantages par rapport à `K-Means`.

L'algorithme de `K-Means` est moins coûteux en complexité que `K-Medoids` ce qui permet d'utiliser plus facilement cette méthode dans une application de recherche d'images "en ligne". Par contre, `K-Means` a besoin du calcul du centroïde qui n'est pas adapté pour certains descripteurs (comme le GPS par exemple). Pour cela, nous avons étudié `K-Medoids` qui utilise des medoïdes ce qui lui permet d'être utilisable avec n'importe quel type de descripteur et donc de pouvoir étudier le clustering plat sur plusieurs descripteurs.

Pour utiliser ces méthodes sur la diversité, nous avons besoin de divers paramètres comme le nombre de clusters, la façon d'initialiser ces clusters, l'ensemble d'images résultats (baseline), le type de descripteur et l'ordre de clusters. De plus, dans la littérature de recherche d'images, nous n'avons pas trouvé des autres travaux qui ont fait une étude approfondie sur ces paramètres pour l'amélioration de la diversité.

Dans les méthodes de clustering, faire le bon choix du **nombre de clusters** est dans la littérature un problème sans réponse unique où les auteurs proposent différentes approches (selon le nombre total de documents, selon la densité des clusters, etc.). Dans nos expérimentations, nous montrons qu'utiliser un nombre `Fixe` des clusters donne des meilleurs résultats que le fait de prendre le vrai nombre des sous-thèmes de la vérité terrain peut-être, car les clusters générés ne sont pas les mêmes aux vrais sous-thèmes de la vérité terrain.

Puis, après avoir testé le découpage `Fixe` des clusters sur plusieurs valeurs de CR, nous avons vu que le bon choix du nombre de clusters pour obtenir la valeur maximale en diversité semblerait être de choisir un nombre fixe clusters égal au nombre d'images à évaluer. De plus, cette valeur maximale devient aussi le point de convergence entre les trois techniques de priorité, ce qui permet à ce choix d'être indépendant de la technique

de priorité. Par contre, si on est dans un cas réel et on utilise des descripteurs moins bons en diversité (comme le texte et le GPS) ce choix peut être faux sur certaines valeurs de CR. Nous pouvons dire donc que notre proposition qui est de choisir un nombre fixe de clusters égal au nombre d'images à afficher est généralisée pour une méthode de clustering hiérarchique et deux méthodes de clustering plat. Cette proposition semble être très intéressante, car ce choix du nombre de clusters semblerait se répéter dans des autres méthodes de clustering.

Par rapport à **l'ordre de clusters**, nous avons comparé une priorité classique par le rang avec deux priorités proposées qui trient les clusters par sa taille. Dans ces expérimentations, nous montrons qu'utiliser une priorité qui donne plus d'importance aux petits clusters semble être intéressant pour augmenter la diversité, car ces petits clusters peuvent être des images originales. Par contre, cette priorité semble être sensible dans un cas réel (car les images non-pertinentes peuvent se trouver aussi dans les petits clusters) et quand on utilise des mauvais descripteurs en diversité (car ces ceux-ci peuvent générer des clusters de mauvaise qualité). Dans ce cas, il semble mieux ordonner les clusters par une priorité classique qui est moins dépendante de ces contraintes.

Dans les deux méthodes de clustering plat nous utilisons une **initialisation aléatoire des clusters** qui a l'inconvénient de ne pas assurer la même qualité de clusters. Pour cela, nous avons lancé ces méthodes plusieurs fois pour un jeu de paramètres donné afin d'éviter des scores abrupts qui puissent fausser les comportements obtenus. Dans ces expérimentations, nous avons montré que cette variation n'est pas très significative (avec des écarts types assez faible) ce qui nous permet d'utiliser une seule fois cette méthode sans le risque de tomber sur un score abrupt qui pourrait fausser par exemple la valeur maximale de diversité.

De même que pour certains descripteurs de couleur (comme HSV ou CN3x3), l'utilisation d'une **arborescence de concepts** comme descripteur semble être aussi intéressante pour augmenter la diversité, en sachant que l'arborescence de concepts est un descripteur très différent de descripteurs de couleur, que "Wu-Palmer" est une mesure de similarité différente de "Euclidean".

Comme la couleur et l'arborescence de concepts ont donné de bons résultats en diversité, il semblerait que le regroupement d'images de la vérité terrain se rapproche le plus à une approche par la couleur ou sémantique que par le texte ou le GPS. De plus, ce comportement est généralisé sur deux approches de clustering différents (un clustering hiérarchique et un clustering plat) ce qui pourrait être intéressant, car il semblerait aussi que ce comportement pourrait être généralisé sur des autres méthodes de clustering ou des autres méthodes de diversité.

Après avoir réalisé de nombreuses expériences sur les différents benchmarks avec les différents descripteurs et en faisant varier divers paramètres, nous avons trouvé une seule exception au comportement de la valeur maximale en diversité dans le cas réel du benchmark `Clef` qui a la particularité d'avoir une baseline avec beaucoup d'images non-pertinentes. Dans cette exception, ce comportement devient instable avec des scores aussi faibles en diversité (au niveau de la baseline) qui est peut-être dû à cause des images non-pertinentes.

TABLE 8.3 – Étude des scores P@10, CR@10 moyens de `K-Medoids` avec les descripteurs HSV, CN3x3, TfIdf et GPS en utilisant la priorité par `Rank` et le découpage `Fixe` à 10 clusters dans un cas réel sur les benchmarks `Clef` et `Media` (sous-ensemble `testset`). Pour chaque descripteur nous avons exécuté 10 fois cette méthode

| | | `Clef` (baseTfIdf) | | `Media` (Baseline) | |
|---|---|---|---|---|---|
| Méthode | | P@10 | CR@10 | P@10 | CR@10 |
| `K-Medoids`(HSV) | min | 0.215 | 0.197 | 0.714 | 0.393 |
| | max | 0.236 | 0.230 | 0.724 | 0.410 |
| | mean | **0.222** | **0.211** | **0.721** | **0.401** |
| | std | 0.006 | 0.011 | 0.003 | 0.005 |
| `K-Medoids`(CN3x3) | min | - | - | 0.719 | 0.397 |
| | max | - | - | 724 | 0.405 |
| | mean | - | - | **0.721** | **0.401** |
| | std | - | - | 0.002 | 0.002 |
| `K-Medoids`(TfIdf) | min | - | - | 0.672 | 0.370 |
| | max | - | - | 0.689 | 0.385 |
| | mean | - | - | **0.683** | **0.377** |
| | std | - | - | 0.005 | 0.004 |
| `K-Medoids`(GPS) | min | - | - | 0.682 | 0.376 |
| | max | - | - | 0.694 | 0.386 |
| | mean | - | - | **0.690** | **0.381** |
| | std | - | - | 0.004 | 0.003 |

TABLE 8.4 – Étude des scores P@10, CR@10 moyens de `K-Medoids` avec les descripteurs HSV, CN3x3, TfIdf, GPS et Tree en utilisant la priorité par `Rank` et le découpage `Fixe` à 10 clusters dans un cas idéal sur les benchmarks `Xilo`, `Clef` et `Media` (sous-ensemble `testset`). Pour chaque descripteur nous avons exécuté 10 fois cette méthode

| | | `Xilo` (idealBaseline) | | `Clef` (runsIdeaux) | | `Media` (idealBaseline) | |
|---|---|---|---|---|---|---|---|
| Méthode | | P@10 | CR@10 | P@10 | CR@10 | P@10 | CR@10 |
| `K-Medoids`(HSV) | min | 1.000 | 0.439 | 0.993 | 0.627 | 0.980 | 0.533 |
| | max | 1.000 | 0.471 | 0.994 | 0.642 | 0.980 | 0.545 |
| | mean | **1.000** | **0.458** | **0.994** | **0.634** | **0.980** | **0.538** |
| | std | 0.000 | 0.011 | 0.000 | 0.004 | 0.000 | 0.003 |
| `K-Medoids`(CN3x3) | min | - | - | - | - | 0.980 | 0.538 |
| | max | - | - | - | - | 0.980 | 0.552 |
| | mean | - | - | - | - | **0.980** | **0.544** |
| | std | - | - | - | - | 0.000 | 0.004 |
| `K-Medoids`(TfIdf) | min | - | - | - | - | 0.980 | 0.529 |
| | max | - | - | - | - | 0.980 | 0.537 |
| | mean | - | - | - | - | **0.980** | **0.533** |
| | std | - | - | - | - | 0.000 | 0.003 |
| `K-Medoids`(GPS) | min | - | - | - | - | 0.980 | 0.523 |
| | max | - | - | - | - | 0.980 | 0.530 |
| | mean | - | - | - | - | **0.980** | **0.526** |
| | std | - | - | - | - | 0.000 | 0.002 |
| `K-Medoids`(Tree) | min | 1.000 | 0.528 | - | - | - | - |
| | max | 1.000 | 0.614 | - | - | - | - |
| | mean | **1.000** | **0.577** | - | - | - | - |
| | std | 0.000 | 0.023 | - | - | - | - |

# Chapitre 9

# Comparaison des méthodes de diversité

Dans les chapitres précédents nous avons étudié une méthode du clustering hiérarchique (`AHC`) et deux méthodes du clustering plat (`K-Means` et `K-Medoids`). Les méthodes de clustering plat sont généralement celles cherchent à optimiser le résultat final, contrairement au clustering hiérarchique qui utilise une optimisation locale. De plus `K-Means` est très rapide en temps de calcul, contrairement au clustering hiérarchique quiest plus lent, car il regroupe de manière hiérarchique les images. Dans ces chapitres nous avons trouvé quelques comportements en commun qui sont :

- **Point de convergence de résultats** : les trois techniques de priorité ont convergé de manière exacte (avec la `AHC`) et de manière inexacte (avec `K-Means` et `K-Medoids`) en utilisant toujours un découpage `Fixe` à $n$ clusters pour CR@n et P@n.
- **Valeur maximale de diversité** : ce point de convergence est devenu aussi la valeur maximale pour CR@n. Il semblerait que le bon choix du nombre de clusters (qui dans la littérature est un problème sans réponse unique) est de choisir un nombre `Fixe` de $n$ clusters égal au nombre d'images à évaluer (CR@n). Par exemple, si on souhaite que les 10 premières images soient les plus diversifies possibles, il faut donc choisir un nombre fixe de 10 clusters.
- **Influence des descripteurs** : nous avons vu que le fait d'utiliser une approche par la couleur semble donner des meilleurs résultats en diversité que par le texte ou le GPS. De plus, le comportement de la valeur maximale de diversité devient moins évident quand on utilise des descripteurs moins bons en diversité comme le texte ou le GPS.

Cependant, dans la littérature de la recherche d'images ils existent aussi d'autres méthodes pour augmenter la diversité. De plus, nous n'avons pas trouvé dans cette littérature des travaux qui ont fait une étude approfondie et comparable sur différentes méthodes classiques de diversité. Pour cela, nous allons étudier un algorithme glouton `Maxmin` qui n'utilise pas du clustering avec l'objectif de savoir l'intérêt d'utiliser du clustering par rapport à une autre approche différent de diversité. Comme `Maxmin` n'utilise pas du clustering, il n'a pas le problème du choix du nombre de clusters qui dans la littérature est un problème sans réponse unique. Par contre, `Maxmin` a une étape d'initialisation qui peut faire varier son résultat final. Nous allons comparer les quatre méthodes de diversité sur les mêmes benchmarks et les mêmes descripteurs afin d'obtenir des comportements les plus généraux possibles et de savoir quelle méthode est la plus adaptée dans un cas d'étude différent de diversité.

Enfin, nous étudions aussi le temps de calcul de ces méthodes de diversité afin de savoir quelles méthodes sont les plus adaptés à être utilisables dans une application de

recherche d'images "en ligne". Le nombre d'images est un paramètre très important à étudier sur le temps de calcul, car la complexité des algorithmes sont souvent liés à ce paramètre. La dimension du descripteur peut augmenter aussi le temps de calcul spécialement quand on utilise un algorithme qui a besoin de parcourir la dimension du descripteur (dans le calcul des centroïdes par exemple).

Dans la section 9.1, nous étudions les paramètres optimaux de `Maxmin` sur la diversité. Dans la section 9.2, nous comparons les quatre méthodes classiques de diversité. Enfin, dans la section 9.3 nous comparons son temps de calcul afin de savoir si ces méthodes sont utilisables dans un moteur de recherche d'images "en ligne" (contrainte industrielle).

## 9.1 Étude de la diversité par `Maxmin`

Jusqu'à maintenant nous avons étudié des méthodes classiques du clustering, cependant dans la littérature de recherche d'images ils existent d'autres approches de diversité qui n'utilisent pas du clustering. Dans cette section nous étudions l'algorithme glouton `Maxmin` qui est un algorithme d'optimisation très utilisée et populaire dans l'état de l'art.

L'algorithme `Maxmin` (décrit dans la section 4.2.2 page 45) possède une complexité de type quadratique. Est-ce que son temps de calcul permettra d'utiliser cette méthode dans une application de recherche en ligne ?.

Contrairement aux méthodes de clustering où on doit indiquer toujours le nombre de clusters (qui dans la littérature est un problème sans réponse unique), dans `Maxmin` on n'a pas besoin d'indiquer ce paramètre, car il ne fait pas de regroupement des données.

Cependant, cette méthode a une étape d'initialisation qui est de choisir d'abord une image pour démarrer l'algorithme. Dans cette thèse nous choisissons le choix classique qui est de prendre d'abord l'image avec le rang le plus fort.

`Maxmin` semble être sensible à la présence des images non-pertinentes dans la baseline, car cet algorithme cherche toujours l'image la plus dissimilaire à l'image déjà choisi. En effet, si la première image choisie est une image pertinente, alors c'est très probable que l'image suivante (la plus dissimilaire) soit une image non-pertinente. En prenant en compte cette contrainte, combien d'images on doit prendre en compte dans l'algorithme `Maxmin` afin d'augmenter la diversité sans être trop pénalisée par les images non-pertinentes présentes dans la baseline ?

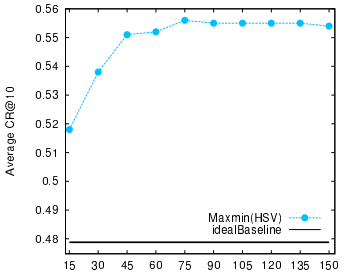
Pour utiliser la méthode `Maxmin` sur la diversité nous avons besoin de divers paramètres comme le type de descripteur, la mesure de similarité et le nombre d'images. Pour être comparable aux expérimentations précédentes, nous étudions cette méthode sur les mêmes benchmarks et les mêmes descripteurs.

**Influence du nombre d'image** Dans cette partie nous comparons la méthode `Maxmin` en utilisant différents nombres d'images sur plusieurs benchmarks et sur plusieurs descripteurs.

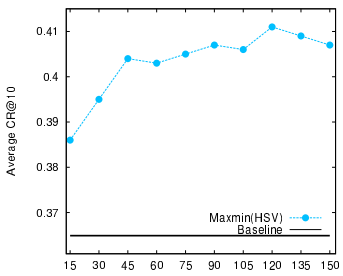
**Comparaison de `Maxmin` sur plusieurs benchmarks** Dans la figure 9.1 nous comparons `Maxmin` avec différents nombres d'images sur plusieurs benchmarks. Les scores CR@10 et P@10 représentent la méthode `Maxmin` avec le descripteur HSV dans un cas idéal et réel.

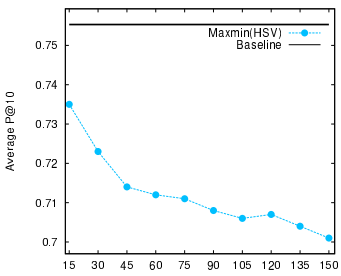
Dans cette figure on peut voir que le fait d'augmenter le nombre d'images, augmente aussi la diversité jusqu'à obtenir une valeur maximale de diversité qui reste stable avec l'augmentation du nombre d'image.

Cependant, la seule exception à ce comportement se trouve dans le cas réel du `Clef` qui a une baseline faible en pertinence (voir figure 9.1(d)). Dans cette figure on voit que

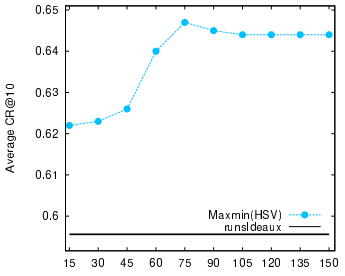



(a) Maxmin(HSV) (`Media`-idéal)

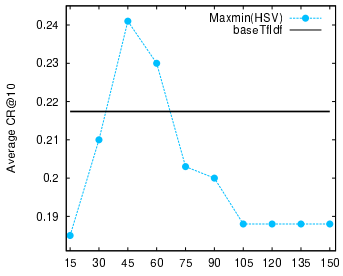


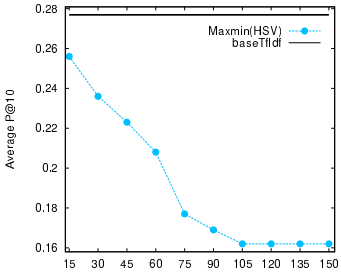




(b) Maxmin(HSV) (`Media`-réel)

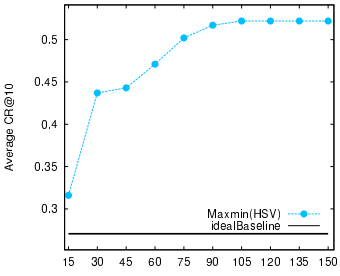



(c) Maxmin(HSV) (`Clef`-idéal)




(d) Maxmin(HSV) (`Clef`-réel)



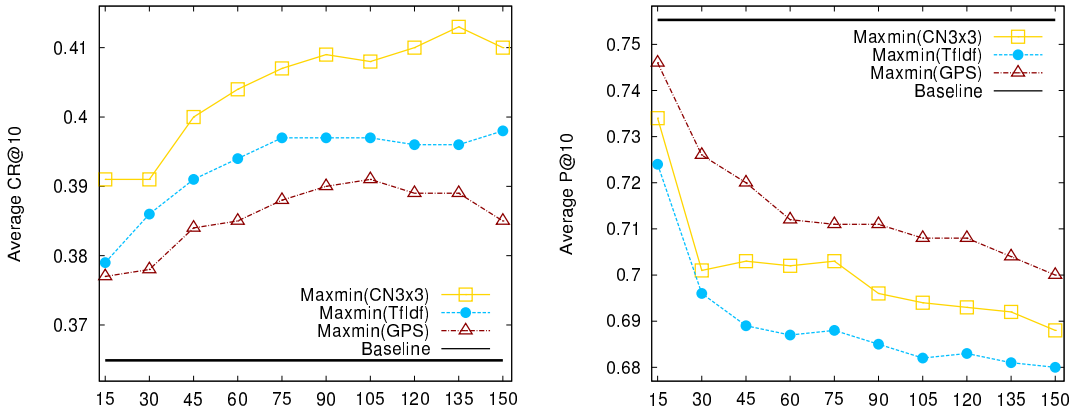

(e) Maxmin(HSV) (`Xilo`-idéal)

FIGURE 9.1 – Comparaison de `Maxmin` avec différents nombres d'images (en abscisse) sur plusieurs benchmarks. Les scores CR@10 et P@10 moyens représentent `Maxmin` avec le descripteur HSV dans un cas idéal et réel sur les benchmarks (a) (b) `Media` (sous-ensemble `testset`), (c) (d) `Clef` et (e) `Xilo`

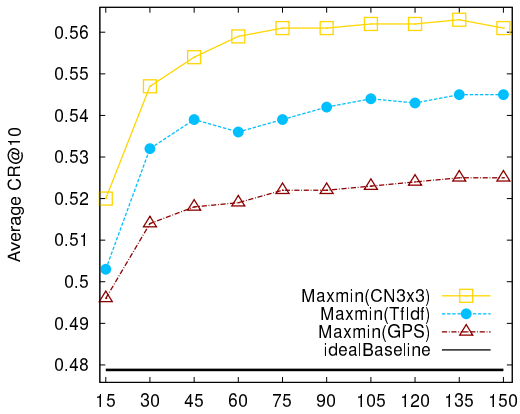
la diversité augmente jusqu'à obtenir la valeur maximale, mais après cette valeur diminue avec l'augmentation du nombre d'images. Comme cette baseline contient beaucoup d'images non pertinentes, il semblerait que `Maxmin` devient trop sensible à cette baseline au point tel de prendre seulement des images non-pertinentes qui sont dissimilaires à celles déjà choisies. Ce qui fait à la fin d'obtenir une liste d'images diversifiées, mais entre les images non-pertinentes présentes dans la baseline.

**Comparaison de `Maxmin` sur plusieurs descripteurs** Dans la figure 9.2 nous comparons `Maxmin` avec différents nombres d'images sur plusieurs descripteurs. Les scores CR@10 et P@10 représentent la méthode `Maxmin` avec les descripteurs CN3x3, TfIdf et GPS sur le benchmark `Media`.

Dans cette figure on voit que, même si nous utilisons `Maxmin` avec différents types de descripteurs, on obtient le même comportement où l'augmentation du nombre d'images, augmente aussi la diversité jusqu'à obtenir une valeur maximale de diversité.




(a) Maxmin (`Media`-réel)




(b) Maxmin (`Media`-idéal)

FIGURE 9.2 – Comparaison de `Maxmin` avec différents nombres d'images (en abscisse) sur plusieurs descripteurs. Les scores CR@10 et P@10 moyens représentent `Maxmin` avec les descripteurs CN3x3, TfIdf et GPS dans un cas idéal et réel sur le benchmark `Media` (sous-ensemble `testset`)

**Discussion** Après avoir compare `Maxmin` avec la variation du nombre d'images sur plusieurs benchmarks et descripteurs, il vaut mieux en général choisir une grande quantité d'images afin d'obtenir un bon score de diversité. Cependant, choisir un nombre d'images assez grande pourrait négliger le temps de calcul de `Maxmin` qui a une complexité de type quadratique, ce qui pourrait faire que cette méthode ne soit pas utilisable dans une application de recherche en ligne.

Nous avons trouvé une seule exception à la variation du nombre d'images avec `Maxmin` dans le cas réel du benchmark `Clef` qui a la particularité d'avoir une baseline avec beaucoup d'images non-pertinentes. Dans cette exception, `Maxmin` devient trop sensible avec la présence des images non-pertinentes de la baseline.

De même que pour les méthodes de clustering, dans `Maxmin` on voit aussi qu'utiliser une approche par la couleur semble être plus intéressant que le texte ou le GPS afin d'augmenter la diversité.

TABLE 9.1 – Paramètres optimaux des méthodes de diversité : `K-Means`, `K-Medoids`, `AHC` et `Maxmin`

| Paramètres | | Clustering plat `K-Means` | Clustering plat `K-Medoids` | Clustering hiérarchique `AHC` | Algorithme Glouton `Maxmin` |
|---|---|---|---|---|---|
| Linkage | | - | - | Average | - |
| Nb clusters | | n clusters pour CR@n | | | - |
| Priorité | | `Rank` | | | - |
| Similarité | Visual | Euclidean | | | |
| | Text | - | Cosinus | | |
| | GPS | - | Haversine | | |
| Nb images | | 150 max | | | |

## 9.2 Comparaison des méthodes de diversité

Nous avons comparé les trois méthodes de clustering en utilisant plusieurs paramètres : différents descripteurs, benchmarks, différentes techniques de priorité de clusters, de découpage de clusters afin d'obtenir des comportements qui soient les plus généraux possibles.

Après avoir étudié de manière indépendante, deux approches différentes de clustering (plat et hiérarchique) et une qui n'utilise pas du clustering (`Maxmin`) sur les mêmes benchmarks et descripteurs nous allons dans cette section comparer ces méthodes afin de savoir laquelle est la plus adaptée pour augmenter la diversité. Dans le tableau 9.1 nous résumons les paramètres optimaux obtenus pour augmenter la diversité sur chaque méthode de diversité.

Pour les méthodes de clustering, nous comparons aussi la courbe de la valeur maximale obtenue dans les section précédentes.

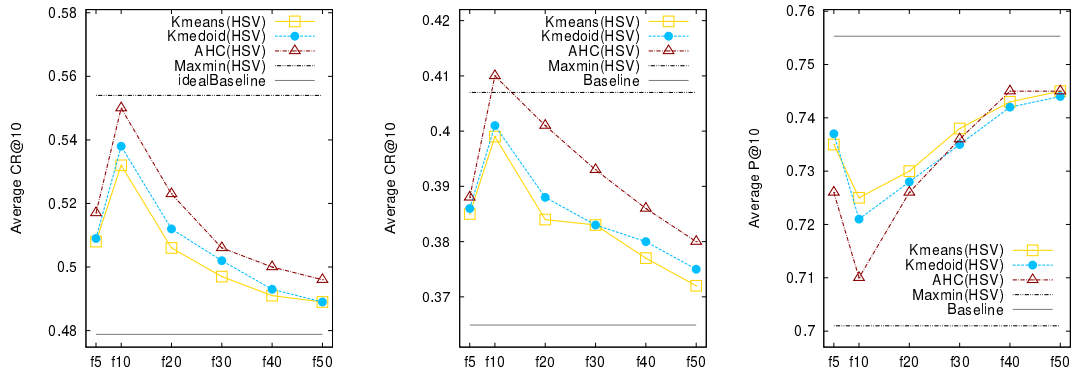
Dans ces expérimentations, nous avons vu que le clustering hiérarchique obtient une meilleure diversité que le clustering plat. Pour expliquer ce comportement, nous étudions d'abord la taille des sous-thèmes de la vérité de terrain et la taille des clusters générés par les méthodes de clustering pour savoir s'il y a une méthode qui se rapproche le plus à la distribution des sous-thèmes de la vérité terrain ce qui pourrait aider cette méthode à générer des meilleurs résultats en diversité.

Par contre, il n'est pas suffisant de se rapprocher à la distribution des sous-thèmes pour obtenir des meilleurs résultats en diversité, car il faut aussi que les clusters générés appartiennent aux mêmes sous-thèmes. Pour cela, nous étudions aussi la qualité des clusters générés qui correspond à savoir si le cluster généré correspond bien à sous-thème de la même taille.

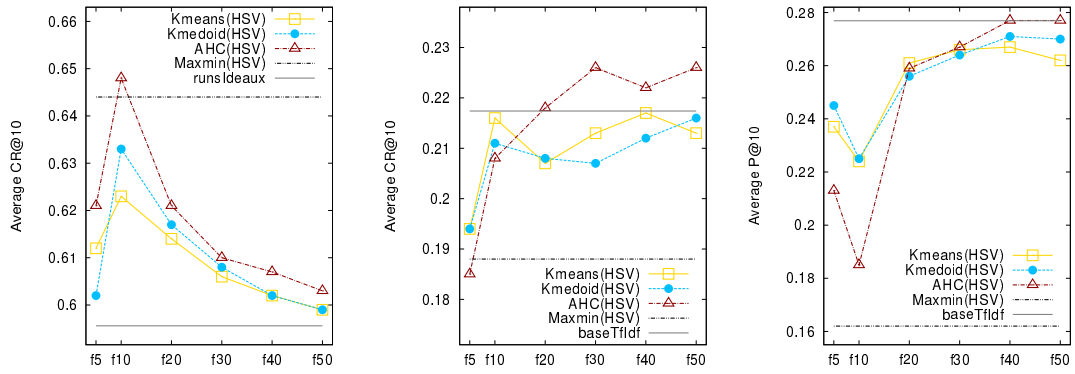
Enfin, nous comparons le temps de calcul des méthodes de diversité afin de savoir si ces méthodes sont utilisables dans une application de recherche en ligne.

### 9.2.1 Influence du nombre de clusters

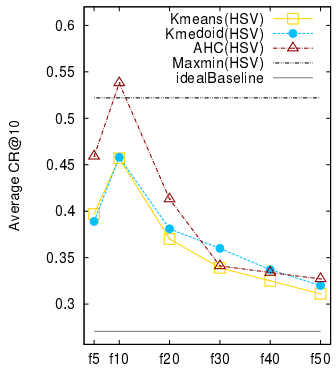
Dans ces expérimentations nous comparons les méthodes de clustering en utilisant différents nombres `Fixe` de $n$ clusters sur plusieurs benchmarks et sur plusieurs descripteurs afin de comparer le comportement de la valeur maximale en diversité de chaque méthode. Il faut remarquer que nous montrons seulement les résultats obtenus avec la priorité classique par `Rank` car les méthodes de clustering ont obtenu des comportements assez similaires en utilisant n'importe quelle technique de priorité.




(a) Clustering(HSV) (`Media`-idéal) (b) Clustering(HSV) (`Media`-réel)



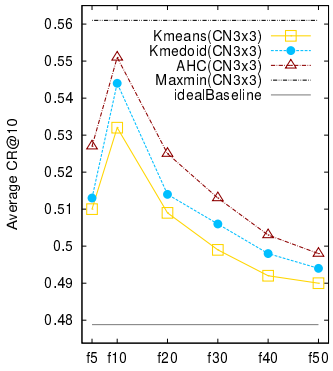

(c) Clustering(HSV) (`Clef`-idéal) (d) Clustering(HSV) (`Clef`-réel)



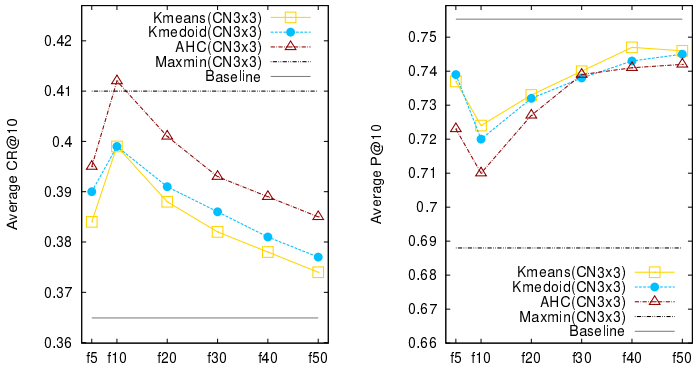

(e) Clustering(HSV) (`Xilo`-idéal)

FIGURE 9.3 – Comparaison des méthodes de diversité sur plusieurs benchmarks. Les scores CR@10 et P@10 moyens représentent les méthodes `K-Means`, `K-Medoids`, `AHC` et `Maxmin` avec le descripteur HSV en utilisant différentes techniques de découpage (en abscisse) et une priorité par `Rank` dans un cas idéal et réel sur les benchmarks (a) (b) `Media` (sous-ensemble `testset`), (c) (d) `Clef` et (e) `Xilo`

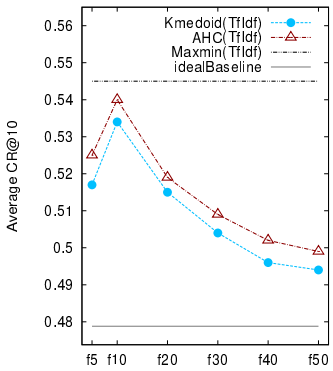
**Comparaison du nombre de clusters sur plusieurs benchmarks** Dans la figure 9.3 nous comparons les trois méthodes de clustering sur plusieurs benchmarks. Les scores CR@10 et P@10 moyens représentent les méthodes de clustering `K-Means`, `K-Medoids` et `AHC` avec le descripteurs HSV en utilisant différentes techniques de décou-



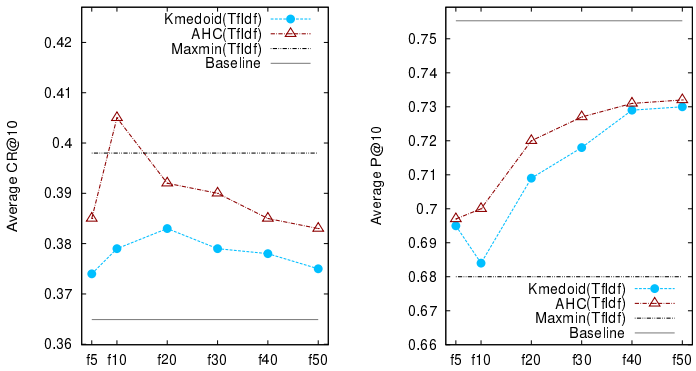

(a) Clustering(CN3x3) (`Media`-idéal)



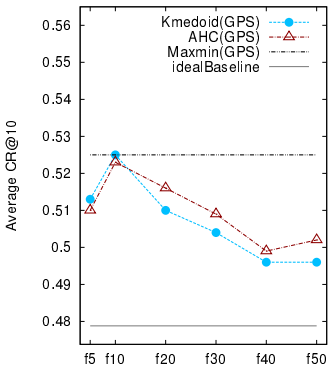

(b) Clustering(CN3x3) (`Media`-réel)



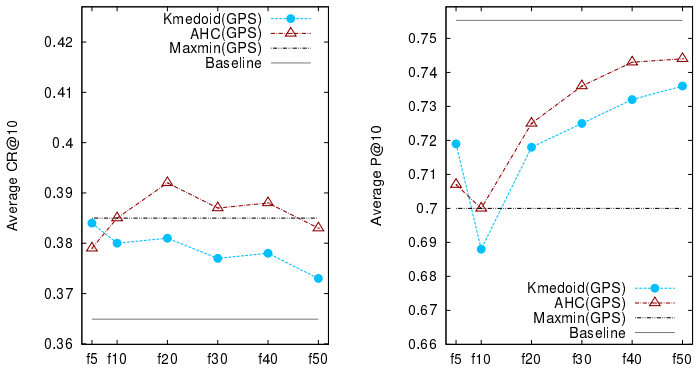

(c) Clustering(TfIdf) (`Media`-idéal)




(d) Clustering(TfIdf) (`Media`-réel)




(e) Clustering(GPS) (`Media`-idéal)




(f) Clustering(GPS) (`Media`-réel)

FIGURE 9.4 – Comparaison des méthodes de clustering sur plusieurs descripteurs. Les scores CR@10 et P@10 moyens représentent les méthodes `K-Means`, `K-Medoids`, `AHC` et `Maxmin` avec les descripteurs (a) (b) CN3x3, (c) (d) TfIdf et (e) (f) GPS en utilisant différentes techniques de découpage (en abscisse) et une priorité par `Rank` dans un cas idéal et réel sur le benchmark `Media` (sous-ensemble `testset`)

page dans un cas idéal et réel.

Dans cette figure on voit que, dans la plupart des cas, la méthode de clustering hiérarchique `AHC` obtient toujours des meilleurs résultats en diversité que les méthodes de clustering plat `K-Means` et `K-Medoids`. Par contre, cette amélioration en diversité

avec la AHC implique aussi une diminution de sa pertinence par rapport au K-Means et K-Medoids.

La seule exception à ce comportement est toujours dans le cas réel du Clef (voir figure 9.3(d)) où les trois méthodes ont des comportements instables à cause de la faible pertinence de sa baseline.

**Comparaison du nombre de clusters sur plusieurs descripteurs** Dans la figure 9.4 nous comparons les trois méthodes de clustering sur plusieurs descripteurs. Les scores CR@10 et P@10 moyens représentent les méthodes de clustering K-Means, K-Medoids et AHC avec les descripteurs CN3x3, TfIdf et GPS en utilisant différentes techniques de découpage dans un cas idéal et réel sur le benchmark Media.

Dans cette figure on voit que lorsqu'on utilise un descripteur de mauvaise qualité en diversité (avec des scores faibles en diversité), la différence entre le clustering hiérarchique et le clustering plat est moins significative. Par exemple si on utilise la couleur on voit que le clustering hiérarchique est meilleur que le clustering plat, par contre si on utilise du GPS les deux méthodes de clustering obtiennent quasiment les mêmes résultats.

Par rapport à la pertinence P@10, on voit que si on utilise la couleur, la pertinence du clustering hiérarchique devient plus faible que le clustering plat. Par contre, on voit le cas contraire quand on utilise des descripteurs de mauvaise qualité en diversité (comme le texte ou le GPS).

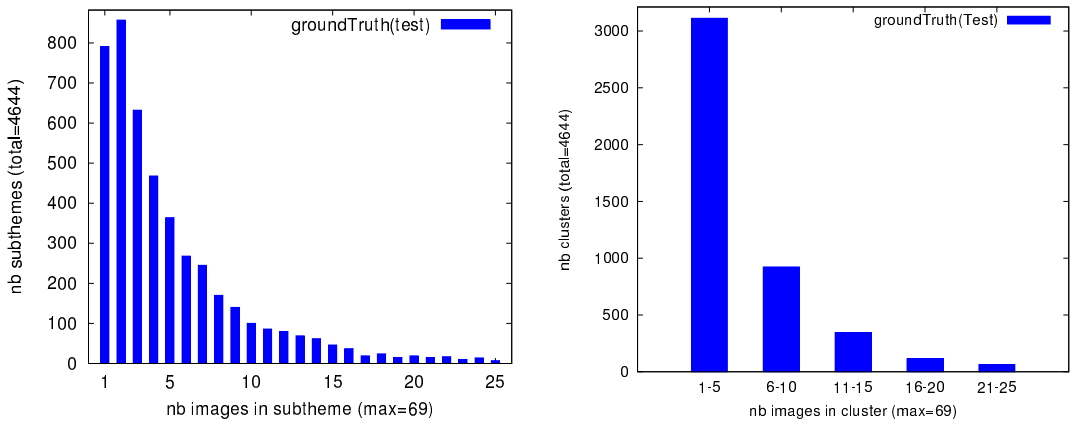
**Discussion** En général, nous avons vu que le fait d'utiliser une approche de clustering hiérarchique (AHC) permet d'obtenir plus de diversité qu'avec une approche de clustering plat (K-Means, K-Medoids). Par contre, cette différence devient moins significative quand on utilise un descripteur de mauvaise qualité en diversité (comme le texte ou le GPS).

Peut-être le clustering hiérarchique est plus adapté à générer des clusters les plus similaires à ceux de la vérité terrain. Il semblerait aussi que le clustering hiérarchique a tendance à générer des clusters de différentes tailles ce qui pourrait aider à mettre les images originales (des images pertinentes qui sont difficiles à retrouver) dans un seul cluster, contrairement au clustering plat qui semble générer des clusters avec des tailles plus homogènes ce qui risque de mélanger dans un cluster des images originales avec des autres types d'images et limiterait la possibilité de prendre cette image originale dans ce cluster.

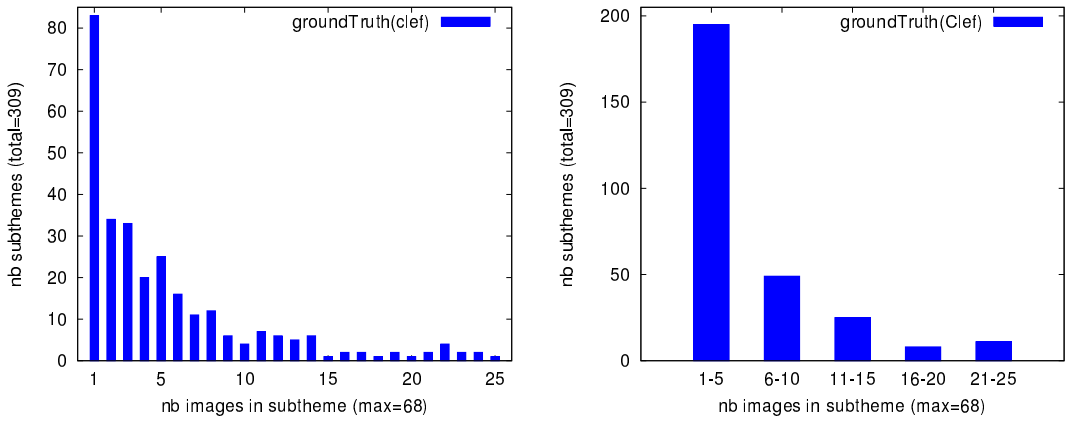
Nous avons fait les mêmes expérimentations avec les scores de CR@20 et P@20 moyens (voir figures A.6 et A.7 dans l'annexe A.3 page 183) et, en général, on voit le même comportement qu'avec les scores CR@10 et P@10 .

### 9.2.2 Étude de la taille des sous-thèmes de la vérité terrain

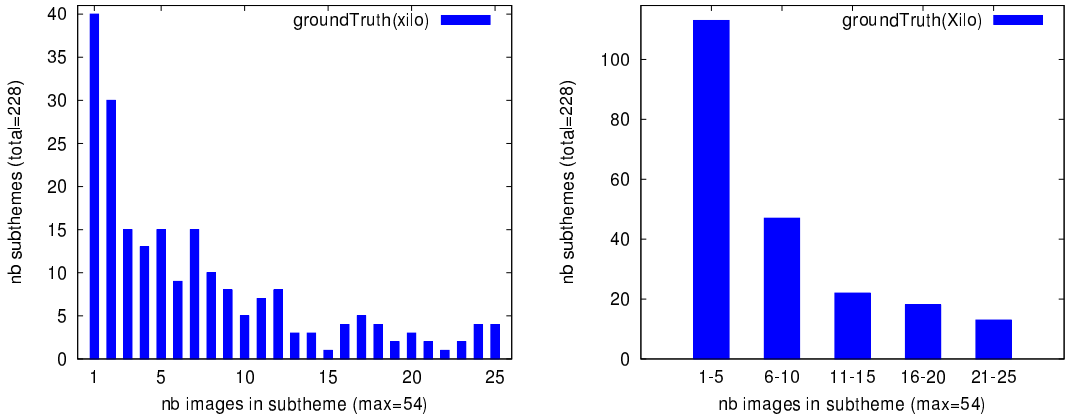
Dans la section précédente nous avons vu que la méthode de clustering hiérarchique AHC donne en général des meilleurs résultats en diversité par rapport aux méthodes de clustering plat K-Means et K-Medoids. En général, une méthode peut obtenir de bons résultats en diversité parce qu'elle arrive bien à détecter la plus grande quantité de sous-thèmes de la vérité terrain. Ces sous-thèmes peuvent être des **petits sous-thèmes** (des sous-thèmes avec peu d'images) qui contiennent souvent des images originales ou rares que l'utilisateur ne s'attend pas à retrouver pour une requête donnée ou des **grands sous-thèmes** (des sous-thèmes avec beaucoup d'images) qui contiennent souvent des images courantes que l'utilisateur s'attend à retrouver pour une requête donnée. Dans cette section, nous étudions la distribution de ces sous-thèmes dans la vérité terrain de




(a) Vérité Terrain du benchmark `Media` (342 requêtes)




(b) Vérité Terrain du benchmark `Clef` (39 requêtes)




(c) Vérité Terrain du benchmark `Xilo` (21 requêtes)

FIGURE 9.5 – Distribution de la taille des sous-thèmes en fonction du nombre d'images par sous-thème sur les benchmarks (a) (b) `Media` (sous-ensemble `testset`), (c) (d) `Clef` et (e) `Xilo`. Pour `Media` le sous-thème qui contient le plus d'images possède 69 images. Nous avons donc fait un histogramme de 69 bins, et dans la figure nous montrons que les 25 premières bins

chaque benchmark afin de savoir la distribution des sous-thèmes et savoir quel type de sous-thème est le plus intéressant à retrouver pour augmenter la diversité.

Dans la figure 9.5 nous montrons la distribution de la taille des sous-thèmes sur les benchmarks `Media` (sous-ensemble `testset`), `Clef` et `Xilo`. Comme la taille maximale des sous-thèmes peut être différent selon chaque benchmark dans ces figures nous montrons que les sous-thèmes qui possèdent jusqu'à 25 images. Par exemple, dans le benchmark `Media` le sous-thème qui contient le plus d'images possède 69 images. Nous avons donc fait un histogramme de 69 bins, et dans la figure nous montrons que les 25 premières bins.

Pour interpréter ces figures, dans l'histogramme à 25 bins de la figure 9.5(a) on peut voir que ils existent environ 800 sous-thèmes qui ont une image dans les 342 requêtes de la vérité terrain de `Media` qui contient un total de 4644 sous-thèmes. Si nous prenons l'histogramme à 5 bins de la même figure on peut voir que ils existent environ 3000 sous-thèmes qui ont entre 1 et 5 images (abrégé en "1-5") dans les mêmes requêtes de `Media`.

Dans cette figure on voit que dans les trois benchmarks sa vérité terrain contient beaucoup de petits sous-thèmes, par exemple les sous-thèmes qui ont entre 1 et 5 images représentent environ la moitie du total des sous-thèmes. Puis, cette quantité commence à diminuer considérablement avec l'augmentation de la taille des sous-thèmes jusqu'à avoir une quantité très faible de grandes sous-thèmes, par exemple les sous-thèmes qui contiennent entre 20 et 25 images représentent environ un 5% du total des sous-thèmes.

D'ici donc l'intérêt d'avoir une méthode qui soit adaptée à retrouver la plus grande quantité de petits sous-thèmes, car ils représentent la moitié des sous-thèmes de la vérité terrain. Par contre, ses petits sous-thèmes sont souvent des images avec des caractéristiques très particulières à la plupart des images de la requête ce qui rend très difficile la tâche de retrouver ces petits sous-thèmes.

### 9.2.3 Comparaison de la taille des clusters

Après avoir étudié la distribution de la taille des sous-thèmes dans la vérité terrain de chaque benchmark, dans cette nouvelle partie nous allons comparer la distribution de la taille des clusters générés par les méthodes de clustering hiérarchique et plat (en fonction du nombre d'images) afin de savoir si ces méthodes permettent de générer une distribution de taille des clusters similaire à ceux de la vérité terrain.

Pour commencer cette étude, dans le tableau 9.2 on peut voir la taille des clusters générés par les méthodes de clustering `K-Means`, `K-Medoids` et `AHC` avec le descripteur HSV en utilisant comme paramètre un découpage `Fixe` à $k$ clusters (10, 20 et 30 clusters) et une priorité par `Rank` dans un cas réel sur le benchmark `Media`. Il faut remarquer qu'en utilisant une valeur $K$ plus grande, les méthodes vont pouvoir générer plus de clusters. Dans ce tableau on voit principalement que celle hiérarchique `AHC` génère beaucoup plus de petits clusters qui représentent déjà plus de la moitié du total de clusters générés par elle par rapport aux méthodes de clustering plat qui semblent générer des clusters plus homogènes.

Pour approfondir cette étude, dans la figure 9.6 nous comparons le taux de la distribution des sous-thèmes de la vérité terrain avec le taux de la distribution des méthodes de clustering avec les mêmes paramètres du tableau 9.2. L'objectif de cette figure c'est de voir mieux si la distribution d'une méthode de clustering s'approche à la distribution des sous-thèmes de la vérité terrain. Dans cette figure on peut voir que pour $K = 10$ la distribution du clustering hiérarchique s'approche le mieux à la distribution de la vérité terrain. Par contre, pour $K = 20, 30$ la distribution du clustering plat s'approche un peu plus à la distribution de la vérité terrain qu'avec celui hiérarchique.

Afin de confirmer ces résultats, dans le tableau 9.3 nous calculons l'intersection de histogramme $inter_{histo}(VT, cluster(k))$ (décrite dans la section 2.3.3 page 23) où $VT$ représente l'histogramme du taux de la distribution des sous-thèmes de la vérité terrain

TABLE 9.2 – Comparaison de la taille des clusters générés par les méthodes `K-Means`, `K-Medoids` et `AHC` avec le descripteur HSV en utilisant différents $K$ clusters dans un cas réel sur le benchmark `Media` (sous-ensemble `testset`). Entre parenthèse nous avons le taux en pourcentage du type de cluster généré par rapport au total des clusters générés par la méthode de clustering

| | CLUSTERS GÉNÉRÉS (avec $K = 10$) (en fonction du nb d'images) | | | | | | |
|---|---|---|---|---|---|---|---|
| Méthode | total | 1-5 | 6-10 | 11-15 | 16-20 | 21-25 | 26+ |
| `AHC` | 3420 | 2060(60%) | 479(14%) | 216(6%) | 147(4%) | 107(3%) | 411(12%) |
| `K-Means` | 3420 | 998(29%) | 903(26%) | 666(19%) | 402(12%) | 244(7%) | 205(6%) |
| `K-Medoids` | 3420 | 1043(30%) | 872(25%) | 610(18%) | 424(12%) | 245(7%) | 226(7%) |

| | CLUSTERS GÉNÉRÉS (avec $K = 20$) (en fonction du nb d'images) | | | | | | |
|---|---|---|---|---|---|---|---|
| Méthode | total | 1-5 | 6-10 | 11-15 | 16-20 | 21-25 | 26+ |
| `AHC` | 6840 | 5071 (74%) | 819 (12%) | 344 (5%) | 204 (3%) | 125 (2%) | 277 (4%) |
| `K-Means` | 6840 | 4019 (59%) | 1958 (29%) | 662 (10%) | 157 (2%) | 33 (0%) | 5 (0%) |
| `K-Medoids` | 6840 | 4099 (60%) | 1794 (26%) | 696 (10%) | 208 (3%) | 35 (1%) | 8 (0%) |

| | CLUSTERS GÉNÉRÉS (avec $K = 30$) (en fonction du nb d'images) | | | | | | |
|---|---|---|---|---|---|---|---|
| Méthode | total | 1-5 | 6-10 | 11-15 | 16-20 | 21-25 | 26+ |
| `AHC` | 10260 | 8452 (82%) | 1012 (10%) | 400 (4%) | 167 (2%) | 119 (1%) | 110 (1%) |
| `K-Means` | 10260 | 7985 (78%) | 1947 (19%) | 295 (3%) | 20 (0%) | 2 (0%) | 0 (0%) |
| `K-Medoids` | 10260 | 7914 (77%) | 1951 (19%) | 344 (3%) | 46 (0%) | 5 (0%) | 0 (0%) |

TABLE 9.3 – Comparaison des scores $inter_{histo}(VT, cluster(k))$ entre l'histogramme des taux de la distribution des sous-thèmes de la vérité terrain $VT$ et les histogrammes des taux de la distribution des clusters générés par les méthodes de clustering $cluster(k)$ avec HSV en utilisant $k$ clusters dans un cas réel sur le benchmark `Media` (sous-ensemble `testset`). Plus la valeur se rapproche de 1, plus les deux histogrammes sont similaires

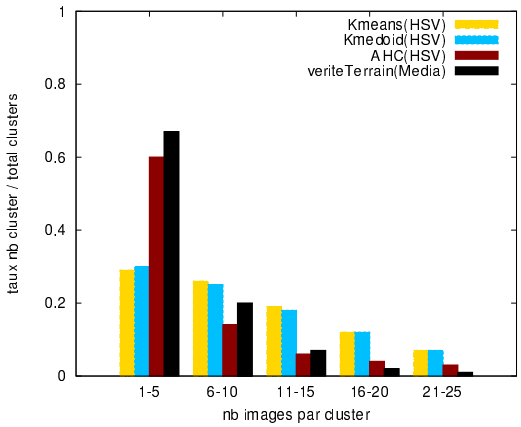
| | $inter_{histo}(VT, clust(k))$ | | |
|---|---|---|---|
| clust(k) | $k = 10$ | $k = 20$ | $k = 30$ |
| `AHC` | 0.86 | 0.89 | 0.85 |
| `K-Means` | 0.62 | 0.88 | 0.89 |
| `K-Medoids` | 0.63 | 0.9 | 0.9 |

et $cluster(k)$ représente l'histogramme du taux de la distribution des méthodes de clustering. Dans cette mesure, plus la valeur se rapproche de 1, plus les deux histogrammes sont similaires.

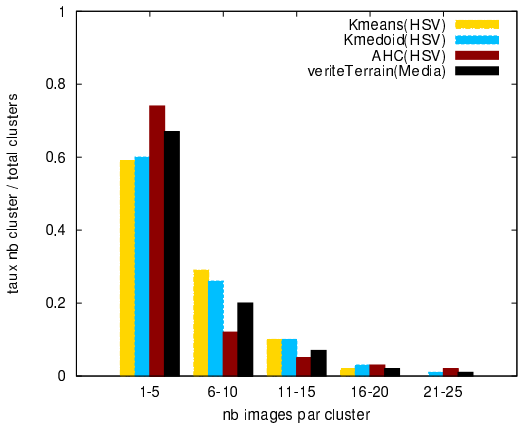
Dans cette figure nous confirmons les comportements obtenus où pour $K = 10$ la distribution de la `AHC` se rapproche le plus à la vérité terrain (avec une valeur de similarité de 0.86), et pour $K = 20, 30$ le clustering plat se rapproche de plus en plus à la distribution de la vérité terrain (avec une valeur de similarité de 0.9).

Nous avons fait les mêmes expérimentations sur plusieurs benchmarks et sur plusieurs descripteurs pour $K = 10$ (voir figures A.11 et A.12 dans l'annexe A.4 page 190). Pour $K = 10$, on peut voir que l'utilisation d'un descripteur de mauvaise qualité en diversité (comme le texte ou le GPS) et l'utilisation d'un cas idéal permet au clustering plat de générer des clusters moins homogènes et de se rapprocher mieux à la distribution de la vérité terrain.

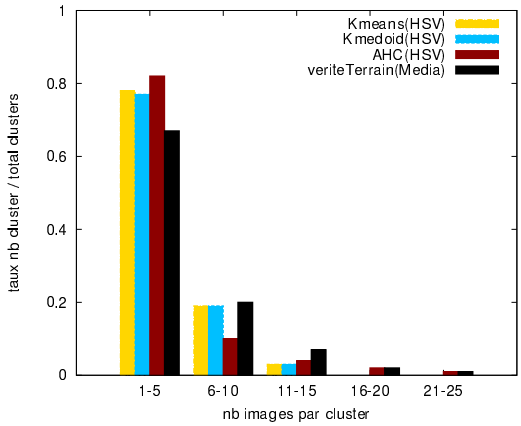
**Discussion** Nous avons vu, en général, que lorsque nous utilisons un nombre de clusters $K = 10$ le clustering hiérarchique génère une distribution plus similaire à la distribution de sous-thèmes de la vérité terrain, tandis que lorsque nous utilisons un




(a) Taille des clusters générés avec $K = 10$




(b) Taille des clusters générés avec $K = 20$




(c) Taille des clusters générés avec $K = 30$

FIGURE 9.6 – Comparaison du taux de la distribution des sous-thèmes de la vérité terrain avec le taux de la distribution des clusters générés par les méthodes `K-Means`, `K-Medoids` et `AHC` avec le descripteur HSV en fonction du nombre d'images par cluster en utilisant différents découpages `Fixe` (a) à 10 clusters, (b) à 20 clusters et (c) à 30 clusters dans un cas réel sur le benchmark `Media` (sous-ensemble `testset`)

nombre de cluster plus grand $K = 20, 30$ le clustering plat commence à se rapprocher de plus en plus de la distribution de sous-thèmes de la vérité terrain.

Nous avons vu aussi que lorsqu'on utilise un descripteur de mauvaise qualité (comme le texte ou le GPS) et que l'on est dans un cas idéal, le clustering plat se rapproche de plus en plus à la distribution de sous-thèmes de la vérité terrain.

### 9.2.4 Étude de la qualité des résultats obtenus

Dans la partie précédente nous avons étudié la distribution de la taille clusters générés afin de savoir la méthode qui se rapproche le plus à la distribution de la taille des sous-thèmes de la vérité terrain. Cependant, il ne suffit pas de se rapprocher de la distribution des sous-thèmes de la vérité terrain pour conclure qu'une méthode donne des meilleures résultats en diversité car on ne sait pas si les clusters générés par cette méthode ont bien contribué à augmenter la diversité.

**Algorithme** Pour étudier plus en détail l'apport en diversité des méthodes étudiées, dans cette partie nous comparons les sous-thèmes retrouvés dans les $n$ premiers

**Algorithm 1** Algorithme pour calculer l'apport des types de sous-thèmes retrouvés dans les $n$ premiers résultats de chaque méthode de diversité.

**Paramètre :** $n \leftarrow$ nombre d'images à évaluer
**Paramètre :** $q \leftarrow$ requête à évaluer
**pour** chaque requête $q$ **faire**
  $result \leftarrow []$
  $bruit \leftarrow redondant \leftarrow utile \leftarrow 0$

  on récupère les $n$ premiers résultats
  **pour** chaque image $i \in n$ **faire**
    on identifie le sous-thème $ST_i$ retrouvé avec l'image $i$
    **si** $ST_i == nil$ **alors**
      $bruit \leftarrow bruit + 1$
    **sinon**
      **si** $i$ correspond à un sous-thème $ST_i$ déjà vu **alors**
        $redondant \leftarrow redondant + 1$
      **sinon**
        $utile \leftarrow utile + 1$
      **fin si**
    **fin si**
  **fin pour**
**fin pour**

résultats de chaque méthode de diversité. Pour cela, dans les $n$ premiers résultats, nous calculons le nombre d'images :

- qui appartiennent aux sous-thèmes différents qui vont augmenter la diversité. Nous appelons ces sous-thèmes de type **"utiles"** ;
- qui appartiennent aux sous-thèmes déjà vus qui ne vont ni augmenter ni diminuer la diversité. Nous appelons ces sous-thèmes de type **"redondants"** ;
- non-pertinentes qui vont pénaliser la diversité. Nous appelons ces sous-thèmes de type **"bruit"**.

L'algorithme 1 montre plus en détail les étapes suivies afin de calculer l'apport en diversité des types de sous-thèmes retrouvés dans les $n$ premières images résultats de chaque méthode de diversité.

**Types de sous-thèmes retrouvés** Dans le tableau 9.4 nous montrons les types de sous-thèmes retrouvés dans les 10 premiers images résultats pour chaque méthode de diversité avec le descripteur HSV en utilisant $K$ clusters (10, 20 et 30 clusters), priorité par `Rank` sur le benchmark `Media` (sous-ensemble `testset`). Dans ce tableau on voit qu'en général, dans les 10 premières images les méthodes de diversité arrivent à retrouver environ 50% d'images qui appartiennent aux sous-thèmes différents et qui vont augmenter la diversité, environ 20% d'images qui sont des sous-thèmes redondants, et environ 30% d'images qui sont des images non-pertinentes. Cependant, on voit aussi que `Maxmin` retrouve plus d'images non pertinentes que les méthodes de clustering, tandis que les méthodes de clustering retrouvent plus des sous-thèmes redondants que `Maxmin`.

En sachant que dans les 10 premiers résultats la valeur maximale de diversité pour les méthodes de clustering est obtenue avec $K = 10$, si on choisi une valeur $K$ différente à 10, le nombre d'images non-pertinentes diminue, mais le nombre de sous-thèmes redondants augmente.

Nous avons fait les mêmes expérimentations en analysant les 20 premières images

TABLE 9.4 – Apport de sous-thèmes retrouvés dans les 10 premiers résultats de chaque méthode de diversité sur le benchmark `Media`-réel (sous-ensemble `testset`)

| $K$ | Méthode | sous-thèmes retrouvés total | utile | redondant | bruit |
|---|---|---|---|---|---|
| - | Baseline | 3420 | 1601(47%) | 982(29%) | 837(25%) |
| - | Maxmin(HSV) | 3420 | **1800(53%)** | **596(17%)** | **1024(30%)** |
| 10 | AHC(HSV) | 3420 | **1803(53%)** | 624(18%) | 993(29%) |
| | Kmeans(HSV) | 3420 | 1772(52%) | 706(21%) | 942(28%) |
| | Kmedoid(HSV) | 3420 | 1797(53%) | 671(20%) | 952(28%) |
| 20 | AHC(HSV) | 3420 | 1778(52%) | 705(21%) | 937(27%) |
| | Kmeans(HSV) | 3420 | 1694(50%) | 809(24%) | 917(27%) |
| | Kmedoid(HSV) | 3420 | 1722(50%) | 762(22%) | 936(27%) |
| 30 | AHC(HSV) | 3420 | 1748(51%) | 769(23%) | 903(26%) |
| | Kmeans(HSV) | 3420 | 1695(50%) | 840(25%) | 885(26%) |
| | Kmedoid(HSV) | 3420 | 1697(50%) | 810(24%) | 913(27%) |

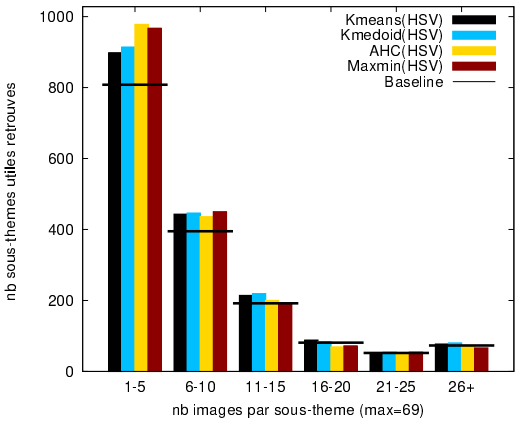



FIGURE 9.7 – Apport des sous-thèmes utiles retrouvés dans les 10 premiers résultats des méthodes de diversité (`K-Means`, `K-Medoids`, `AHC` et `Maxmin`) avec le descripteur HSV en utilisant $K = 10$ et une priorité par `Rank` sur benchmark `Media` (sous-ensemble `testset`)

résultats (voir tableau A.7 dans l'annexe A.5 page 193) où on voit, en général, que le pourcentage des sous-thèmes redondants augmente (environ 30%) contre une diminution de ceux utiles (environ 40%). Dans ce nouveau tableau, il semblerait que `Maxmin` arrive à retrouver plus de bruit et moins des sous-thèmes redondants que les méthodes de clustering. Il semblerait aussi que celles-ci avec une valeur $K$ différent à celui de la valeur maximale de diversité, sont pénalisées par la quantité des sous-thèmes redondants qui se trouvent dans les $n$ premiers résultats.

**Type de sous-thèmes utiles retrouvés** Dans la figure 9.7 nous montrons plus en détail l'apport des sous-thèmes utiles retrouvés dans les 10 premiers résultats des méthodes de diversité avec le descripteur HSV en utilisant $K = 10$, une priorité par `Rank` dans le cas réel du benchmark `Media` (sous-ensemble `testset`). Dans cette figure on voit que les méthodes de diversité sembleraient obtenir des meilleurs résultats en diversité que la baseline, car ils retrouvent plus de sous-thèmes avec un nombre petit et moyen d'images (sous-thèmes de type '1-5' et '6-10').

Par rapport aux méthodes de diversité, on voit aussi que `Maxmin` et `AHC` arrivent à retrouver plus de sous-thèmes qui contiennent peu d'images (sous-thèmes de type '1-5')

par rapport aux méthodes de clustering plat.

**Type de sous-thèmes utiles retrouvés sur plusieurs benchmarks** Nous faisons les mêmes expérimentations, mais en utilisant différents benchmarks. Pour cela,la figure 9.8 montre l'apport des sous-thèmes utiles retrouvés dans les 10 premiers résultats dans les méthodes de diversité avec le descripteur HSV en utilisant $K = 10$ et une priorité par `Rank` dans un cas idéal et réel. Dans cette figure on voit qu'en général, `Maxmin` et `AHC` retrouvent plus de sous-thèmes qui contiennent peu d'images, dans les sous-thèmes de type '1-5' pour `Media`, dans les sous-thèmes de type '1-5' et '6-10' pour `Xilo` par rapport aux méthodes de clustering plat. Par contre, nous avons la seule exception à ce comportement dans le cas réel du benchmark `Clef` (voir figure 9.8(d)) qui a la particularité d'avoir une baseline faible en pertinence. Dans cette figure on voit que les méthodes de diversité retrouvent moins de sous-thèmes utiles que la baseline.

**Type de sous-thèmes utiles retrouvés sur plusieurs descripteurs** Nous faisons les mêmes expérimentations, mais en utilisant différents types de descripteurs. Pour cela, la figure 9.9 montre l'apport des sous-thèmes utiles retrouvés dans les 10 premiers résultats des méthodes de diversité avec les descripteurs CN3x3, TfIdf et GPS en utilisant $K = 10$ et une priorité par `Rank` dans un cas idéal et réel sur le benchmark `Media` (sous-ensemble `testset`). Dans cette figure on observe le même comportement avec le descripteur visuel CN3x3, de la même famille que HSV. Cependant, quand on utilise des mauvais descripteur en diversité (comme TfIdf ou GPS) la différence des sous-thèmes retrouvés est moins significative.

**Discussion** Dans cette partie nous avons étudié la qualité du résultat obtenu par les méthodes de diversité. Nous mesurons cette qualité par l'apport de sous-thèmes retrouvés dans les $n$ premières images des méthodes de diversité. Une image peut appartenir soit à un sous-thème différent (sous-thème utile) ou déjà vu (sous-thème redondant) soit ne pas être pertinente (sous-thème bruit).

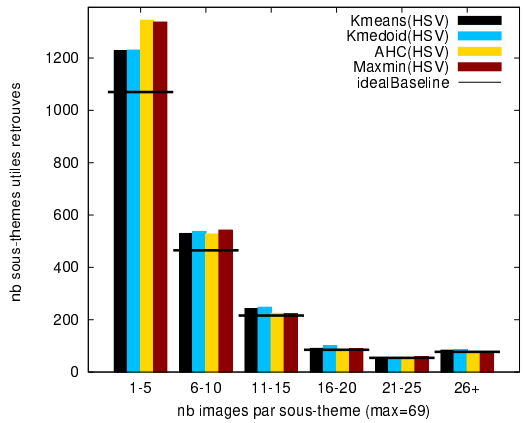
En étudiant l'apport des sous-thèmes dans les 10 premières images des méthodes de diversité, nous avons vu qu'en général, les méthodes de diversité arrivent à retrouver environ 50% d'images qui appartiennent aux sous-thèmes différents. On voit aussi que `Maxmin` retrouve plus d'images non pertinentes par rapport aux méthodes de clustering tandis que les méthodes de clustering retrouvent plus des sous-thèmes redondants que l'utilisateur a déjà vu par rapport au `Maxmin`.

En étudiant plus en détail l'apport des sous-thèmes utiles dans les 10 premiers résultats dans les méthodes de diversité, nous avons vu que les méthodes de diversité sembleraient obtenir des meilleurs résultats en diversité que la baseline, car ces méthodes retrouvent plus de sous-thèmes avec un nombre petit et moyen d'images (sous-thèmes de type '1-5' et '6-10').

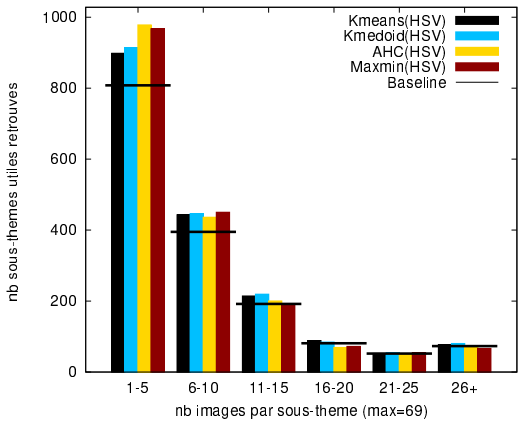
Par rapport aux méthodes de diversité, on voit aussi que `Maxmin` et `AHC` arrivent à retrouver plus de sous-thèmes qui contiennent peu d'images (sous-thèmes de type '1-5') par rapport aux méthodes de clustering plat

Il faut dire aussi que ce comportement devient un peu moins évident quand on utilise un descripteur de mauvaise qualité en diversité. Il semblerait donc qu'utiliser une approche par la couleur pourrait être intéressante pour retrouver plus de sous-thèmes avec peu d'images.

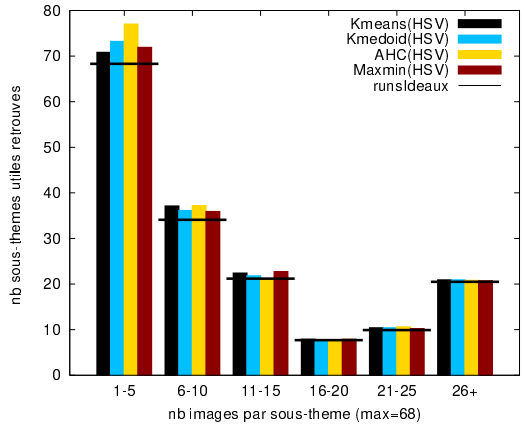
Nous avons fait les mêmes expérimentations, en étudiant l'apport des sous-thèmes utiles dans les 20 premiers résultats pour les méthodes de clustering avec 20 clusters (voir figures A.13 et A.14 dans l'annexe A.5 page 193) et, en général, on retrouve le même comportement qu'avec l'étude des sous-thèmes retrouvés dans les 10 premiers résultats.



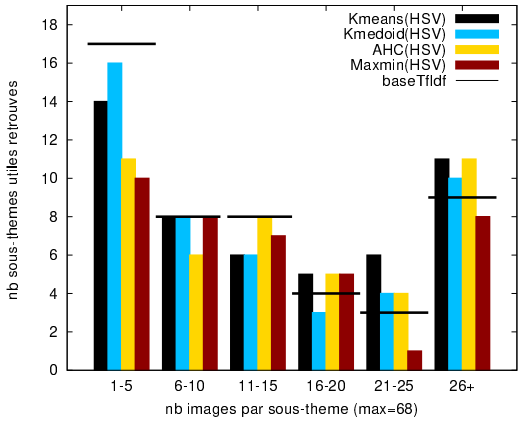

(a) Clustering(HSV) (`Media`-idéal)



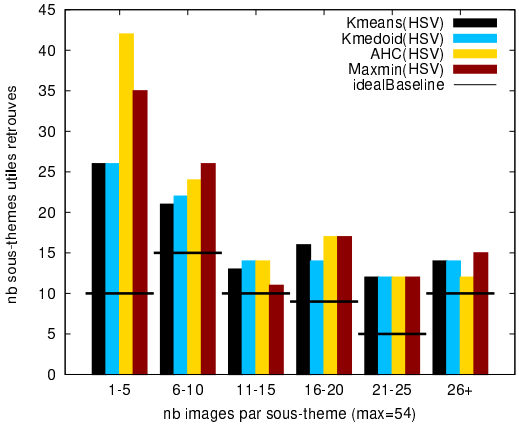

(b) Clustering(HSV) (`Media`-réel)




(c) Clustering(HSV) (`Clef`-idéal)

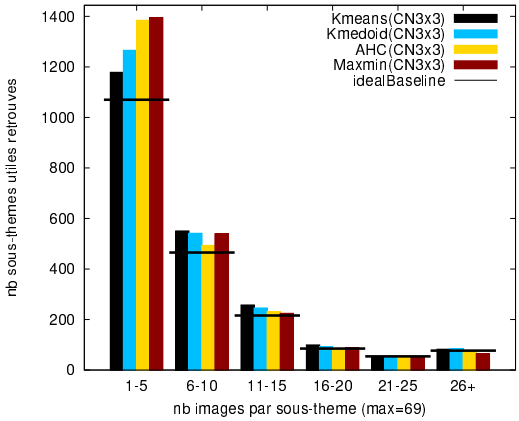



(d) Clustering(HSV) (`Clef`-réel)

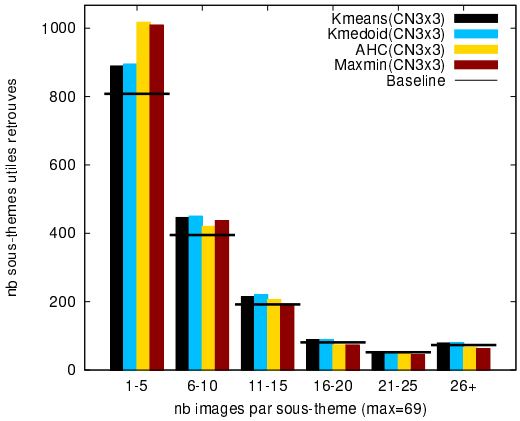



(e) Clustering(HSV) (`Xilo`-idéal)

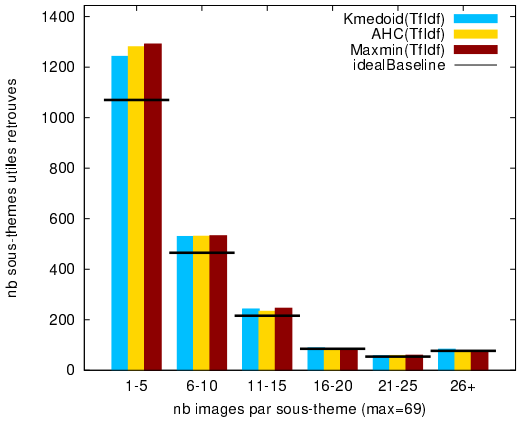
FIGURE 9.8 – Apport des sous-thèmes utiles retrouvés dans les 10 premiers résultats des méthodes `K-Means`, `K-Medoids`, `AHC` et `Maxmin` avec le descripteur HSV en fonction du nombre d'images par cluster en utilisant un découpage `Fixe` à 10 clusters et une priorité par `Rank` dans un cas idéal et réel sur les benchmarks (a) (b) `Media` (sous-ensemble `testset`), (c) (d) `Clef` et (e) `Xilo`



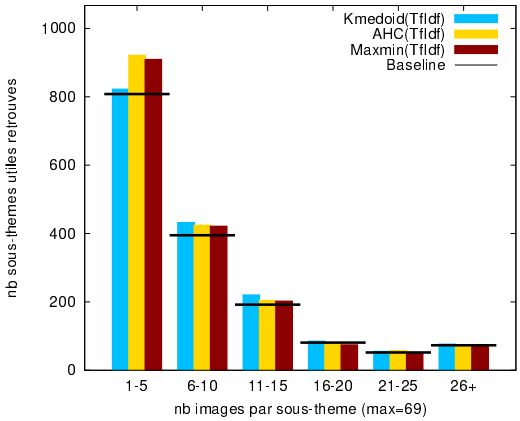

(a) Clustering(CN3x3) (`Media`-idéal)



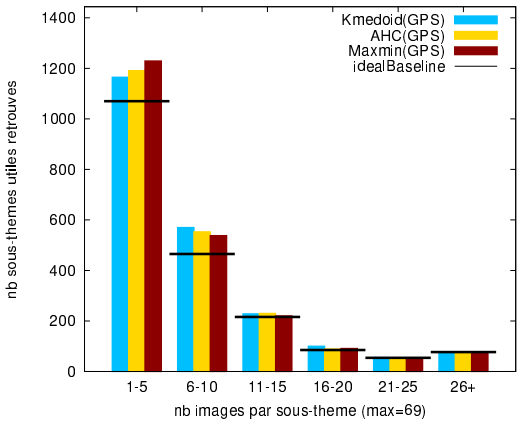

(b) Clustering(CN3x3) (`Media`-réel)



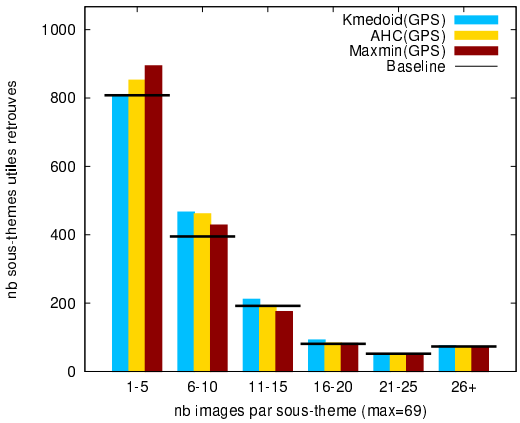

(c) Clustering(TfIdf) (`Media`-idéal)




(d) Clustering(TfIdf) (`Media`-réel)




(e) Clustering(GPS) (`Media`-idéal)




(f) Clustering(GPS) (`Media`-réel)

FIGURE 9.9 – Apport des sous-thèmes utiles retrouvés dans les 10 premiers résultats des méthodes `K-Means`, `K-Medoids`, `AHC` et `Maxmin` avec les descripteurs (a) (b) CN3x3, (c) (d) TfIdf et (e) (f) GPS en utilisant un découpage `Fixe` à 10 clusters et une priorité par `Rank` dans un cas idéal et réel sur le benchmark `Media` (sous-ensemble `testset`)

### 9.2.5 Comparaison de la valeur maximale de diversité

Jusqu'à maintenant, nous avons étudié quatre méthodes de diversité : deux de clustering plat (`K-Means` et `K-Medoids`), une de clustering hiérarchique (`AHC`) et une qui n'utilise

pas du clustering (`Maxmin`). Dans ces méthodes nous avons vu un comportement en commun où le fait d'utiliser une approche par la couleur ou la sémantique semblerait être plus intéressante pour augmenter la diversité que le fait d'utiliser une approche par le texte ou par le GPS.

Dans cette partie, nous allons comparer la valeur maximale de diversité des différentes méthodes de diversité étudiées. Pour cela, dans le tableau 9.1 nous montrons les paramètres optimaux de diversité de chaque méthode.

#### 9.2.5.1 Comparaison des scores en général

Dans la figure 9.10 nous comparons les méthodes de diversité sur plusieurs benchmarks. Les scores CR@10 et P@10 représentent les méthodes de diversité `K-Means`, `K-Medoids`, `AHC` et `Maxmin` avec le descripteur HSV dans un cas idéal et réel. Dans cette figure on voit qu'en général, `Maxmin` et la `AHC` obtiennent les meilleurs résultats en diversité, malgré la différence entre les deux méthodes, car ils utilisent des approches différentes pour augmenter la diversité.

Dans la figure 9.11 nous comparons les méthodes de diversité sur plusieurs descripteurs. Les scores CR@10 et P@10 représentent les méthodes de diversité `K-Means`, `K-Medoids`, `AHC` et `Maxmin` avec les descripteurs CN3x3, TfIdf et GPS dans un cas idéal et réel sur le benchmark `Media`. Dans cette figure on voit que dans un cas idéal `Maxmin` semble être le plus intéressant pour augmenter la diversité tandis que dans un cas réel la `AHC` devient plus intéressante pour augmenter la diversité. Il semblerait que `Maxmin` est plus sensible aux images non-pertinentes, car dans un cas réel, l'image la plus dissimilaire à la première image déjà choisi peut être une image non-pertinente.

Nous avons fait les mêmes expérimentations avec les scores de CR@20 et P@20 moyens (voir figures A.8 et A.9 dans l'annexe A.3 page 183) où nous confirmons que `Maxmin` et la `AHC` obtiennent aussi les meilleurs résultats en diversité que les méthodes de clustering plat.

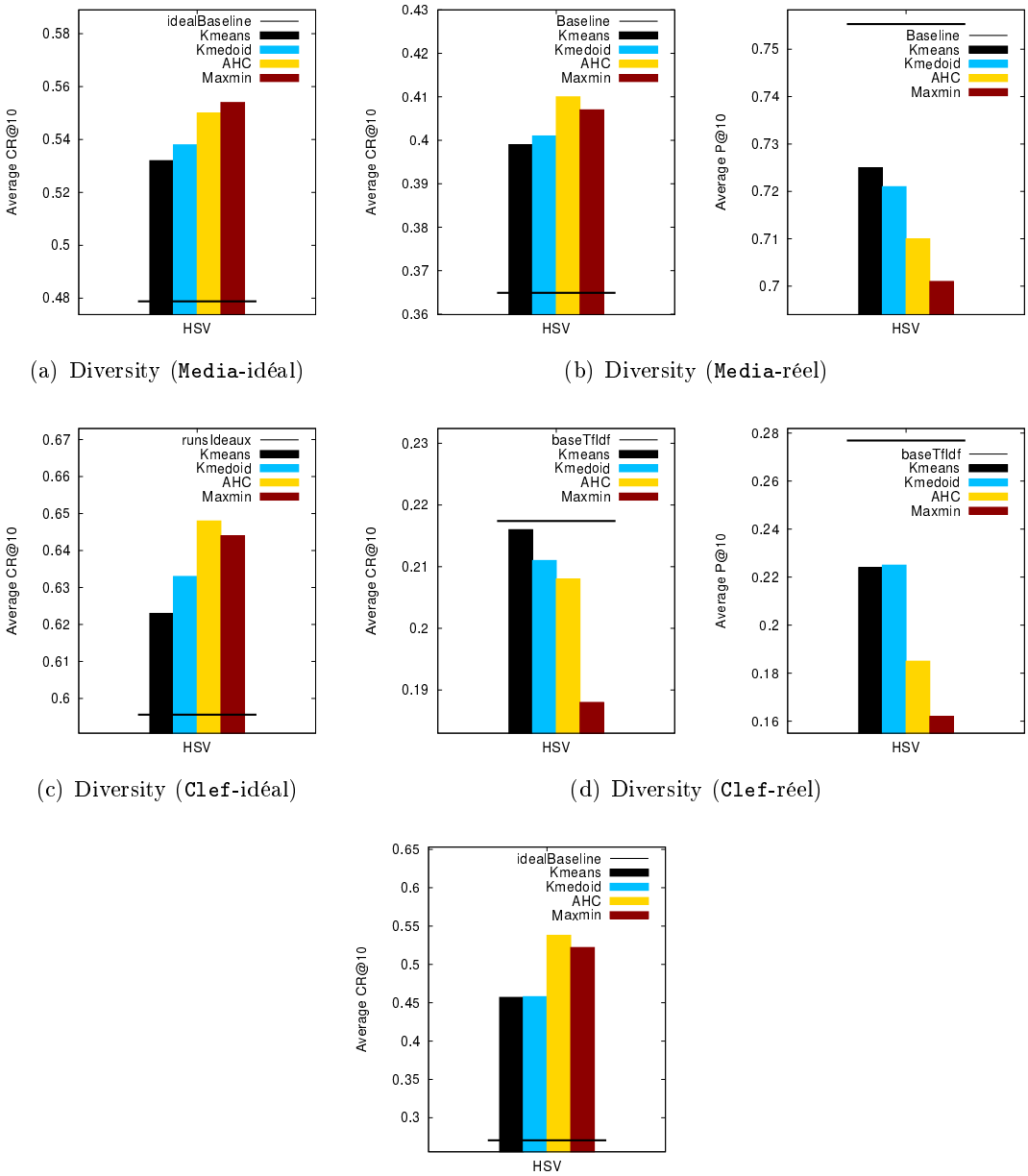
#### 9.2.5.2 Comparaison des scores par requête

Dans les expériences précédentes, nous avons vu que les deux méthodes `Maxmin` et la `AHC` donnent les meilleurs résultats en diversité. Afin d'étudier plus en détail ces méthodes de diversité, nous comparons les scores obtenus par requête. Pour cela, nous prenons seulement le sous-ensemble `testset` du benchmark `Media` car il contient beaucoup de requêtes.

Dans la figure 9.12 nous comparons les scores par requêtes entre "`Maxmin` versus baseline" et "`AHC` versus baseline". En ordonnée, les scores CR@10 et P@10 par requête représentent les méthodes de diversité `AHC` et `Maxmin` avec le descripteur HSV dans un cas idéal et réel sur le benchmark `Media`. Dans ces figures, les points représentent les scores de chaque requête. Les scores des méthodes de diversité (en ordonnée) et les scores de la baseline (en abscisse) sont séparés par une diagonale. Si le point est au-dessus de la diagonale, cela veut dire que la méthode de diversité obtient un meilleur score que la baseline. Dans le cas contraire, la baseline obtient un meilleur score que la méthode de diversité. Si le point est dans la diagonale, cela veut dire que la baseline obtient le même score que la méthode de diversité.

Dans la comparaison des deux méthodes de diversité contre la baseline on voit de manière générale que :

— pour environ 50% des requêtes dans le cas réel et environ 60% des requêtes dans le cas idéal, les méthodes de diversité ont réussi à augmenter la diversité par rapport à la baseline




(a) Diversity (`Media`-idéal)

(b) Diversity (`Media`-réel)

(c) Diversity (`Clef`-idéal)

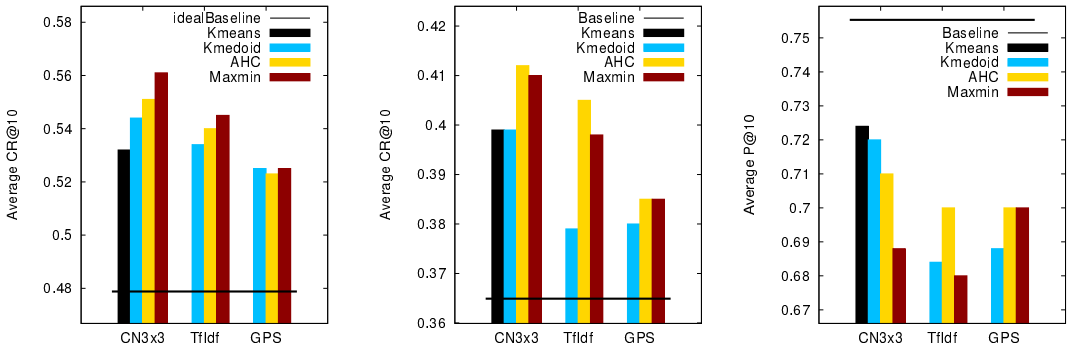
(d) Diversity (`Clef`-réel)

(e) Diversity (`Xilo`-idéal)

FIGURE 9.10 – Comparaison des méthodes de diversité sur plusieurs benchmarks. Les scores CR@10 et P@10 moyens représentent les méthodes `K-Means`, `K-Medoids`, `AHC` et `Maxmin`avec le descripteur HSV en utilisant les paramètres mentionnées dans le tableau 9.1 dans un cas idéal et réel sur les benchmarks (a) (b) `Media` (sous-ensemble `testset`), (c) (d) `Clef` et (e) `Xilo`

— pour environ 25% des requêtes dans le cas réel et environ 15% des requêtes dans le cas idéal, les méthodes de diversité ont diminué la diversité par rapport à la baseline
— pour environ 25% des requêtes dans les deux cas, les méthodes de diversité ont obtenu les mêmes résultats que la baseline.

Ces résultats confirment que les deux méthodes de diversité obtiennent globalement des meilleurs résultats en diversité par rapport à la baseline, avec une différence beaucoup plus significative dans le cas idéal.




(a) Diversity (`Media`-idéal) (b) Diversity (`Media`-réel)

FIGURE 9.11 – Comparaison des méthodes de diversité sur plusieurs descripteurs. Les scores CR@10 et P@10 moyens représentent les méthodes `K-Means`, `K-Medoids`, `AHC` et `Maxmin`avec les descripteurs CN3x3, TfIdf et GPS en utilisant les paramètres mentionnées dans le tableau 9.1 dans un cas idéal et réel sur le benchmark `Media` (sous-ensemble `testset`)

TABLE 9.5 – Calcul de la distance sur la diagonale entre les scores en CR@20 de "`Maxmin` versus Baseline" et "`AHC` versus Baseline" en utilisant le descripteur HSV et les paramètres mentionnées dans le tableau 9.1 dans un cas réel et idéal sur le benchmark `Media` (sous-ensemble `testset`)

| Méthodes | distance sur la diagonale (cas réel) | (cas idéal) |
|---|---|---|
| `Maxmin`(HSV) vs Baseline | 31.98 | 26.87 |
| `AHC`(HSV) vs Baseline | 26.30 | 24.08 |

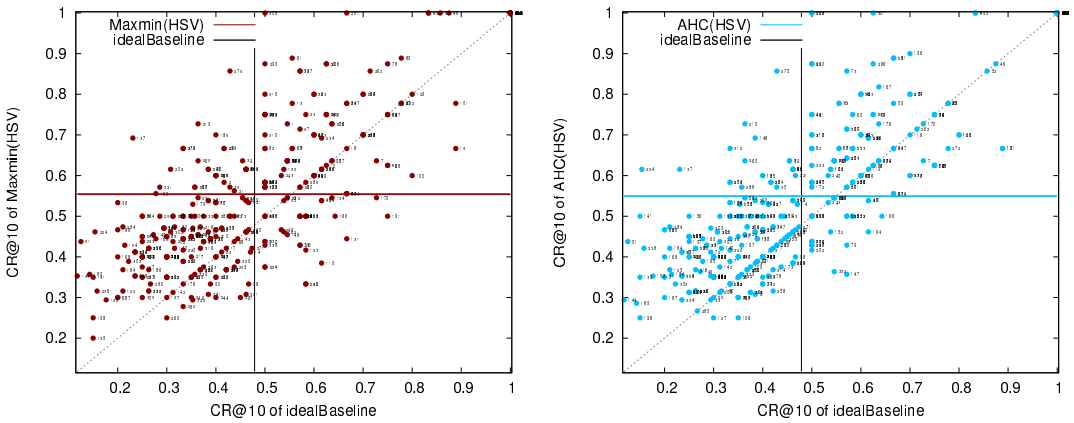
Dans ces figures on voit aussi que la plupart des points sont autour de la diagonale. Les points très éloignés de la diagonale correspondent souvent aux requêtes qui représentent des cas particuliers. Les scores de ces types de requêtes peuvent fausser l'augmentation moyenne en diversité des méthodes de diversité sur la baseline. Afin de mesurer ces types de requêtes, nous calculons la distance de chaque requête sur la diagonale afin de savoir quelle méthode contient des requêtes avec les scores les plus stables. Dans le tableau 9.5 nous montrons la somme de la distance sur la diagonale entre les scores en CR@10 de chaque méthode de diversité versus la Baseline où on voit que dans `Maxmin` obtient en général une distance plus grande qu'avec la `AHC`. Il semblerait donc que la `AHC` obtient des scores plus stables qu'avec `Maxmin`.

Nous confirmons ce comportement en calculant la somme de la distance sur la diagonale entre les scores en CR@20 de chaque méthode de diversité versus la Baseline (voir le tableau A.6 dans l'annexe A) où nous confirmons que la `AHC` obtient toujours des scores plus stables que `Maxmin`.

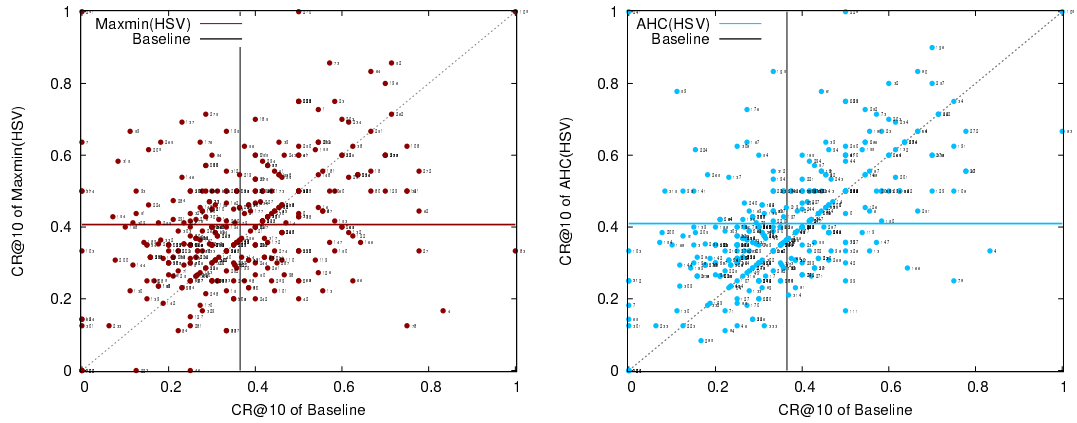
### 9.2.6 Discussion

Jusqu'à présent nous avons comparé quatre méthodes classiques de diversité : une méthode de clustering hiérarchique (`AHC`), deux méthodes de clustering plat (`K-Means` et `K-Medoids`) et un algorithme glouton (`Maxmin`).

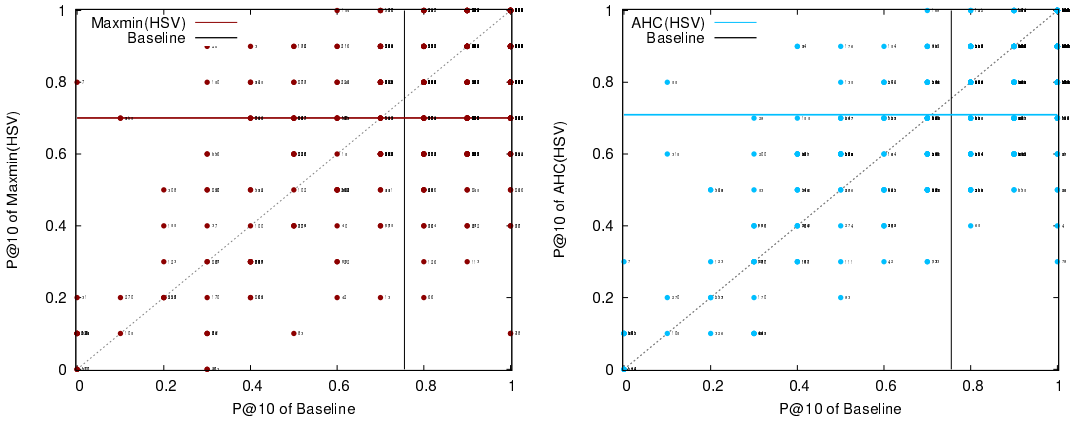
**Clustering hiérarchique vs clustering plat** Les méthodes clustering plat sont généralement des méthodes qui cherchent à optimiser le résultat final, contrairement au




(a) Scores CR@10 des méthodes de diversité vs idealBaseline (`Media`-idéal)




(b) Scores CR@10 des méthodes de diversité vs Baseline (`Media`-réel)




(c) Scores P@10 des méthodes de diversité vs Baseline (`Media`-réel)

FIGURE 9.12 – Comparaison des scores par requête entre "`Maxmin` versus baseline" (à gauche) et "`AHC` versus baseline" (à droite). En ordonnée, les scores CR@10, P@10 par requête représentent les méthodes `AHC` et `Maxmin` avec le descripteur HSV en utilisant les paramètres mentionnées dans le tableau 9.1 dans un cas idéal et réel sur le benchmark `Media` (sous-ensemble `testset`)

clustering hiérarchique qui utilise une optimisation locale. En comparant ces méthodes de clustering, nous avons vu qu'utiliser un clustering hiérarchique (`AHC`) permet d'obtenir souvent une meilleure diversité que les méthodes de clustering plat. Cependant, cette

différence devient moins évident quand on utilise un descripteur de mauvaise qualité en diversité (comme le texte ou le GPS) et plus évident quand on utilise la couleur qui est un descripteur de bonne qualité en diversité.

Pour étudier plus en détail ce comportement, nous avons étudié la distribution du nombre d'images par cluster générées par les méthodes de clustering afin de savoir la distribution qui s'approche le plus à la distribution du nombre d'images dans les sous-thèmes de la vérité terrain. Nous avons vu que lorsqu'on utilise peu de clusters ($K = 10$) la distribution du nombre d'images par cluster généré par la `AHC` se rapproche le plus à la distribution du nombre d'images dans les sous-thèmes de la vérité terrain. Cependant, quand le nombre de clusters augmente ($K = 20$ ou $K = 30$), la distribution de clusters des méthodes de clustering plat se rapproche le plus à la distribution de sous-thèmes de la vérité terrain. Nous avons vu aussi que les méthodes de clustering plat semblent générer une distribution des clusters moins homogène quand ces méthodes utilisent un descripteur de mauvaise qualité comme le texte ou le GPS.

Dans la distribution de sous-thèmes de la vérité terrain nous avons vu aussi qu'environ 50% des sous-thèmes sont de petits sous-thèmes qui ont entre un et cinq images. Les images de petits sous-thèmes sont souvent des images originales qui sont difficiles à retrouver, car ces images ont des caractéristiques très différentes que celles de la plupart des images pertinentes pour une requête donnée. Donc, le fait de détecter la plus grande quantité de petits sous-thèmes pourrait jouer un rôle très important dans la diversité.

**Diversité par clustering vs diversité par optimisation** En comparant les méthodes de diversité, nous avons vu que dans un cas idéal `Maxmin` devient très intéressante pour augmenter la diversité, car l'approche de rechercher toujours l'image la plus dissimilaire peut avoir plus de probabilité à trouver des images originales qui sont normalement difficiles à retrouver. Par contre, dans un cas réel, la `AHC` dévient plus intéressant pour augmenter la diversité, car `Maxmin` semble être trop pénalisé par les images non-pertinentes présentes dans ce cas réel.

Pour étudier plus en détail ces méthodes, nous avons analysé d'abord les sous-thèmes retrouvés par les méthodes de diversité et en analysant les 10 premières images retrouvées, nous avons vu qu'en général, les méthodes de diversité arrivent à retrouver environ 50% d'images qui appartiennent aux sous-thèmes différents. Cependant, nous avons vu que `Maxmin` retrouve plus d'images non pertinentes (environ 30% pour `Maxmin` et environ 20% pour le clustering) tandis que les méthodes de clustering retrouvent plus de sous-thèmes redondants que l'utilisateur a déjà vu (environ 30% pour le clustering et environ 20% pour `Maxmin`).

Après avoir étudié les scores par requête nous avons vu la difficulté d'utiliser l'écart type, car ils existent des requêtes avec des scores abruptes qui peuvent fausser la valeur finale de l'écart type pour chaque méthode. Cependant, en calculant la distance sur la diagonale entre chaque méthode et la baseline, il semblerait que la `AHC` devient plus robuste que `Maxmin` car dans `Maxmin` les scores des requêtes sont plus éloignes de la diagonale que dans le clustering hiérarchique.

## 9.3 Temps de calcul

Après avoir étudié différentes méthodes de diversité classiques, dans cette section nous étudions le temps de calcul de ces méthodes afin de savoir si elles sont adaptées à être utilisables dans un système de recherche d'images en ligne.

Nous décrivons la complexité de ces méthodes de diversité dans le chapitre 3 page 31 pour le clustering et dans la section 4.2.2 page 45 pour `Maxmin` et ses paramètres optimaux sont décrites dans le tableau 9.1 page 139.

En général, le nombre d'images est un paramètre très important, car si on utilise plus d'images on a plus de probabilité d'augmenter la diversité, mais on risque aussi de ramener plus de bruit (des images non-pertinentes) et d'augmenter aussi son temps de calcul. Donc, on doit trouver le bon compromis de choisir un nombre d'images qui puisse nous donner de bons scores en diversité, mais en évitant de ramener trop de bruit et en gardant un temps de calcul raisonnable pour être utilisable dans une application de recherche en ligne.

Il faut remarquer aussi que normalement dans un système de recherche d'images "en ligne" un utilisateur moyen pourrait attendre environ jusqu'à 2 ou 3 secondes afin obtenir la réponse du système.

Nous avons programmé les algorithmes de diversité en langage Ruby et nous avons lancé ces expérimentations dans un processeur Intel i7 de 8 cœurs à 3GHz avec une mémoire de 16GB. Afin d'optimiser le temps de calcul dans les algorithmes de diversité nous avons utilisé des tables d'hachage afin d'éviter de recalculer les distances à chaque fois. De plus, pour optimiser le temps de calcul de l'algorithme de la `AHC`, nous avons généré la hiérarchie de clusters jusqu'à $k$ clusters.

Afin d'étudier les facteurs qui peuvent affecter le temps de calcul, dans cette section nous étudions l'influence de la dimension des descripteurs, le nombre d'images et le nombre de clusters pour les méthodes de clustering.

### 9.3.1 Influence de la dimension des descripteurs

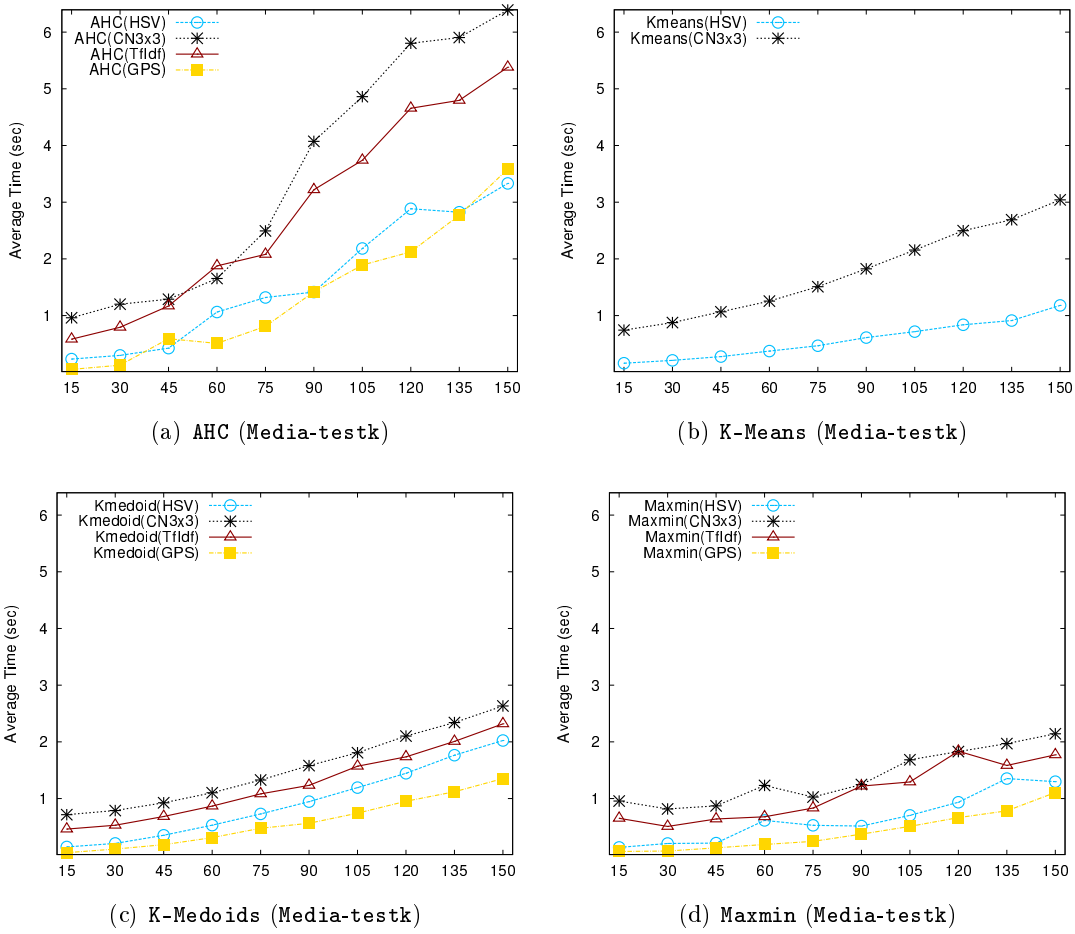
Dans les chapitres précédents, nous avons étudié les scores des méthodes de diversité sur plusieurs descripteurs, mais nous n'avons pas encore étudié l'influence de la dimension du descripteur sur le temps de calcul. Ce paramètre peut augmenter de manière considérable le temps de calcul, spécialement quand on utilise un algorithme qui a besoin de parcourir la dimension du descripteur à chaque itération.

Pour ces expérimentations, nous utilisons quatre descripteurs de dimensions différents. : deux de couleur avec une dimension de 33 bins (HSV) et une dimension de 99 bins (CN3x3), un textuel avec une dimension d'environ 77 bins (TfIdf) et des coordonnées GPS avec une dimension de 2 bins. Dans ces expérimentations, nous mesurons le temps de calcul depuis le début de l'algorithme jusqu'à l'obtention de la liste d'images résultats.

Dans la figure 9.13 nous comparons les temps de calcul des méthodes de diversité avec plusieurs descripteurs. Le temps de calcul représente la moyenne des 132 requêtes du benchmark `Media` (sous-ensemble `testk`) en utilisant différents nombres d'images. Dans cette figure on voit que généralement la dimension du descripteur influe sur le temps de calcul. De plus, on voit que cette influence est plus significative sur les méthodes `K-Means` et `AHC` et moins significative sur `Maxmin` et `K-Medoids`. En effet, dans `K-Means` le coût du calcul du vecteur moyen (linkage centroid) dépend de la dimension du descripteur, contrairement aux méthodes `Maxmin` et `K-Medoids` qui utilisent les mêmes distances à chaque itération. Dans la `AHC` on utilise aussi les mêmes distances, mais on en rajoute des distances provenant de la fusion de chaque cluster, comportement qui semblerait influencer sur le temps de calcul. Donc, il est toujours préférable de choisir un descripteur avec peu de dimensions, spécialement dans la `AHC` ou `K-Means` qui sont des algorithmes plus sensibles à la dimension.

### 9.3.2 Influence du nombre de clusters

Un paramètre en commun dans les méthodes de clustering est le nombre de clusters à utiliser pour regrouper les images. Dans cette partie nous étudions l'influence de ce paramètre sur le temps de calcul en utilisant plusieurs nombres d'images (en abscisse)




(a) `AHC` (`Media-testk`) (b) `K-Means` (`Media-testk`)

(c) `K-Medoids` (`Media-testk`) (d) `Maxmin` (`Media-testk`)

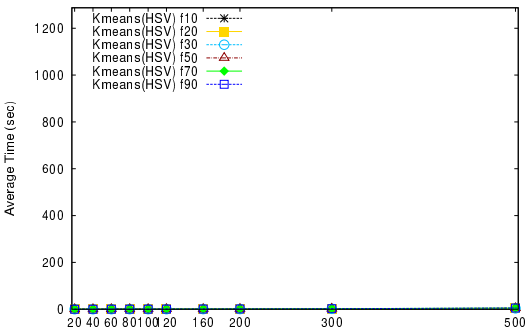
FIGURE 9.13 – Comparaison du temps de calcul des méthodes de diversité sur plusieurs descripteurs. Le temps de calcul représente la moyenne des 132 requêtes du benchmark `Media` (sous-ensemble `testk`) sur les méthodes de diversité (a) `AHC`, (b) `K-Means`, (c) `K-Medoids` et (d) `Maxmin` avec les descripteurs HSV, CN3x3, TfIdf et GPS en utilisant différents nombres d'images (en abscisse)

et différentes tailles des clusters. Afin de lancer ces méthodes dans les mêmes conditions nous mesurons le temps de calcul depuis le début de l'algorithme jusqu'à l'obtention des clusters.

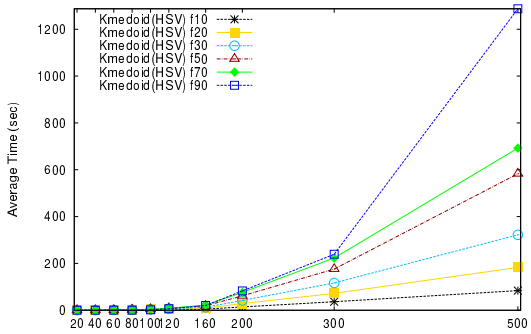
Dans la figure 9.14 nous comparons le temps de calcul des méthodes de clustering avec le descripteur HSV en utilisant différentes tailles des clusters et différents nombres d'images (en abscisse) sur le benchmark `Clef`. Dans cette figure on voit que dans `K-Medoids` la variation du nombre des clusters affecter le temps de calcul, spécialement quand on utilise un grand nombre d'images tandis que dans la `AHC` et dans `K-Means` on voit que cette est moins significative.

Dans la complexité des algorithmes du clustering plat nous avons le nombre d'itérations $I$ qui pourrait influencer sur le temps de calcul. Pour cette raison, dans la figure 9.15 nous montrons le temps de calcul et le nombre d'itérations pour les méthodes de clustering plat avec le descripteur HSV en utilisant différents nombres des clusters et d'images sur le benchmark `Clef`. Dans cette figure on voit que dans `K-Medoids` le fait d'augmenter le nombre de clusters, augmente aussi le nombre d'itérations. De plus, le fait de générer plus d'itérations pénalise énormément le temps de calcul de cet algorithme, car sa complexité sans compter le nombre d'itération est déjà de type quadratique.

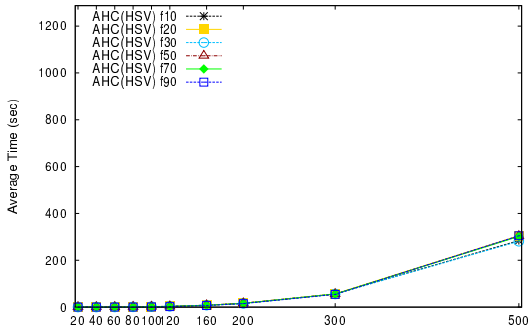
Cependant, dans `K-Means` on voit un comportement différent où le fait d'augmenter le




(a) `K-Means` (`Clef`). Dans la figure 9.15(a) nous montrons la même figure avec une échelle plus petite




(b) `K-Medoids` (`Clef`)




(c) `AHC` (`Clef`)

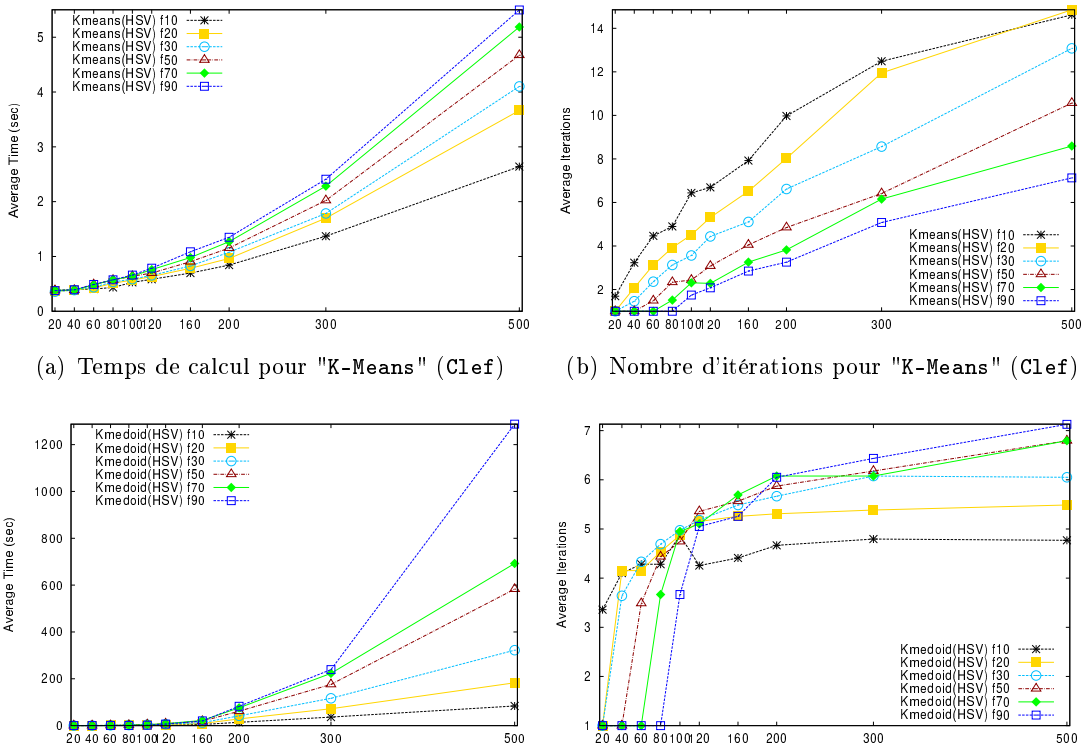
FIGURE 9.14 – Comparaison du temps de calcul des méthodes de clustering sur plusieurs nombres des clusters. Le temps de calcul représente la moyenne des 39 requêtes du benchmark `Clef` sur (a) `K-Means`, (b) `K-Medoids` et (c) la `AHC` avec le descripteur HSV en utilisant différents nombres d'images (en abscisse)

nombre de clusters, diminue le nombre d'itérations. Il semblerait que lorsqu'on utilise plus de clusters, l'algorithme `K-Means` converge plus rapidement ce qui génère en conséquence moins d'itérations. Seulement lorsqu'on utilise de grandes quantités d'images, on peut commencer à voir l'influence de la taille de clusters avec `K-Means`.

Ce comportement pourrait expliquer pourquoi la méthode `K-Means` est moins sensible au nombre des clusters ce qui pourrait être intéressant à utiliser dans une application de recherche d'images où on est limité par le nombre d'images.

### 9.3.3 Influence des méthodes de diversité

Finalement, dans la figure 9.16 nous comparons le temps de calcul des méthodes de diversité en faisant varier le nombre d'images. Le temps de calcul représente la moyenne des 39 requêtes du benchmark `Clef` sur les quatre méthodes de diversité avec le descripteur HSV. Dans cette figure on voit que les méthodes `K-Means` et `Maxmin` sont plus rapides que les méthodes `K-Medoids` et `AHC` avec la variation du nombre d'images. Il faut faire attention à la méthode `K-Means` car nous avons vu déjà que son temps de calcul dépend aussi de la dimension du descripteur ce qui pourrait augmenter son temps de calcul avec un descripteur de dimension plus grande (comme CN3x3 par exemple). De même, si les méthodes `K-Medoids`, `AHC` et `Maxmin` n'ont pas besoin de recalculer la distance, les itérations réalisées dans `Maxmin` sembleraient être plus rapides que les itérations réalisées dans les méthodes de clustering `K-Medoids` et `AHC`.




(a) Temps de calcul pour "`K-Means`" (`Clef`) (b) Nombre d'itérations pour "`K-Means`" (`Clef`)

(c) Temps de calcul pour "`K-Medoids`" (`Clef`) (d) Nombre d'itérations pour "`K-Medoids`" (`Clef`)

FIGURE 9.15 – Comparaison du temps de calcul et du nombre d'itérations des méthodes de clustering sur plusieurs nombre de clusters. Le temps de calcul et le nombre d'itérations représentent la moyenne des 39 requêtes du benchmark `Clef` sur (a) (b) `K-Means` et (c) (d) `K-Medoids` avec le descripteur HSV en utilisant différents nombres d'images (en abscisse)

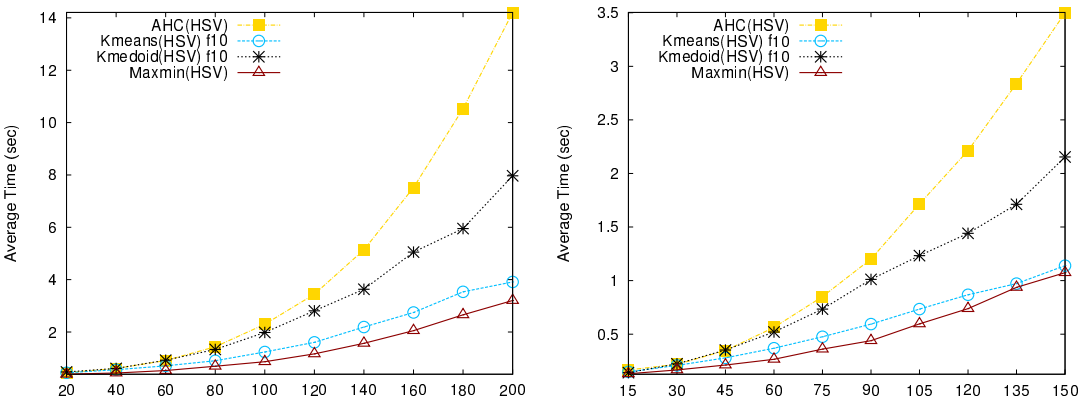
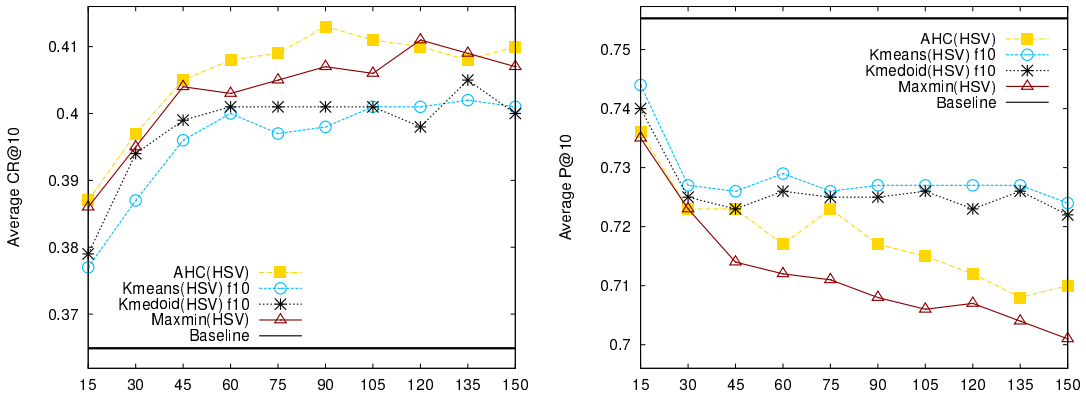
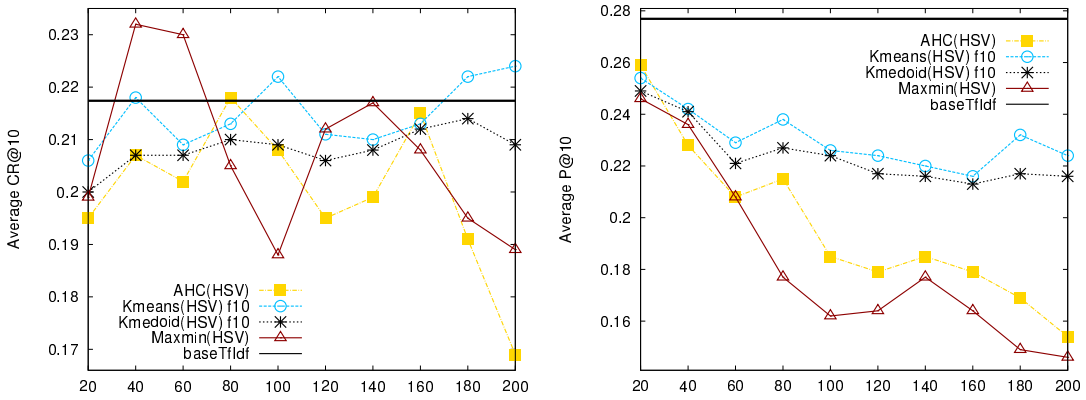



FIGURE 9.16 – Comparaison du temps de réponse des méthodes de diversité. Le temps de réponse représente la moyenne des 39 requêtes du benchmark `Clef` sur les méthodes de diversité `AHC`, `K-Means`, `K-Medoids` et `Maxmin` avec le descripteur HSV en utilisant différents nombres d'images (en abscisse)

Dans la figure 9.17 nous comparons aussi les scores des méthodes de diversité avec la variation du nombre d'images. Les scores CR@10 et P@10 moyens représentent les méthodes de diversité avec le descripteur HSV en utilisant différents nombres d'images dans le cas réel sur les benchmarks (a) `Media` (sous-ensemble `testset`) et (b) `Clef`.




(a) Scores CR@10 et P@10 des méthodes de diversité sur `Media-testset`




(b) Scores CR@10 et P@10 des méthodes de diversité sur `Clef`

FIGURE 9.17 – Comparaison des scores des méthodes de diversité. Les scores CR@10 et P@10 représentent les méthodes de diversité `AHC`, `K-Means`, `K-Medoids` et `Maxmin` avec le descripteur HSV en utilisant différents nombres d'images (en abscisse) dans un cas réel sur le (a) benchmark `Media` (sous-ensemble `testset`) et (b) `Clef`

Dans la figure 9.17(a) du benchmark `Media` on voit que lorsque les méthodes prennent un nombre d'images trop grand l'augmentation de la diversité devient moins significative. En effet, en sachant que les résultats pertinents se trouvent dans les premières images résultats, il n'est donc pas nécessaire de lancer ces méthodes sur beaucoup d'images, il suffit de la lancer sur les premiers résultats, ce qui nous permettrait d'utiliser ces méthodes en un temps raisonnable. En regardant les scores de pertinence, on voit que si on augmente le nombre d'images, la diminution de la pertinence est plus significative dans `Maxmin` et la `AHC` que dans les méthodes de clustering plat.

Donc, il semblerait mieux choisir une méthode `Maxmin` ou la `AHC` qui permettent d'obtenir une meilleure diversité, mais sans prendre un nombre d'images trop élevé afin d'éviter d'être pénalise par les images non-pertinentes et d'avoir un temps de calcul acceptable afin d'être utilisable dans une application de recherche en ligne.

Cependant, dans la figure 9.17(b) nous montrons une exception obtenue dans le cas réel du benchmark `Clef` où nous avons la particularité d'avoir une baseline avec beaucoup d'images non-pertinentes où on voit que les quatre méthodes obtient des faibles scores en diversité. Dans cette figure on voit aussi que les méthodes du clustering plat ont des scores plus stables en diversité que `Maxmin` et `AHC`. Il semblerait que le clustering plat est moins sensible aux images non-pertinentes que les méthodes `Maxmin` et `AHC`.

**Discussion** Après avoir étudié les différentes méthodes de diversité, dans cette section nous avons étudié son temps de calcul afin de savoir si ces méthodes pourraient être utilisables dans une application de recherche en ligne. Dans ces méthodes nous avons plusieurs paramètres qui peuvent influencer sur le temps de calcul comme le nombre d'images, la dimension du descripteur, le nombre d'itération, le nombre de clusters, etc.

Par rapport aux méthodes de clustering, on voit que dans `K-Means` lorsqu'on utilise un nombre réduit d'images, le taux de croissance du temps de calcul ne semble pas être affecté par la taille du cluster, car quand on utilise plus de clusters on fait moins d'itérations. Seulement lorsqu'on commence à utiliser de grandes quantités d'images, on peut commencer à voir l'influence de la taille de clusters dans `K-Means`. Donc, utiliser `K-Means` dans une application de recherche d'images "en ligne" pourrait être totalement envisageable lorsqu'on utilise peu d'images, car la taille de clusters se compense avec le nombre d'itération.

La dimension du descripteur est un paramètre important dans le temps de calcul, car si on utilise un descripteur de grande dimension, ce descripteur pourrait être de bonne qualité pour augmenter la diversité, mais on risque aussi d'augmenter son temps de calcul. En général, nous avons vu que la taille de la dimension du descripteur influe sur le temps de calcul, en plus, cette influence est plus significative sur les méthodes `K-Means` et `AHC` et moins significative sur `Maxmin` et `K-Medoids`. Donc, il est toujours préférable de choisir un descripteur avec peu de dimensions, spécialement avec la `AHC` ou `K-Means` qui sont des algorithmes plus sensibles à la dimension du descripteur.

Le nombre d'images est un autre paramètre très important dans le temps de calcul, car si on utilise plus d'images on a plus de probabilité d'augmenter la diversité, mais on risque aussi de ramener plus de bruit (des images non-pertinentes) et d'augmenter aussi son temps de calcul. En général nous avons vu que les méthodes `K-Means` et `Maxmin` sont plus rapides que les méthodes `K-Medoids` et `AHC` avec l'augmentation du nombre d'images. Cependant, il semblerait mieux choisir `Maxmin` ou la `AHC` car ils utilisent des stratégies plus avantageuses pour augmenter la diversité.

De plus, nous avons vu aussi que lorsque les méthodes prennent un nombre d'images trop grande l'augmentation de la diversité devient moins significative. En effet, en sachant que les résultats pertinents se trouvent dans les premiers résultats, il n'est donc pas nécessaire de lancer ces méthodes sur beaucoup de résultats, il suffit de la lancer sur les premiers résultats, ce qui nous permettrait d'utiliser ces méthodes en un temps raisonnable.

De même, si la `AHC` semble être coûteuse en temps de calcul lorsqu'on utilise un grand nombre d'images, ils existent plusieurs stratégies pour réduire ce temps de calcul. Par exemple, on peut calculer à l'avance les distances entre les images (étape offline), on peut aussi générer la hiérarchie de clusters jusqu'à $n$ clusters ou encore mieux on peut gagner en itérations en fusionnant toutes les distances inférieures à un seuil.

## 9.4 Bilan et discussion

Dans les chapitre précédentes, nous avons utilisé des méthodes de diversité par clustering afin d'augmenter la diversité. Afin de montrer l'intérêt de ces méthodes sur la diversité, nous lui comparons avec une méthode de diversité par optimisation (algorithme `Maxmin`) qui est une approche de diversité totalement différente de la diversité par clustering.

**Clustering hiérarchique vs clustering plat** En comparant les méthodes de clustering, nous avons vu qu'en général, utiliser un clustering hiérarchique (`AHC`) permet d'obtenir une meilleure diversité que les méthodes de clustering plat. Cependant, cette

différence devient moins évident quand on utilise un descripteur de mauvaise qualité en diversité (comme le texte ou le GPS) et plus évident quand on utilise la couleur qui est un descripteur de bon qualité en diversité.

Pour étudier plus en détail le comportement obtenu, nous avons étudié la distribution des clusters générés par ces méthodes en fonction du nombre d'images par cluster. Nous avons vu que lorsqu'on utilise peu de clusters ($K = 10$) la distribution du nombre d'images par cluster généré par la `AHC` se rapproche le plus à la distribution des sous-thèmes de la vérité terrain. Cependant, quand le nombre de clusters augmente ($K > 10$), la distribution de clusters des méthodes de clustering plat se rapproche le plus de la distribution de sous-thèmes de la vérité terrain.

**Diversité par clustering vs diversité par optimisation** En comparant les méthodes de diversité, nous avons vu que dans un cas idéal `Maxmin` devient très intéressante pour augmenter la diversité, car rechercher toujours l'image la plus dissimilaire peut avoir plus de probabilité à trouver des images originales qui sont difficiles à retrouver. Par contre, dans un cas réel, la `AHC` dévient plus intéressant pour augmenter la diversité, car `Maxmin` semble être trop pénalisé par les images non-pertinentes présentes dans ce cas réel.

Pour étudier plus en détail ces méthodes, nous avons analysé d'abord les sous-thèmes retrouvés par les méthodes de diversité et en analysant les 10 premières images retrouvées, nous avons vu qu'en général, les méthodes de diversité arrivent à retrouver environ 50% d'images qui appartiennent aux sous-thèmes différents. Cependant, nous avons vu que `Maxmin` retrouve plus d'images non pertinentes (environ 30% pour `Maxmin` et environ 20% pour le clustering) tandis que les méthodes de clustering retrouvent plus de sous-thèmes redondants que l'utilisateur a déjà vu (environ 30% pour le clustering et environ 20% pour `Maxmin`).

Après avoir étudié les scores par requête nous avons vu la difficulté d'utiliser l'écart type, car il existe des requêtes avec des scores abruptes qui peuvent fausser la valeur finale de l'écart type pour chaque méthode. Cependant, en calculant la distance sur la diagonale entre chacune et la baseline, il semblerait que la `AHC` devient plus robuste que `Maxmin` où les scores des requêtes sont plus éloignés de la diagonale que dans le clustering hiérarchique.

**Temps de calcul** Enfin, nous avons étudié le temps de calcul des méthodes de diversité afin de savoir quelle méthode est la plus adaptée à être utilisable dans une application de recherche d'images en ligne.

Premièrement, nous avons étudié l'influence de la **dimension des descripteurs** sur le temps de calcul. Nous avons vu que la dimension du descripteur influe sur le temps de calcul, en plus, cette influence est plus significative sur les méthodes `K-Means` et `AHC` et moins significative sur `Maxmin` et `K-Medoids`. Donc, il est toujours préférable de choisir un descripteur avec peu de dimensions, spécialement dans la `AHC` et `K-Means` qui sont des algorithmes plus sensibles à la dimension du descripteur.

Deuxièmement, nous avons étudié l'influence du **nombre de clusters** $K$ et le **nombre d'itération** $I$ sur le temps de calcul. Nous avons vu que, contrairement aux autres méthodes de clustering, dans `K-Means` quand on utilise un nombre réduit d'images, le taux de croissance du temps de calcul ne semble pas être affecté par la taille du cluster $K$ car quand on utilise plus de clusters, `K-Means` semblerait faire moins d'itérations $I$ (et vice-versa). Seulement quand on utilise de grandes quantités d'images, on peut commencer à voir l'influence de la taille de clusters avec `K-Means`.

Enfin, nous avons étudié l'influence du **nombre d'images** sur le temps de calcul. Nous avons vu que les méthodes `K-Means` et `Maxmin` sont plus rapides que les méthodes

`K-Medoids` et `AHC` avec l'augmentation du nombre d'images. Cependant, nous avons vu aussi que lorsque les méthodes prennent un nombre d'images trop grande l'augmentation de la diversité devient moins significative. En effet, en sachant que les résultats pertinents se trouvent dans les premiers résultats, il n'est donc pas nécessaire de lancer ces méthodes sur beaucoup de résultats, il suffit de la lancer sur les premiers résultats, ce qui nous permettrait d'utiliser ces méthodes en un temps raisonnable.

Donc, il semblerait mieux choisir `Maxmin` ou la `AHC` qui sont des méthodes adaptées pour augmenter la diversité, mais sans prendre un nombre d'images trop élevé afin d'éviter d'être pénalise par les images non-pertinentes et le temps de calcul.

# Chapitre 10

# Conclusions et perspectives

Les moteurs de recherche traditionnels offrent à l'utilisateur des résultats de plus en plus pertinents, mais, dans la plupart des cas, les résultats similaires ont tendance à se regrouper. L'utilisateur peut être intéressé par retrouver des documents qui soient certes tous pertinents par rapport à sa requête, mais aussi qui soient différents les uns des autres. Ce problème de la diversité est très vaste et est observé dans plusieurs domaines.

Dans cette thèse, placée dans un cadre CIFRE avec l'entreprise Xilopix, nous avons considéré le problème de la diversité pour les systèmes de recherche d'images. Nous avons apporté un cadre théorique qui permet d'encadrer, conceptualiser et comparer les différentes approches et nous avons focalisé notre attention sur la diversité par l'exploitation du clustering. Nous avons étudié de manière approfondie et verticale, théorique et expérimentale, ce que ce choix implique en considérant en particulier les questions suivantes : comment obtenir les clusters (approche hiérarchique vs plat) ? Quel est le nombre de clusters optimal ? Comment gérer les clusters obtenus pour ensuite diversifier par réordonnancement des images ? Comment comparer les images et leurs descripteurs pour créer les clusters ? Quels sont les avantages et inconvénients de ce type d'approches dans un contexte opérationnel dans l'industrie ?

Ce chapitre résume les principales contributions de la thèse avant de décrire les perspectives qu'elle ouvre.

## 10.1 Cadre théorique de la diversité par clustering

### 10.1.1 Proposition d'une méthode de diversité basée sur le clustering hiérarchique

Pour résoudre le problème de la diversité, nous avons proposé d'utiliser une méthode de clustering, plus spécialement, le clustering hiérarchique (`AHC`) car sa hiérarchie des clusters peut bien correspondre à la nature intrinsèquement hiérarchique de la diversité.

Cependant, l'exploitation du clustering pour la diversité n'est pas évidente à faire et pose un certain nombre de difficultés qui nous avons prises en compte pour obtenir les résultats optimaux en diversité. Pour résoudre ces problèmes, nous avons proposé un cadre théorique qui peut être synthétisé dans une chaîne de traitement composée de deux parties principales : la première a pour objectif d'augmenter la pertinence, la deuxième l'objectif d'augmenter la diversité. Ce cadre nous a permis d'étudier deux cas d'utilisation : le cas idéal où tous les documents retrouvés sont pertinents et le cas réel où parmi les documents retrouvés un certain nombre est non-pertinent.

Dans ce cadre général, nous avons aussi proposé des solutions aux difficultés d'utilisation d'une méthode du clustering pour la diversité.

**Proposition d'une approche d'ordre des clusters** En sachant que le clustering ne fournit pas d'ordre sur les clusters ni sur les images, comment déterminer l'ordre des clusters et l'ordre des images pour améliorer la diversité des résultats ? Une approche classique consiste à trier les clusters en fonction du rang des images.

Dans cette thèse, nous avons proposé deux approches qui trient les clusters selon leur taille (en fonction du nombre d'images), car nous pensons par exemple qu'un cluster avec peu d'images, peut représenter des images originales qui sont normalement des images difficiles à retrouver et qui peuvent augmenter la diversité.

Dans nos expériences, nous avons remarqué que le fait de donner plus d'importance aux clusters avec peu d'images (priorité `Increasing`) semble être intéressante seulement dans un cas idéal et avec l'utilisation des bons descripteurs en diversité (comme la couleur ou l'arborescence de concepts). Cependant, nos résultats expérimentaux ont montré que pour obtenir la valeur maximale de diversité, il serait suffisant d'utiliser une approche classique par le rang.

**Proposition d'un réordonnancement hiérarchique** Une autre difficulté est de déterminer comment réordonner les résultats après le clustering. Le réordonnancement classique d'images consiste le plus souvent à alterner les images de chaque cluster.

Afin d'utiliser les différents niveaux de granularité fournis par la `AHC`, nous avons proposé un réordonnancement hiérarchique (`HierO`) qui diversifie les images dans chaque cluster avant de réordonner les images.

Dans nos expérimentations, nous avons observé que même si l'approche classique est suffisante pour d'obtenir la valeur maximale de diversité, notre approche hiérarchique s'avère d'être une méthode intéressante dans certains cas.

**Paramétrisation pour maximiser la diversité** L'étude expérimentale des performances des méthodes de clustering avec différents paramètres et lors de nombreux tests, a montré un comportement général très intéressant pour obtenir la valeur maximale en diversité, liée au choix du nombre de clusters : elle a permis d'établir que le bon choix de paramètres pour obtenir la valeur maximale en diversité est de choisir un nombre fixe de clusters égal au nombre d'images à présenter à l'utilisateur.

De plus, le point de la valeur maximale de diversité devient le point de convergence pour les techniques de priorité ou ordre de clusters et les techniques réordonnancement des images ce qui fait que la valeur maximale de diversité dépend seulement du choix du nombre de clusters.

### 10.1.2 Mesure de similarité pour un descripteur basé sur une arborescence de concepts

Dans cette thèse, nous avons aussi proposé une nouvelle approche basée sur l'utilisation d'une arborescence de concepts, et plus particulièrement celle de Xilopix, comme descripteur alternatif au texte ou visuel. Nous pensons que l'arborescence de concepts est une ressource très riche en information et on espère que sa structure hiérarchique corresponde à la structure hiérarchique de la diversité.

Cependant, l'utilisation de cette approche dans les méthodes de diversité pose un certain nombre de difficultés. Dans cette thèse, nous avons développé des solutions afin de résoudre ces dernières.

**Proposition d'une méthode de généralisation pour le descripteur arborescent** Dans une arborescence de concepts, les images peuvent être associées à un seul concept de l'arborescence, mais aussi à plusieurs concepts. Dans la littérature, il existe

plusieurs mesures classiques qui calculent la similarité entre deux images associées à un seul concept. Cependant, nous n'avons pas trouvé de mesures dans le cas où les images sont associées à plusieurs concepts. Pour résoudre ce problème, nous avons proposé une méthode de généralisation qui calcule la similarité entre deux images décrites par plusieurs concepts (voir section 5.1.4.1 page 55).

**Proposition d'un critère d'agrégation adapté au descripteur arborescent** Pour construire la hiérarchie de clusters dans la `AHC`, nous avons besoin de choisir un critère d'agrégation.Dans la littérature, il existe des critères d'agrégation classiques qui sont très adaptés aux descripteurs classiques (visuels ou textuels par exemple), mais qui ne tirent pas profit de la structure hiérarchique d'un descripteur basé sur l'utilisation d'une arborescence de concepts. En conséquence, en nous inspirant du critère centroïde, nous avons proposé un critère d'agrégation plus adapté à l'utilisation d'une arborescence de concepts comme descripteur (voir section 7.1.1 page 97). De plus, cette proposition est très rapide en temps de calcul ce qui la rend très appropriée dans le cadre d'applications en ligne.

## 10.2 Étude expérimentale exhaustive et comparative de la diversité

Il existe de nombreux travaux dans le domaine de la diversité, dont certains travaux ont déjà travaillé avec les algorithmes de clustering que nous utilisons, cependant l'un des apports de cette thèse est d'avoir cherché à obtenir des résultats qui ne soient pas dépendants d'un benchmark ni de certains cas particuliers qui peuvent fausser le vrai comportement des méthodes de diversité. Pour cela, dans cette thèse, nous avons mis au point des protocoles expérimentaux afin de réaliser une comparaison exhaustive avec toutes sortes de méthodes de diversité, afin de comparer différentes approches dans les mêmes conditions expérimentales.

Pour cela, nous avons volontairement choisi de travailler sur trois benchmarks assez différents, en nombre d'images, en type de descripteurs et provenant de contextes différents afin d'obtenir des comportements les plus généraux possibles. De plus, nous avons généré un type de descripteur commun sur tous les benchmarks afin de comparer de manière globale les résultats expérimentaux sur ces trois benchmarks.

En outre, à notre connaissance, il n'existe pas de benchmark adapté pour faire des expériences utilisant une arborescence de concepts pour la diversité. Pour cette raison, nous avons construit le benchmark `Xilo` avec la collaboration de l'entreprise Xilopix.

### 10.2.1 Mise en évidence de l'intérêt des approches par clustering hiérarchique

Pour montrer l'intérêt du clustering hiérarchique (`AHC`) par rapport à d'autres méthodes de diversité, nous avons comparé cette méthode avec deux méthodes classiques de clustering plat (`K-Means` et `K-Medoids`) ainsi qu'avec une méthode de diversité par optimisation (`Maxmin`) qui utilise une approche de diversité totalement différente de l'approche de diversité par clustering.

**Clustering plat vs clustering hiérarchique** Dans nos expériences, la `AHC` donne globalement de meilleurs résultats en diversité par rapport aux méthodes classiques de clustering plat. Ces résultats semblent confirmer qu'une approche de clustering hiérarchique est plus adaptée pour augmenter la diversité que l'approche de clustering plat.

Afin d'étudier plus en détail ces méthodes de clustering, nous avons examiné la distribution des clusters générés en fonction du nombre d'images par cluster. Les résultats obtenus montrent que la distribution des tailles de clusters obtenues par la `AHC` est celle qui se rapproche plus de la distribution du nombre d'images dans les sous-thèmes de la vérité terrain.

Nous avons aussi étudié le nombre des sous-thèmes retrouvés dans les 10 premières images retrouvées par chaque méthode de clustering, afin d'examiner si les clusters générés permettent d'offrir des images qui appartiennent aux sous-thèmes différents. Les résultats expérimentaux ne montrent pas de différence significative entre les deux approches de clustering.

**Diversité par clustering vs diversité par optimisation** Dans nos expériences, nous avons observé que la `AHC` et `Maxmin` obtiennent en général de bons résultats en diversité. Cependant, dans le cas où toutes les images sont pertinentes (cas idéal), `Maxmin` devient très intéressant pour augmenter la diversité, car son approche de rechercher toujours l'image la plus dissimilaire a plus de probabilité de trouver des images originales (particulières) qui sont des images difficiles à retrouver. Par contre, dans un cas réel où un grand nombre d'images non-pertinentes sont retrouvées, la `AHC` devient plus intéressante pour augmenter la diversité, car `Maxmin` semble être trop pénalisé par les images non-pertinentes.

Pour approfondir plus en détail cette étude, nous avons analysé les sous-thèmes retrouvés dans les 10 premières images retrouvées par chaque méthode. Les résultats expérimentaux montrent qu'en général, les méthodes de diversité arrivent seulement à retrouver environ 50% d'images qui appartiennent aux sous-thèmes différents, environ 20% d'images qui sont des sous-thèmes redondants (que l'utilisateur a déjà vus) et environ 30% d'images qui sont des images non-pertinentes. Nous remarquons aussi que `Maxmin` retrouve un peu plus d'images non-pertinentes que les méthodes de clustering, tandis que celles-ci retrouvent plus de sous-thèmes redondants que `Maxmin`. Ces résultats semblent confirmer le comportement précédent dans les cas idéal et réel des méthodes de la diversité.

En analysant les scores des méthodes de diversité par requête, nous remarquons aussi que la `AHC` obtient des scores par requête moins dispersés que `Maxmin`. Donc, il semble que la `AHC` est une méthode plus robuste que `Maxmin` concernant la régularité des performances en termes de diversité.

### 10.2.2 Étude des rôles relatifs de différents descripteurs d'images

Dans nos expériences sur le benchmark `Xilo`, nous avons observé que l'arborescence de concepts comme descripteur sur nos méthodes proposées permet d'obtenir de meilleurs résultats en diversité par rapport à une approche par la couleur, mais aussi par rapport à l'utilisation directe de l'arborescence de concepts (sans clustering) qui est une technique classique pour augmenter la diversité avec cette ressource. Pour le benchmark `Media` nous avons constaté que nos méthodes basées sur la couleur ont donné de meilleurs résultats en diversité par rapport au texte et aux coordonnées GPS.

De manière général, dans nos expériences, la couleur et notre approche d'utiliser une arborescence de concepts comme descripteur donnent globalement de meilleurs résultats en diversité par rapport au texte et aux coordonnées GPS. De plus, les résultats expérimentaux ont montré que l'arborescence de concepts comme descripteur donne des comportements très similaires à la couleur en sachant que cette arborescence génère un descripteur totalement différent des descripteurs de couleurs.

### 10.2.3 Influence de la pertinence initiale sur la diversité

Dans le benchmark `Media`, les résultats expérimentaux ont montré que le fait d'augmenter la pertinence initiale de la référence (baseline) par le texte, permet d'augmenter encore plus la diversité par un clustering visuel, par rapport à l'utilisation directe de la `AHC` : il semble que si l'on arrive à augmenter la pertinence initiale de baseline on pourrait augmenter encore plus la diversité de nos méthodes proposées. Il semblerait donc que la pertinence initiale de la baseline pourrait influer sur le comportement de la diversité de nos méthodes proposées.

D'autre part, quand nous étudié le comportement de la valeur maximale des méthodes de diversité, nous avons trouvé une seule exception à ce comportement, dans le cas réel du benchmark `Clef` qui a la particularité d'avoir une baseline avec beaucoup d'images non-pertinentes. Dans ce cas, le comportement devient instable, peut-être à cause de les images non-pertinentes.

## 10.3 Développement d'un système industriel

Un des résultats de notre thèse est le développement d'un prototype complet avec en particulier prise en compte des contraintes fortes de temps de calcul ce qui le rend adapté pour être utilisable dans le moteur de recherche de l'entreprise. Le prototype proposé a été développé dans le langage de l'entreprise et son déploiement dans le moteur de recherche de l'entreprise est en cours. La problématique industrielle du temps de calcul consiste à fournir une réponse rapide à une requête dans une application de recherche d'images en ligne. Les contraintes étudiées qui peuvent influencer le temps de calcul et la diversité sont les suivantes.

**Influence de la dimension des descripteurs** Nous avons étudié l'influence de la dimension de différents descripteurs sur le temps de calcul. Dans ces expérimentations, nous avons observé que l'influence de la dimension est plus significative pour les méthodes `K-Means` et `AHC` et moins significative sur `Maxmin` et `K-Medoids`. Donc, il est toujours préférable de choisir un descripteur avec peu de dimensions, spécialement pour `K-Means` et `AHC`.

**Influence du nombre d'itérations vs nombre de clusters** Pour les méthodes de clustering, nous avons aussi étudié l'influence du nombre de clusters $K$ et le nombre d'itération sur le temps de calcul. Dans ces expérimentations, nous avons observé que, contrairement aux autres méthodes de clustering, dans `K-Means` quand on utilise un nombre réduit d'images, le taux de croissance du temps de calcul ne semble pas être affecté par la taille du cluster $K$. En effet, quand on utilise plus de clusters, `K-Means` semble faire moins d'itérations (et vice-versa).

Toutefois quand on utilise de grandes quantités d'images, on peut voir l'influence de la taille de clusters avec `K-Means`. Donc, si on utilise un nombre de clusters $K$ assez grand, la méthode `K-Means` semblerait être le meilleur choix pour obtenir un temps de calcul raisonnable.

**Influence du nombre d'images** L'étude de l'influence du nombre d'images a montré que les méthodes `K-Means` et `Maxmin` sont les plus rapides en temps de calcul avec l'augmentation du nombre d'images par rapport aux méthodes `K-Medoids` et `AHC`. Cependant, nous avons vu aussi que quand les méthodes prennent un nombre d'images trop grand, l'augmentation de la diversité devient moins significative. En effet, en sachant que les résultats pertinents se trouvent dans les premiers résultats, il n'est pas nécessaire

de lancer ces méthodes sur beaucoup de résultats : il suffit de les lancer sur les premiers résultats, ce qui nous permet d'utiliser ces méthodes en un temps raisonnable.

Si l'on prend en compte ces contraintes, il semble préférable de choisir `Maxmin` ou `AHC`, qui sont les meilleures méthodes pour augmenter la diversité, mais sans prendre un nombre d'images trop élevé afin d'éviter d'être pénalisé par les images non-pertinentes et le temps de calcul

Dans le cas d'`AHC`, même si sa complexité dans le pire cas est quadratique, il existe plusieurs stratégies pour réduire ce temps de calcul : on peut calculer à l'avance les distances entre les images (étape offline) ou générer la hiérarchie de clusters jusqu'à $K$ clusters. On peut également diminuer le nombre d'itérations de la génération de la hiérarchie de clusters en fusionnant les couples de clusters qui ont des distances inférieures à un seuil déterminé.

## 10.4 Perspectives

Les résultats expérimentaux montrent que notre approche exploitant l'arborescence de concepts de Xilopix comme descripteur sur nos méthodes proposées ont permis d'obtenir de bons résultats en diversité. Cependant, ce comportement a été obtenu sur le seul benchmark `Xilo` que nous avons construit, car dans la littérature nous n'avons pas trouvé un benchmark public qui dispose d'une arborescence de concepts. Une proposition pour approfondir cette étude, est d'utiliser l'arborescence de concepts de Xilopix sur d'autres benchmarks afin de confirmer les résultats obtenus de manière plus exhaustive. Il faut souligner que cette proposition génère aussi d'autres difficultés comme l'utilisation d'une indexation automatique et efficace qui puisse associer les images de ces benchmarks dans l'arborescence de concepts de Xilopix afin de générer les descripteurs respectifs : le traitement automatique génère souvent des erreurs qui peuvent affecter les résultats obtenus en diversité par cette ressource.

Les résultats expérimentaux ont montré aussi qu'en général notre approche utilisant une arborescence de concepts comme descripteur obtient des comportements similaires à une approche par la couleur, même si l'arborescence de concepts génère un descripteur totalement différent des descripteurs de couleurs. Il serait donc intéressant d'étudier l'utilisation de la fusion de ces descripteurs afin d'examiner si cette approche permet d'augmenter encore plus la diversité.

Afin d'adapter l'utilisation de l'arborescence de concepts comme descripteur dans la recherche d'images, nous avons proposé une méthode de généralisation dans le cas où une image est associée à plusieurs concepts. Dans cette méthode, nous faisons l'hypothèse que, pour un univers donné, chaque image est décrite tout au plus par un seul concept. Il serait intéressant de proposer une méthode de similarité plus générale avec ce type de descripteur, s'affranchissant de cette hypothèse.

En ce qui concerne la stratégie d'ordonnancement des résultats après le clustering, afin d'utiliser les différents niveaux de granularité fournis par la `AHC`, nous avons proposé un réordonnancement hiérarchique où les images sont réordonnées en prenant dans l'ordre des clusters et en alternant les sous-clusters (obtenus en coupant une deuxième fois la hiérarchie de clusters), une image de chaque cluster. Même si, l'approche de réordonnancement classique semble suffisante pour obtenir la valeur maximale de diversité, notre approche de réordonnancement hiérarchique s'avère d'être une méthode intéressante pour augmenter la diversité dans certains cas. Il serait être intéressant d'approfondir cette étude, proposant une stratégie de coupure de la hiérarchie de clusters plus riche : on peut par exemple considérer ce type de réordonnancement hiérarchique, mais en alternant les sous-clusters provenant des différents niveaux de cette hiérarchie.

Dans le benchmark `Media`, nous avons vu que diversifier directement les images ré-

sultats de la baseline par la couleur obtient des bons résultats en diversité. Cependant, nous avons vu aussi que le fait d'augmenter la pertinence de la baseline par le texte, et puis diversifier les résultats par la couleur permet d'augmenter encore plus la diversité. Ces résultats soulignent la question de savoir si la variation de la pertinence initiale de la baseline pourrait influencer la diversité. Afin d'approfondir cette étude, nous souhaitons comparer différentes approches de diversité en utilisant différents runs (baselines) de pertinences initiales différentes. Une piste intéressante pour générer ces runs est de dégrader de manière automatique la pertinence initiale d'une baseline, par exemple, en rajoutant des images non-pertinentes. Une autre piste pourrait être de réordonner les images résultats de la baseline en utilisant différents types de descripteurs (du texte, du visuel, etc.) ou en alternant de manière aléatoire les images résultats de cette baseline.

# Annexe A

# Annexe Expérimentale

Dans cette annexe nous décrivons les informations et les résultats expérimentaux supplémentaires que nous avons réalisés afin de valider les comportements obtenus.

## A.1 Information générale sur chaque benchmark

Dans cette partie les tableaux A.2 et A.3 montrent les informations générales du benchmark `Media` et le tableau A.1 montre les informations générales des benchmarks `Xilo` et `Clef`. Entre les informations disponibles dans ces tableaux nous avons le nombre d'images par requête, le nombre d'images pertinentes par requête, le nombre de sous-thèmes par requête, la moyenne d'images par sous-thème, le nombre d'images dans chaque sous-thème, etc.

TABLE A.1 – Information générale sur la vérité terrain des experts des benchmarks `Xilo` et `Clef`

| | | `Xilo` | `Clef` (runsIdeaux) |
|---|---|---|---|
| Une étude par requête | | | |
| nombre de requêtes | | 21 | 39 |
| images | min | 100 | 18 |
| résultats | max | 100 | 184 |
| par requête | mean | **100.0** | **60.5** |
| | std | 0.0 | 30.8 |
| images | min | 100 | 18 |
| pertinentes | max | 100 | 184 |
| par requête | mean | **100.0** | **60.5** |
| | std | 0.0 | 30.8 |
| nb sous-thèmes | min | 6 | 2 |
| par requête | mean | **10.9** | **7.9** |
| | max | 16 | 23 |
| | std | 2.5 | 5.0 |
| moyenne | min | 6.3 | 2.5 |
| d'images | mean | **9.8** | **9.4** |
| par sous-thème | max | 16.7 | 31.0 |
| | std | 2.8 | 5.6 |

TABLE A.2 – Bencharmk `Media` : Information générale sur la vérité terrain des experts (sous-ensembles : `devset` et `testset`)

| | | devset | | | testset | | |
|---|---|---|---|---|---|---|---|
| | | devkGPS | devk | all | testkGPS | testk | all |
| Une étude par requête | | | | | | | |
| nombre de requêtes | | 25 | 25 | 50 | 210 | 132 | 342 |
| numeros de requêtes | | 1 à 25 | 26 à 50 | 1 à 50 | 51 à 261 | 262 à 396 | 51 à 396 |
| | | - | - | - | (sauf les requêtes 81 298 305 367) | | |
| images | min | 45 | 30 | 30 | 35 | 30 | 30 |
| résultats | max | 150 | 150 | 150 | 150 | 150 | 150 |
| par requête | mean | **113.5** | **91.2** | **102.4** | **117.2** | **100.2** | **110.6** |
| | std | 41.5 | 50.1 | 46.9 | 43.1 | 45.4 | 44.7 |
| | nbImg≤30 | 0 | 1 | 1 | 0 | 1 | 1 |
| | nbImg≤50 | 3 | 8 | 11 | 32 | 27 | 59 |
| | nbImg≤100 | 11 | 15 | 26 | 68 | 65 | 133 |
| images | min | 21 | 3 | 3 | 1 | 1 | 1 |
| pertinentes | mean | **89.2** | **62.3** | **75.8** | **87.6** | **56.7** | **75.7** |
| par requête | max | 142 | 142 | 142 | 150 | 148 | 150 |
| | std | 38.1 | 45.6 | 43.7 | 42.0 | 39.5 | 43.7 |
| | nbImgPert=1 | 0 | 0 | 0 | 1 | 1 | 2 |
| | nbImgPert≤5 | 0 | 2 | 2 | 2 | 3 | 5 |
| | nbImgPert≤10 | 0 | 2 | 2 | 4 | 12 | 16 |
| | nbImgPert≤20 | 0 | 4 | 4 | 10 | 20 | 30 |
| | nbImgPert≤50 | 6 | 13 | 19 | 59 | 69 | 128 |
| nb sous-thèmes | min | 7 | 3 | 3 | 1 | 1 | 1 |
| par requête | mean | **12.8** | **10.4** | **11.6** | **14.5** | **12.1** | **13.6** |
| | max | 19 | 18 | 19 | 20 | 20 | 20 |
| | std | 3.4 | 4.3 | 4.0 | 4.3 | 4.8 | 4.6 |
| | nbSubTheme=1 | 0 | 0 | 0 | 1 | 1 | 2 |
| | nbSubTheme≤3 | 0 | 2 | 2 | 3 | 4 | 7 |
| | nbSubTheme≤5 | 0 | 3 | 3 | 5 | 10 | 15 |
| | nbSubTheme≤10 | 8 | 12 | 20 | 38 | 54 | 92 |
| | nbSubTheme≤15 | 21 | 22 | 43 | 118 | 95 | 213 |
| moyenne | min | 2.1 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| d'images | mean | **7.4** | **5.4** | **6.4** | **5.9** | **4.3** | **5.3** |
| par sous-thème | max | 18.7 | 11.3 | 18.7 | 13.6 | 9.2 | 13.6 |
| | std | 4.1 | 2.8 | 3.6 | 2.4 | 2.0 | 2.4 |
| nb de requêtes qui ont | | | | | | | |
| un sous-thème | nbImg=1 | 15 | 16 | 31 | 171 | 124 | 295 |
| avec : | nbImg=2 | 20 | 19 | 39 | 193 | 119 | 312 |
| | nbImg≥10 | 20 | 14 | 34 | 170 | 76 | 246 |
| | nbImg≥15 | 16 | 8 | 24 | 126 | 48 | 174 |
| | nbImg≥20 | 13 | 8 | 21 | 94 | 29 | 123 |
| | nbImg≥30 | 6 | 4 | 10 | 43 | 9 | 52 |
| Une étude pour toutes les requêtes | | | | | | | |
| nb images résultats | | 2837 | 2281 | 5118 | 24607 | 13220 | 37827 |
| nb images pertinentes | | 2231 | 1558 | 3789 | 18403 | 7489 | 25892 |
| nb sous-thèmes | | 319 | 261 | 580 | 3050 | 1595 | 4645 |
| nombre d'images | min | 1 | 1 | 1 | 1 | 1 | 1 |
| dans chaque | mean | **7.0** | **6.0** | **6.5** | **6.0** | **4.7** | **5.6** |
| sous-thème | max | 47 | 46 | 47 | 69 | 56 | 69 |
| | std | 7.1 | 6.3 | 6.8 | 6.6 | 5.0 | 6.1 |
| nombre de | nbImg=1 | 22 | 27 | 49 | 414 | 377 | 791 |
| sous-thèmes | nbImg=2 | 49 | 42 | 91 | 552 | 305 | 857 |
| | nbImg=3 | 35 | 46 | 81 | 432 | 200 | 632 |
| | nbImg≤5 | 182 | 175 | 357 | 1972 | 1140 | 3112 |
| | nbImg≤10 | 266 | 225 | 491 | 2596 | 1439 | 4035 |
| | nbImg≤20 | 301 | 249 | 550 | 2928 | 1564 | 4492 |
| | nbImg≤30 | 311 | 256 | 567 | 3011 | 1587 | 4598 |

TABLE A.3 – Bencharmk `Media` : Information générale sur le **Crowd-Sourcing** (sous-ensembles `GT1`, `GT2` et `GT3`). Remarques : pour `GT1`, il y a 14 requêtes sur les 49 qui ont un seul cluster. `GT1`, `GT2`, `GT3` ont les mêmes images pertinentes

| | | Crowd-Sourcing | | | testset | | |
|---|---|---|---|---|---|---|---|
| | | GT1 | GT2 | GT3 | GT0 | testkGPS | testk |
| Une étude par requête | | | | | | | |
| nombre de requêtes | | 49 | | | | 210 | 135 |
| numeros de requêtes | | 51, 52, 54, 56-60, 62, 64-71, 74, 75, 77-80, 82-88, 93-95, 97, 109, 113, 134, 140, 153, 160, 164, 236, 255, 281, 308, 327, 336, 384, 392 (sauf 81) | | | | 51 à 261 (sauf 81 298 305 367) | 262 à 396 |
| images | min | 35 | | | | 35 | 30 |
| résultats | mean | **123.8** | | | | **117.2** | **100.2** |
| par requête | max | 150 | | | | 150 | 150 |
| | std | 40.6 | | | | 43.1 | 45.4 |
| nbImg≤30 | | 0 | | | | 0 | 1 |
| nbImg≤50 | | 5 | | | | 32 | 27 |
| nbImg≤100 | | 13 | | | | 68 | 65 |
| images | min | 15 | | | 2 | 1 | 1 |
| pertinentes | mean | **85.9** | | | **97.4** | **87.6** | **56.7** |
| par requête | max | 143 | | | 150 | 150 | 148 |
| | std | 38.0 | | | 42.2 | 42.0 | 39.5 |
| nbImgPert≤5 | | 0 | | | 1 | 2 | 3 |
| nbImgPert≤10 | | 0 | | | 1 | 4 | 12 |
| nbImgPert≤20 | | 2 | | | 1 | 10 | 20 |
| nbImgPert≤50 | | 9 | | | 10 | 59 | 69 |
| nb sous-thèmes | min | 1 | 1 | 1 | 2 | 1 | 1 |
| par requête | mean | **3.5** | **4.3** | **6.3** | **14.9** | **14.5** | **12.1** |
| | max | 14 | 12 | 16 | 20 | 20 | 20 |
| | std | 3.1 | 2.5 | 4.0 | 4.0 | 4.3 | 4.8 |
| nbSubTheme=1 | | **14 (29%)** | **10 (20%)** | 4 (8%) | 0 (0%) | 1 (0%) | 1 (1%) |
| nbSubTheme≤3 | | **33 (67%)** | 18 (37%) | 17 (35%) | 1 (2%) | 3 (1%) | 4 (3%) |
| nbSubTheme≤5 | | 40 (82%) | 33 (67%) | 23 (47%) | 2 (4%) | 5 (2%) | 10 (8%) |
| nbSubTheme≥10 | | 4 (8%) | **1 (2%)** | 13 (27%) | **47 (96%)** | **185 (88%)** | 87 (66%) |
| nbSubTheme≥15 | | 0 (0%) | 0 (0%) | 2 (4%) | 24 (49%) | 105 (50%) | 41 (31%) |
| moyenne | min | 5.6 | 5.3 | 2.6 | 1.0 | 1.0 | 1.0 |
| d'images par | mean | **43.6** | **30.8** | **24.3** | **6.5** | **5.9** | **4.3** |
| sous-thème | max | 136.0 | 124.0 | 139.0 | 13.6 | 13.6 | 9.2 |
| | std | 37.5 | 29.6 | 28.4 | 2.8 | 2.4 | 2.0 |
| nombre de requêtes qui ont un sous-thème avec : | | | | | | | |
| nbImg=1 | | 3 (6%) | 2 (4%) | 11 (22%) | 41 (84%) | 171 (81%) | 124 (94%) |
| nbImg≤3 | | 10 (20%) | 9 (18%) | 20 (41%) | 48 (98%) | 209 (100%) | 132 (100%) |
| nbImg≥20 | | 40 (82%) | 38 (78%) | 32 (65%) | 29 (59%) | 94 (45%) | 29 (22%) |
| nbImg≥40 | | 24 (49%) | 24 (49%) | 15 (31%) | 5 (10%) | 16 (8%) | 2 (2%) |
| Une étude pour toutes les requêtes | | | | | | | |
| nb images résultats | | 6067 | | | | 24607 | 13220 |
| nb images pertinentes | | 4211 | | | 4772 | 18403 | 7489 |
| nb sous-thèmes | | 173 | 210 | 310 | 729 | 3050 | 1595 |
| nombre d'images | min | 1 | 1 | 1 | 1 | 1 | 1 |
| dans chaque | mean | **24.5** | **20.1** | **13.7** | **6.5** | **6.0** | **4.7** |
| sous-thème | max | 136 | 124 | 139 | 69 | 69 | 56 |
| | std | 28.7 | 20.8 | 17.2 | 7.4 | 6.6 | 5.0 |
| nombre de sous-thèmes | nbImg=1 | 4 | 3 | 20 | 103 | 414 | 377 |
| | nbImg=2 | 8 | 4 | 12 | 137 | 552 | 305 |
| | nbImg=3 | 12 | 4 | 18 | 98 | 432 | 200 |
| | nbImg≤5 | 35 | 38 | 100 | 455 | 1972 | 1140 |
| | nbImg≤10 | 66 | 88 | 176 | 598 | 2596 | 1439 |
| | nbImg≤20 | 110 | 148 | 257 | 688 | 2928 | 1564 |
| | nbImg≤30 | 131 | 174 | 281 | 712 | 3011 | 1587 |

**Scores de la baseline des benchmarks** Dans cette partie nous montrons les scores de la méthode de référence (ou Baseline) dans chaque ensemble de données sur le benchmark `Media`. Les tableaux A.4 et A.5 montrent en détail les scores P@10, P@20, CR@10, CR@20 des méthodes de référence.

TABLE A.4 – Benchmark `Media` : Étude des scores de la baseline sur les sous-ensembles : `devset` et `testset`

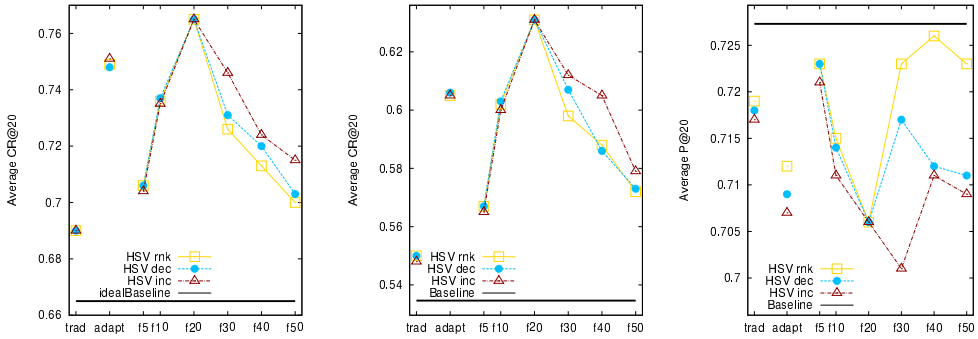
| | | devset | | | testset | | |
|---|---|---|---|---|---|---|---|
| | | keywordsGPS | keywords | all | keywordsGPS | keywords | all |
| nombre requêtes | | 25 | 25 | 50 | 210 | 132 | 342 |
| P@10 | min | 0.100 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| | max | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| | mean | **0.860** | **0.688** | **0.774** | **0.787** | **0.705** | **0.755** |
| | std | 0.196 | 0.306 | 0.269 | 0.242 | 0.300 | 0.268 |
| P@10=0 | | 0 (0%) | 1 (4%) | 1 (2%) | 3 (1%) | 8 (6%) | 11 (3%) |
| P@10≤0.3 | | 1 (4%) | 6 (24%) | 7 (14%) | 15 (7%) | 21 (16%) | 36 (11%) |
| P@10≥0.7 | | 23 (92%) | 14 (56%) | 37 (74%) | 165 (79%) | 87 (66%) | 252 (74%) |
| P@10=1 | | 10 (40%) | 7 (28%) | 17 (34%) | 73 (35%) | 33 (25%) | 106 (31%) |
| P@20 | min | 0.050 | 0.150 | 0.050 | 0.000 | 0.000 | 0.000 |
| | max | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| | mean | **0.864** | **0.674** | **0.769** | **0.770** | **0.660** | **0.727** |
| | std | 0.190 | 0.287 | 0.259 | 0.236 | 0.290 | 0.263 |
| P@20=0 | | 0 (0%) | 0 (0%) | 0 (0%) | 3 (1%) | 5 (4%) | 8 (2%) |
| P@20≤0.3 | | 1 (4%) | 5 (20%) | 6 (12%) | 16 (8%) | 21 (16%) | 37 (11%) |
| P@20≥0.7 | | 24 (96%) | 16 (64%) | 40 (80%) | 158 (75%) | 75 (57%) | 233 (68%) |
| P@20=1 | | 6 (24%) | 1 (4%) | 7 (14%) | 35 (17%) | 8 (6%) | 43 (13%) |
| CR@10 | min | 0.100 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| | max | 0.750 | 0.800 | 0.800 | 0.778 | 1.000 | 1.000 |
| | mean | **0.412** | **0.464** | **0.438** | **0.344** | **0.398** | **0.365** |
| | std | 0.170 | 0.202 | 0.187 | 0.150 | 0.196 | 0.171 |
| CR@10=0 | | 0 (0%) | 1 (4%) | 1 (2%) | 3 (1%) | 8 (6%) | 11 (3%) |
| CR@10≤0.2 | | 2 (8%) | 2 (8%) | 4 (8%) | 33 (16%) | 17 (13%) | 50 (15%) |
| CR@10≤0.3 | | 6 (24%) | 7 (28%) | 13 (26%) | 99 (47%) | 41 (31%) | 140 (41%) |
| CR@10≥0.5 | | 7 (28%) | 13 (52%) | 20 (40%) | 36 (17%) | 40 (30%) | 76 (22%) |
| CR@10≥0.7 | | 3 (12%) | 3 (12%) | 6 (12%) | 7 (3%) | 10 (8%) | 17 (5%) |
| CR@10=1 | | 0 (0%) | 0 (0%) | 0 (0%) | 0 (0%) | 2 (2%) | 2 (1%) |
| CR@20 | min | 0.100 | 0.182 | 0.100 | 0.000 | 0.000 | 0.000 |
| | max | 0.900 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| | mean | **0.608** | **0.663** | **0.635** | **0.510** | **0.574** | **0.535** |
| | std | 0.198 | 0.232 | 0.215 | 0.177 | 0.234 | 0.203 |
| CR@20=0 | | 0 (0%) | 0 (0%) | 0 (0%) | 3 (1%) | 5 (4%) | 8 (2%) |
| CR@20≤0.3 | | 2 (8%) | 1 (4%) | 3 (6%) | 25 (12%) | 17 (13%) | 42 (12%) |
| CR@20≥0.5 | | 19 (76%) | 19 (76%) | 38 (76%) | 111 (53%) | 94 (71%) | 205 (60%) |
| CR@20≥0.7 | | 8 (32%) | 9 (36%) | 17 (34%) | 28 (13%) | 40 (30%) | 68 (20%) |
| CR@20≥0.9 | | 1 (4%) | 6 (24%) | 7 (14%) | 7 (3%) | 11 (8%) | 18 (5%) |
| CR@20=1 | | 0 (0%) | 5 (20%) | 5 (10%) | 3 (1%) | 8 (6%) | 11 (3%) |

TABLE A.5 – Benchmark `Media` : Étude des scores de la baseline sur le **Crowd-Sourcing**

| | | Crowd-Sourcing | | | `testset` | | |
|---|---|---|---|---|---|---|---|
| | | `GT1` | `GT2` | `GT3` | `GT0` | `keywordsGPS` | `keywords` |
| nombre de requêtes | | 49 | 49 | 49 | 49 | 210 | 132 |
| P@10 | min | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| | max | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| | mean | **0.682** | **0.682** | **0.682** | **0.798** | **0.787** | **0.705** |
| | std | 0.303 | 0.303 | 0.303 | 0.244 | 0.242 | 0.300 |
| | P@10=0 | 2 (4%) | 2 (4%) | 2 (4%) | 1 (2%) | 3 (1%) | 8 (6%) |
| | P@10≤0.3 | 9 (18%) | 9 (18%) | 9 (18%) | 2 (4%) | 15 (7%) | 21 (16%) |
| | P@10≥0.7 | 31 (63%) | 31 (63%) | 31 (63%) | 37 (76%) | 165 (79%) | 87 (66%) |
| | P@10=1 | 11 (22%) | 11 (22%) | 11 (22%) | 20 (41%) | 73 (35%) | 33 (25%) |
| P@20 | min | 0.150 | 0.150 | 0.150 | 0.000 | 0.000 | 0.000 |
| | max | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| | mean | **0.706** | **0.706** | **0.706** | **0.804** | **0.770** | **0.660** |
| | std | 0.247 | 0.247 | 0.247 | 0.225 | 0.236 | 0.290 |
| | P@20=0 | 0 (0%) | 0 (0%) | 0 (0%) | 1 (2%) | 3 (1%) | 5 (4%) |
| | P@20≤0.3 | 5 (10%) | 5 (10%) | 5 (10%) | 3 (6%) | 16 (8%) | 21 (16%) |
| | P@20≥0.7 | 30 (61%) | 30 (61%) | 30 (61%) | 39 (80%) | 158 (75%) | 75 (57%) |
| | P@20=1 | 4 (8%) | 4 (8%) | 4 (8%) | 12 (24%) | 35 (17%) | 8 (6%) |
| CR@10 | min | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| | max | 1.000 | 1.000 | 1.000 | 0.643 | 0.778 | 1.000 |
| | mean | **0.754** | **0.667** | **0.572** | **0.334** | **0.344** | **0.398** |
| | std | 0.325 | 0.305 | 0.283 | 0.149 | 0.150 | 0.196 |
| | CR@10=0 | 2 (4%) | 2 (4%) | 2 (4%) | 1 (2%) | 3 (1%) | 8 (6%) |
| | CR@10≤0.1 | 2 (4%) | 2 (4%) | 2 (4%) | 3 (6%) | 8 (4%) | 8 (6%) |
| | CR@10≤0.2 | 3 (6%) | 4 (8%) | 2 (4%) | 9 (18%) | 33 (16%) | 17 (13%) |
| | CR@10≤0.3 | 9 (18%) | 6 (12%) | 7 (14%) | 23 (47%) | 99 (47%) | 41 (31%) |
| | CR@10≥0.5 | 38 (78%) | 37 (76%) | 29 (59%) | 9 (18%) | 36 (17%) | 40 (30%) |
| | CR@10≥0.7 | 30 (61%) | 25 (51%) | 14 (29%) | 0 (0%) | 7 (3%) | 10 (8%) |
| | CR@10=1 | 28 (57%) | 17 (35%) | 11 (22%) | 0 (0%) | 0 (0%) | 2 (2%) |
| CR@20 | min | 0.333 | 0.250 | 0.333 | 0.000 | 0.000 | 0.000 |
| | max | 1.000 | 1.000 | 1.000 | 0.923 | 1.000 | 1.000 |
| | mean | **0.874** | **0.805** | **0.757** | **0.501** | **0.510** | **0.574** |
| | std | 0.210 | 0.215 | 0.231 | 0.192 | 0.177 | 0.234 |
| | CR@20=0 | 0 (0%) | 0 (0%) | 0 (0%) | 1 (2%) | 3 (1%) | 5 (4%) |
| | CR@20≤0.3 | 0 (0%) | 1 (2%) | 0 (0%) | 9 (18%) | 25 (12%) | 17 (13%) |
| | CR@20≥0.5 | 48 (98%) | 45 (92%) | 42 (86%) | 25 (51%) | 111 (53%) | 94 (71%) |
| | CR@20≥0.7 | 37 (76%) | 32 (65%) | 28 (57%) | 6 (12%) | 28 (13%) | 40 (30%) |
| | CR@20=1 | 35 (71%) | 22 (45%) | 19 (39%) | 0 (0%) | 3 (1%) | 8 (6%) |

## A.2 Scores complémentaires de la AHC sur la diversité

**Scores CR@20 des techniques de priorité avec la AHC** Dans cette section nous comparons les trois techniques de priorité avec la AHC en utilisant différentes techniques de découpage. Le tableau A.1 montre les résultats obtenus de la AHC avec le descripteur HSV sur plusieurs benchmarks. Le tableau A.2 montrent les résultats obtenus de la AHC avec le descripteur Tree sur le benchmark Xilo et les descripteurs CN3x3, TfIdf, GPS sur le benchmark Media.

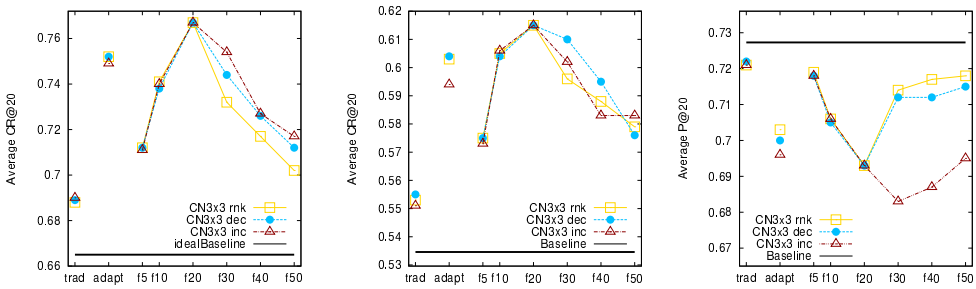



(a) AHC(HSV) (Media-idéal) (b) AHC(HSV) (Media-réel)

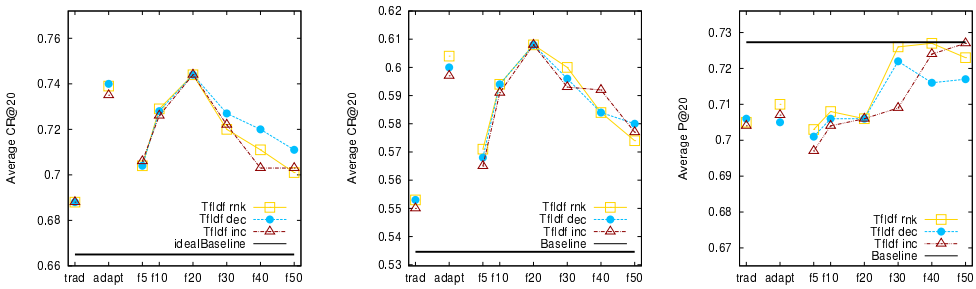
(c) AHC(HSV) (Clef-idéal) (d) AHC(HSV) (Clef-réel)

(e) AHC(HSV) (Xilo-idéal)

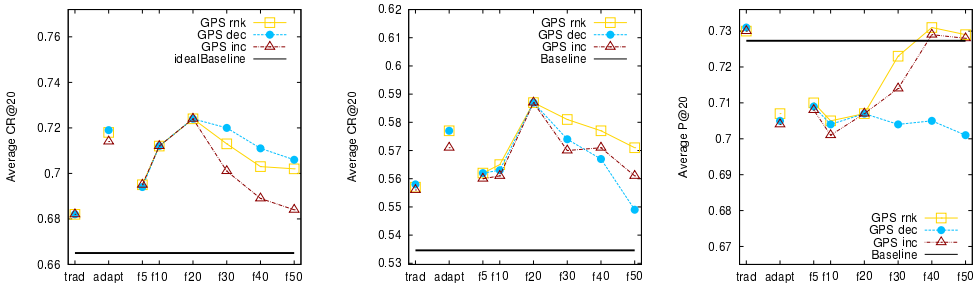
FIGURE A.1 – Comparaison des techniques de priorité sur plusieurs benchmarks. Les scores CR@20 et P@20 moyens représentent la AHC avec le descripteur HSV en utilisant différentes techniques de découpage (en abscisse) dans un cas idéal et réel sur les benchmarks (a) (b) Media (sous-ensemble testset), (c) (d) Clef et (e) Xilo



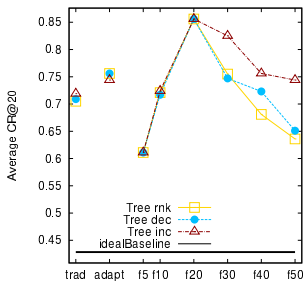

(a) AHC(CN3x3) (Media-idéal) (b) AHC(CN3x3) (Media-réel)




(c) AHC(TfIdf) (Media-idéal) (d) AHC(TfIdf) (Media-réel)



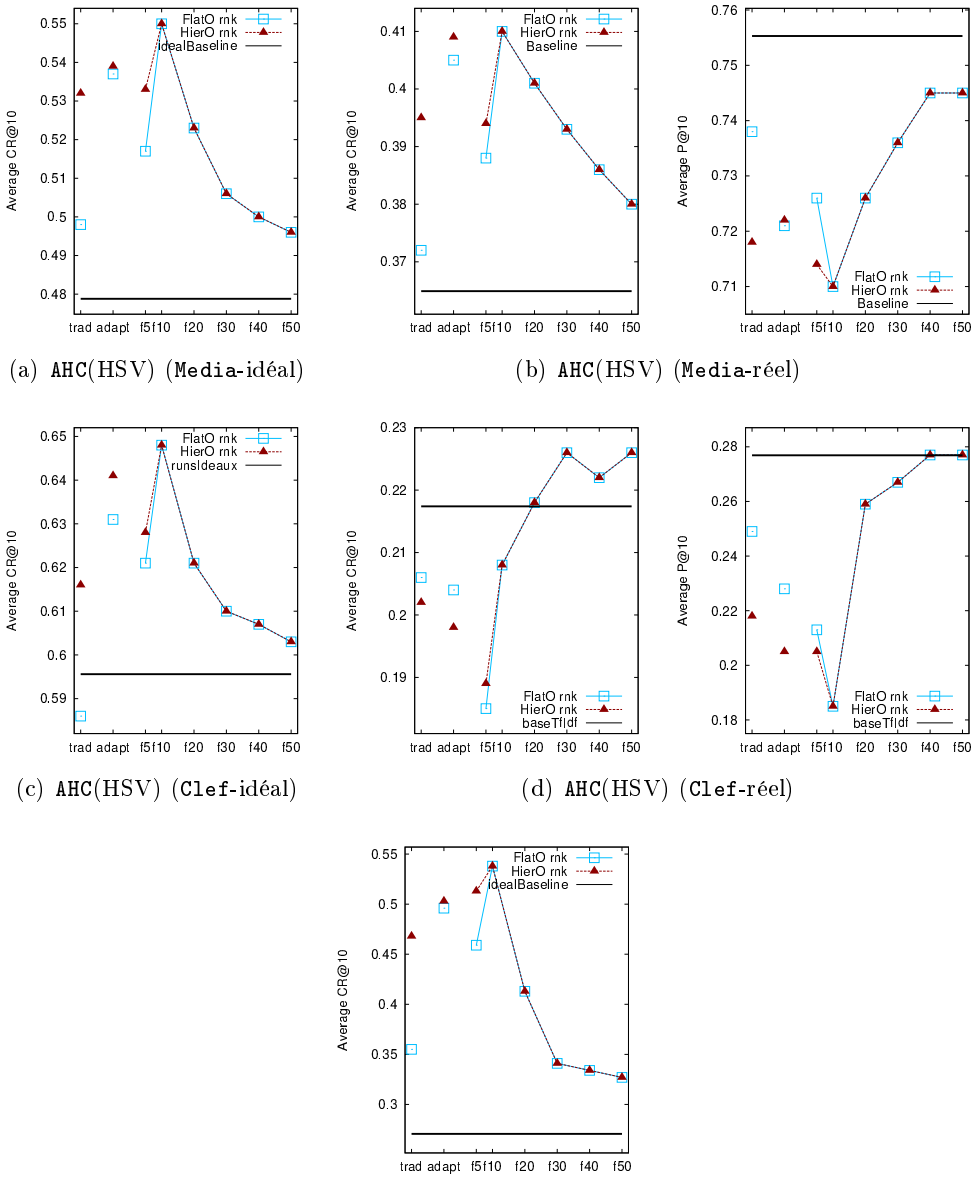

(e) AHC(GPS) (Media-idéal) (f) AHC(GPS) (Media-réel)




(g) AHC(Tree) (Xilo-idéal)

FIGURE A.2 – Comparaison des techniques de priorité sur plusieurs types de descripteurs. Les scores CR@20 et P@20 moyens représentent la AHC avec le descripteur (g) Tree sur le benchmark Xilo et les descripteurs (a) (b) CN3x3, (c) (d) TfIdf et (e) (f) GPS sur le benchmark Media en utilisant le découpage Fixe (en abscisse) dans un cas idéal et réel

**Scores CR@10, CR@40 et CR@50 du réordonnancement avec `AHC`** Dans cette section nous comparons les deux techniques de réordonnancement (`FlatO` et `HierO`) avec la `AHC` en utilisant différentes techniques de découpage sur plusieurs benchmarks. Les tableaux A.3, A.4 et A.5 montrent les résultats CR@10, CR@40 et CR@50 de la `AHC` avec le descripteur HSV.

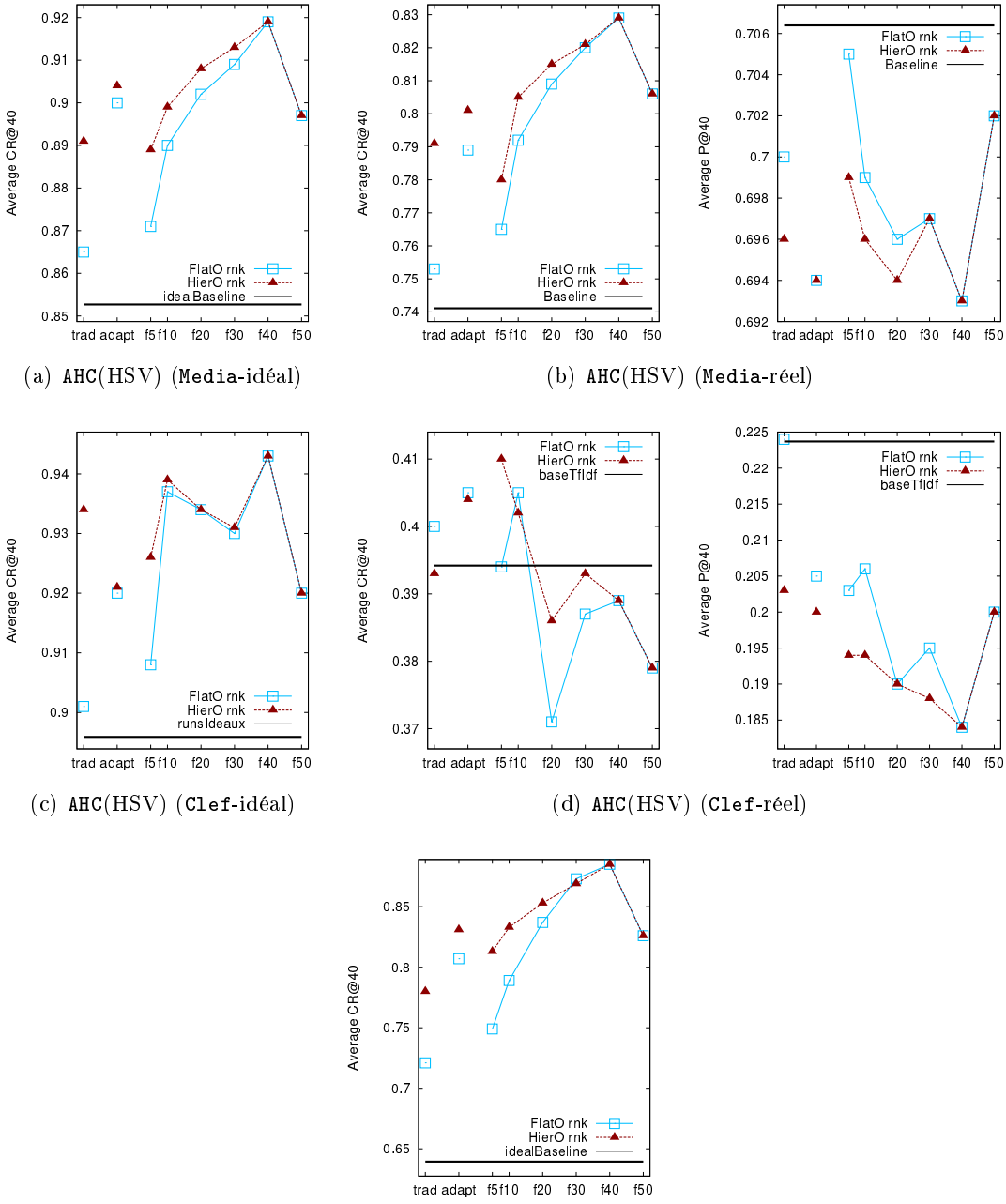



(a) `AHC`(HSV) (`Media`-idéal) (b) `AHC`(HSV) (`Media`-réel)

(c) `AHC`(HSV) (`Clef`-idéal) (d) `AHC`(HSV) (`Clef`-réel)

(e) `AHC`(HSV) (`Xilo`-idéal)

FIGURE A.3 – Comparaison des techniques de réordonnancement sur plusieurs benchmarks pour CR@10. Les scores CR@10 et P@10 moyens représentent la `AHC` avec le descripteur HSV en utilisant différentes techniques de découpage (en abscisse) et une priorité classique `Rank` dans un cas idéal et réel sur les benchmarks (a) (b) `Media` (sous-ensemble `testset`), (c) (d) `Clef` et (e) `Xilo`




(a) AHC(HSV) (Media-idéal)

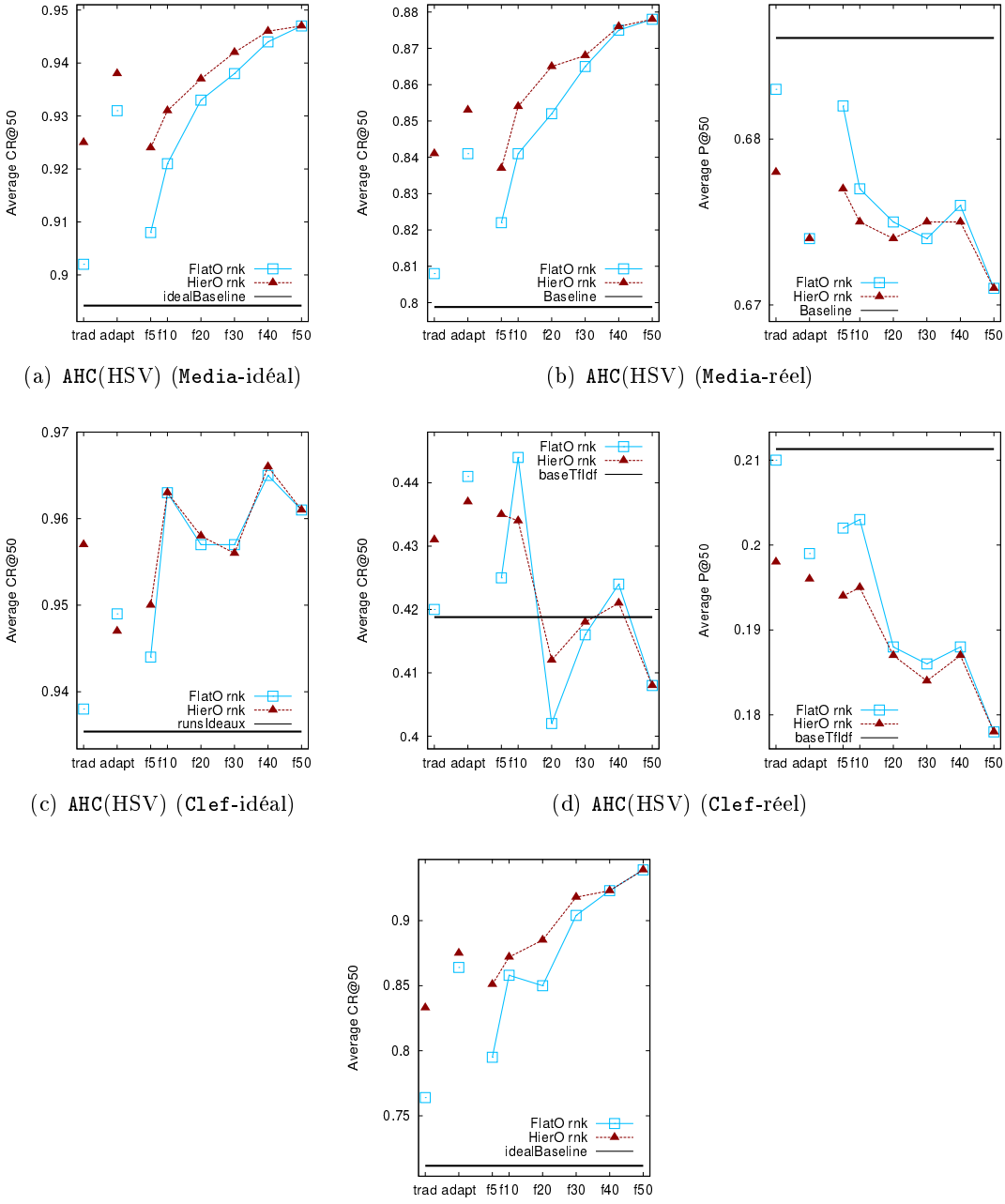
(b) AHC(HSV) (Media-réel)

(c) AHC(HSV) (Clef-idéal)

(d) AHC(HSV) (Clef-réel)

(e) AHC(HSV) (Xilo-idéal)

FIGURE A.4 – Comparaison des techniques de réordonnancement sur plusieurs benchmarks pour CR@40. Les scores CR@40 et P@40 moyens représentent la AHC avec le descripteur HSV en utilisant différentes techniques de découpage (en abscisse) et une priorité classique Rank dans un cas idéal et réel sur les benchmarks (a) (b) Media (sous-ensemble testset), (c) (d) Clef et (e) Xilo




(a) `AHC`(HSV) (`Media`-idéal)

(b) `AHC`(HSV) (`Media`-réel)

(c) `AHC`(HSV) (`Clef`-idéal)

(d) `AHC`(HSV) (`Clef`-réel)

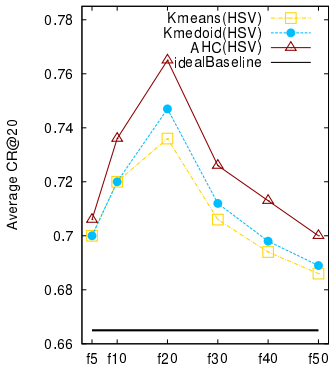
(e) `AHC`(HSV) (`Xilo`-idéal)

FIGURE A.5 – Comparaison des techniques de réordonnancement sur plusieurs benchmarks pour CR@50. Les scores CR@50 et P@50 moyens représentent la `AHC` avec le descripteur HSV en utilisant différentes techniques de découpage (en abscisse) et une priorité classique `Rank` dans un cas idéal et réel sur les benchmarks (a) (b) `Media` (sous-ensemble `testset`), (c) (d) `Clef` et (e) `Xilo`

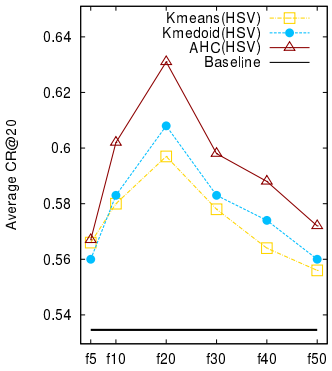
## A.3 Scores complémentaires de la valeur maximale de diversité

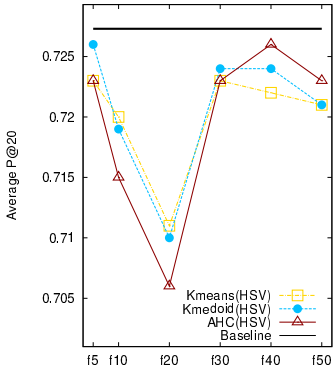
Dans cette section nous comparons les scores CR@20 et P@20 des quatre méthodes de diversité : une méthode de clustering hiérarchique (`AHC`), deux méthodes de clustering plat (`K-Means` et `K-Medoids`) et un algorithme glouton (`Maxmin`) en utilisant les paramètres de la valeur maximale de diversité (voir tableau 9.1 page 139).

**Scores CR@20 du comportement de la valeur maximale de diversité** Les figures A.6 et A.7 montrent les comportements pour obtenir la valeur maximale de diversité dans les méthodes de diversité sur plusieurs benchmarks et sur plusieurs descripteurs.

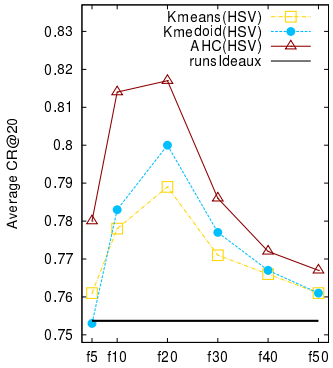



(a) Clustering(HSV) (`Media`-idéal)

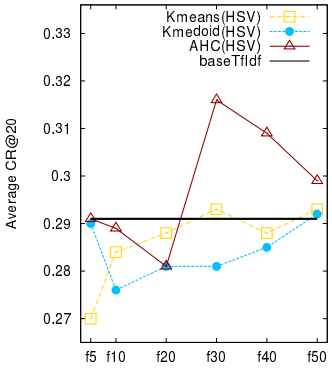


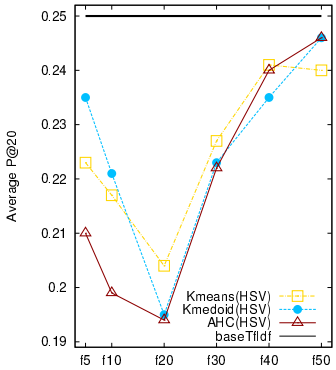




(b) Clustering(HSV) (`Media`-réel)

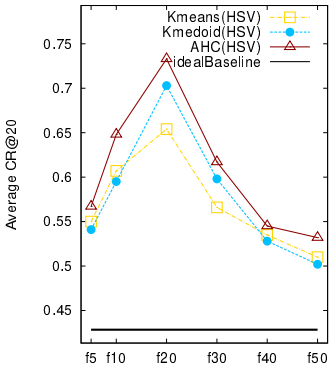



(c) Clustering(HSV) (`Clef`-idéal)



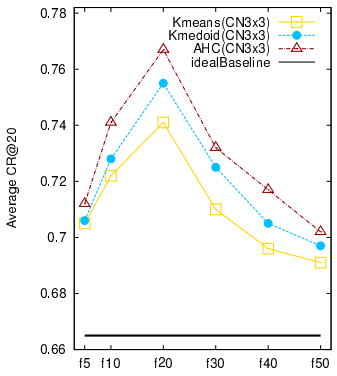




(d) Clustering(HSV) (`Clef`-réel)

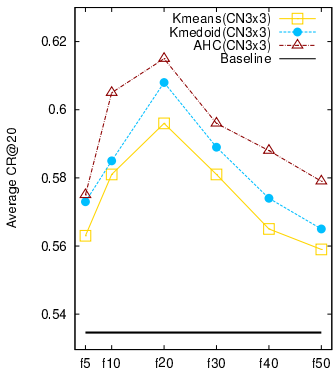


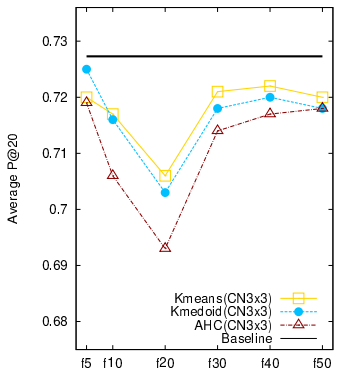

(e) Clustering(HSV) (`Xilo`-idéal)

FIGURE A.6 – Comparaison des méthodes de clustering sur plusieurs benchmarks. Les scores CR@20, P@20 moyens représentent les méthodes `K-Means`, `K-Medoids` et `AHC` avec HSV en utilisant différents découpages (en abscisse) et une priorité `Rank` dans un cas idéal et réel sur les benchmarks (a) (b) `Media-testset`, (c) (d) `Clef` et (e) `Xilo`

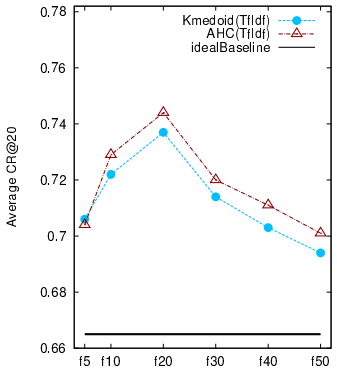



(a) Clustering(CN3x3) (`Media`-idéal)

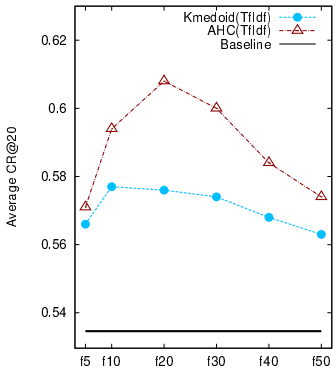


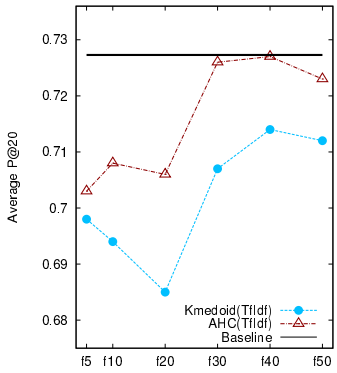




(b) Clustering(CN3x3) (`Media`-réel)

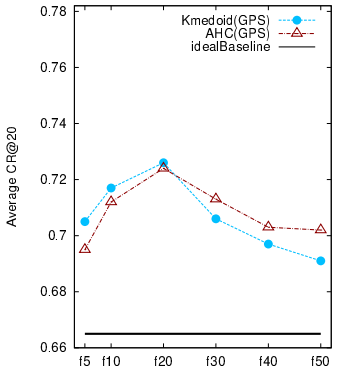



(c) Clustering(TfIdf) (`Media`-idéal)

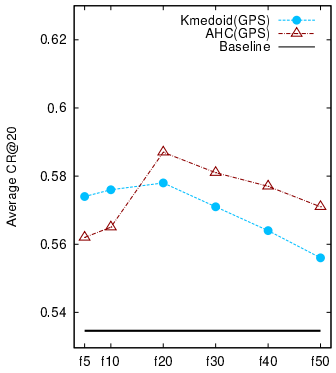


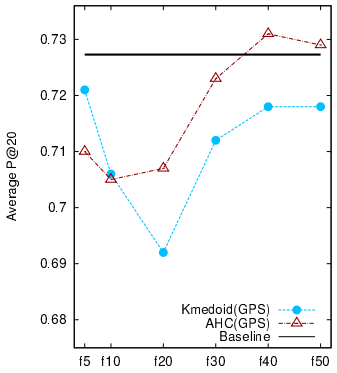




(d) Clustering(TfIdf) (`Media`-réel)




(e) Clustering(GPS) (`Media`-idéal)

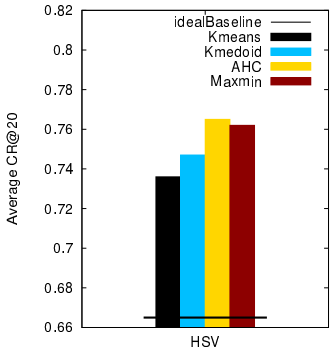


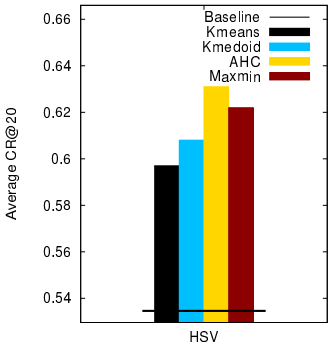

(f) Clustering(GPS) (`Media`-réel)

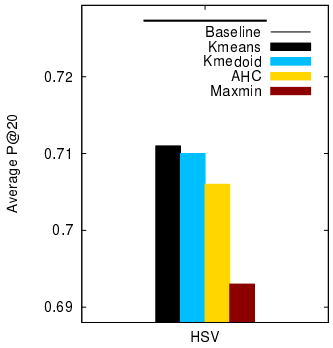
FIGURE A.7 – Comparaison des méthodes de clustering sur plusieurs descripteurs. Les scores CR@20 et P@20 moyens représentent les méthodes `K-Means`, `K-Medoids` et `AHC` avec les descripteurs (a) (b) CN3x3, (c) (d) TfIdf et (e) (f) GPS en utilisant différentes techniques de découpage (en abscisse) et une priorité par `Rank` dans un cas idéal et réel sur le benchmark `Media` (sous-ensemble `testset`)

**Scores CR@20 de la valeur maximale de diversité en général** Dans les figures A.8 et A.9, nous montrons les scores de la valeur maximale de diversité obtenus dans les méthodes de diversité sur plusieurs benchmarks et sur plusieurs descripteurs.

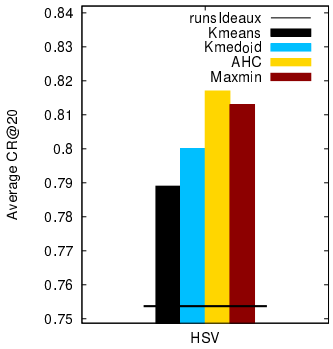



(a) Diversity (`Media`-idéal)

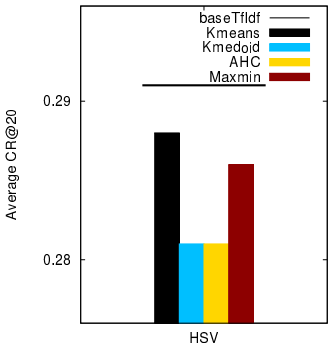


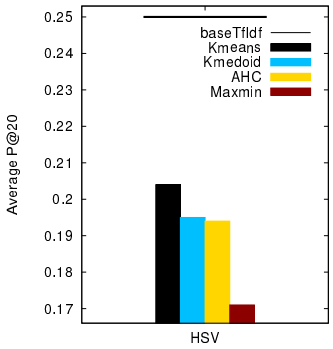




(b) Diversity (`Media`-réel)

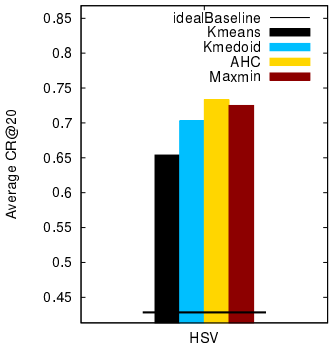



(c) Diversity (`Clef`-idéal)



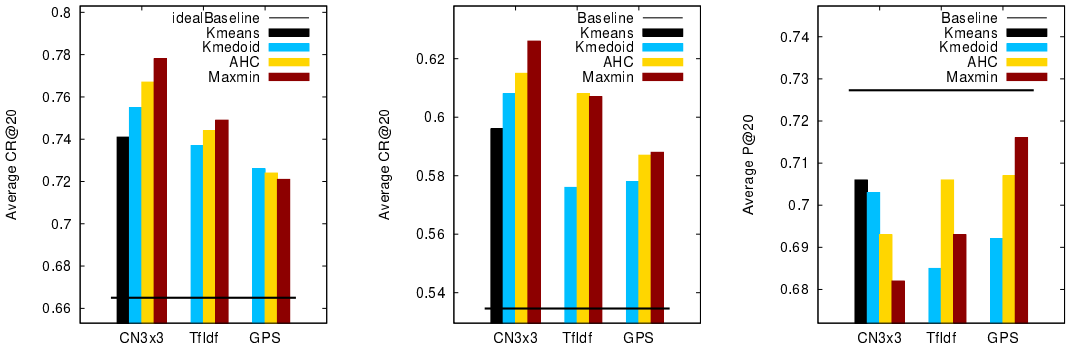




(d) Diversity (`Clef`-réel)




(e) Diversity (`Xilo`-idéal)

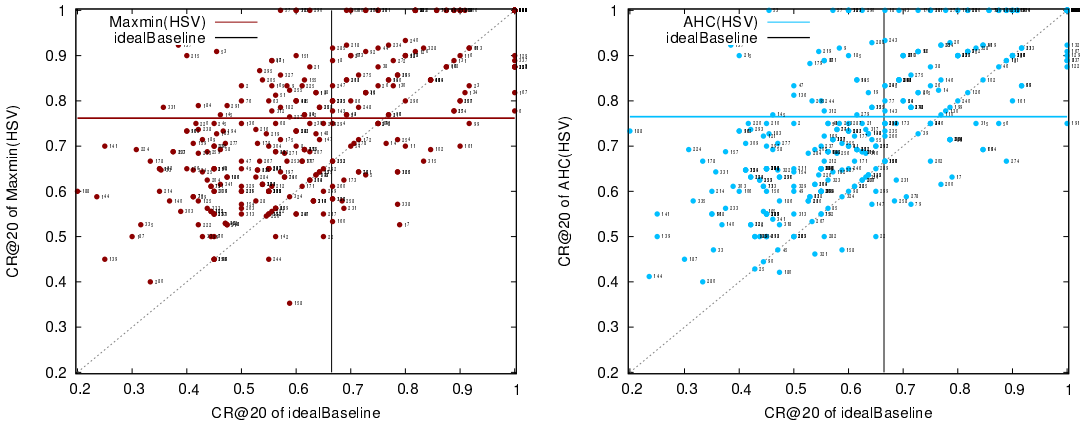
FIGURE A.8 – Comparaison des méthodes de diversité sur plusieurs benchmarks. Les scores CR@20 et P@20 moyens représentent les méthodes `K-Means`, `K-Medoids`, `AHC` et `Maxmin`avec le descripteur HSV en utilisant les paramètres mentionnées dans le tableau 9.1 page 139 dans un cas idéal et réel sur les benchmarks (a) (b) `Media` (sous-ensemble `testset`), (c) (d) `Clef` et (e) `Xilo`



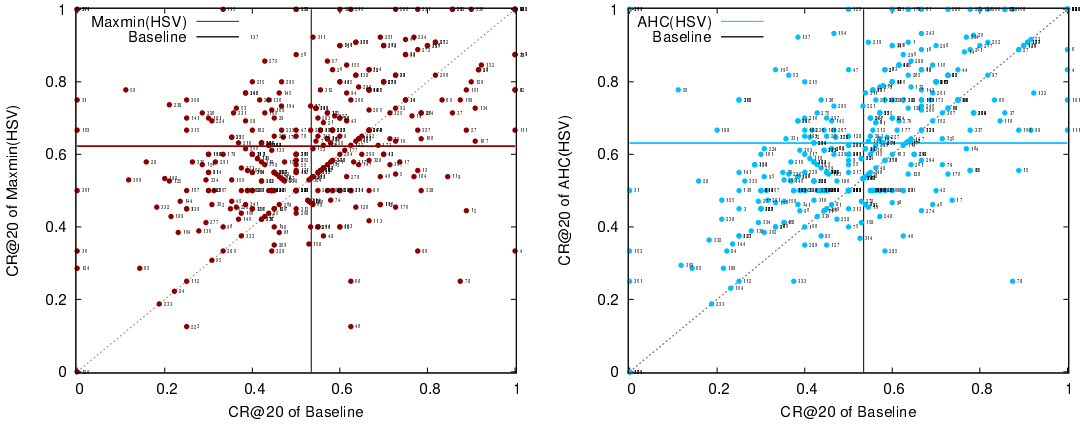

(a) Diversity (`Media`-idéal)

(b) Diversity (`Media`-réel)

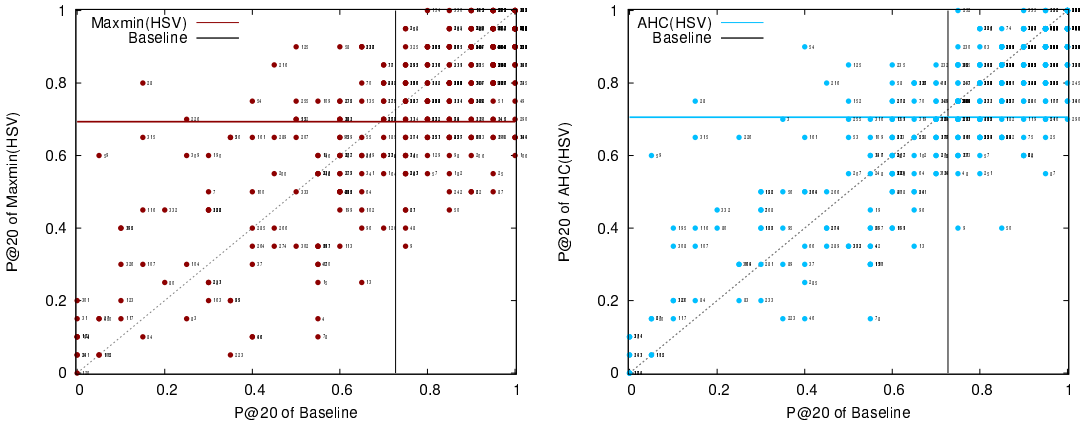
FIGURE A.9 – Comparaison des méthodes de diversité sur plusieurs descripteurs. Les scores CR@20 et P@20 moyens représentent les méthodes `K-Means`, `K-Medoids`, `AHC` et `Maxmin`avec les descripteurs (a) (b) CN3x3, (c) (d) TfIdf et (e) (f) GPS en utilisant les paramètres mentionnées dans le tableau 9.1 page 139 dans un cas idéal et réel sur le benchmark `Media` (sous-ensemble `testset`)




(a) Scores CR@10 des méthodes de diversité vs idealBaseline (`Media`-idéal)




(b) Scores CR@10 des méthodes de diversité vs Baseline (`Media`-réel)




(c) Scores P@10 des méthodes de diversité vs Baseline (`Media`-idéal)

FIGURE A.10 – Comparaison des scores par requêtes entre "`Maxmin` versus Baseline" (à gauche) et "`AHC` versus Baseline" (à droite). En ordonnée, les scores CR@20, P@20 par requête représentent les méthodes `Maxmin` et `AHC` avec le descripteur HSV en utilisant la priorité `Rank` et le découpage `Fixe` à 20 clusters dans un cas idéal et réel sur le benchmark `Media` (sous-ensemble `testset`)

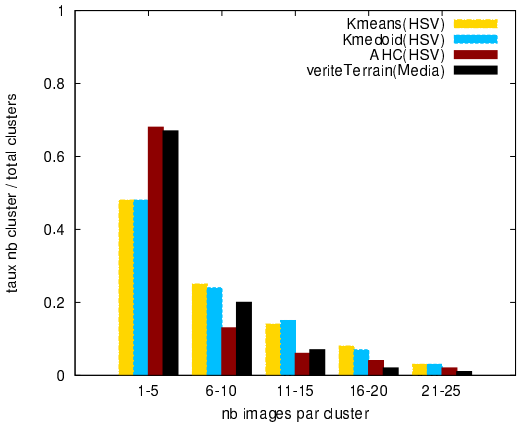
**Scores CR@20 de la valeur maximale de diversité par requête**

TABLE A.6 – Calcul de la distance sur la diagonale entre les scores CR@20 de "`Maxmin` versus Baseline" et "`AHC` versus Baseline" en utilisant le descripteur HSV, le découpage `Fixe` à 20 clusters et une priorité par `Rank` dans un cas réel et idéal sur le benchmark `Media` (sous-ensemble `testset`)

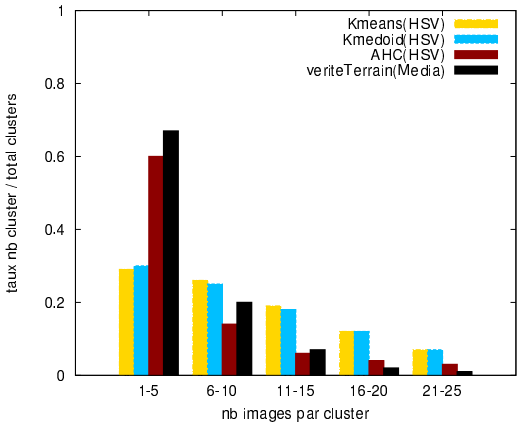
| Méthodes | distance sur la diagonale (cas réel) | (cas idéal) |
|---|---|---|
| `Maxmin`(HSV) vs Baseline | 39.25 | 30.77 |
| `AHC`(HSV) vs Baseline | 34.81 | 29.20 |

## A.4 Distribution de clusters générés dans les méthodes de clustering

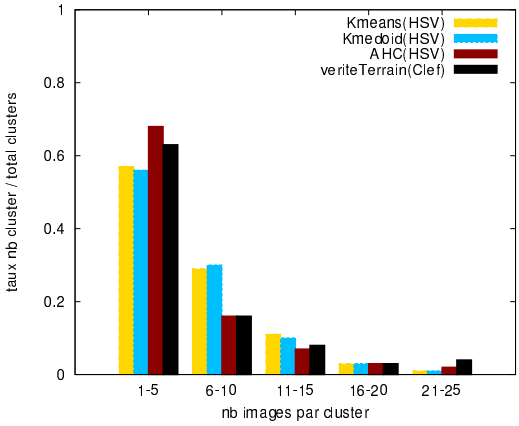
Dans cette section nous comparons le taux de la distribution de la taille des sous-thèmes (en fonction du nombre d'images) de la vérité terrain avec celui de la distribution des clusters générés par les méthodes de clustering en utilisant 10 clusters. Les figures A.11 et A.12 montrent la comparaison des distributions de clusters dans les méthodes de clustering avec le descripteur HSV sur plusieurs benchmarks et sur plusieurs descripteurs.



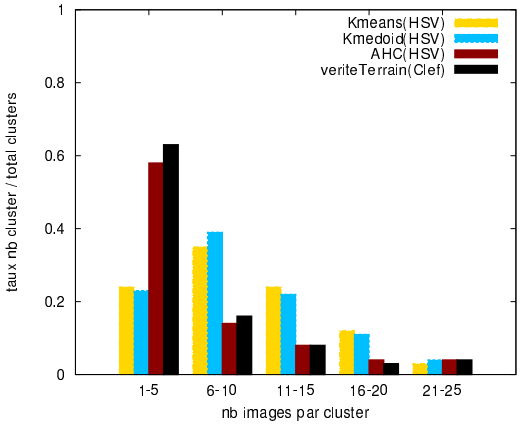

(a) Clustering(HSV) (`Media`-idéal)



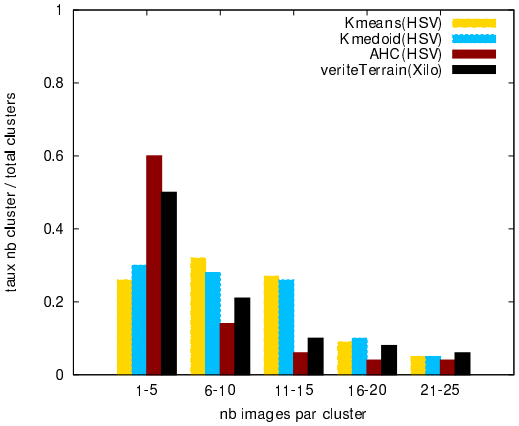

(b) Clustering(HSV) (`Media`-réel)




(c) Clustering(HSV) (`Clef`-idéal)

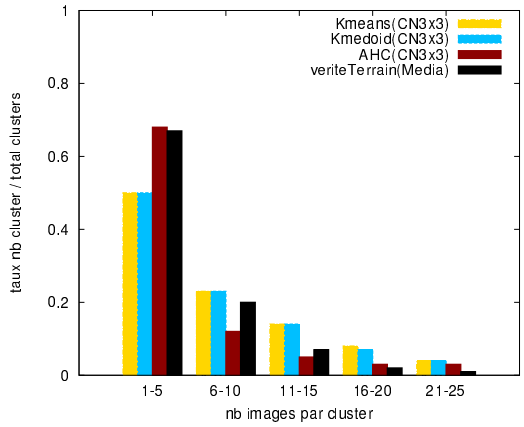



(d) Clustering(HSV) (`Clef`-réel)

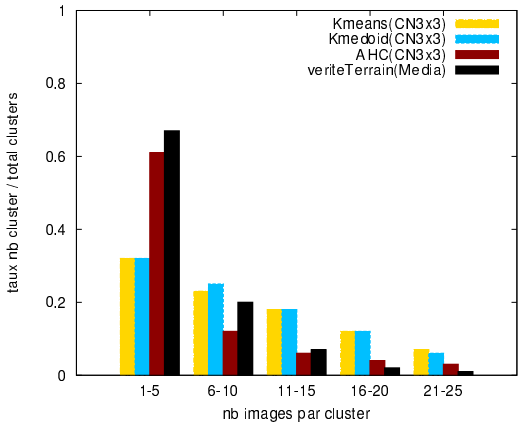



(e) Clustering(HSV) (`Xilo`-idéal)

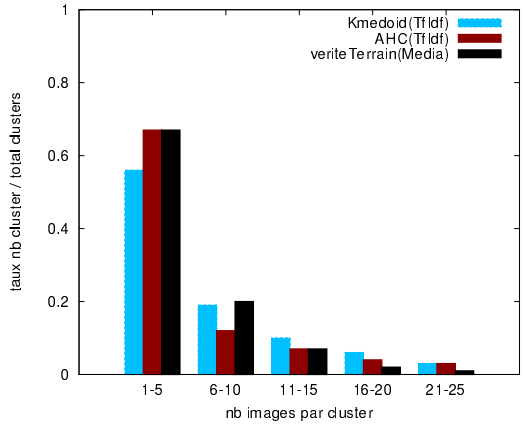
FIGURE A.11 – Comparaison du taux de distribution des sous-thèmes de la vérité terrain avec le taux de distribution des clusters générés par les méthodes `K-Means`, `K-Medoids` et `AHC` avec le descripteur HSV et $K = 10$ dans un cas idéal et réel sur les benchmarks (a) (b) `Media-testset`, (c) (d) `Clef` et (e) `Xilo`



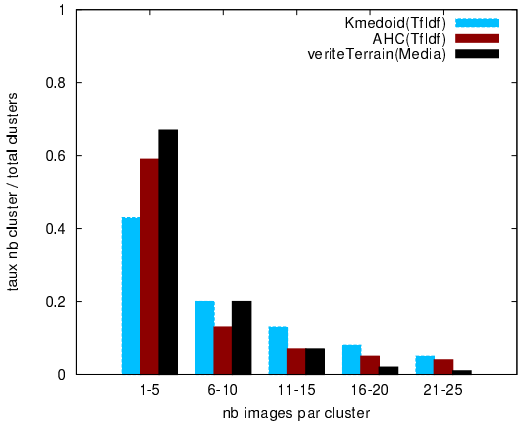

(a) Clustering(CN3x3) (`Media`-idéal)



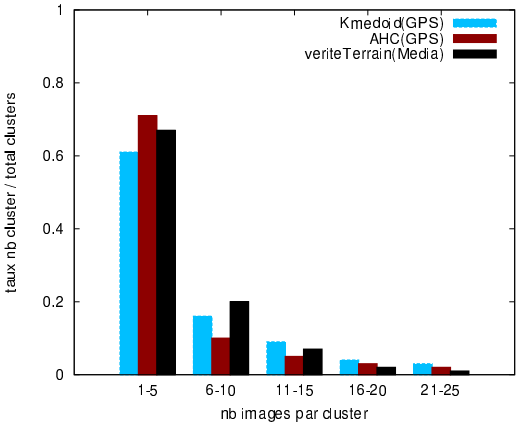

(b) Clustering(CN3x3) (`Media`-réel)



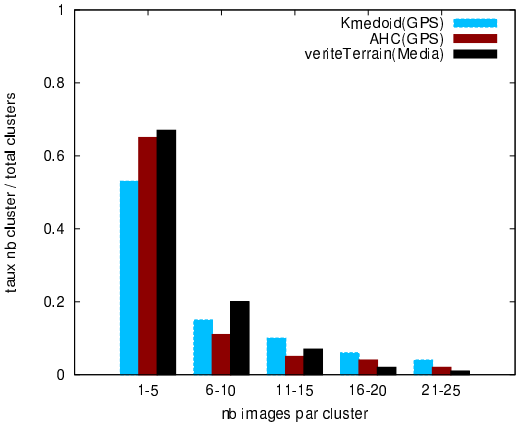

(c) Clustering(TfIdf) (`Media`-idéal)




(d) Clustering(TfIdf) (`Media`-réel)




(e) Clustering(GPS) (`Media`-idéal)




(f) Clustering(GPS) (`Media`-réel)

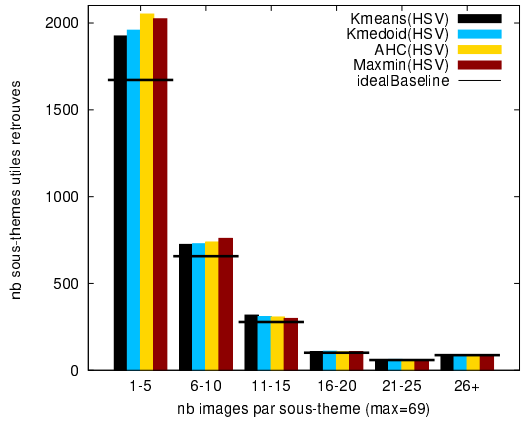
FIGURE A.12 – Comparaison du taux de distribution des sous-thèmes de la vérité terrain avec le taux de distribution des clusters générés par les méthodes `K-Means`, `K-Medoids` et `AHC` avec les descripteurs (a) (b) CN3x3, (c) (d) TfIdf et (e) (f) GPS en utilisant un découpage `Fixe` à 10 clusters dans un cas idéal et réel sur le benchmark `Media` (sous-ensemble `testset`)

## A.5 Apport des sous-thèmes retrouvés dans les méthodes de diversité

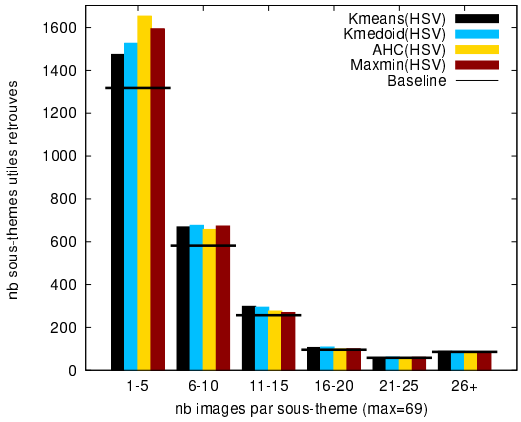
Dans cette section nous comparons l'apport de sous-thèmes retrouvés dans les 20 premiers résultats des méthodes de diversité en utilisant les paramètres de la valeur maximale de diversité (voir tableau 9.1 page 139). Le tableau A.7 montre l'apport de sous-thèmes retrouvés dans `Maxmin`et dans les méthodes de clustering avec 10, 20 et 30 clusters. Les figures A.13 et A.14 montrent un apport plus spécifique de sous-thèmes utiles retrouvés (par exemple '1-5' représente les sous-thèmes retrouvés qui ont entre 1 et 5 images) dans les méthodes de clustering avec 20 clusters sur plusieurs benchmarks et sur plusieurs descripteurs.

TABLE A.7 – Apport de sous-thèmes retrouvés dans les 20 premiers résultats de chaque méthode de diversité sur le benchmark `Media`-réel (sous-ensemble `testset`)

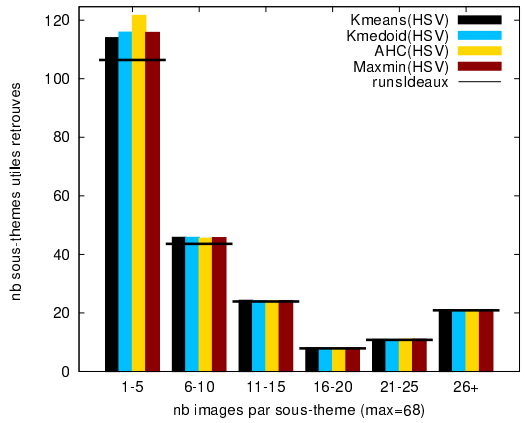
| | | sous-thèmes retrouvés | | | |
|---|---|---|---|---|---|
| $K$ | Méthode | total | utile | redondant | bruit |
| - | Baseline | 6840 | 2397(35%) | 2578(38%) | 1865(27%) |
| - | Maxmin(HSV) | 6840 | **2782(41%)** | **1961(29%)** | **2097(31%)** |
| 10 | AHC(HSV) | 6840 | 2692(39%) | 2202(32%) | 1946(28%) |
| | Kmeans(HSV) | 6840 | 2609(38%) | 2307(34%) | 1924(28%) |
| | Kmedoid(HSV) | 6840 | 2628(38%) | 2274(33%) | 1938(28%) |
| 20 | AHC(HSV) | 6840 | **2827(41%)** | **2000(29%)** | 2013(29%) |
| | Kmeans(HSV) | 6840 | 2694(39%) | 2183(32%) | 1963(29%) |
| | Kmedoid(HSV) | 6840 | 2753(40%) | 2113(31%) | 1974(29%) |
| 30 | AHC(HSV) | 6840 | 2717(40%) | 2225(33%) | 1898(28%) |
| | Kmeans(HSV) | 6840 | 2612(38%) | 2333(34%) | 1895(28%) |
| | Kmedoid(HSV) | 6840 | 2648(39%) | 2290(34%) | 1902(28%) |



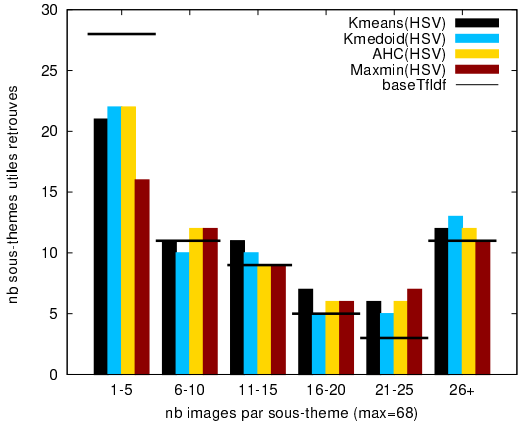

(a) Clustering(HSV) (`Media`-idéal)



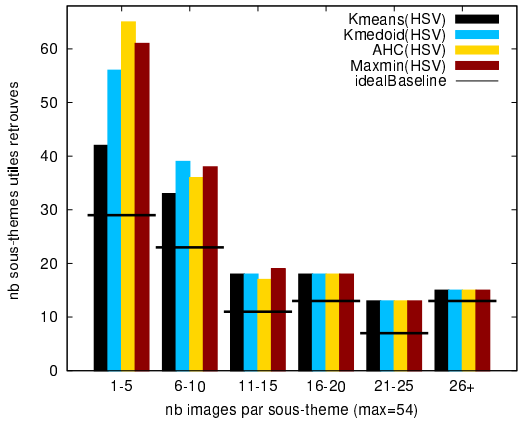

(b) Clustering(HSV) (`Media`-réel)




(c) Clustering(HSV) (`Clef`-idéal)

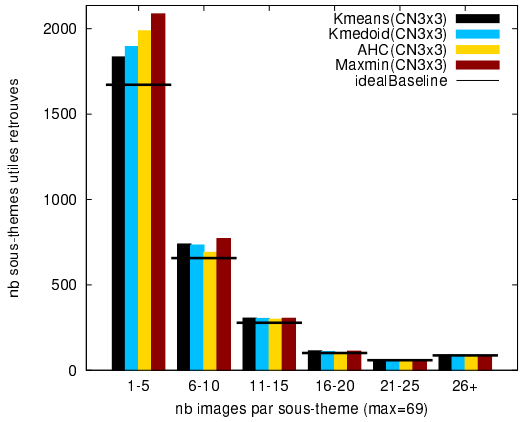



(d) Clustering(HSV) (`Clef`-réel)

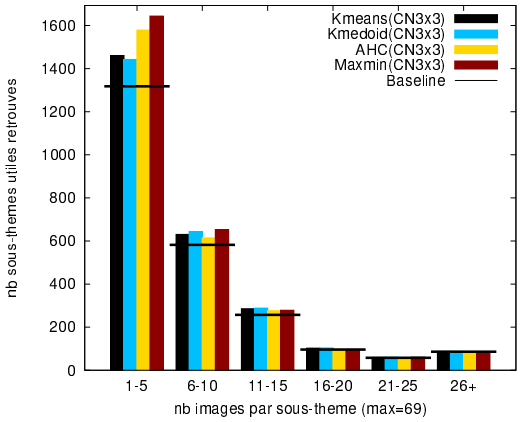



(e) Clustering(HSV) (`Xilo`-idéal)

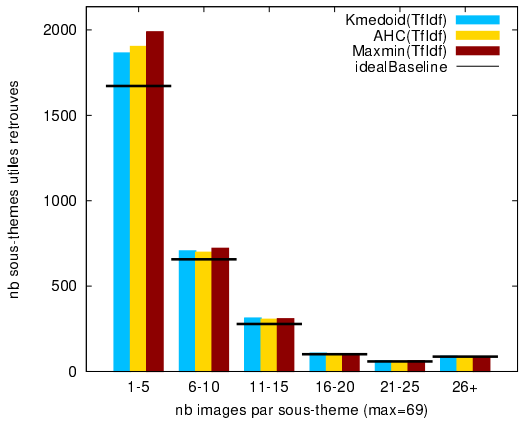
FIGURE A.13 – Apport des sous-thèmes utiles retrouvés dans les 20 premiers résultats des méthodes `K-Means`, `K-Medoids`, `AHC` et `Maxmin` avec le descripteur HSV en fonction du nombre d'images par cluster en utilisant un découpage `Fixe` à 20 clusters et une priorité par `Rank` dans un cas idéal et réel sur les benchmarks (a) (b) `Media` (sous-ensemble `testset`), (c) (d) `Clef` et (e) `Xilo`



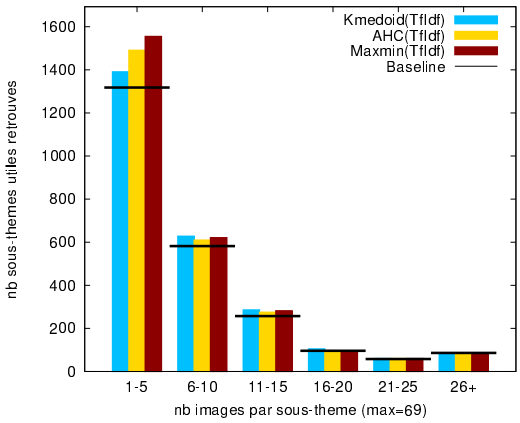

(a) Clustering(CN3x3) (`Media`-idéal)



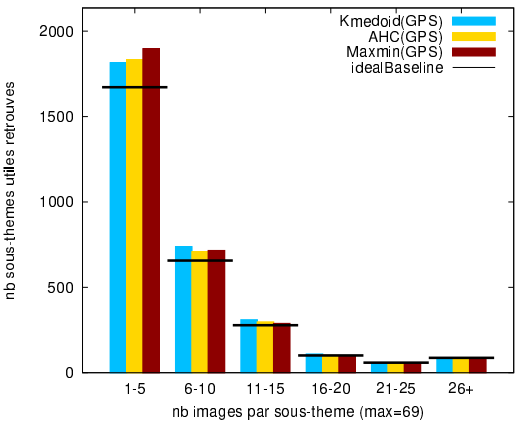

(b) Clustering(CN3x3) (`Media`-réel)



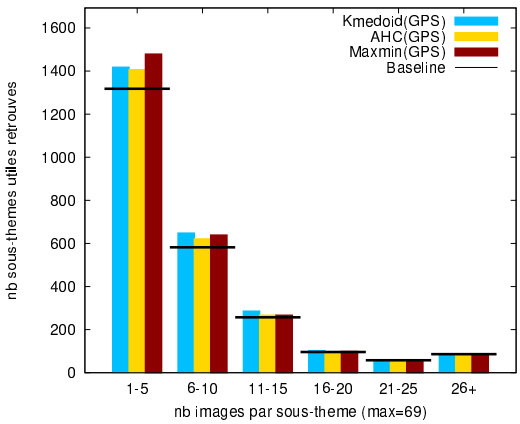

(c) Clustering(TfIdf) (`Media`-idéal)




(d) Clustering(TfIdf) (`Media`-réel)




(e) Clustering(GPS) (`Media`-idéal)




(f) Clustering(GPS) (`Media`-réel)

FIGURE A.14 – Apport des sous-thèmes utiles retrouvés dans les 20 premiers résultats des méthodes `K-Means`, `K-Medoids`, `AHC` et `Maxmin` avec les descripteurs (a) (b) CN3x3, (c) (d) TfIdf et (e) (f) GPS en fonction du nombre d'images par cluster en utilisant un découpage `Fixe` à 20 clusters et une priorité par `Rank` dans un cas idéal et réel sur le benchmark `Media` (sous-ensemble `testset`)

# Annexe B

# Nouveaux descripteurs de couleurs (palette subjective)

Pour obtenir des comportements généraux, dans cette thèse, nous avons utilisé plusieurs benchmarks et plusieurs descripteurs. Cependant, il est très rare de trouver des benchmarks publics disponibles qui disposent des mêmes descripteurs. Pour résoudre cette difficulté, nous avons généré un descripteur en commun sur tous les benchmarks afin que les comportements obtenus soient les plus comparables possibles. Pour générer ces descripteurs, on a créé un histogramme de couleurs qui représente la fréquence de chaque couleur dans l'image. Pour créer cet histogramme, on a besoin de définir une palette de couleurs qui ne doit pas être très grande parce que le temps de calcul des mesures de similarité est sensible à la taille de l'histogramme, et parce que nous souhaitons utiliser ces histogrammes en ligne.

## B.1 Génération de l'histogramme de couleurs

On a d'abord décidé de prendre comme base l'espace de représentation HSV, parce que ses dimensions expriment d'une manière plus claire la distribution des couleurs que possède une image. Par exemple, dans la dimension "teinte" de l'espace HSV, nous pouvons savoir quelle couleur est la plus dominante dans une image. Puis, on a divisé les dimensions de cet espace de représentation dans le but de créer une palette avec la plus petite quantité de couleurs.

### B.1.1 Dimension Teinte (en anglais "Hue")

Cette dimension est représentée par les couleurs de l'arc-en-ciel et a été initialement divisée en 12 régions (bins) pour être après réduite à 9 bins par la fusion des couleurs les plus similaires.

### B.1.2 Dimension Saturation et Value

Ces dimensions représentent le niveau de saturation et le niveau de luminosité que peut avoir une couleur et ces dimensions ont été stratégiquement divisés en 7 régions (bins). Quatre de ces 7 bins représentent les couleurs neutres (noir, blanc et gris) et les trois bins restants représentent les couleurs obscures, claires et nettes. La figure B.1 montre la représentation graphique de la division des dimensions dans l'espace HSV.

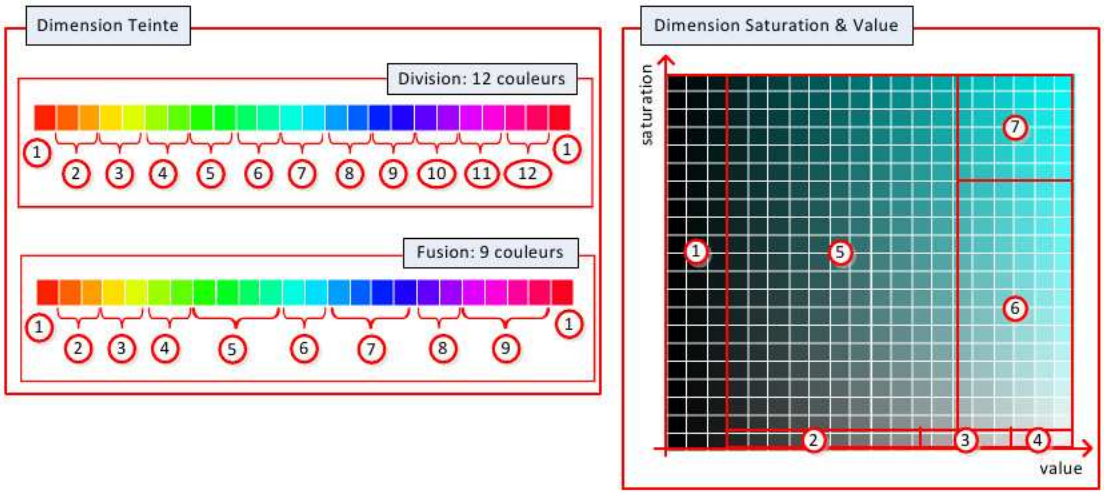



FIGURE B.1 – Espace de représentation HSV : Dimension teinte (à gauche) et dimension saturation et value (à droite).

TABLE B.1 – Règles pour générer une palette de 31 couleurs.

| Saturation | Value | Teinte | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | [0.96-0.04> | [0.04-0.13> | [0.13-0.21> | [0.21-0.29> | [0.29-0.46> | [0.46-0.54> | [0.54-0.71> | [0.71-0.79> | [0.79-0.96> |
| [0-1] | [0-0.15] | | | | | | | | | |
| [0-0.05] | <0.15-0.62] | | | | | | | | | |
| | <0.62-0.85] | | | | | | | | | |
| | <0.85-1] | | | | | | | | | |
| <0.05-1] | <0.15-0.72] | | | | | | | | | |
| <0.05-0.68] | <0.72-1] | | | | | | | | | |
| <0.68-1] | | | | | | | | | | |

## B.2 Règles

À partir des partitionnements effectués dans les dimensions de l'espace HSV, on peut produire des règles qui vont permettre de faire la réduction de la palette de couleurs. Ces règles sont montrées dans le tableau B.1.

## B.3 Algorithme

Une fois créée la palette de 31 couleurs, nous avons produit un histogramme de couleurs pour chaque image qui se trouve dans la banque d'images. La procédure de la création de chaque histogramme de couleur est la suivante :

1. Premièrement, on a capturé les pixels d'une image.
2. Deuxièmement, on a effectué la conversion de la valeur RGB de chaque pixel vers sa valeur HSV.
3. Troisièmement, on a groupé les couleurs les plus similaires grâce aux règles. Avec ce regroupement, on réduit les couleurs vers une palette de 31 couleurs.
4. Quatrièmement, on a créé l'histogramme de l'image à partir de la quantité de pixels qui appartient à chaque couleur et après l'histogramme est normalisé.

La figure B.2 montre le diagramme de flux qui représente la procédure de la création des histogrammes de couleurs.

Le temps d'exécution dans la création d'un histogramme est d'une seconde par image.

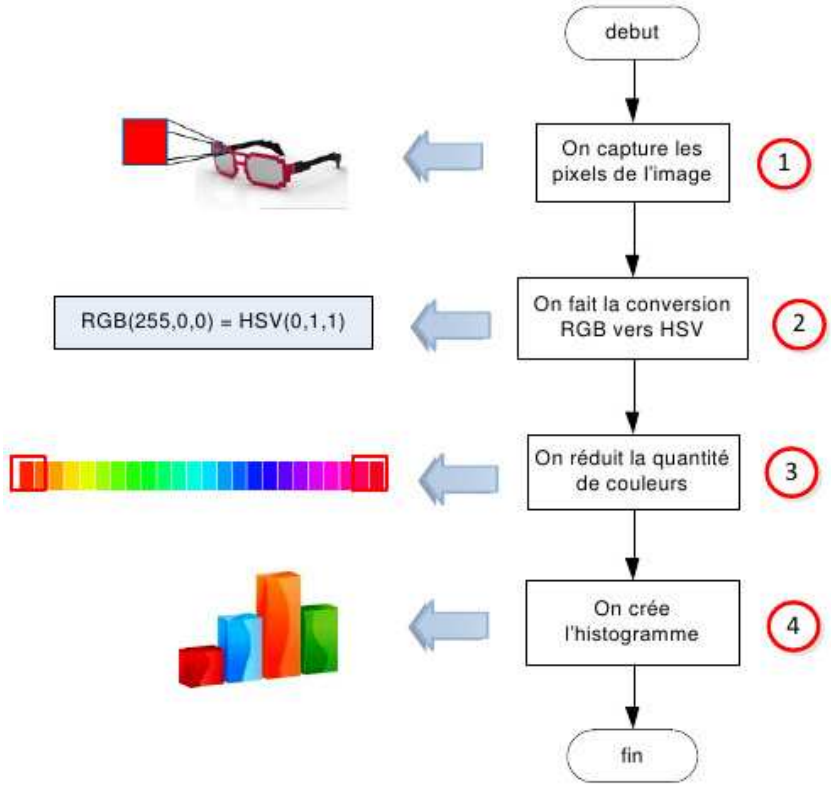



FIGURE B.2 – Procédure pour la création des histogrammes de couleur.

# Annexe C

# Diagramme de classes du système proposé

Dans cette annexe, nous montrons un aperçu de la structure du logiciel que nous avons développé pour résoudre le problème de la diversité en prenant en compte la problématique industrielle du temps de calcul et les difficultés abordées dans cette thèse. Ce logiciel a été développé dans le langage utilisé dans l'entreprise et il est disponible pour être utilisable dans le moteur de recherche de l'entreprise.

Dans la figure C.1 nous montrons le diagramme de classes UML de notre système principal pour résoudre le problème de la diversité. Dans une phase "hors-ligne", la classe *Descriptor* nous sert à générer les descripteurs de chaque image. Cette classe prend comme entrée le type de descripteur à générer (soit du visuel, du texte, des coordonnées GPS, etc). Pour les descripteurs visuels, dans la figure B.2 page 199 nous montrons l'algorithme utilisé pour générer les histogrammes de couleur HSV, Le benchmark `Media` fourni aussi certains descripteurs visuels qui sont mentionnées dans le tableau 6.5 page 73. Pour les descripteurs textuels, nous expliquons la procédure pour générer ces descripteurs dans la section 6.3.1.2 page 73.

Dans une phase en ligne, la classe *Reranking* représente la chaîne de traitement proposée. La première étape de cette chaîne est représentée par la classe *Relevance* où les images sont réordonnées par la similarité à la requête où l'objectif c'est d'augmenter la pertinence. Puis, la deuxième étape est représentée par les classes *DivClustering* (diversité par clustering) et *Maxmin* (diversité par optimisation) où les images sont réordonnées avec l'objectif d'augmenter la diversité.

Afin d'utiliser du clustering sur la diversité, la classe *DivClustering* est divisée en deux étapes. Une première étape représentée par la classe *Clustering* où les images sont regroupées en utilisant une méthode de clustering. Une deuxième étape représentée par la classe *RankCluster* où les clusters et les images à l'intérieur de chaque cluster sont triés par un critère de priorité et puis les images sont réordonnées en alternant les images de différents clusters.

Enfin, dans la figure C.2 nous montrons le diagramme de classes UML des méthodes utilisées par notre système principal. Dans cette figure nous pouvons voir la classe *similarity* qui est utilisée par la plupart des méthodes (par les méthodes de clustering, par `Maxmin`, etc.) où la mesure de similarité choisi dépend fortement du type de descripteur utilisé. Par exemple, si nous utilisons des coordonnées GPS, la mesure de similarité la plus adaptée est la distance "Haversine". Dans cette figure, nous pouvons voir aussi les classes *priority* et *cutting* qui sont utilisées par les méthodes de clustering ; et la classe *linkage* qui est plus utilisée par le clustering hiérarchique.

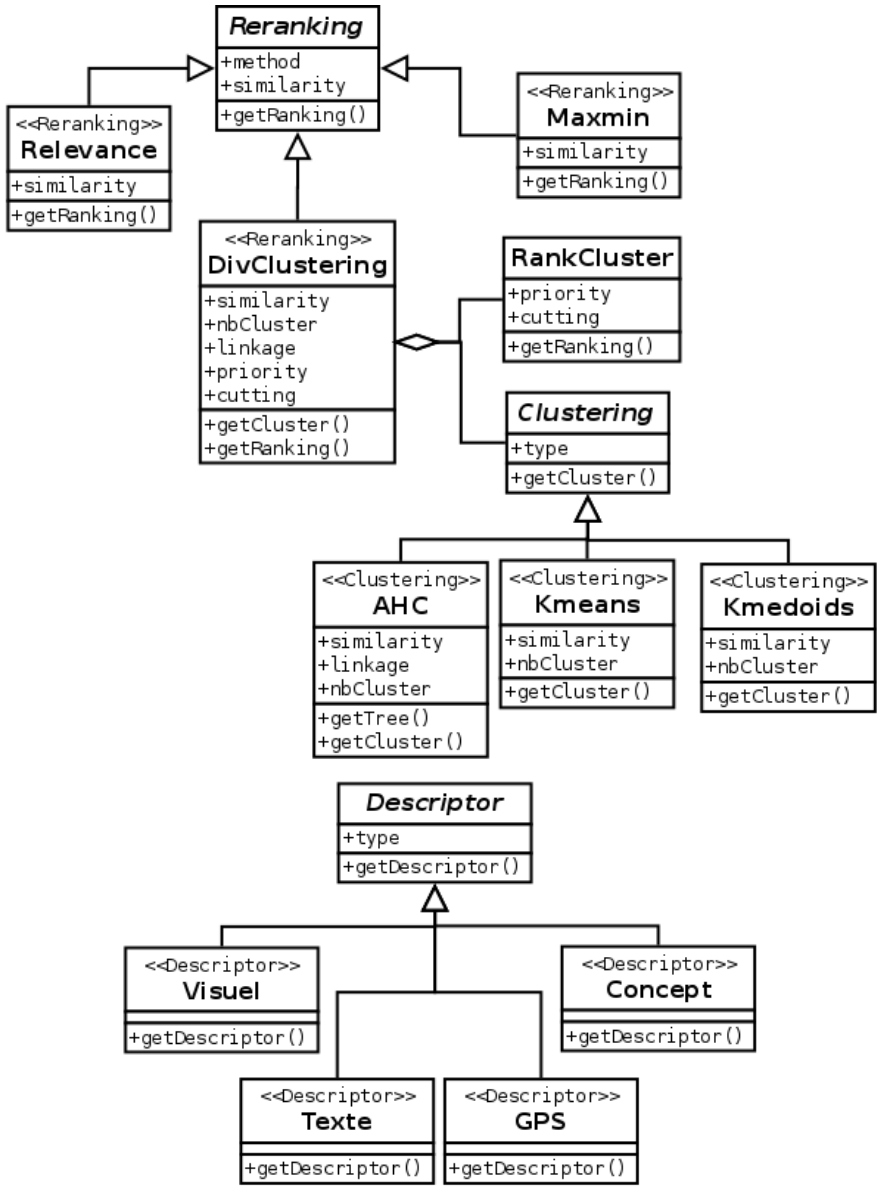



FIGURE C.1 – Diagramme de classes UML de notre système principal.

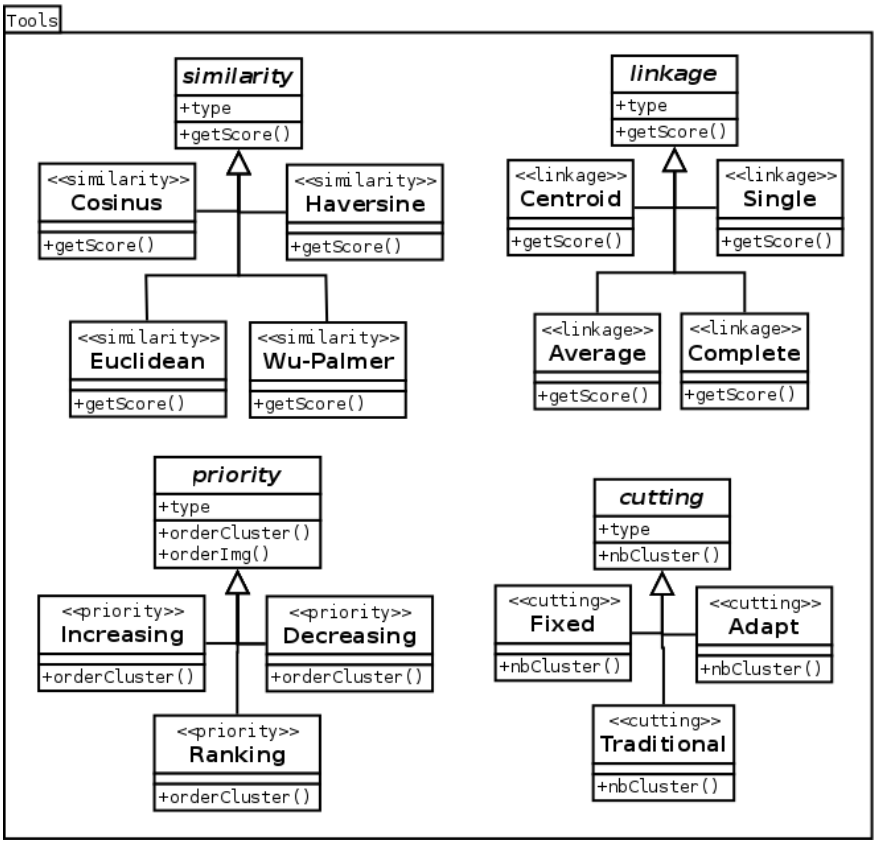



FIGURE C.2 – Diagramme de classes UML des outils de notre système.

# Bibliographie


[1] J. Ah-Pine, S. Clinchant, G. Csurka, F. Perronnin, and J.-M. Renders. Leveraging image, text and cross-media similarities for diversity-focused multimedia retrieval. In *ImageCLEF : Experimental Evaluation in Visual Information Retrieval*, 2010.

[2] J. April, F. Glover, J. P. Kelly, and M. Laguna. Simulation-based optimization : practical introduction to simulation optimization. In *Proceedings of the 35th Winter Simulation Conference : Driving Innovation*, WSC '03, pages 71–78, New Orleans, Louisiana, USA, December 7-10 2003.

[3] A. Armagan, A. Popescu, and P. Duygulu. MUCKE Participation at Retrieving Diverse Social Images Task of MediaEval 2013. In *Proceedings of the MediaEval 2013 Multimedia Benchmark Workshop*, volume 1043 of *CEUR Workshop Proceedings*, Barcelona, Spain, October 18-19 2013.

[4] T. Arni, P. D. Clough, M. Sanderson, and M. Grubinger. Overview of the ImageCLEFphoto 2008 Photographic Retrieval Task. In *Evaluating Systems for Multilingual and Multimodal Information Access, 9th Workshop of the Cross-Language Evaluation Forum, CLEF 2008*, volume 5706 of *Lecture Notes in Computer Science*, pages 500–511, Aarhus, Denmark, September 17-19 2008.

[5] P. Berkhin. A Survey of Clustering Data Mining Techniques. In *Grouping Multidimensional Data - Recent Advances in Clustering*, pages 25–71. 2006.

[6] B. Boteanu, I. Mironică, and B. Ionescu. Hierarchical Clustering Pseudo-Relevance Feedback for Social Image Search Result Diversification. In *International Workshop on Content-Based Multimedia Indexing - CBMI*, pages 29–34, June 10-12 2015.

[7] R. T. Calumby, V. P. Santana, F. S. Cordeiro, O. A. B. Penatti, L. T. Li, G. Chiachia, and R. da Silva Torres. Recod @ MediaEval 2014 : Diverse Social Images Retrieval. In *Working Notes Proceedings of the MediaEval 2014 Workshop*, volume 1263 of *CEUR Workshop Proceedings*, Barcelona, Spain, October 16-17 2014.

[8] J. G. Carbonell and J. Goldstein. The Use of MMR, Diversity-Based Reranking for Reordering Documents and Producing Summaries. In *SIGIR '98 : Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 335–336, August 24-28 1998.

[9] C. L. Clarke, M. Kolla, G. V. Cormack, O. Vechtomova, A. Ashkan, S. Büttcher, and I. MacKinnon. Novelty and Diversity in Information Retrieval Evaluation. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '08, pages 659–666, 2008.

[10] D. Corney, C. J. Martín, A. Göker, E. S. Xioufis, S. Papadopoulos, Y. Kompatsiaris, L. M. Aiello, and B. Thomee. SocialSensor : Finding Diverse Images at MediaEval 2013. In *Proceedings of the MediaEval 2013 Multimedia Benchmark Workshop*, volume 1043 of *CEUR Workshop Proceedings*, Barcelona, Spain, October 18-19 2013.

[11] D. Dang-Nguyen, L. Piras, G. Giacinto, G. Boato, and F. G. B. D. Natale. Retrieval of Diverse Images by Pre-filtering and Hierarchical Clustering. In *Working Notes*

*Proceedings of the MediaEval 2014 Workshop*, volume 1263 of *CEUR Workshop Proceedings*, Barcelona, Spain, October 16-17 2014.

[12] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and F. Li. ImageNet : A large-scale hierarchical image database. In *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009)*, pages 248–255, Miami, Florida, USA, 20-25 June 2009.

[13] T. Deselaers, T. Gass, P. Dreuw, and H. Ney. Jointly optimising relevance and diversity in image retrieval. In *Proceedings of the 8th ACM International Conference on Image and Video Retrieval, CIVR 2009*, pages 08–10, Santorini Island, Greece, July 8-10 2009.

[14] C. Fellbaum. *WordNet : An Electronic Lexical Database*. MIT Press, Cambridge, MA, May 1998.

[15] M. Ferecatu and H. Sahbi. TELECOMParisTech at ImageClefphoto 2008 : Bi-Modal Text and Image Retrieval with Diversity Enhancement. In *Working Notes for CLEF 2008 Workshop co-located with the 12th European Conference on Digital Libraries (ECDL 2008)*, Aarhus, Denmark, September 17-19 2008.

[16] J. Fournier, M. Cord, and S. Philipp-Foliguet. RETIN : A Content-Based Image Indexing and Retrieval System. *Pattern Analysis and Applications*, 4(2-3) :153–173, 2001.

[17] A. Gînsca, A. Popescu, and N. Rekabsaz. CEA LIST's Participation at the MediaEval 2014 Retrieving Diverse Social Images Task. In *Working Notes Proceedings of the MediaEval 2014 Workshop*, volume 1263 of *CEUR Workshop Proceedings*, Barcelona, Spain, October 16-17 2014.

[18] J. L. Hafner, H. S. Sawhney, W. Equitz, M. Flickner, and W. Niblack. Efficient Color Histogram Indexing for Quadratic Form Distance Functions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(7) :729–736, 1995.

[19] M. Halvey, P. Punitha, D. Hannah, R. Villa, F. Hopfgartner, A. Goyal, and J. M. Jose. Diversity, Assortment, Dissimilarity, Variety : A Study of Diversity Measures Using Low Level Features for Video Retrieval. In *Advances in Information Retrieval, 31th European Conference on IR Research, ECIR 2009*, pages 126–137, Toulouse, France, April 6-9 2009.

[20] J. A. Hartigan and M. A. Wong. Algorithm AS 136 : A k-means clustering algorithm. *Applied Statistics*, 28(1) :100–108, 1979.

[21] J. Howe. The Rise of Crowdsourcing. *Wired Magazine*, 14(6), 06 2006.

[22] M. Inoue and P. Grover. Query Types and Visual Concept-Based Post-retrieval Clustering. In *Evaluating Systems for Multilingual and Multimodal Information Access, 9th Workshop of the Cross-Language Evaluation Forum, CLEF 2008*, pages 661–668, Aarhus, Denmark, September 17-19 2008.

[23] B. Ionescu, A. Popescu, M. Lupu, A. L. Gînscă, B. Boteanu, and H. Müller. Div150Cred : A Social Image Retrieval Result Diversification with User Tagging Credibility Dataset. In *Proceedings of the 6th ACM Multimedia Systems Conference*, MMSys '15, pages 207–212, 2015.

[24] B. Ionescu, A. Popescu, A.-L. Radu, and H. Müller. Result Diversification in Social Image Retrieval : A Benchmarking Framework. *Multimedia Tools and Applications*, pages 1–31, 2014.

[25] B. Ionescu, A.-L. Radu, M. Menéndez, H. Müller, A. Popescu, and B. Loni. Div400 : A Social Image Retrieval Result Diversification Dataset. In *Proceedings of the 5th ACM Multimedia Systems Conference*, MMSys '14, pages 29–34, 2014.

[26] N. Jain, J. S. Hare, S. Samangooei, J. Preston, J. Davies, D. Dupplaw, and P. H. Lewis. Experiments in Diversifying Flickr Result Sets. In *Proceedings of the MediaEval 2013 Multimedia Benchmark Workshop*, volume 1043 of *CEUR Workshop Proceedings*, Barcelona, Spain, October 18-19 2013.

[27] L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data : An Introduction to Cluster Analysis*. John Wiley, New York, NY, 9th edition, Mars 1990.

[28] C.-C. Kuo, F. Glover, and K. S. Dhir. Analyzing and Modeling the Maximum Diversity Problem by Zero-One Programming. *Decision Sciences*, 24(6) :1171–1185, 1993.

[29] C. Kuoman, S. Tollari, and M. Detyniecki. Diversité par Classification Ascendante Hiérarchique dans un moteur de recherche d'images. *Stage de master 2 réalisée dans l'entreprise Xilopix et le laboratoire LIP6*, 2011.

[30] C. Kuoman, S. Tollari, and M. Detyniecki. UPMC at MediaEval 2013 : Relevance by Text and Diversity by Visual Clustering. In *Proceedings of the MediaEval 2013 Multimedia Benchmark Workshop*, volume 1043 of *CEUR Workshop Proceedings*, Barcelona, Spain, October 18-19 2013.

[31] G. N. Lance and W. T. Williams. A General Theory of Classificatory Sorting Strategies : 1. Hierarchical Systems. *The Computer Journal*, 9(4) :373–380, 1967.

[32] D. Lin. An Information-Theoretic Definition of Similarity. In *Proceedings of the Fifteenth International Conference on Machine Learning (ICML 1998)*, pages 296–304, Madison, Wisconsin, USA, July 24-27 1998.

[33] G. Liu and J. Yang. Content-based image retrieval using color difference histogram. *Pattern Recognition*, 46(1) :188–198, January 2013.

[34] O. Ludwig, D. Delgado, V. Goncalves, and U. Nunes. Trainable classifier-fusion schemes : An application to pedestrian detection. In *Intelligent Transportation Systems, 2009. ITSC '09. 12th International IEEE Conference on*, pages 1–6, Oct 2009.

[35] L. Maisonnasse, P. Mulhem, É. Gaussier, and J. Chevallet. LIG at ImageCLEF 2008. In *Evaluating Systems for Multilingual and Multimodal Information Access, 9th Workshop of the Cross-Language Evaluation Forum, CLEF 2008*, pages 704–711, Aarhus, Denmark, September 17-19 2008.

[36] B. S. Manjunath, J. Ohm, V. V. Vasudevan, and A. Yamada. Color and texture descriptors. *IEEE Transactions on Circuits and Systems for Video Technology*, 11(6) :703–715, 2001.

[37] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to information retrieval*. Cambridge University Press, 2008.

[38] G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. J. Miller. Introduction to WordNet : an on-line lexical database. *International Journal of Lexicography*, 3(4) :235–244, 1990.

[39] G. Milligan and M. Cooper. An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, 50(2) :159–179, June 1985.

[40] S. Nelson and J.-L. Schulman. Orthopaedic Literature and MeSH. *Clinical Orthopaedics and Related Research*, 468(10) :2621–2626, 2010.

[41] T. Ojala, M. Pietikainen, and D. Harwood. Performance Evaluation of Texture Measures with Classification Based on Kullback Discrimination of Distributions. In *IAPR International Conference on Pattern Recognition*, volume 1, pages 582–585, 1994.

[42] J. R. M. Palotti, N. Rekabsaz, M. Lupu, and A. Hanbury. TUW @ Retrieving Diverse Social Images Task 2014. In *Working Notes Proceedings of the MediaEval 2014 Workshop*, volume 1263 of *CEUR Workshop Proceedings*, Barcelona, Spain, October 16-17 2014.

[43] M. L. Paramita, J. Tang, and M. Sanderson. Generic and Spatial Approaches to Image Search Results Diversification. In *Advances in Information Retrieval, 31th European Conference on IR Research, ECIR 2009*, pages 603–610, Toulouse, France, April 6-9 2009.

[44] J. M. Ponte and W. B. Croft. A Language Modeling Approach to Information Retrieval. In *SIGIR '98 : Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 275–281, August 24-28 1998.

[45] A. Popescu. CEA LIST's Participation at the MediaEval 2013 Retrieving Diverse Social Images Task. In *Proceedings of the MediaEval 2013 Multimedia Benchmark Workshop*, volume 1043 of *CEUR Workshop Proceedings*, Barcelona, Spain, October 18-19 2013.

[46] A. Popescu and G. Grefenstette. Social media driven image retrieval. In *Proceedings of the 1st International Conference on Multimedia Retrieval, ICMR 2011*, pages 33 :1–33 :8, April 18 - 20 2011.

[47] D. C. Porumbel, J. Hao, and F. Glover. A simple and effective algorithm for the MaxMin diversity problem. *Annals OR*, 186(1) :275–293, 2011.

[48] A. Radu, B. Boteanu, O. Ples, and B. Ionescu. LAPI @ Retrieving Diverse Social Images Task 2013 : Qualitative Photo Retrieval using Multimedia Content. In *Proceedings of the MediaEval 2013 Multimedia Benchmark Workshop*, volume 1043 of *CEUR Workshop Proceedings*, Barcelona, Spain, October 18-19 2013.

[49] S. E. Robertson. Readings in Information Retrieval. chapter The Probability Ranking Principle in IR, pages 281–286. Morgan Kaufmann Publishers Inc., 1997.

[50] J. Rocchio. *Relevance Feedback in Information Retrieval*, pages 313–323. The SMART Retrieval System Experiments in Automatic Document Processing, Prentice Hall, Englewood Cliffs NJ, 1971.

[51] G. Salton and M. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill Book Company, 1984.

[52] J. Savoy. A Learning Scheme for Information Retrieval in Hypertext. *Inf. Process. Manage.*, 30(4) :515–534, 1994.

[53] K. Song, Y. Tian, W. Gao, and T. Huang. Diversifying the Image Retrieval Results. In *Proceedings of the 14th Annual ACM International Conference on Multimedia*, MULTIMEDIA '06, pages 707–710, 2006.

[54] M. A. Stricker and M. Orengo. Similarity of Color Images. In *Storage and Retrieval for Image and Video Databases (SPIE)*, pages 381–392, 1995.

[55] M. J. Swain and D. H. Ballard. Color indexing. *International Journal of Computer Vision*, 7(1) :11–32, 1991.

[56] G. Szücs, Z. Paroczi, and D. M. Vincz. BMEMTM at MediaEval 2013 Retrieving Diverse Social Images Task : Analysis of Text and Visual Information. In *Proceedings of the MediaEval 2013 Multimedia Benchmark Workshop*, volume 1043 of *CEUR Workshop Proceedings*, Barcelona, Spain, October 18-19 2013.

[57] X. Tang. Texture information in run-length matrices. *IEEE Transactions on Image Processing*, 7(11) :1602–1609, 1998.

[58] C. Thériault, N. Thome, and M. Cord. Extended Coding and Pooling in the HMAX Model. *IEEE Transactions on Image Processing*, 22(2) :764–777, 2013.

[59] S. Tollari, M. Detyniecki, A. Fakeri-Tabrizi, C. Marsala, M. Amini, and P. Gallinari. Using Visual Concepts and Fast Visual Diversity to Improve Image Retrieval. In *Evaluating Systems for Multilingual and Multimodal Information Access, 9th Workshop of the Cross-Language Evaluation Forum, CLEF 2008*, volume 5706 of *Lecture Notes in Computer Science*, pages 577–584, Aarhus, Denmark, September 17-19 2008.

[60] S. Tollari, P. Mulhem, M. Ferecatu, H. Glotin, M. Detyniecki, P. Gallinari, H. Sahbi, and Z. Zhao. A Comparative Study of Diversity Methods for Hybrid Text and Image Retrieval Approaches. In *Evaluating Systems for Multilingual and Multimodal Information Access, 9th Workshop of the Cross-Language Evaluation Forum, CLEF 2008*, pages 585–592, Aarhus, Denmark, September 17-19 2008.

[61] J. van de Weijer, C. Schmid, J. Verbeek, and D. Larlus. Learning Color Names for Real-world Applications. *IEEE Trans. on Image Processing*, 18(7) :1512–1523, July 2009.

[62] R. van Zwol, V. Murdock, L. G. Pueyo, and G. Ramírez. Diversifying image search with user generated content. In *Proceedings of the 1st ACM SIGMM International Conference on Multimedia Information Retrieval, MIR 2008*, pages 67–74, Vancouver, British Columbia, Canada, October 30-31 2008.

[63] B. Vandersmissen, A. Tomar, F. Godin, W. D. Neve, and R. V. de Walle. Ghent University-iMinds at MediaEval 2013 Diverse Images : Relevance-Based Hierarchical Clustering. In *Proceedings of the MediaEval 2013 Multimedia Benchmark Workshop*, volume 1043 of *CEUR Workshop Proceedings*, Barcelona, Spain, October 18-19 2013.

[64] E. Vee, U. Srivastava, J. Shanmugasundaram, P. Bhat, and S. A. Yahia. Efficient computation of diverse query results. In *Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on*, pages 228–236, 2008.

[65] R. C. Veltkamp and M. Hagedoorn. State of the Art in Shape Matching. In *Principles of Visual Information Retrieval*, pages 87–119. Springer, 2001.

[66] I. H. Witten and E. Frank. *Data Mining : Practical Machine Learning Tools and Techniques, Second Edition (Morgan Kaufmann Series in Data Management Systems)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.

[67] W. A. Woods. Conceptual Indexing : A Better Way to Organize Knowledge. Technical report, Mountain View, CA, USA, April 1997.

[68] Z. Wu and M. Palmer. Verbs Semantics and Lexical Selection. In *Proceedings of the 32Nd Annual Meeting on Association for Computational Linguistics*, ACL '94, pages 133–138, 1994.

[69] E. S. Xioufis, S. Papadopoulos, Y. Kompatsiaris, and I. P. Vlahavas. SocialSensor : Finding Diverse Images at MediaEval 2014. In *Working Notes Proceedings of the MediaEval 2014 Workshop, Barcelona, Catalunya, Spain, October 16-17, 2014.*, volume 1263 of *CEUR Workshop Proceedings*, Barcelona, Spain, October 16-17 2014.

[70] N. Ye. *The handbook of data mining*. Lawrence Erlbaum, 2003.

[71] M. Zaharieva and P. Schwab. A Unified Framework for Retrieving Diverse Social Images. In *Working Notes Proceedings of the MediaEval 2014 Workshop*, volume 1263 of *CEUR Workshop Proceedings*, Barcelona, Spain, October 16-17 2014.

[72] C. Zhai, W. W. Cohen, and J. D. Lafferty. Beyond independent relevance : methods and evaluation metrics for subtopic retrieval. In *SIGIR 2003 : Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 10–17, Toronto, Canada, July 28 - August 1 2003.

[73] B. Zhang, H. Li, Y. Liu, L. Ji, W. Xi, W. Fan, Z. Chen, and W.-Y. Ma. Improving Web Search Results Using Affinity Graph. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '05, pages 504–511, 2005.

[74] C.-N. Ziegler, S. M. McNee, J. A. Konstan, and G. Lausen. Improving Recommendation Lists Through Topic Diversification. In *Proceedings of the 14th International Conference on World Wide Web*, WWW '05, pages 22–32, 2005.

# Index

# Références de l'auteur

C. Kuoman, S. Tollari, and M. Detyniecki. Using tree of concepts and hierarchical reordering for diversity in image retrieval. In *11th International Workshop on Content-Based Multimedia Indexing, CBMI 2013*, pages 251–256, Veszprém, Hungary, June 17-19 2013.

C. Kuoman, S. Tollari, and M. Detyniecki. UPMC at MediaEval 2013 : Relevance by Text and Diversity by Visual Clustering. In *Proceedings of the MediaEval 2013 Multimedia Benchmark Workshop*, volume 1043 of *CEUR Workshop Proceedings*, Barcelona, Spain, October 18-19 2013.

C. Kuoman, S. Tollari, and M. Detyniecki. Diversité hiérarchique et utilisation d'arbres de concepts pour la recherche d'images. In *CORIA 2013 - Conférence en Recherche d'Infomations et Applications - 10th French Information Retrieval Conference*, pages 47–62, Neuchâtel, Suisse, April 3-5 2013.

C. Kuoman, S. Tollari, and M. Detyniecki. Diversité par Classification Ascendante Hiérarchique dans un moteur de recherche d'images. *Stage de master 2 réalisée dans l'entreprise Xilopix et le laboratoire LIP6*, 2011.