Apprentissage interactif de règles d'extraction d'information textuelle

Sondes Bannour

École Doctorale Galilée

Thèse de doctorat

Discipline : Informatique

présentée par

Sondes Bannour

Apprentissage interactif de règles d'extraction d'information textuelle

dirigée par Henry Soldano
encadrée par Laurent Audibert

M. Laurent Audibert	Université Paris 13	encadrant
M. Jean-Gabriel Ganascia	UPMC	examinateur
M^{me} Adeline Nazarenko	Université Paris 13	examinatrice
M^{me} Claire Nédellec	MaIAGE-INRA	rapportrice
M^{me} Pascale Sébillot	IRISA/INSA de Rennes	rapportrice
M. Henry Soldano	Université Paris 13	directeur

Université Paris 13
Ecole Doctorale Galilée
99 avenue Jean-Baptiste Clément
93430 Villetaneuse

Remerciements

Je tiens à remercier, en premier lieu, Laurent Audibert, mon encadrant, et Henry Soldano, mon directeur de thèse, pour la confiance et la liberté qu'ils m'ont accordées, pour m'avoir guidée dans les moments de doute et pour m'avoir apporté les moyens d'arriver au bout de ce long et difficile chemin. Je tiens aussi à remercier particulièrement Adeline Nazarenko pour m'avoir accueillie au sein de l'équipe RCLN (Représentation des Connaissances et Langage Naturel) et pour avoir pris sur son temps chaque fois que je sollicitais ses précieux conseils.

J'adresse une pensée particulière à ma soeur et confidente Ines qui égayait mes journées au LIPN, à Haïfa pour son amitié et ses conseils précieux et à mes amies Sarra, Nouha, Leila, Aïcha, Hanène, Ines et Nada pour tous les moments de bonheur et de solidarité que nous avons partagés au LIPN.

Un immense merci à mon époux Sami et à ma petite Sarah qui illuminent chacun de mes jours.

Je remercie enfin mes parents et mes soeurs pour leur amour et leur soutien.

Résumé

L'Extraction d'Information est une discipline qui a émergé du Traitement Automatique des Langues afin de proposer des analyses fines d'un texte écrit en langage naturel et d'améliorer la recherche d'informations spécifiques. Les techniques d'extraction d'information ont énormément évolué durant les deux dernières décennies. Les premiers systèmes d'extraction d'information étaient des systèmes à base de règles écrites manuellement. L'écriture manuelle des règles étant devenue une tâche fastidieuse, des algorithmes d'apprentissage automatique de règles ont été développés. Ces algorithmes nécessitent cependant la rédaction d'un guide d'annotation détaillé, puis l'annotation manuelle d'une grande quantité d'exemples d'entraînement.

Pour minimiser l'effort humain requis dans les deux familles d'approches de mise au point de règles, nous avons proposé, dans ce travail de thèse, une approche hybride qui combine les deux en un seul système interactif qui procède en plusieurs itérations. Ce système que nous avons nommé IRIES permet à l'utilisateur de travailler de manière duale sur les règles d'extraction d'information et les exemples d'apprentissage. Pour mettre en place l'approche proposée, nous avons proposé une chaîne d'annotation linguistique du texte et l'utilisation d'un langage de règles expressif pour la compréhensibilité et la généricité des règles écrites ou inférées, une stratégie d'apprentissage sur un corpus réduit pour ne pas discriminer les exemples positifs non encore annotés à une itération donnée, la mise en place d'un concordancier pour l'écriture de règles prospectives et la mise en place d'un module d'apprentissage actif (IAL4Sets) pour une sélection intelligente d'exemples.

Ces propositions ont été mises en place et évaluées sur deux corpus : le corpus de BioNLP-ST 2013 et le corpus SyntSem. Une étude de différentes combinaisons de traits linguistiques utilisés dans les expressions des règles a permis de voir l'impact de ces traits sur les performances des règles. L'apprentissage sur un corpus réduit a permis un gain considérable en temps d'apprentissage sans dégradation de performances. Enfin, le module d'apprentissage actif proposé (IAL4Sets) a permis d'améliorer les performances de l'apprentissage actif de base de l'algorithme WHISK grâce à l'introduction de la notion de distance ou de similarité distributionnelle qui permet de proposer à l'utilisateur des exemples sémantiquement proches des exemples positifs déjà couverts.

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction

1.1 Contexte et motivations

Internet a connu un essor fulgurant dans les années 1990. Très vite, la quantité d'information disponible sur le réseau est devenue très importante, mais son hétérogénéité et son manque de structuration rendait l'accès à cette information très difficile. L'Extraction d'Information (EI), une discipline du Traitement Automatique des Langues (TAL), a alors vu le jour afin de proposer des analyses fines d'un texte écrit en langage naturel et d'améliorer la recherche d'informations spécifiques. Une tâche d'extraction d'information consiste, en effet, à prendre en entrée du texte non structuré écrit en langage naturel, à en extraire des entités et des évènements, pour en produire des données non ambigües dans un format structuré (template, formulaire, etc.), qui sont soit présentées directement à l'utilisateur soit stockées pour des traitements ultérieurs (indexation dans des applications de recherche d'information, etc.). Les travaux dans ce domaine se sont développés grâce aux campagnes MUC (Message Understanding Conferences) qui ont défini un cadre à cette discipline en précisant les différentes tâches qu'elle comporte et en proposant des protocoles d'évaluation associés à ces tâches.

Les techniques d'extraction d'information ont énormément évolué durant les deux dernières décennies. Les premiers systèmes d'extraction d'information étaient des systèmes à base de règles écrites manuellement par des experts. L'écriture manuelle des règles étant devenue une tâche fastidieuse, des algorithmes d'apprentissage automatique de règles ont été développés (Aitken, 2002 ; Califf & Mooney, 1998 ; Ciravegna, 2001 ; Riloff, 1996 ; S. Soderland, Cardie, & Mooney, 1999). D'autres techniques ont ensuite vu le jour comme l'apprentissage statistique et les techniques de construction de grammaires car les algorithmes d'apprentissage de règles ont été jugés fragiles face au bruit dans les textes non structurés.

Malgré la diversité des techniques d'extraction d'information, les approches à base de règles écrites manuellement et les approches à base d'apprentissage automatique (apprentissage de règles et apprentissage statistique) continuent à être utilisées en parallèle. En effet, chacune de ces familles d'approches possède ses avantages et ses inconvénients. Les approches à base de règles écrites manuellement sont généralement précises et flexibles mais nécessitent une expertise du domaine d'étude ainsi que des compétences techniques et linguistiques importantes pour l'écriture de

patrons assez robustes. Ces approches sont également coûteuses à adapter à de nouveaux domaines. Les approches à base d'apprentissage, d'un autre coté, obtiennent des performances comparables aux approches à base de règles écrites manuellement, avec habituellement une précision inférieure mais un meilleur rappel. Cependant, ces approches nécessitent la rédaction d'un guide d'annotation détaillé, puis l'annotation d'une grande quantité d'exemples d'entrainement afin de pouvoir mettre en œuvre des techniques d'apprentissage. Actuellement, alors que les systèmes à base de règles, écrites manuellement ou inférées de manière automatique, dominent le monde industriel, les recherches académiques semblent plutôt se pencher du côté des approches statistiques (Chiticariu, Li, & Reiss, 2013).

Les systèmes d'extraction d'information sont généralement coûteux à mettre en place pour des utilisateurs qui ne disposent pas d'une quantité suffisante de données d'apprentissage annotées ou qui ne sont pas des experts en ingénierie de connaissances. Des systèmes d'extraction d'information interactifs ont donc vu le jour d'une part pour réduire ce coût et d'autre part pour permettre à l'utilisateur d'investiguer les erreurs faites par le système d'extraction d'information et de les corriger. L'extraction d'information interactive a créé de nouveaux besoins dans les systèmes d'extraction d'information (Kristjansson et al., 2004). Pour faciliter l'expérience des utilisateurs, un système d'extraction d'information interactif doit pouvoir attribuer des indices de confiance aux valeurs des champs extraits et prendre en considération de manière optimale les corrections de l'utilisateur. Même si les approches à base de règles sont moins adaptées au paradigme d'extraction d'information interactive que les approches statistiques car elles ne peuvent ni estimer la confiance ni incorporer de manière naturelle les annotations et les corrections de l'utilisateur, des travaux montrent qu'il est tout à fait possible d'introduire la notion de confiance dans des systèmes d'extraction d'information à base de règles sans pour autant utiliser les méthodes conventionnelles d'estimation de confiance tirées de l'état de l'art de l'apprentissage actif (S. Soderland et al., 1999 ; Thompson, Califf, & Mooney, 1999 ; T. Wu & Pottenger, 2005). L'interaction avec l'utilisateur dans ces systèmes à base de règles peut se faire de différentes manières. Certains travaux se contentent d'entraîner de manière interactive le système d'extraction d'information (Cardie & Pierce, 1998). D'autres disposent d'une interface interactive qui guide l'utilisateur dans la construction du système d'extraction d'information mais ne disposent pas d'un module d'apprentissage automatique (Li et al., 2012), ce qui les rend coûteux en termes d'effort humain. D'autres permettent à un utilisateur initié de développer un extracteur d'information de bout en bout mais ne disposent pas d'un module qui facilite le choix des exemples à annoter par l'utilisateur (Akbik et al., 2013). Un besoin de développer un système d'EI interactif générique qui permet de guider l'utilisateur aussi bien dans l'écriture des règles que dans le choix des exemples à annoter pour inférer les règles se fait ressentir. Un tel système fonctionnerait idéalement avec n'importe quel langage de règles et n'importe quel algorithme d'apprentissage de règles. Cependant, l'absence de langage de règles standard (Chiticariu et al., 2013) et d'algorithmes d'apprentissage de règles qui s'appuient sur des langages standard complique l'interaction de l'utilisateur avec le système d'EI dans la mesure où il faut soit maîtriser le langage de règles utilisé par l'algorithme d'apprentissage de règles, soit adapter l'algorithme d'apprentissage de règles au langage de règles maîtrisé par

l'utilisateur. Le système TextRuler (Kluegl, Atzmueller, Hermann, & Puppe, 2009) est, par exemple, un système de développement semi-automatique d'applications d'EI à base de règles contenant des implémentations d'algorithmes d'apprentissage de règles de la littérature adaptées au langage Ruta.

1.2 Contributions

Nous nous intéressons, dans ce travail, aux approches à base de règles car les règles sont plus faciles à manipuler et à interpréter par un être humain. Ainsi, nous proposons une approche d'EI à base de règles qui a l'avantage d'être :

- **hybride** dans la mesure où elle combine une approche d'EI à base de règles écrites manuellement et une approche d'EI à base d'apprentissage de règles permettant ainsi à l'utilisateur de choisir l'opération qu'il juge la plus adaptée entre l'annotation d'exemples d'apprentissage et l'écriture de règles ;
- **interactive** dans la mesure où l'utilisateur interagit avec le module d'apprentissage de règles en lui communiquant un *feedback* sur la pertinence des règles inférées (soit en annotant les exemples couverts par ces règles soit en modifiant les règles elles-mêmes) ;
- **itérative** car elle nécessite plusieurs itérations pour une construction progressive de la base des règles de manière à minimiser l'effort humain requis.

L'approche interactive que nous proposons permet à l'utilisateur d'assurer la généricité des règles et de contenir leur complexité. Pour assurer ses objectifs, l'approche proposée doit lever certains verrous liés aux propriétés suivantes.

- **La compréhensibilité des règles :** pour pouvoir modifier et améliorer des règles existantes, l'utilisateur a besoin de comprendre les règles qui doivent avoir une expression symbolique, être assez courtes, être en nombre réduit et s'appuyer sur des termes intuitifs.
- **La généricité des règles :** pour être générique, une règle doit reposer sur des annotations textuelles elles mêmes génériques. La généricité des règles améliore leur compréhensibilité dans la mesure où elle permet d'obtenir des expressions de règles plus courtes et un ensemble de règles plus réduit.
- **La stabilité des règles :** le module d'apprentissage doit prendre en compte les règles écrites ou modifiées par l'utilisateur et essayer de les étendre sans modifier complètement leurs expressions pour permettre à l'utilisateur de suivre l'évolution des règles au fil des itérations et de garder le contrôle sur leur complexité.
- **La sélection d'exemples :** un module de sélection d'exemples pertinents permet à l'utilisateur d'annoter moins d'exemples.
- **Le temps d'apprentissage :** pour pouvoir interagir avec le module d'apprentissage, l'utilisateur ne doit pas attendre longtemps avant d'avoir une réponse de l'algorithme.

Le système qui met en place cette approche s'appelle IRIES. IRIES étend l'interface de visualisation du système TextRuler (Kluegl, Atzmueller, Hermann, & Puppe, 2009) et l'algorithme WHISK basé sur le langage Ruta implémenté dans TextRuler. Il contient des modules qui lui permettent de répondre aux spécifications de l'approche proposée : une chaîne d'annotation linguistique et un langage de règles

expressif pour la compréhensibilité et la généricité des règles d'EI écrites ou inférées, une stratégie d'apprentissage sur un corpus réduit qui permet de ne pas considérer comme négatifs les exemples positifs non encore annotés par l'utilisateur à une itération donnée et de réduire le temps d'apprentissage, un concordancier pour pouvoir écrire des règles prospectives et réduire son espace de travail et enfin, deux versions d'un module d'apprentissage actif (IAL4Sets et IAL3Sets) pour une sélection intelligente des exemples à annoter. Les modules proposés ont été mis en place et évalués sur deux corpus : le corpus de BioNLP-ST 2013 et le corpus SyntSem.

1.3 Organisation

Nous organisons notre rapport de la manière suivante :

Le chapitre 2 est consacré à l'état de l'art en EI. Après une brève présentation de ce qu'est l'EI, des tâches d'EI et de l'évaluation en EI, ce chapitre décrit les approches d'EI : les approches à base de règles (écrites manuellement ou inférées automatiquement) et les approches statistiques, en mettant l'accent sur les approches à base de règles. Nous clôturons ce chapitre par la présentation des nouvelles tendances en EI, à savoir l'utilisation des ontologies et le paradigme de l'extraction d'information ouverte.

Dans le chapitre 3, nous faisons la lumière sur l'extraction d'information interactive, un paradigme qui a vu le jour pour permettre à l'utilisateur de comprendre, d'explorer et d'améliorer les résultats des systèmes d'extraction d'information. Nous décrivons, dans ce chapitre, 4 types de systèmes d'EI interactifs : les systèmes entraînés de manière interactive, les systèmes basés sur l'apprentissage actif, les systèmes dotés d'interfaces de visualisation et les systèmes d'aide au développement d'extracteurs d'information à base de règles. Ce chapitre montre qu'il est nécessaire de combiner les avantages de plusieurs systèmes interactifs existants.

Le chapitre 4 décrit la plateforme UIMA choisie pour mettre en place la chaîne d'annotation linguistique des corpus d'étude, le langage Ruta choisi pour écrire et induire des règles d'EI et la plateforme TextRuler utilisée dans l'approche proposée.

Le chapitre 5 présente la problématique, explicite les objectifs et détaille les enjeux de ce travail. Nous présentons dans ce chapitre les propriétés caractéristiques d'un système d'EI interactif : la compréhensibilité des règles, la généricité des règles, la stabilité des règles, la sélection d'exemples et le temps d'apprentissage. Le chapitre 5 présente également les corpus utilisés pour évaluer l'approche proposée dans ce travail : le corpus de BioNLP-ST 2013 et le corpus SyntSem.

Le chapitre 6 détaille les différentes méthodes et stratégies proposées dans ce travail pour mettre en place l'approche interactive d'apprentissage de règles d'EI. Les détails de leur mise en œuvre et leur implémentation figurent dans le chapitre 7.

Le chapitre 8 contient la description de toutes les expérimentations réalisées dans le cadre de ce travail pour valider les différentes propositions.

Le chapitre 9 clôture ce rapport en dressant un bilan du travail réalisé et en donnant quelques perspectives.

Chapitre 2

Extraction d'information

Nous présentons dans ce chapitre un état de l'art autour de l'extraction d'information (EI). Avant d'être interactive, l'approche que nous mettons en place dans ce travail est une approche d'EI. Par conséquent, il est primordial de comprendre ce qu'est l'EI, en quoi consistent les tâches d'EI, de quoi est composé un système d'EI type, comment on évalue les systèmes d'EI et surtout quelles sont les différentes approches d'EI, leurs avantages et leurs inconvénients. Nous tâchons, dans ce chapitre, d'éclaircir ces différents points.

2.1 Définition

L'extraction d'information (EI) est un sous-domaine du Traitement Automatique des Langues (TAL) qui a pour but d'identifier dans des textes relatifs à un domaine spécifique un ensemble de concepts prédéfinis pour en produire des données non ambigües dans un format structuré (*template*, formulaire, etc.). Autrement dit, l'EI permet de dériver des informations structurées à partir de données non structurées. Considérons un exemple d'extraction d'information à propos d'une catastrophe naturelle :

Xynthia, une violente tempête a traversé la France le 28 février entre 0h et 17h. Le dernier bilan national fait état de 53 morts, 7 blessés graves et 72 blessés plus légèrement. Au total, plus de 500 000 personnes ont été sinistrées à des degrés divers.

Champ à remplir	**Valeur**
TYPE	Catastrophe naturelle
SOUS-TYPE	Tempête
LIEU	France
NOMBRE DE MORTS	53
NOMBRE DE BLESSÉS	79
DATE	le 28 février entre 0h et 17h

FIGURE 2.1 – Exemples d'informations extraites à partir d'un article de journal à propos d'une catastrophe naturelle.

La figure 2.1 montre un extrait d'un article de journal qui parle d'une catastrophe naturelle et les informations structurées dérivées de cet extrait sous forme de paires champ-valeur. Les champs correspondent aux attributs à remplir dans le formulaire cible et les valeurs aux segments de texte extraits pour remplir les champs.

2.2 Architecture type d'un système d'extraction d'information

D'après Appelt (1999), un système d'EI typique contient un ensemble de phases pour assurer une tokenisation des données d'entrée, une analyse lexicale et morphologique, une analyse syntaxique de base et une identification des informations relatives à l'application en question (voir figure 2.2).

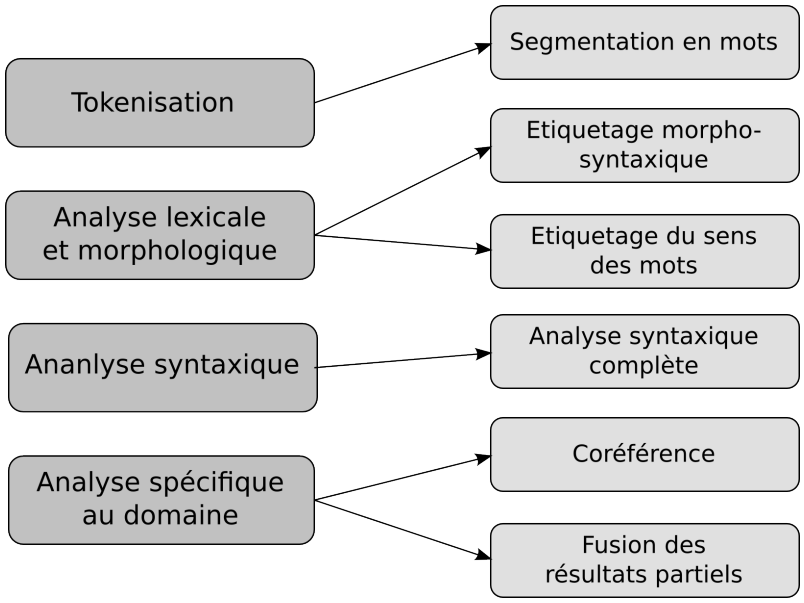


FIGURE 2.2 – Architecture type d'un système d'extraction d'information. Schéma tiré de (Appelt, 1999).

Certaines phases peuvent ne pas être nécessaires à certaines applications. En plus des modules principaux figurant dans la partie gauche de la figure, certains systèmes d'EI comportent des modules supplémentaires (partie droite de la figure) selon les besoins de l'application.

Tokenisation - Elle permet de segmenter le texte d'entrée en unités appelées *tokens* et de les classer selon leur type (mots écrits en majuscules, mots écrits en miniscules, mots contenant un trait d'union, signes de ponctuation, nombres, etc.).

Analyse lexicale et morphologique - Elle permet d'extraire à partir des *tokens* des informations morphologiques comme la forme canonique des mots (le lemme) et les parties de discours (nom, verbe, adjectif, etc.). Une désambiguïsation est réalisée pour certains mots qui sont ambigus face à certaines catégories morphologiques.

Analyse syntaxique - Elle a pour but d'identifier la structure syntaxique du document analysé. La plupart des systèmes d'EI réalisent une analyse superficielle de certains fragments de texte mais certains n'en font pas. Les systèmes d'EI s'intéressent à des informations spécifiques dans le texte et ignorent les portions de texte qu'ils ne jugent pas pertinentes pour leur tâche. Il est, par conséquent, inutile d'analyser ces portions.

Analyse spécifique au domaine - Ce module est le cœur de tout système d'EI. Les modules précédents permettent de préparer le texte pour l'analyse de domaine. Ce module permet de remplir des templates ou formulaires qui sont généralement construits sous forme de paires attribut-valeur ou champ-valeur. Les templates sont, en effet, des collections de champs ou d'attributs qui doivent être remplis chacun par une ou plusieurs valeurs. Les champs d'un template peuvent, par exemple, être remplis par des dates, des noms d'organisations, etc.

Le système d'EI que nous proposons dans ce travail ne contient pas de module d'analyse syntaxique. Nous nous contentons donc des modules de tokenisation, d'analyse lexicale et morphologique et d'analyse spécifique au domaine.

2.3 Tâches d'extraction d'information

L'EI a comme but l'identification dans un texte écrit en langage naturel des instances d'une classe particulière d'entités, de relations ou d'évènements ainsi que l'extraction de leurs propriétés. Les informations extraites sont ensuite stockées dans des structures prédéfinies par l'utilisateur (formulaire, template, etc.) contenant un ensemble de champs ou d'attributs qui sont instanciés par le système d'EI.

Les tâches classiques d'EI sont les suivantes.

Reconnaissance d'entités nommées (REN) – La REN est la plus simple et la plus fiable des tâches d'EI. Les systèmes de REN permettent d'identifier les entités nommées comme les personnes, les organisations, les dates, les montants, etc. Les tâches de REN peuvent être accomplies avec une précision de 95%. Étant donné que les annotateurs humains n'atteignent pas un niveau de précision égal à 100%, nous pouvons dire que les tâches de REN atteignent des niveaux de performances humaines. L'inconvénient majeur des systèmes de REN réside dans leur dépendance vis-à-vis du domaine étudié. Le changement du domaine d'étude implique le plus souvent des changements majeurs.

Résolution de coréférence (CO) – Cette tâche permet d'identifier toutes les mentions de la même entité. Les mentions en question sont soit des entités identifiées par la tâche de REN, soit des références anaphoriques à ces entités. « GIT » et « gastrointestinal tract » sont, par exemple, deux mentions d'une même entité. La CO est d'une grande utilité pour la tâche d'extraction d'évènements car elle permet d'associer des informations descriptives dispersées dans les textes aux entités auxquelles elles réfèrent. Contrairement à la tâche de REN, la CO est une tâche accomplie avec des valeurs de précision

faibles surtout quand il s'agit de résoudre des références anaphoriques. Elle est également dépendante du domaine d'étude.

Extraction de relations (ER) – Elle permet de détecter les relations entre les entités identifiées dans le texte. Même s'il existe généralement beaucoup de relations intéressantes dans un texte, les relations à extraire dans une tâche donnée sont prédéfinies et font partie intégrante des spécifications de la tâche. « EmployéDe » peut être, par exemple, une relation entre une personne et une organisation. L'extraction de relations est une tâche qui dépend peu du domaine et peut atteindre environ 75% de précision (Cunningham, 2005).

Extraction d'évènements (EE) – Cette tâche permet d'identifier des évènements dans un texte non structuré et de les exprimer de manière structurée. Elle inclut l'extraction de plusieurs entités et les relations entre elles. L'extraction d'informations à propos d'une attaque terroriste, par exemple, implique l'identification des acteurs, des victimes, du nombre de blessés/tués, des armes utilisées et du lieu de l'attaque.

Dans le programme ACE, le successeur des campagnes MUCs, les systèmes d'EI n'extraient généralement pas des templates du type de ceux des MUCs. Toutes les tâches sont, en effet, confondues dans une seule tâche (Doddington et al., 2004).

L'augmentation de l'utilisation des ontologies et du web des données a conduit à une nouvelle définition des entités au regard de ces ressources dans les systèmes d'EI. Le terme « annotation sémantique » est parfois utilisé pour désigner l'annotation du texte avec des sémantiques définies dans des ressources externes comme les ontologies. Ce processus est également appelé l'extraction d'information basée sur les ontologies (Bontcheva, Tablan, Maynard, & Cunningham, 2004) (voir section 2.8).

Avec le développement des réseaux sociaux et l'attractivité des informations qu'ils diffusent, une nouvelle génération d'extraction d'information a vu le jour : l'extraction d'information ouverte (Open Information Extraction, OIE) (Banko, Cafarella, Soderland, Broadhead, & Etzioni, 2007). Ce type d'extraction d'information ne présente pas les mêmes objectifs et défis que l'EI classique (voir section 2.9).

Nous ne visons pas, dans ce travail, une tâche particulière d'EI, le but étant de proposer une approche générique d'extraction d'information interactive qui peut s'appliquer dans le cadre de n'importe quelle tâche d'extraction d'information. Il suffit pour cela d'adapter le module d'EI à la tâche cible.

2.4 Évaluation en extraction d'information

2.4.1 Campagnes d'évaluation

La recherche dans le domaine de l'EI a souvent été suivie et évaluée à travers une série de campagnes d'évaluation sponsorisées par le gouvernement des États-Unis. Au milieu des années 80, il y a eu plusieurs tentatives pour extraire des informations à partir de journaux (DeJong, 1982) et de rapports médicaux (Sager, Friedman, & Lyman, 1987) mais l'évaluation était limitée, surtout quand il s'agissait de comparer des systèmes entre eux car il était difficile de dire quel progrès avait été fait.

Pour mieux évaluer le progrès, la DARPA (agence de projets de recherche avancés dans le domaine de la défense) a initié une série de campagnes d'évaluation (MUCs) [1]. Ces campagnes ont été lancées dans le but de coordonner les différents groupes de recherche qui cherchaient à améliorer les techniques d'EI et de recherche d'information (Grishman, 1996). Les MUCs ont défini plusieurs tâches d'EI et ont invité la communauté scientifique à adapter leurs systèmes à ces tâches. Elles fournissaient aux participants une description détaillée des tâches à accomplir, des données d'apprentissage et une période de temps limitée (de 1 à 6 mois) pour adapter leurs systèmes. Durant la phase de test, chaque participant reçoit un nouvel ensemble de documents de test, applique son système d'EI sur ces documents et renvoie les templates remplis aux organisateurs de la MUC. Les résultats sont ensuite comparés à des templates remplis manuellement par des annotateurs humains.

Les deux premières MUCs (1987-1989) se sont concentrées sur l'analyse automatique de messages militaires contenant des informations textuelles sur les batailles navales où le template à remplir contenait 10 champs (attributs). La MUC-3 (Lehnert, Cardie, Fisher, Riloff, & Williams, 1991) s'est intéressée à l'extraction à partir d'articles de presse des informations concernant des activités terroristes, des fondations internationales, des évènements de succession de gestion d'entreprises et des lancements de missiles et de véhicules spatiaux. Les structures des templates à remplir s'est complexifiée au fil du temps. À partir de MUC-5, la structure imbriquée des templates et l'EI multilingue ont été introduites. Les dernières MUCs ont défini plusieurs sous-tâches d'EI dans le but de faciliter l'évaluation et l'identification de sous-composants d'EI utilisables immédiatement. Les tâches d'EI génériques définies dans MUC-7 (1998) ont fourni progressivement des informations de haut niveau sur les textes.

Le programme ACE [2] lancé en 1999 dans la continuité des campagnes MUCs a comme but l'aide au développement de techniques d'extraction automatique de contenu pour le traitement automatique du langage naturel dans le texte depuis différentes sources (Doddington et al., 2004). Ce programme a défini de nouvelles tâches d'EI plus complexes centrées autour de l'extraction d'entités, de relations et d'évènements (voir section 2.3). La complexité de ces tâches est due essentiellement à :

— la considération de différentes sources d'information (journaux, pages web, groupes de discussion, etc.) et de la qualité des données d'entrée (transcriptions de conversations téléphoniques par exemple) ;
— l'introduction de types d'entités plus fins (aménagements, entités géopolitiques, etc.), de structures de templates et de types de relations ;
— l'élargissement de la portée des tâches d'EI de base. La tâche classique de REN a, par exemple, été convertie en tâche de détection et de suivi d'entités qui permet de détecter toutes les mentions d'entités dans le texte qu'elles soient nommées, nominales ou pronominales.

Le programme ACE a été assisté par le consortium de données linguistiques [3] (LDC) qui a permis de fournir des manuels d'annotation, des corpus et d'autres

1. http://www-nlpir.nist.gov/related_projects/muc/
2. http://www.itl.nist.gov/iad/mig/tests/ace/
3. https://www.ldc.upenn.edu/

ressources linguistiques qui ont été utilisés pour évaluer les systèmes d'EI d'une manière similaire à celle utilisée dans les campagnes MUCs. Des efforts ont été faits pour préparer les données pour des langues, autres que l'Anglais, comme l'Espagnol, le Chinois et l'Arabe.

MUC et ACE sont d'une grande importance dans le domaine d'EI. Ils ont fourni un ensemble de corpus disponibles à la communauté scientifique pour évaluer les systèmes et les approches d'EI.

Les systèmes et approches d'EI ont également été évaluées de manière plus élargie. La conférence CONLL a, par exemple, organisé des compétitions (*shared tasks*) autour des tâches de REN indépendantes du langage [4]. La conférence TAC [5] dédie également depuis 2009 une tâche appelée KBP (peuplement de bases de données) à l'évaluation des tâches centrées autour de la découverte d'informations sur les entités (extraction des attributs des entités) dans les corpus et leur incorporation dans une base de données. D'autres tâches liées à l'extraction d'information ont été évaluées dans Senseval [6] dont le but est d'évaluer les systèmes d'analyse sémantique.

2.4.2 Fonctions de mesure de performance

La précision et le rappel sont des mesures adoptées par la communauté de recherche d'information pour évaluer la sortie d'un système d'EI. Elles permettent de mesurer l'efficacité du système d'un point de vue utilisateur. Autrement dit, elles permettent de déterminer jusqu'à quel point le système produit toutes les sorties appropriées (rappel) et seulement les sorties appropriées (précision). La précision et le rappel peuvent ainsi être perçus comme des mesures d'exhaustivité et d'exactitude.

Pour définir ces mesures de manière formelle, nous posons les notations suivantes.

— GS : le nombre total de champs à remplir au regard d'un corpus de référence annoté qui représente un *gold-standard*.
— C : le nombre de champs remplis correctement dans la réponse du système.
— I : le nombre de champs remplis incorrectement dans la réponse du système.

On dit qu'un champ ou un attribut est rempli de manière incorrecte s'il ne correspond à aucun champ du *gold-standard* ou si la valeur qui lui est attribuée est incorrecte.

La précision et le rappel sont donc définis comme suit :

$$Précision = \frac{C}{C+I} \qquad Rappel = \frac{C}{GS}$$

Pour évaluer de manière plus approfondie les performances des systèmes d'EI, la précision et la rappel sont souvent calculés pour chaque type de champ séparément. La F-mesure définie par la formule suivante est la moyenne harmonique pondérée de la précision et du rappel.

$$F - mesure = \frac{(\beta^2 + 1) \times précision \times rappel}{(\beta^2 \times précision) + rappel}$$

4. http://www.clips.ua.ac.be/conll2002/ner/
5. http://www.nist.gov/tac/
6. http://www.senseval.org/

où β est une valeur positive. Elle est égale à 1 quand le même poids est attribué aux deux mesures et inférieure à 1 quand la précision est privilégiée.

D'autres mesures sont également utilisées dans la littérature pour évaluer les systèmes d'EI comme la mesure SER (*Slot Error Rate*) (Makhoul, Kubala, Schwartz, & Weischedel, 1999).

$$SER = \frac{I + M}{GS}$$

où M représente le nombre de champs dans le *gold-standard* qui ne correspondent à aucun champ dans la réponse du système d'extraction d'information.

La mesure SER représente le rapport entre le nombre total d'erreurs dans l'extraction des champs et le nombre total des champs dans le corpus de référence.

Nous réutilisons, dans ce travail, les fonctions précision, rappel et F-mesure pour évaluer le module d'EI proposé dans le cadre de notre approche interactive.

2.5 Rôle des informations linguistiques en extraction d'information

Les informations linguistiques peuvent jouer un rôle très important dans l'identification de contenus pertinents. L'analyse linguistique offre, en effet, des informations supplémentaires qui permettent de caractériser les données extraites ainsi que leurs contextes. Plusieurs faits sont caractérisés par une certaine structure syntaxique, par des propriétés morphologiques ou représentés par des entités nommées. Pour pouvoir les identifier, des informations sur leurs propriétés linguistiques sont donc essentielles.

La quantité et les types d'informations linguistiques utilisées diffèrent d'un système d'EI à un autre. Dans le système BWI (Freitag & Kushmerick, 2000), un système d'EI à base de règles utilisé dans plusieurs tâches conventionnelles d'EI, les auteurs utilisent uniquement des informations de capitalisation, un asterisque et un gazetier (*gazetteer*) pour généraliser. Le système RAPIER (Califf & Mooney, 2003), un autre système à base de règles d'EI évalué sur des annonces d'emploi et des annonces de séminaires, utilise une méthodologie plus linguistique : il permet aux patrons d'exprimer des contraintes sur les mots, les étiquettes morpho-syntaxiques et les classes sémantiques attribuées aux cibles à extraire et aux éléments de contexte. Ciravegna (2001, 2003) montre, à travers l'algorithme $(LP)^2$, le rôle très important des informations linguistiques dans le processus d'induction de règles d'EI. $(LP)^2$ utilise des informations linguistiques pour généraliser sur les formes fléchies des mots (expression exacte). Il utilise des informations morphologiques pour résoudre le problème de dispersion de données, des informations morpho-syntaxiques pour généraliser sur les catégories lexicales et un module de reconnaissance d'entités nommées pour généraliser sur certaines classes sémantiques définies par l'utilisateur. Le concepteur de l'algorithme oppose plusieurs versions de son algorithme basées sur différentes combinaisons d'informations linguistiques et montre expérimentalement sur des tâches d'EI concernant des annonces de séminaires et des annonces d'emploi que plus on ajoute des informations linguistiques, meilleures sont les performances.

Opposé à certains systèmes de l'état de l'art qui le précèdent, $(LP)^2$ a été classé premier sur ces tâches. Pour extraire des informations (nom, dosage, mode, fréquence, durée, raison) sur les médicaments utilisés par des patients à partir de leurs rapports médicaux, Spasic, Sarafraz, Keane, et Nenadic (2010) utilisent une approche à base de règles s'appuyant sur les informations morphologiques, lexicales et syntaxiques de la cible à extraire (segmentation en mots et en phrases, étiquetage morpho-syntaxique, analyse syntaxique). Leur système a obtenu la meilleure F-mesure pour les champs durée et raison dans le cadre du challenge i2b2 2009.

Le rôle des informations linguistiques a également été étudié dans des travaux relatifs à la Reconnaissance d'Entités Nommées (REN).

Les informations morphologiques (capitalisation, digitalisation, formation des mots) ont largement été utilisées dans le domaine de la presse (Bikel, Schwartz, & Weischedel, 1999 ; Chieu & Ng, 2002 ; G. Zhou & Su, 2002) et dans le domaine biomédical (Collier, Nobata, & Tsujii, 2000 ; Gaizauskas, Demetriou, & Humphreys, 2000 ; Kazama, Makino, Ohta, & Tsujii, 2002 ; Nigel, Collier, & ichi Tsujii, 1999 ; Takeuchi & Collier, 2002). Dans le domaine de la presse, ces informations ont été jugées très utiles pour détecter les frontières des entités. Elles ont, en revanche, été jugées beaucoup moins utiles dans le domaine biomédical.

L'étiquetage morpho-syntaxique a été jugé non utile dans le domaine de la presse car l'utilisation des étiquettes morpho-syntaxiques s'est avérée incompatible avec l'utilisation d'autres informations plus fiables (capitalisation) pour la détection des frontières des entités (Bikel et al., 1999 ; G. Zhou & Su, 2002). Dans le domaine biomédical, même si la plupart des entités sont écrites en miniscules et par conséquent, les informations de capitalisation ne sont pas trop pertinentes, l'étiquetage morpho-syntaxique n'apporte aucune amélioration significative (Collier et al., 2000 ; Takeuchi & Collier, 2002).

La lemmatisation a fait l'étude de plusieurs travaux (Brunet, s. d. ; Xu & Croft, 1998) mais il ne semble pas y avoir des conclusions claires à son sujet. Dans certains travaux, elle permet d'améliorer les résultats (M. P. Jones & Martin, 1997) et dans d'autres elle n'améliore pas voire dégrade les résultats (Bilotti, Katz, & Lin, 2004).

Il est vrai que la lemmatisation présente l'avantage de réduire le nombre de formes à considérér et d'augmenter le nombre d'occurrences de chaque forme mais elle peut également entraîner une perte d'informations en remplaçant un mot par son lemme. Lemaire (2008), par exemple, pense que cette perte d'informations est préjudiciable aux algorithmes qui utilisent le contexte des mots puisque ce dernier n'est pas le même selon les formes. Il recommande donc d'être prudent dans l'utilisation de la lemmatisation quand les mots du corpus sont étudiés en fonction de leur contexte.

Nous évaluons, dans ce travail, l'impact de l'étiquetage morpho-syntaxique et de la lemmatisation sur les performances des règles inférées.

2.6 Approches d'extraction d'information à base de règles

Les méthodes d'EI peuvent être présentées de plusieurs manières. Sarawagi (2008) oppose, par exemple, d'un côté, les approches à base de règles écrites manuellement

aux approches à base d'apprentissage artificiel et, d'un autre côté, les approches à base de règles, qu'elles soient écrites manuellement ou apprises, aux approches à base d'apprentissage statistique. Reeve et Han (2005) rejoignent Sarawagi (2008) sur la première classification tandis que Hobbs et Riloff (2010) la rejoignent sur la deuxième classification. Siefkes et Siniakov (2005) distinguent trois types d'approches d'EI : les approches à base de règles, les approches basées sur la connaissance et les approches statistiques. Ils les comparent en s'appuyant sur le type de la tâche d'EI, le type du texte traité, les différentes propriétés et annotations considérées et le niveau d'annotation du texte exigé. Jiang (2012) et Grishman (2012), eux, classent les approches d'EI selon la tâche à accomplir, en approches pour l'extraction d'entités et en approches pour l'extraction de relations.

Dans ce chapitre, nous avons choisi de distinguer les approches à base de règles des approches à base d'apprentissage statistique tout en mettant l'accent sur les approches à base de règles qui nous intéressent dans ce travail.

Les premiers systèmes d'EI utilisent une approche d'ingénierie de connaissances (Appelt, 1999) où la création des connaissances linguistiques sous forme de règles ou de patrons est réalisée par des experts humains qui inspectent les corpus en se fiant à leur intuition. Ce procédé est généralement exécuté de manière itérative en commençant par un petit ensemble de règles d'extraction testées sur le corpus qui sont ensuite étendues jusqu'à ce qu'un bon compromis entre la précision et le rappel soit trouvé. La plupart des anciens systèmes d'EI souffrent d'un inconvénient majeur : ils sont monolingues et difficilement adaptables à de nouveaux scénarios. Ils ont cependant montré que des techniques de TAL relativement simples peuvent suffire pour résoudre certaines tâches d'EI.

Des recherches menées vers la fin des années 90 et au début du 21e siècle ont fait émerger des systèmes d'EI modulaires capables de traiter de manière robuste et efficace de grandes masses de données dans une multitude de langues. Plusieurs parmi ces systèmes s'appuient sur des règles et utilisent un méchanisme de *pattern-matching* pour identifier dans le texte des éléments qui ont une certaine signification sémantique vis-à-vis de la tâche d'EI. Ces règles spécifiques au domaine sont comprises plus facilement quand elles sont écrites de manière déclarative. Des langages pour écrire des règles ont donc émergé.

2.6.1 Formalismes de règles

Plusieurs formalismes ont été développés pour permettre d'écrire des règles d'extraction d'information. Nous pouvons par exemple citer :

- le langage CSPL (*Common Pattern Specification Language*) de Hobbs, Bear, Israel, et Tyson (1993) qui représente l'une des premières tentatives pour formaliser un langage de règles ;
- les éléments de patrons dans RAPIER (Califf & Mooney, 1998) ;
- les expressions régulières dans le système WHISK (S. Soderland et al., 1999) ;
- les expressions SQL comme dans Avatar (Jayram, Krishnamurthy, Raghavan, Vaithyanathan, & Zhu, 2006 ; Reiss, Raghavan, Krishnamurthy, Zhu, & Vaithyanathan, 2008) ;
- les expressions Datalog comme dans DBLife (Shen, Doan, Naughton, & Ra-

makrishnan, 2007) ;
— le langage JAPE (*Java Annotations Pattern Engine*) de Cunningham, Maynard, et Tablan (2000), dérivé de CSPL et introduit dans la plateforme GATE (Cunningham, 2002) afin d'extraire des informations en s'appuyant sur les annotations produites par cette plateforme ;
— ou encore plus récemment le langage Ruta (Atzmueller, Kluegl, & Puppe, 2008 ; Kluegl, Atzmueller, & Puppe, 2009) qui est en quelque sorte le pendant de JAPE pour la plateforme UIMA (Ferrucci & Lally, 2004).

Plusieurs langages d'extraction d'information déclaratifs dans la communauté des bases de données ont été développés comme AQL (Chiticariu, Krishnamurthy, et al., 2010 ; Li, Reiss, & Chiticariu, 2011), xLog (Shen et al., 2007), des extensions de SQL (Jain, Ipeirotis, & Gravano, 2009 ; Wang, Michelakis, Franklin, Garofalakis, & Hellerstein, 2010). Ces langages ont montré que des formalismes d'extraction d'information à base de règles sont possibles (Fagin, Kimelfeld, Reiss, & Vansummeren, 2013). Ils restent cependant peu connus dans la communauté de TAL.

Chiticariu et al. (2013) pensent qu'il est temps d'établir un langage de règles standard pour l'EI qui s'inspire des propositions et des expériences des 30 dernières années. Pour ce faire, les chercheurs doivent répondre aux questions suivantes :
— quel est le modèle de données approprié pour exploiter le texte, les annotations du texte et leurs propriétés ?
— peut-on établir un langage de règles déclaratif standard extensible qui permettra de traiter des données dans ce modèle avec un ensemble de constructions suffisamment expressives pour résoudre la plupart des tâches d'EI rencontrées ?

Les auteurs s'inspirent, dans leur article nommé « Rule-based information extraction is dead ! Long live rule-based information extraction systems ! », du succès de SQL dans la communauté de bases de données pour proposer des instructions pour réduire l'écart entre le monde industriel où les systèmes d'EI à base de règles dominent et le monde académique où ces systèmes ont été délaissés au profit des systèmes à base d'apprentissage statistique.

Sarawagi (2008) propose une définition générique d'une règle d'extraction qui est la suivante :

$$\textit{Patron contextuel} \rightarrow \textit{Action}$$

Le patron contextuel est constitué d'une ou plusieurs expressions régulières définies en s'appuyant sur les propriétés des *tokens* dans le texte ainsi que leurs contextes. L'action permet de décrire les différents types de marquage dans le texte :
— l'étiquetage d'une séquence de *tokens* ;
— l'étiquetage d'une entité ;
— l'insertion d'une étiquette de début et de fin à une entité ;
— l'attribution de multiples étiquettes à une entité ;
— l'étiquetage simultané de plusieurs entités ;
— l'étiquetage de relations en repérant les entités impliquées dans la relation à étiqueter, etc.

Dans certains langages évolués comme JAPE et Ruta, la partie action de la règle peut accéder aux différentes propriétés utilisées dans le patron contextuel et

s'en servir pour ajouter de nouvelles propriétés au fragment de texte annoté qui, elles-mêmes, vont servir à d'autres règles par la suite.

JAPE Le langage JAPE (*Java Annotation Patterns Engine*) (Cunningham et al., 2000) est une version du langage CSPL (Hobbs et al., 1993). Il fournit un transducteur à états finis sur les annotations qui s'appuie sur des expressions régulières.

La grammaire JAPE consiste en un ensemble de phases dont chacune est composée d'un ensemble de règles sous forme patron/action. Les phases s'exécutent séquentiellement et forment une cascade de transducteurs à états finis sur les annotations fournies par la plateforme GATE (Cunningham, 2002). La partie gauche des règles (*Left-Hand-Side*, LHS) décrit un patron d'annotation et la partie droite (*Right-Hand-Side*, RHS) contient des instructions de manipulation d'annotations. La RHS peut référer à des annotations couvertes par la LHS à travers des étiquettes associées aux éléments du patron.

Considérons l'exemple suivant :

```
1 Phase: Jobtitle
2 Input: Lookup
3 Options: control= appelt bebug= true
4
5 Rule: Jobtitle1
6 (
7  {Lookup.MajorType == jobtitle}
8  (
9   {Lookup.MajorType == jobtitle}
10 )?
11 )
12 :jobtitle
13 -->
14  :jobtitle.JobTitle = {rule = "JobTitle1"}
```

La LHS est la partie qui précède le signe `-->` et la RHS est la partie qui le suit. La LHS spécifie un patron à appliquer sur un texte annoté avec la plateforme GATE alors que la RHS spécifie l'action à exécuter sur le texte couvert par le patron. Dans l'exemple donné, une règle nommée `Jobtitle1` (ligne 5) cherche des correspondances dans un texte contenant des annotations de type `Lookup` avec un attribut `MajorType` égal à `jobtitle` (ligne 7), suivi optionnellement par un texte annoté avec le type `Lookup` avec un attribut `MajorType` égal à `jobtitle` (ligne 9). Une fois un segment de texte couvert par la règle est identifié, cette dernière lui attribue l'étiquette ou le label `jobtitle` (ligne 12). Dans la RHS, on trouve une référence au segment de texte concerné par le label `jobtitle` dans la LHS. Une annotation de type `JobTitle` est attribuée à ce segment de texte (ligne 14).

Dans la grammaire JAPE, on commence par donner un nom de phase à la grammaire (`Phase: Jobtitle` (ligne 1), par exemple). Les grammaires JAPE peuvent être cascadées, chaque grammaire étant ainsi considérée comme une phase. Le nom

de phase fait partie du nom de la classe Java pour les actions RHS compilées [7].

Une liste des types d'annotation utilisés dans la grammaire est également fournie. Dans le cas de l'exemple fourni précédemment, `Input: Lookup` (ligne 2) signifie que seul le type d'annotation `Lookup` est utilisé dans la LHS. Si aucune annotation n'est définie, toutes les annotations sont considérées.

Plusieurs options peuvent également être utilisées :

— Le contrôle (`control`) – Cette option définit la méthode de mise en correspondance de la règle (`appelt` dans l'exemple fourni).

— Le débogage (`debug`) – Quand cette option est fixée à vrai (true), si la grammaire est utilisée dans un mode `appelt` et s'il y a plus d'une correspondance, les conflits sont affichés sur la sortie standard.

Une large panoplie de fonctionnalités peuvent être utilisées avec JAPE, ce qui en fait un système puissant.

Ruta Le langage Ruta (anciennement appelé TextMarker (Atzmueller et al., 2008 ; Kluegl, Atzmueller, & Puppe, 2009)) est un langage de script générique qui ressemble au langage JAPE (Cunningham et al., 2000) mais qui repose sur la plateforme UIMA (Ferrucci & Lally, 2004) plutôt que GATE (Cunningham, 2002). Selon les expériences de certains utilisateurs, il est plus complet que JAPE qui est l'un des langages de règles les plus connus.

Ce langage est détaillé dans la section 4.3.

2.6.2 Stratégies de résolution de conflits entre les règles

Les systèmes d'EI à base de règles reposent sur une base de règles pouvant avoir des recouvrements. D'où la nécessité de résoudre les conflits entre ces différentes règles.

Plusieurs stratégies sont proposées dans la littérature pour résoudre le problème des conflits entre les règles dans un système d'EI à base de règles (Sarawagi, 2008). Ces stratégies sont généralement non standardisées et impliquent plusieurs heuristiques et la gestion d'exceptions. Nous pouvons citer dans cette section quelques pratiques communes :

Règles non ordonnées – Une stratégie commune consiste à considérer les règles comme une collection de disjonctions. Chaque règle s'applique indépendamment des autres. Il y a conflit quand deux segments de texte différents qui se chevauchent sont couverts par deux règles différentes. Dans ce cas, plusieurs politiques de résolution de conflits peuvent s'appliquer comme le fait de favoriser la règle qui couvre le segment de texte le plus long (c'est le cas dans GATE (Cunningham, 2002)), ou la fusion des segments de texte qui se chevauchent dans le cas où les règles correspondantes partagent la même action. L'avantage majeur de cette méthode d'organisation des règles est la flexibilité offerte à l'utilisateur pour définir ses règles sans se soucier des éventuels conflits avec les règles existantes.

7. Le nom de phase doit contenir uniquement des caractères alphanumériques et des tirets bas et ne doit pas commencer par un nombre.

Règles organisées comme un ensemble ordonné – Il s'agit, dans cette stratégie, de définir un ordre de priorité entre les différentes règles. Dans le cas d'un conflit entre deux règles, la règle la plus prioritaire est favorisée (Maynard, Tablan, Ursu, Cunningham, & Wilks, 2001). Dans les systèmes d'apprentissage de règles d'EI, la priorité entre les règles est fixée en s'appuyant sur la précision et la couverture des règles sur les données d'entraînement. Une pratique commune consiste à ordonner les règles dans l'ordre décroissant de la précision de la règle sur l'ensemble d'entraînement. L'un des avantages de cette stratégie est qu'elle permet de définir de nouvelles règles sur la base des actions de règles antérieures.

Règles exprimées à l'aide d'automates à états finis – Les deux formes de règles présentées précédemment peuvent être exprimées sous forme d'un automate déterministe à états finis. Cependant, au moment de la définition des règles, l'utilisateur est dispensé des détails de la formation de l'automate. Parfois, l'utilisateur souhaite définir explicitement l'automate complet pour contrôler la séquence des déclenchements des règles. C'est la stratégie adoptée, par exemple, dans Softmealy (Hsu & Dung, 1998) où chaque entité est représentée par un nœud dans un transducteur à états finis. Les nœuds sont connectés entre eux par des arêtes orientées. chaque arête est associée à une règle en s'appuyant sur les *tokens* qui doivent être satisfaits pour que l'arête soit sélectionnée. Par conséquent, chaque déclenchement d'une règle doit correspondre à un chemin dans le transducteur à états finis. Aussi longtemps qu'il existe un seul chemin entre un état de départ et un état final pour chaque séquence de *tokens*, il n'y a pas d'ambiguïté concernant l'ordre des déclenchements des règles. Cependant, pour augmenter le rappel, Softmealy permet le déclenchement de plusieurs règles dans un nœud. Il est donc nécessaire de rédiger manuellement un ensemble de décisions pour arbitrer entre elles.

La politique de résolution de conflits que nous utiliserons pour organiser notre ensemble de règles dépendra beaucoup de l'algorithme d'apprentissage de règles que nous réutiliserons car l'algorithme d'apprentissage de règles peut avoir sa politique propre.

2.6.3 Apprentissage de règles d'extraction d'information

Dans un système d'EI à base de règles, les règles sont souvent écrites par un expert du domaine. Cependant, ces règles peuvent être apprises de manière automatique à partir d'exemples grâce à des algorithmes d'apprentissage de règles.

Les approches d'apprentissage de règles regroupent les méthodes d'apprentissage inductives qui permettent d'acquérir des règles à partir de documents d'entraînement. Les règles apprises sont soit des règles *single-slot* qui permettent d'extraire des fragments de texte correspondant à un seul champ du template à remplir, soit des règles *multi-slot* capables d'extraire des fragments de texte correspondant à tous les champs du template à remplir.

Il existe plusieurs critères pour classer les algorithmes d'apprentissage de règles. Turmo, Ageno, et Català (2006) proposent, par exemple, comme critères le degré

de supervision, le type des règles apprises, le type du document d'entraînement, le paradigme d'apprentissage et la stratégie d'apprentissage. Hobbs et Riloff (2010) utilisent le degré de supervision comme critère principal. Sarawagi (2008), quant à elle, classe les algorithmes d'apprentissage de règles en deux catégories : les algorithmes ascendants et les algorithmes descendants.

Nous nous appuyons, dans ce chapitre, sur le degré de supervision comme critère de classification.

Algorithmes d'apprentissage supervisé de règles d'extraction d'information On dit qu'une méthode d'apprentissage de règles est supervisée quand elle requiert une intervention humaine pendant le processus d'apprentissage. Cette intervention consiste généralement à fournir des exemples d'entraînement. Ces exemples peuvent être fournis au début du processus dans la phase de prétraitement ou de manière dynamique dans un processus en ligne.

Les méthodes d'apprentissage de règles supervisées visent à réduire l'ingénierie requise pour développer un système d'EI adaptable à de nouveaux domaines, en proposant à un utilisateur, qui a un minimum de connaissances sur le domaine étudié, d'annoter des documents d'entraînement au lieu d'effectuer le travail complexe d'écriture de règles à la main.

Les algorithmes d'apprentissage de règles suspervisés peuvent utiliser plusieurs stratégies d'apprentissage comme l'élimination de candidats (PALKA (Kim & Moldovan, 1995)), la force brute (TIMES (Chai, Biermann, & Guinn, 1999)), des heuristiques de généralisation (l'algorithme de Basili, Pazienza, et Vindigni (2000)) ou de spécialisation (AutoSlog (Riloff, 1993) et AutoSlog-TS (Riloff, 1996)) (Hobbs & Riloff, 2010) mais la plus commune est la couverture.

Plusieurs systèmes d'EI s'appuient sur des algorithmes couvrants (séparer et conquérir) (Fürnkranz, 1999), effectuant une forme d'apprentisage inductif. Ces systèmes exigent le plus souvent une structure prédéfinie de la cible à extraire. À part l'algorithme WHISK (S. Soderland et al., 1999) qui utilise une technique d'apprentissage actif, ils requièrent généralement des corpus d'apprentissage complètement annotés où tous les fragments de texte qui correspondent à un champ de la structure cible sont marqués. Après l'apprentissage de règles couvrant une partie de ces instances d'apprentissage, ces algorithmes suppriment (séparent) ces instances de l'ensemble d'apprentissage et continuent d'apprendre des règles qui couvrent (conquièrent) certaines des instances restantes jusqu'à ce que toutes les instances d'apprentissage ou presque soient couvertes. La définition d'une instance et des propriétés utilisées dépendent de l'algorithme utilisé.

Parmi les algorithmes couvrants, nous pouvons citer Crystal (S. Soderland, Fisher, Aseltine, & Lehnert, 1995 ; S. G. Soderland, 1997), WHISK (S. Soderland et al., 1999) et $(LP)^2$ (Ciravegna, 2001, 2003).

Dans cette section, nous décrivons deux algorithmes d'apprentissage de règles d'EI qui ont fait leurs preuves : WHISK qui adopte une stratégie de couverture descendante et $(LP)^2$ qui adopte une stratégie de couverture ascendante.

WHISK est un système d'apprentissage de règles d'EI conçu pour travailler sur des textes non structurés comme ceux qu'on trouve dans les journaux et les

livres, des textes semi-structurés comme ceux qu'on trouve sur le web et des textes structurés. WHISK est capable d'apprendre aussi bien des règles *single-slot* que des règles *multi-slot* sous forme d'expressions régulières. Les patrons des règles de WHISK peuvent contenir du texte, des classes de caractères (chiffre, nombre, etc.) et le caractère « * » qui permet de sauter des caractères jusqu'à ce que la partie suivante du patron trouve une correspondance dans le texte. Des classes sémantiques de termes équivalents peuvent être définies par l'utilisateur et utilisées dans les règles. La classe `Bdrm`, par exemple, dans la règle qui suit correspond aux différentes formes et abbréviations du terme *Bedroom* dans les annonces de location qui correspondent aux données d'apprentissage.

```
ID::2
Pattern:: *(Digit)~»Bdrm*'$'(Number)
Output:: Rental {Bedrooms $1}{Price $2}
```

Les parenthèses délimitent les segments à extraire. La partie `Output` indique quels champs remplir avec les segments extraits. La règle précédente extrait deux champs : le nombre de chambres et leur prix dans une annonce de location.

WHISK utilise un algorithme couvrant descendant. Il commence par construire la règle la plus générale et la spécialise par la suite. Pour évaluer la qualité des règles, WHISK utilise un Laplacien défini comme : $Laplacien = \frac{e+1}{n+1}$ où n représente le nombre d'extractions de la règle et e le nombre d'erreurs parmi les extractions. En cas d'égalité de Laplacien entre deux règles, la règle la plus générale est utilisée.

Les règles de WHISK peuvent ne pas être optimales car WHISK adopte une méthode de descente en gradient (*hill climbing*). En effet, en spécialisant les règles, l'algorithme ajoute les termes un par un alors que parfois l'ajout de deux termes permet de créer une règle plus fiable. Bien éviemment, si WHISK ajoute le mauvais terme, il rate la règle la plus fiable mais il continue à ajouter des termes jusqu'à ce que la règle obtienne de bons résultats sur l'ensemble d'apprentissage. Une telle règle est plus limitée que la règle optimale car elle a tendance à avoir une faible couverture sur des instances non vues. Pour ne pas faire de surapprentissage, WHISK utilise un seuil de Laplacien pour arrêter l'expansion d'une règle (pré-élagage). Une étape de post-élagage est également utilisée pour éliminer les règles qui ont une mauvaise couverture sur l'ensemble d'apprentissage.

Pour réduire le nombre d'exemples d'apprentissage à étiqueter à l'avance par l'utilisateur, WHISK utilise une technique d'apprentissage actif. Il alterne le processus d'annotation avec l'apprentissage. À chaque itération, il propose à l'utilisateur un ensemble d'instances à étiqueter pour remplir les champs à extraire et induit par la suite un ensemble de règles à partir de l'ensemble d'apprentissage étendu. Les règles de WHISK n'utilisent ni des niveaux de confiance ni des schémas de vote pour sélectionner des instances à annoter. Le système propose des exemples à annoter à l'utilisateur parmi les 3 ensembles d'exemples suivants :

1. Instances couvertes par des règles existantes (pour améliorer la précision des règles) ;
2. Instances qui représentent des *near misses* couvertes par des généralisations minimales des règles existantes (pour améliorer le rappel des règles) ;

3. Instances non couvertes par aucune règle (pour vérifier s'il reste des règles à découvrir).

La proportion des instances à tirer au hasard parmi ces trois ensembles peut être définie par l'utilisateur. Par défaut, elle est égale à un tiers de chaque ensemble.

(LP)^2 apprend des règles qui ajoutent des étiquettes ou des balises SGML/XML au texte. Il s'appuie sur des règles d'annotation qui insèrent uniquement une balise (de début ou de fin) dans le texte. Autrement dit, chaque règle d'annotation se charge de reconnaître soit le début soit la fin d'un segment de texte qui correspond à un champ à extraire et non de reconnaître ou d'annoter le segment en entier en une seule fois comme la plupart des systèmes.

Les règles d'annotation de (LP)^2 sont inférées de manière ascendante à partir d'un corpus d'apprentissage étiqueté manuellement. L'algorithme commence par construire une règle initiale qui couvre une instance. Les k meilleures généralisations de chaque règle initiale sont stockées dans un ensemble appelé « ensemble des meilleures règles ». Étant donné que (LP)^2 est un algorithme couvrant, les exemples d'apprentissage couverts par une règle appartenant à cet ensemble sont supprimés de l'ensemble d'apprentissage.

(LP)^2 procède en 4 étapes :

1. Les règles d'annotation appartenant à l'ensemble des meilleures règles sont appliquées.
2. Des règles contextuelles sont appliquées au texte résultant. Il s'agit de règles dont la fiabilité n'était pas suffisante pour faire partie des meilleures règles, mais qui ont de meilleures performances quand elles sont restreintes aux étiquettes insérées dans la première étape (par exemple, une règle qui insère une balise de fin peut être appliquée étant donné qu'une balise de début correspondante a été insérée quelques mots avant).
3. Des règles de correction sont appliquées. Elles n'ont pas pour but d'ajouter ou de supprimer des étiquettes mais plutôt de changer la position d'une étiquette en la déplaçant de quelques mots en avant ou en arrière.
4. Enfin, les étiquettes invalides (balise non fermée par exemple) sont supprimées dans une étape de validation.

(LP)^2 est utilisé dans le système Amilcare (Ciravegna, Dingli, Wilks, & Petrelli, 2002) où la quantité des informations linguistiques utilisées peut être ajustée de manière dynamique. L'algorithme induit, tout d'abord, des règles qui n'utilisent aucune information linguistique. Ensuite, il itère en ajoutant des informations linguistiques (fournies par des composants tiers) et s'arrête quand l'efficacité des règles générées ne s'améliore plus. La quantité convenable d'informations linguistiques est apprise pour chaque type de champ à extraire séparément (reconnaître le nom d'une personne, par exemple, peut demander plus d'informations linguistiques que reconnaître une date).

Apprentissage peu supervisé de règles d'extraction d'information Les techniques d'apprentissage de règles supervisées ont permis de réduire l'effort humain requis pour la création d'un système d'EI adaptable à un nouveau domaine.

Cependant, l'annotation manuelle des exemples d'entraînement peut être coûteuse en temps.

Pour contourner ces inconvénients, plusieurs méthodes qui visent la réduction du degré de supervision des algorithmes d'apprentissage de règles ont vu le jour. Parmi ces méthodes, nous pouvons par exemple citer le *bootstrapping* (Brin, 1999 ; Yangarber, 2001, 2003). Il s'agit, en effet, d'un apprentissage qui se base sur un ensemble d'exemples graines (seed examples) ou de patrons graines (seed patterns) à partir duquel il aquiert des conditions contextuelles qui vont permettre d'annoter, par la suite, de nouveaux exemples qui, eux-mêmes, vont permettre d'appendre de nouvelles conditions contextuelles et ainsi de suite.

Nous décrivons, dans cette section, deux algorithmes d'apprentissage peu supervisé de règles d'EI : AutoSlog-TS de Riloff (1996) et Ex-Disco de Yangarber, Grishman, Tapanainen, et Huttunen (2000).

AutoSlog-TS (Riloff, 1996) est une extension d'AutoSlog (Riloff, 1993). Il nécessite uniquement un corpus pré-classé au regard de la pertinence de chaque document par rapport à la tâche d'intérêt. Il génère des patrons d'extraction pour chaque syntagme nominal dans le corpus d'apprentissage en utilisant des heuristiques. Il évalue ensuite les patrons d'extraction en analysant le corpus une deuxième fois et en générant des statistiques de pertinence pour chaque patron.

AutoSlog-TS procède en deux étapes. Dans la première étape, un analyseur de phrases identifie les syntagmes nominaux. Pour chaque syntagme nominal, les règles heuristiques génèrent un patron d'extraction. AutoSlog-TS utilise l'ensemble des règles heuristiques utilisées par AutoSlog plus deux autres. Dans la deuxième étape, le corpus d'apprentissage est traité une deuxième fois en utilisant les nouveaux patrons d'extraction. L'analyseur de phrases active tous les patrons applicables dans chaque phrase. Des statistiques de pertinence (taux de pertinence) pour chaque patron sont ensuite calculées.

Ex-Disco (Yangarber et al., 2000) applique une technique de *bootstrapping mutuel*. Il s'appuie sur l'hypothèse que la présence de documents pertinents annonce de bons patrons et les bons patrons peuvent trouver de bons documents. Étant donné un corpus non annoté et quelques patrons graines, l'ensemble des documents est divisé en un ensemble de documents pertinents contenant au moins une instance de patron et un ensemble de documents non pertinents ne contenant aucun patron graine. Des patrons candidats sont générés à partir des clauses dans les documents et classés en corrélation avec les documents pertinents. Le meilleur patron est ajouté à l'ensemble des patrons et chaque document est reclassé en utilisant le nouvel ensemble de patrons. Le document en entier est encore divisé en documents pertinents et documents non pertinents et le système continue à itérer.

2.7 Approches d'extraction d'information statistiques

Les approches statistiques sont plus récentes dans le domaine d'EI. Elles permettent de construire un modèle, appelé également classifieur, capable d'attribuer automatiquement les classes aux éléments du texte. Sarawagi (2008) trouve que les approches à base de règles, qu'elles soient écrites à la main ou inférées automatiquement, sont plus faciles à interpréter et à développer et plus utiles dans des domaines où l'intervention humaine est disponible et nécessaire alors que les approches statistiques sont plus robustes face au bruit dans des textes non structurés et plus utiles dans des textes comme les transcriptions de parole, les blogs, etc.

Plusieurs approches d'EI exploitent de plus en plus des méthodes d'apprentissage statistique bien connues mais qui n'étaient pas conçues initialement pour des tâches d'EI. Ces méthodes peuvent être divisées en deux catégories. La première catégorie regroupe les méthodes génératives. Le Modèle de Markov Caché (*Hidden Markov Model*, HMM) est l'un des classifieurs basés sur un modèle génératif les plus utilisés pour la reconnaissance d'entités nommées (REN). La deuxième catégorie regroupe les méthodes discriminantes. Les classifieurs discriminants qui modélisent directement la distribution à posteriori des attributs donnés d'une classe d'étiquettes comme les Séparateurs à Vaste Marge (*Support Vector Machines*, SVM) (Isozaki & Kazawa, 2002) et le modèle d'Entropie Maximale (*Maximum Entropy*, ME) pour la REN (Chieu & Ng, 2003) obtiennent généralement des résultats meilleurs que les classifieurs basés sur les modèles génératifs. Plus récemment, les Champs Conditionnels Aléatoires (*Conditional Random Fields*, CRF) (McCallum & Li, 2003 ; Peng, Feng, & McCallum, 2004) qui sont des classifieurs discriminants ont été proposés pour les problèmes d'étiquetage de séquences.

Nous donnons, dans cette section, une description brève de ces différentes méthodes.

2.7.1 Modèles de Markov Cachés

Les modèles de Markov cachés (HMM) sont des modèles génératifs qui s'appuient sur l'estimation de la probabilité de génération de l'observation à partir de l'état courant étant donné l'état précédent. Les HMM ont été introduits par Baum et Petrie (1966) puis repris par Rabiner (1989) pour des traitements acoustiques. Dans les processus markoviens, l'évolution d'un état e_t à un instant t vers l'état e_{t+1} à l'instant $t+1$ ne dépend que de l'état courant e_t. Autrement dit, le futur est indépendant du passé connaissant le présent. Dans le cas d'un HMM, la chaine de Markov n'est pas observable : on peut observer seulement les variables aléatoires reliées à la chaine.

Plusieurs travaux se sont appuyés sur l'apprentissage de différentes variantes de HMM pour l'extraction de fragments pertinents à partir des documents disponibles sur le web. Les HMM ont largement été appliqués à différentes tâches de TAL (étiquetage morpho-syntaxique, reconnaissance d'entités nommées, reconnaissance de la parole) et plus récemment à des tâches d'EI. Les HMM fournissent des outils probabilistes efficaces et robustes. Néanmoins, ils nécessitent une connaissance à

priori de la structure du modèle (le nombre des états et des transitions entre les états).

Leek (1997) est l'un des premiers à avoir introduit les HMM dans des tâches d'EI. Il construit manuellement une architecture spécifique de HMM pour l'extraction d'informations dans des corpus biomédicaux. Zaragoza et Gallinari (1998) utilisent un HMM pour extraire des informations dans un corpus de journaux financiers. L'extraction se fait en deux étapes. Dans la première étape, un classifieur à base de réseaux de neurones multi-couches est appliqué aux phrases pour repérer celles qui sont susceptibles de contenir de l'information pertinente. Dans la deuxième étape, un HMM permet l'identification précise de l'information à l'intérieur des phrases sélectionnées.

Freitag et Mccallum (1999) utilisent les HMM pour extraire des informations dans un corpus d'annonces de séminaires. Ils proposent une méthodologie dans laquelle un HMM séparé est construit manuellement pour chaque champ à extraire. La structure du HMM modélise les préfixe et suffixe immédiats ainsi que la structure interne de chaque champ. Pour chaque HMM, les probabilités de transition d'état et d'émission de mot sont apprises à partir des données annotées. Dans une extension de cette approche (Freitag & McCallum, s. d.), la même tâche d'EI est reprise mais en se focalisant sur l'apprentissage d'une structure de HMM pour chaque champ à partir de données d'apprentissage spécifiques et limitées. Démarrant avec un modèle simple, un processus de *hill climbing* est lancé dans l'espace des structures possibles à chaque étape en appliquant les 7 opérations possibles définies (division d'états, ajout d'états, etc.) sur le modèle et en sélectionnant la structure qui obtient le meilleur score comme prochain modèle. Les résultats expérimentaux montrent que cette approche est plus performante que l'ancienne.

Seymore, Mccallum, et Rosenfeld (1999) explorent l'utilisation des HMM dans des tâches d'EI en se focalisant sur comment apprendre la structure du modèle à partir des données et comment utiliser au mieux les données étiquetées et non étiquetées. Ils introduisent le concept des « données étiquetées à distance » (distantly labeled data) qui sont des données étiquetées provenant d'un autre domaine dont les étiquettes se chevauchent partiellement avec celles du domaine cible. Ils prouvent qu'un modèle construit manuellement qui contient plusieurs états par champ extrait est plus performant qu'un modèle avec un seul état par champ.

D'autres approches utilisent non seulement les mots mais aussi des attributs additionnels attachés aux mots (étiquettes morpho-syntaxiques, capitalisation, position dans le document, etc.) et des attributs relatifs aux séquences de mots (longueur, indentation, nombre de caractères blancs, etc.). C'est le cas de l'approche présentée par McCallum, Freitag, et Pereira (2000) dans laquelle la tâche de la segmentation des questions fréquemment posées en leurs constituants est accomplie. L'approche introduit le modèle de Markov à entropie maximale (Maximum Entropy Markov Model, MEMM), un modèle à états finis à probabilité conditionnelle dans lequel les paramètres du modèle HMM génératif sont remplacés par une seule fonction qui combine les paramètres de transition et d'émission.

Ray et Craven (2001) sont les premiers à avoir utilisé les HMM dans des tâches d'EI à partir du texte non structuré pour extraire des relations n-aires (multi-slot). Ils explorent une approche qui tente d'incorporer les informations sur la structure

grammaticale des phrases dans les architectures de HMM. Les états dans le HMM représentent des segments annotés d'une phrase (déjà analysée syntaxiquement). Les auteurs adoptent une méthode d'apprentissage qui maximise la probabilité d'attribuer les bonnes étiquettes aux différentes parties des phrases traitées au lieu de maximiser la vraisemblance des phrases elles-mêmes.

Skounakis, Craven, et Ray (2003) étendent le travail de Ray et Craven (2001) en définissant les modèles de Markov cachés hiérarchiques (Hierarchical Hidden Markov Models, HHMM), des HMM avec plus qu'un niveau d'états. Ces modèles sont utilisés pour définir une représentation grammaticale multi-niveaux plus riche des phrases. Les HHMM ont ensuite été étendus pour y introduire des informations de contexte.

L'un des inconvénients des HMM consiste dans le fait que ce modèle génératif fait des hypothèses indépendantes sur ses observations en limitant les connaissances que peuvent fournir les observations antérieures et futures.

2.7.2 Modèles d'Entropie Maximale

Le principe de l'entropie maximale (ME) dit que parmi toutes les distributions qui satisfont des contraintes sur les attributs, il faut choisir la distribution avec la plus grande entropie puisqu'elle fait le moins d'hypothèses à propos des données et a, par conséquent, une meilleure capacité de généralisation sur les données non vues.

Les modèles de ME estiment des probabilités en essayant de faire le moins d'hypothèses possibles, autres que les contraintes imposées. Ils peuvent facilement combiner plusieurs attributs. Ces attributs peuvent être assez complexes et permettent d'exploiter des connaissances à priori concernant le type d'informations susceptibles d'être importantes pour la classification. La ME est largement utilisée dans les tâches de TAL et de fouille de données comme l'étiquetage morpho-syntaxique, la reconnaissance d'entités nommées et l'extraction de relations. Comme McCallum et al. (2000), Chieu (2002) utilise le modèle de ME mais au lieu de s'appuyer sur des modèles de Markov, il utilise une approche basée sur la classification. Il développe deux techniques : l'une pour une extraction d'information *single-slot* sur du texte semi-structuré (annonces de séminaires) et l'autre pour une extraction d'information *multi-slot* sur du texte non structuré (sur la succession de gestion). L'auteur utilise la ME pour classer les mots en champs. Le mot qui précède et le mot qui suit le mot étiqueté dans l'ensemble d'apprentissage sont utilisés pour la classification. Pour limiter le problème de la séquence de classes inadmissibles, une probabilité de transition est définie entre les classes de mots, ensuite l'algorithme Viterbi (Viterbi, 1967) est appliqué pour sélectionner la séquence de classes de mots qui obtient la meilleure probabilité. Dans la tâche *multi-slot*, la ME est utilisée pour classer les relations entre les champs à extraire en relations positives (existantes) et relations négatives (non existantes).

Kambhatla (2004) utilise le modèle de ME pour prédire le type de relation entre chaque paire d'entités dans chaque phrase. Cette tâche de prédiction est modélisée comme un problème de classification avec jusqu'à deux classes pour chaque sous-type de relation défini par la campagne ACE (la plupart des relations ne sont pas symétriques) et une classe supplémentaire au cas où il n'existe aucune relation entre les deux entités. Le modèle de ME est entraîné en utilisant des combinaisons d'attri-

buts lexicaux, sémantiques et syntaxiques. Il permet une extension facile du nombre et du type des attributs considérés.

2.7.3 Champs Aléatoires Conditionnels

Le modèle des champs aléatoires conditionnels (CRF) (Lafferty, McCallum, & Pereira, 2001) est un modèle graphique linéaire non dirigé qui représente un cadre probabiliste discriminant utilisé pour la segmentation et l'étiquetage des données séquentielles. Les CRF modélisent directement la distribution conditionnelle d'une variable cible étant donné la variable observée. Autrement dit, aucune ressource de modélisation n'est gaspillée dans la modélisation de structures de corrélation complexes dans les séquences observées. Comme les HMM, les CRF permettent de prédire l'étiquette d'une séquence en incorporant l'aspect temporel. L'avantage que présentent les CRF par rapport aux modèles markoviens classiques consiste dans le fait de prendre en compte le problème du biais des étiquettes. En effet, les transitions des états ne dépendent pas seulement des états concernant la transition (états voisins) mais aussi des états du modèle global. D'autre part, les CRF peuvent prendre plusieurs paramètres en entrée (attributs des éléments d'entrée comme la position syntaxique des mots dans un texte). Ceci permet d'utiliser plusieurs niveaux hiérarchiques d'étiquetage.

Dans la littérature, le modèle de CRF est considéré comme le système de reconnaissance d'entités nommées supervisé qui produit les meilleurs résultats dans les tâches de REN classiques (McCallum & Li, 2003 ; Peng et al., 2004), dans les tâches de REN dans les textes biomédicaux (Settles, 2004) et dans plusieurs langues autres que l'Anglais comme le Bengali (Ekbal & Bandyopadhyay, 2008).

Aramaki, Eiji and Imai, Takeshi and Miyo, Kengo and Ohe, Kazuhiko (2006) ont participé au challenge i2b2 2006 avec un système d'anonymisation qui s'appuie sur le CRF. Les meilleurs résultats dans ce challenge ont été obtenus par Wellner et al. (2007) qui combinent deux outils : l'un basé sur les CRF (Carafe) et l'autre sur les HMM (LingPipe).

Gardner et Xiong (2008) ont mis en place un système nommé « Hide » pour anonymiser 4 catégories de termes : les noms, les âges, les dates et les identifiants numériques. Ce système a été conçu comme un système de REN qui s'appuie sur des CRF.

Plusieurs modifications ont été apportées aux CRF pour prendre en considération les dépendances non locales (Krishnan & Manning, 2006) ou un contexte plus large que les données d'apprentissage (Du, Zhang, Yan, Cui, & Chen, 2010).

2.7.4 Séparateurs à Vaste Marge

Les séparateurs à vaste marge (SVM) sont largement utilisés dans les problèmes de classification. Cependant, plusieurs travaux ont tenté de les appliquer à des tâches d'EI. Pour ce faire, il faut représenter le problème d'EI comme un problème de classification. Une fois le problème transformé, plusieurs méthodes d'apprentissage peuvent être appliquées.

Les SVM sont à la base des classifieurs discriminants à deux classes. Cependant, ils peuvent être étendus pour être utilisés dans des problèmes multi-classes. Le principe de base des SVM repose sur le fait que l'algorithme cherche à calculer l'hyperplan qui sépare le mieux un espace en 2 classes (Vapnik, 1995). L'algorithme intègre généralement une fonction noyau qui permet de transposer un espace de données dans un autre espace linéairement séparable. Une optimisation est réalisée pour maximiser la distance entre l'hyperplan séparateur et les points les plus proches de chaque classe (Wisniewski & Gallinari, 2007). Les données se retrouvent ainsi le plus loin possible de l'hyperplan.

Sun, Naing, Lim, et Lam (2003) ont été les premiers à avoir utilisé les SVM dans des tâches d'EI. Pour permettre la prise en considération de plus d'informations de contexte dans l'apprentissage de patrons d'extraction, les auteurs proposent de modéliser les informations de contenu et de contexte de l'entité à extraire comme un ensemble d'attributs. Un modèle de classification est ensuite construit pour chaque catégorie d'entités en utilisant les SVM. Une approche similaire est utilisée dans le système ELIE (Finn & Kushmerick, 2004). Cette approche considère l'identification des positions de début et de fin d'un fragment comme des tâches de classification de *tokens* distinctes. Le système étudie également la contribution de différents ensembles d'attributs dans ses performances. Ekbal et Bandyopadhyay (2008) ont développé un système de REN à base de SVM pour les langues indienne et Bengali.

2.8 Extraction d'information et ontologies

La détection et l'extraction d'informations pertinentes dans les documents textuels dépend du degré de compréhension de ces ressources. Le rôle de la sémantique dans l'EI est souvent limité à un simple étiquetage sémantique superficiel. L'analyse sémantique est considérée plus comme un moyen de désambiguïser des phases syntaxiques que comme un moyen de construire une interprétation conceptuelle. La plupart des systèmes d'EI qui incluent une anlayse sémantique n'exploitent réellement qu'une petite partie des connaissances du domaine et des connaissances fournies par les tâches d'EI, autrement dit les entités nommées. Cependant, le besoin croissant d'adapter les applications d'EI à des domaines complexes qui demandent une compréhension plus profonde des textes pousse vers l'utilisation de ressources sémantiques plus sophistiquées et, entre autres, vers les ontologies considérées comme des modèles conceptuels.

Les ontologies permettent de modéliser et de formaliser les connaissances d'un domaine dans un langage lisible par une machine. Une ontologie est définie comme « une spécification formelle et explicite d'une conceptualisation » (Gruber, 1993 ; Studer, Benjamins, & Fensel, 1998). Il s'agit, en effet, d'un modèle abstrait (conceptualisation) qui permet d'identifier des concepts pertinents de manière explicite pour représenter un phénomène donné. Ce modèle doit être interprétable par une machine (formel).

Les ontologies fournissent des connaissances riches sur lesquelles on peut fonder la compréhension du texte et permettent d'extraire des informations pertinentes. Ceci permet de construire des systèmes d'EI plus flexibles et plus adaptatifs.

L'apparition du web sémantique (Berners-Lee, Hendler, & Lassila, 2001) a permis

le développement d'ontologies dans plusieurs domaines. De nos jours, des milliers d'ontologies de domaine sont disponibles gratuitement sur le web (Ding et al., 2004). Des ontologies polyvalentes bien détaillées ont également été développées (WordNet par exemple (Fellbaum, Christiane, 1998)).

Nédellec et Nazarenko (2005) pensent que l'EI et les ontologies sont impliquées dans deux tâches reliées :

— L'ontologie est utilisée pour l'EI – L'EI a besoin des ontologies comme une partie du processus de compréhension pour extraire des informations pertinentes.
— L'EI est utilisée pour peupler et améliorer l'ontologie – Les textes sont des sources de connaissances utiles pour concevoir et enrichir les ontologies.

Ces tâches peuvent être combinées dans un processus cyclique. Les ontologies sont utilisées pour interpréter le texte et en extraire des informations et l'EI permet d'extraire des nouvelles connaissances à partir du texte pour les intégrer dans l'ontologie.

Que les connaissances ontologiques soient utilisées pour interpréter du langage naturel ou pour exploiter les textes pour créer ou mettre à jour les ontologies, dans les deux cas, l'ontologie doit être liée à des phénomènes linguistiques. Plusieurs systèmes d'EI traditionnels ont tenté de définir des règles d'extraction qui permettent cet ancrage. Dans des systèmes d'EI plus puissants, les connaissances ontologiques sont déclarées de manière plus explicite dans les règles qui permettent de réduire l'écart entre le niveau des mots et l'interprétation du texte. L'ontologie n'est donc pas un modèle purement conceptuel, elle représente un modèle associé à un vocabulaire et une grammaire spécifique au domaine. Ce vocabulaire et cette grammaire sont considérés, dans les systèmes d'EI, comme une partie de l'ontologie même quand ils sont intégrés dans les règles d'extraction. La complexité de l'ancrage linguistique des connaissances ontologiques est un problème bien connu. Un concept peut, en effet, être exprimé avec différents termes dont certains peuvent être ambigus.

2.8.1 Extraction d'information guidée par les ontologies

L'évolution des outils de TAL, permettant d'analyser les textes de plus en plus finement et le développement de modèles sémantiques permettant de représenter plusieurs domaines, a permis l'émergence d'une nouvelle génération d'outils d'EI que sont les outils OBIE (*Ontology-Based Information Extraction*) (Wimalasuriya & Dou, 2010). En effet, Nédellec, Nazarenko, et Bossy (2009) montrent que l'EI est une activité du TAL basée sur les ontologies et explicitent les différentes étapes nécessaires pour les tâches d'EI : reconnaissance des entités nommées, analyse des termes, typage sémantique et identification des relations spécifiques. L'EI basée sur les ontologies est donc naturellement devenue une nouvelle approche de l'EI.

Les systèmes d'EI basés sur les ontologies ou guidés par les ontologies sont des systèmes qui s'appuient sur des ontologies prédéfinies dans une ou plusieurs étapes du processus d'EI. Contrairement aux systèmes d'EI traditionnels, ceux qui sont basés sur les ontologies sont capables d'exprimer leurs sorties dans les termes d'une ontologie formelle pré-existante. Ces systèmes utilisent généralement des ontologies de domaine. Cependant, un système est considéré comme indépendant du domaine

s'il fonctionne sans modification sur des ontologies qui couvrent plusieurs domaines.

Wimalasuriya et Dou (2010) définissent un système OBIE comme un système qui « traite du texte non structuré et semi-structuré à travers un mécanisme guidé par des ontologies afin d'extraire certains types d'informations et présente les résultats en utilisant des ontologies ». Selon eux, un système OBIE doit avoir les propriétés suivantes :

- Traiter des textes non structurés ou semi-structurés écrits en langage naturel – L'OBIE est un sous-domaine de l'EI qui, elle-même, est un sous-doamine du TAL. Il est, par conséquent, raisonnable de se limiter aux textes écrits en langage naturel comme textes d'entrée. Ces textes peuvent être non structurés (fichiers textuels) ou semi-structurés (pages web utilisant un certain template comme les pages Wikipédia).
- Présenter la sortie en utilisant les ontologies – Li et Bontcheva (2007) considèrent que le fait d'utiliser une ontologie formelle comme l'une des entrées du système et en même temps comme sortie cible est l'une des caractéristiques qui distinguent un système OBIE des systèmes d'EI traditionnels. Cependant, certains systèmes construisent eux-mêmes l'ontologie qu'ils utilisent ensuite dans le processus d'EI. Pour ne pas empêcher que ces systèmes soient considérés comme des systèmes OBIE, il est plus raisonnable de garder la contrainte uniquement sur la sortie du système qui doit s'exprimer en termes d'ontologies.
- Utiliser un processus d'EI guidé par une ontologie – Dans les systèmes OBIE, le processus d'EI est guidé par une ontologie pour extraire des classes, des propriétés et des instances. Autrement dit, aucun nouveau processus d'EI n'est inventé mais un processus existant est adapté pour identifier des composants de l'ontologie.

Les auteurs expliquent que le terme OBIE est récent mais que les travaux dans ce domaine ont débuté bien avant son apparition et insistent sur le rôle primordial des ontologies dans un processus d'EI. Ils précisent également que ce processus consiste à créer du contenu sémantique, élément essentiel au développement du web sémantique, et peut permettre d'améliorer la qualité des ontologies elles-mêmes.

Le module d'EI peut soit faire partie de l'ontologie soit être indépendant. Selon certains, les extracteurs d'informations doivent être considérés comme une partie de l'ontologie quand des règles linguistiques sont utilisées comme technique d'apprentissage (Buitelaar, Cimiano, Haase, & Sintek, 2009 ; Embley, 2004 ; Maedche, Neumann, & Staab, 2003 ; Yildiz & Miksch, 2007). Cette technique repose, en effet, sur des expressions régulières qui indiquent la présence de concepts ontologiques dans le texte. Wimalasuriya et Dou (2010) penchent plutôt pour l'utilisation des extracteurs d'informations comme modules à part pour deux raisons. D'un coté, les règles linguistiques peuvent contenir des erreurs et rendre l'ontologie moins formelle et d'un autre coté, il est difficile d'argumenter le choix d'inclure les extracteurs d'information à base de règles dans les ontologies et d'exclure ceux basés sur d'autres techniques des ontologies.

Il est également important de mentionner que le terme *Ontology-Driven Information Extraction* est utilisé à la place d'OBIE dans certaines publications (McDowell, 2006 ; F. Wu, Hoffmann, & Weld, 2008). Ce terme est souvent synonyme du terme

OBIE. Cependant, Yildiz et Miksch (2007) font la distinction entre les deux termes et considèrent que les systèmes d'EI guidés par les ontologies sont guidés par une ontologie alors que l'ontologie n'est qu'un composant de plus dans les systèmes OBIE. Nous sommes de l'avis de Wimalasuriya et Dou (2010) qui pensent que cette distinction ne doit pas être faite.

2.8.2 Architecture d'un système OBIE

Bien que les détails d'implémentation diffèrent d'un système OBIE à un autre, Wimalasuriya et Dou (2010) décèlent une architecture commune entre les systèmes OBIE composée des modules suivants (voir figure 2.3).

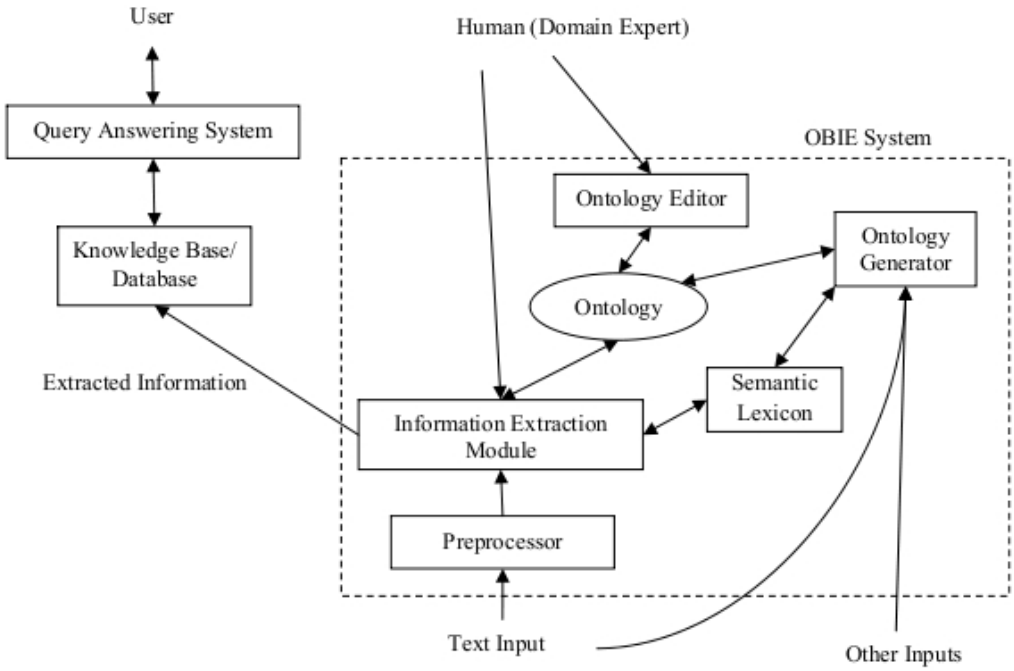


Figure 2.3 – Architecture générale d'un système OBIE(Wimalasuriya & Dou, 2010).

— Un **générateur d'ontologies** qui permet de construire une ontologie, éventuellement à partir d'une ontologie existante. Ce composant peut être absent dans le cas de l'utilisation d'une ontologie préexistante.
— Un **éditeur d'ontologies** permettant à un expert du domaine d'éditer les ontologies ou de les valider.
— Un **lexique sémantique** utilisé pour une langue donnée afin d'assister la génération d'ontologies, comme par exemple WordNet (Miller, 1995) pour l'Anglais. Certains systèmes OBIE utilisent ce lexique à la place de l'ontologie.
— Un **module de prétraitement** pour convertir le texte au format d'entrée du module d'EI.
— Un **module d'EI** qui alimente la base de connaissances ou de données avec les informations extraites du texte en se basant sur une ontologie ou un lexique

sémantique. Les informations extraites peuvent être représentées en utilisant un langage de définition d'ontologies comme le OWL. L'expert du domaine peut intervenir pour valider les informations extraites.

Un module OBIE peut faire partie d'un système de question-réponse qui utilise les informations extraites par le module OBIE pour répondre aux requêtes de l'utilisateur.

Nous notons que l'OBIE est similaire à l'annotation sémantique. En effet, les annotations sont des métadonnées spécifiques qui lient des entités apparaissant dans des ressources aux concepts de domaine modélisés dans une ontologie.

Notre but n'étant pas d'extraire simplement des entités nommées ou des relations définies dans des tâches d'EI mais plutôt d'extraire des concepts plus généraux qui peuvent provenir d'une ontolgie ou d'un modèle sémantique quelconque, nous pouvons nous inscrire dans le cadre des systèmes OBIE même si nous nous occupons essentiellement dans ce travail du module de prétraitement et du module d'EI. Nous pensons comme Wimalasuriya et Dou (2010) que le module d'EI doit être indépendant de l'ontologie ou du modèle sémantique utilisé pour garder le coté formel de ces modèles et ne pas percuter les erreurs qui peuvent être faites par exemple au niveau des règles d'EI sur eux. La base de règles que nous construisons dans ce travail peut donc constituer une extension du modèle sémantique utilisé qui en soit indépendante (Ma, Audibert, & Nazarenko, 2009).

2.9 Extraction d'information ouverte (OIE)

Dans les deux dernières décennies, nous avons noté une évolution des systèmes d'EI, de systèmes monolingues, dépendants du domaine, basés sur la connaissance vers des systèmes multilingues, entraînables qui utilisent des techniques d'apprentissage peu supervisées (Piskorski & Yangarber, 2013). Le paradigme d'extraction d'information ouverte (*Open Information Extraction*, OIE) (Banko et al., 2007 ; Etzioni, Banko, Soderland, & Weld, 2008) a été introduit pour faciliter la découverte de relations dans les textes indépendamment du domaine et le passage à l'échelle des gros corpus hétérogènes tels que le web. Contrairement aux systèmes d'EI traditionnels où les relations d'intérêt doivent être spécifiées à l'avance, les systèmes d'OIE prennent en entrée uniquement un corpus textuel sans aucune connaissance ou spécification des relations d'intérêt et produit un ensemble de relations extraites.

L'un des exemples les plus représentatifs des systèmes d'OIE est le système TextRunner (Yates et al., 2007). TextRunner apprend tout d'abord un modèle général de la manière dont sont exprimées les relations dans un langage particulier en utilisant un CRF. Il parcourt ensuite chaque phrase du corpus et utilise le modèle construit pour attribuer à chaque mot des étiquettes qui dénotent le début/fin d'une entité ou d'une chaine de caractères représentant une relation. Le modèle utilise uniquement des attributs linguistiques de bas niveau comme les étiquettes morpho-syntaxiques, capitalisation, etc., ce qui est attrayant dans le sens où il permet de traiter la diversité de genre et différents langages. Pour chaque phrase, le système retourne un ou plusieurs triplets qui représentent chacun une relation binaire entre deux entités (par exemple, (Paris, CapitaleDe, France)) avec une probabilité de l'exactitude du triplet (relation) qui s'appuie sur des informations liées à la fréquence du triplet sur

le web. Une évaluation du système TextRunner (Banko & Etzioni, 2008) révèle qu'il atteint en moyenne une précision de 75%.

Banko et Etzioni (2008) comparent TextRunner à un algorithme d'extraction de relations traditionnel basé sur le même modèle CRF mais qui a été entraîné sur des données étiquetées manuellement et qui utilise des attributs lexicaux plus riches. Des expérimentations sur l'extraction de relations comme « les aquisitions d'entreprises » ou « les inventeurs de produits » ont montré que les deux systèmes obtiennent des valeurs de précision comparables (environ 75%), mais le système d'EI traditionnel obtient un rappel largement meilleur (60% contre 20% pour le système d'OIE). Piskorski et Yangarber (2013) pensent que dans le cas où le rappel est privéligié dans le contexte d'extraction de relations binaires, l'utilisation d'un système d'extraction de relations traditionnel est de loin plus efficace même si les systèmes d'OIE peuvent réduire la quantité de données d'apprentissage étiquetées manuellement. Si, en revanche, la précision est privilégiée et le nombre de relations à extraire est grand, un système d'OIE peut potentiellement être une bonne alternative. Les auteurs citent également 3 erreurs que font souvent les premiers systèmes d'OIE comme TextRunner :

— des extractions incohérentes (la relation extraite n'a aucun sens) ;
— des extractions non informatives (des informations cruciales omises) ;
— des arguments incorrects.

Des travaux plus récents ont introduit plus d'heuristiques pour améliorer la qualité des relations extraites (Etzioni, Fader, Christensen, Soderland, & Mausam, 2011 ; Fader, Soderland, & Etzioni, 2011). Fader et al. (2011) proposent, par exemple, les deux heuristiques suivantes :

— Une relation multi-mots doit commencer par un verbe, finir par une préposition et être une séquence de mots contigus dans une phrase.
— Une relation binaire doit apparaître avec au moins un nombre minimum de paires d'arguments dans un grand corpus.

Ces deux heuristiques ont permis d'obtenir de meilleurs résultats.

L'utilisation des systèmes d'OIE a beaucoup augmenté ces dernières années pour des besoins de passage à l'échelle et d'indépendance vis à vis du domaine. Cependant, dans des domaines comme le domaine médical, le besoin d'applications d'EI d'une précision extrême se fait encore ressentir. Nous nous inscrivons dans le cadre de ces applications qui ne cherchent pas à extraire toute relation possible dans les corpus mais plutôt des concepts et des relations bien spécifiques à un domaine donné.

Conclusion

Nous avons établi dans ce chapitre un état de l'art autour de l'extraction d'information (EI) en nous focalisant sur les approches d'EI. Notre but étant de mettre en place une approche interactive d'apprentissage de règles d'extraction d'information, nous avons mis l'accent sur les approches d'EI à base de règles. Mêmes si les approches statistiques sont plus récentes dans le domaine d'EI, les approches à base de règles continuent à être utilisées car elles sont faciles à interpréter et à développer. Elles sont également appréciées dans les systèmes qui demandent une intervention de l'utilisateur car il est plus facile pour l'utilisateur de comprendre des

règles symboliques que les sorties d'un algorithme statistique. Tout comme les systèmes d'EI basés sur les ontologies (OBIE), nous cherchons à extraire des concepts et des relations définies dans des modèles sémantiques comme les ontologies tout en assurant l'indépendance entre le module d'EI proposé et le modèle sémantique utilisé pour garder la cohérence et le coté formel de ce dernier. Contrairement aux systèmes d'extraction d'information ouverte qui visent à extraire tout type de relations qui peuvent être extraites à partir de gros corpus comme le web, nous cherchons à extraire des informations spécifiques à un domaine donné comme le font les systèmes d'EI traditionnels. Nous faisons la lumière, dans le chapitre qui suit, sur l'EI interactive et l'intérêt de ce type d'extraction d'information.

Chapitre 3

Extraction d'information interactive

Les systèmes d'extraction d'information (EI) ont largement été déployés pour extraire des informations structurées à partir de données non structurées. Plusieurs de ces systèmes ont atteint de très bonnes performances. Cependant, ces systèmes sont généralement coûteux à mettre en place pour des utilisateurs qui ne disposent pas d'une quantité suffisante de données d'apprentissage annotées ou qui ne sont pas des experts en ingénierie de connaissances. Des systèmes d'EI interactifs ont vu le jour d'une part pour réduire ce coût et d'autre part pour permettre à l'utilisateur d'investiguer les erreurs faites par le système d'EI et de les corriger. Nous définissons dans ce chapitre l'EI interactive et nous expliquons la différence entre différents types de systèmes d'EI interactifs en détaillant le fonctionnement de certains systèmes à titre d'exemples.

3.1 Définition

Même s'il n'existe pas de définition formelle de l'extraction d'information interactive, nous pouvons dire qu'il s'agit d'un paradigme qui a pour but d'aider l'utilisateur à mener à bien une tâche d'EI tout en l'impliquant dans le processus d'extraction. L'interaction entre l'utilisateur et le système d'EI peut se faire de différentes manières. Dans certains travaux, elle se limite à un entraînement interactif d'un système d'EI (Cardie & Pierce, 1998 ; Caruana, Hodor, & Rosenberg, 2000). Dans d'autres systèmes, elle se fait à travers des interfaces qui facilitent la visualisation des résultats et la détection et l'investigation d'éventuelles erreurs (Eichler et al., 2008 ; Sarma, Jain, & Srivastava, 2010). Dans les systèmes d'EI interactifs, il est important de connaître les scores de confiance attribués aux champs extraits pour pouvoir corriger d'éventuelles erreurs. Certains travaux se sont donc intéressés aux méthodes d'estimation de confiance utilisées dans les méthodes d' apprentissage actif (Culotta, Kristjansson, McCallum, & Viola, 2006 ; Kristjansson et al., 2004 ; Probst & Ghani, 2007). Les systèmes d'aide au développement d'extracteurs d'information à base de règles (Akbik et al., 2013 ; Li et al., 2012) représentent une nouvelle génération de systèmes d'EI interactifs. Ils définissent des *workflows* plus complets qui guident l'utilisateur étape par étape depuis l'annotation des exemples

jusqu'à la livraison et le déploiement du système. Nous détaillons ces 4 types de systèmes interactifs dans les sections qui suivent.

3.2 Systèmes d'extraction d'information entraînés de manière interactive

Pour construire des bases de règles d'EI pour un domaine donné, les anciens systèmes d'EI requièrent un effort manuel énorme pour l'écriture des règles et sont par conséquent coûteux à mettre en place en termes de temps. L'apparition des méthodes d'apprentissage de règles d'EI a permis de réduire l'effort humain requis dans l'écriture des règles mais pas de réduire le coût de la mise en place des systèmes d'EI car ces méthodes requièrent des corpus annotés pour chaque tâche d'EI. Certains travaux proposent de réduire ce coût d'annotation de corpus en faisant coopérer l'utilisateur et le système d'EI. Cette coopération se traduit par une annotation interactive et progressive des exemples d'apprentissage.

Le système I^{2}E^2 (Interactive Information Extraction Environment) (Cardie & Pierce, 1998) est l'un des systèmes qui permet une annotation interactive des exemples d'apprentissage. La figure 3.1 schématise l'approche générale de ce système.

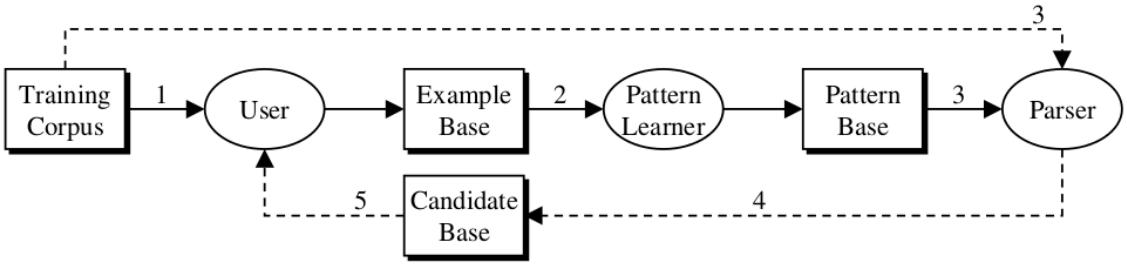


FIGURE 3.1 – Modèle du système I^{2}E^2 (Cardie & Pierce, 1998).

L'utilisateur initie le processus d'entraînement. Il entre ensuite dans une boucle d'apprentissage interactif avec le système (analyseur (*parser*) et module d'apprentissage de règles (*pattern learner*)). Les flèches numérotées sur la figure 3.1 correspondent aux étapes suivantes :

1. L'utilisateur annote des instances d'un certain concept et les ajoute à la base des exemples d'apprentissage.
2. Quand l'utilisateur le décide, le module d'apprentissage commence à induire des patrons à partir des exemples fournis dans la base des exemples.
3. L'analyseur collecte les instances candidates couvertes par les patrons appris dans le corpus d'apprentissage.
4. Les instances candidates sont présentées à l'utilisateur pour révision. Ce dernier rejette les instances incorrectes et accepte les instances correctes. Il fournit ainsi au système des exemples positifs et des exemples négatifs à un coût relativement bas.

5. Le module d'apprentissage de règles se sert des instances positives et négatives pour adapter sa base de patrons. Il garde, en définitive, l'ensemble des règles qui maximise le nombre d'instances positives couvertes et minimise le nombre d'instances négatives couvertes.

L'utilisateur peut ne pas réviser toutes les instances candidates. Sa stratégie peut varier d'une simple utilisation du système comme un outil d'annotation (répétition continue de l'étape 1) jusqu'à la révision d'une seule instance à la fois en guidant le système à travers la modification d'un seul patron et en utilisant le module d'apprentissage de règles pour trouver autant d'instances positives additionnelles que possible (boucle sur les étapes 2, 3, 4 et 5). La stratégie la plus efficace se situe entre ces deux stratégies extrêmes.

Le processus prend fin quand l'utilisateur est satisfait des performances du système ou quand la précision des patrons est maximisée sur un corpus de test annoté.

Le système I^2E^2 s'appuie sur 3 concepts clés : la capacité de l'analyseur à trouver rapidement des instances d'un patron candidat, la capacité de l'utilisateur à juger leur pertinence et la capacité du module d'apprentissage de règles à adapter les patrons candidats pour se conformer aux besoins de l'utilisateur.

3.3 Systèmes d'extraction d'information basés sur l'apprentissage actif

L'apprentissage actif est une approche qui inclut l'utilisateur dans la boucle d'apprentissage. Il est utilisé pour soutenir les méthodes d'apprentissage supervisées en extraction d'information et mettre en place par conséquent des approches hybrides de construction de systèmes d'EI. L'utilisateur reste impliqué dans le processus d'ingénierie de connaissances mais de manière moins directe en étant guidé par le module d'apprentissage. Nous expliquons plus en détails dans cette section ce qu'est l'apprentissage actif et nous donnons quelques exemples de systèmes d'EI interactifs qui utilisent soit l'apprentissage actif soit les méthodes d'estimation de confiance tirées de l'état de l'art des méthodes d'apprentissage actif.

3.3.1 Apprentissage actif

L'apprentissage actif est un domaine de l'apprentissage artificiel qui a pour but de réduire la quantité d'exemples d'apprentissage à annoter par l'utilisateur en participant « activement » dans le processus d'apprentissage. L'idée est que l'annotateur fournit au départ un petit ensemble d'exemples sur la base desquels le module d'apprentissage décide quels exemples, parmi un grand ensemble d'exemples candidats, annoter ensuite pour maximiser le gain. Intuitivement, ceci signifie la sélection des exemples à propos desquels le module d'apprentissage a le moins de certitude. Ces exemples sont en effet les exemples dont le module d'apprentissage peut bénéficier le plus étant donné que l'utilisateur n'est prêt à consacrer qu'une durée de temps limitée à l'annotation d'exemples. Ceci peut également être vu d'une autre manière : le module d'apprentissage tire moins de bénéfices si l'annotateur fournit des exemples qui se ressemblent et des informations redondantes.

Étant donné que le rôle de l'apprentissage actif consiste à sélectionner les exemples à annoter par l'utilisateur qui sont susceptibles d'être informatifs et d'améliorer la précision du modèle, la question qui se pose est : comment décider quels sont les exemples les plus informatifs ? On distingue deux types d'approches standard de *selective sampling* ou d'échantillonnage sélectif (Cohn, Atlas, & Ladner, 1994).

1. Approches fondées sur la certitude (*certainty-based approaches*) (Lewis & Catlett, 1994) : le système est entraîné sur un petit ensemble d'exemples annotés pour apprendre un classifieur initial. Le système examine ensuite les exemples non annotés et attribue des scores de confiance aux annotations prédites pour ces exemples. Les k exemples qui ont les scores de confiance les plus bas sont ensuite présentés à l'utilisateur pour annotation et réapprentissage. Plusieurs méthodes d'attribution de scores de confiance existent. La plupart d'entre elles tentent d'estimer la probabilité qu'un classifieur disposant de données d'apprentissage antérieures arrive à classer correctement un nouvel exemple.
2. Approches fondées sur un comité (*committee-based approaches*) (Dagan & Engelson, 1995) : un comité de classifieurs est créé à partir d'un petit ensemble d'exemples annotés. Chaque membre de ce comité essaie ensuite d'étiqueter des exemples supplémentaires. Les exemples dont l'annotation entraîne le plus de désaccord entre les membres du comité sont présentés à l'utilisateur pour annotation et réapprentissage. Un comité diversifié qui dispose de données d'apprentissage antérieures produit le plus grand désaccord sur les exemples dont l'étiquette est la plus incertaine au regard des classifieurs possibles qui peuvent être obtenus en apprenant sur ces données.

Même si les deux méthodes d'échantillonnage sélectif citées sont les plus utilisées dans la littérature, des chercheurs à l'instar de S. Soderland et al. (1999) ont utilisé des méthodes moins courantes mais qui peuvent être tout aussi performantes.

Plusieurs systèmes d'EI qui utilisent l'apprentissage actif ont montré que cette approche permet de réduire le nombre d'exemples d'apprentissage à annoter par l'utilisateur (Muslea, Minton, & Knoblock, 2000 ; Probst & Ghani, 2007 ; Thompson et al., 1999). (Settles, 2009) et (Olsson, 2009) donnent un état d'art plus détaillé sur l'apprentissage actif.

3.3.2 Exemples de systèmes interactifs utilisant l'apprentissage actif

Nous présentons dans cette section quelques systèmes d'EI qui utilisent l'apprentissage actif et qui mettent en avant le caractère interactif de ce dernier.

(Kristjansson et al., 2004) proposent un système d'EI à base de Champs Aléatoires Conditionnels (CRF). Ce système a deux particularités : il introduit une plateforme d'EI interactive et deux nouveaux algorithmes d'estimation de la confiance attribuée aux champs dans le cadre de l'apprentissage actif avec les CRF. La plateforme interactive inclut une interface utilisateur qui surligne le label attribué à chaque champ du document non structuré et signale les labels qui ont un score de confiance bas. Cette interface permet également la correction rapide en utilisant un

mécanisme de glisser-déposer et la propagation des corrections de champs de manière à ce qu'une seule correction puisse corriger plusieurs erreurs. Les algorithmes d'estimation de confiance introduits permettent d'incorporer les contraintes dans le processus de décodage Viterbi. Ces contraintes proviennent soit des corrections des champs incorrects, soit des nouveaux labels ajoutés par l'utilisateur aux champs. La figure 3.2 présente l'interface utilisateur qui facilite l'EI interactive. Les champs à peupler figurent dans la partie gauche et le texte source est introduit par l'utilisateur dans la partie droite.

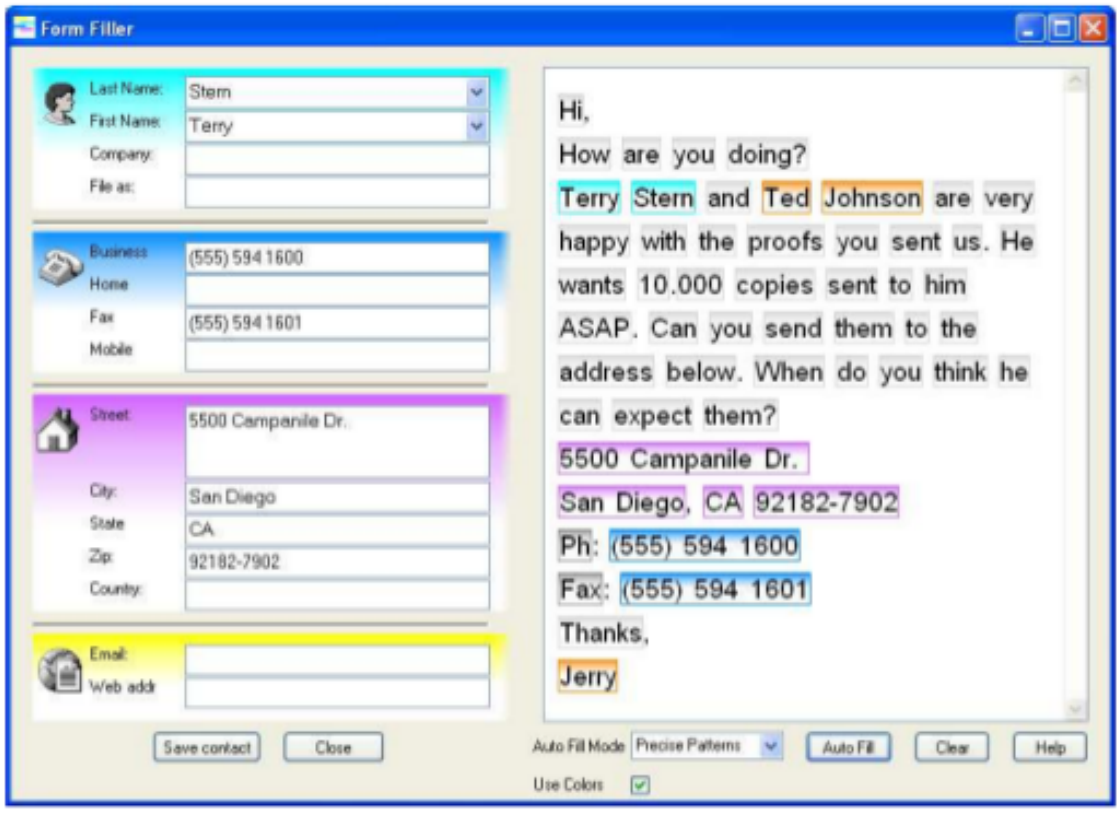


Figure 3.2 – Interface utilisateur pour la saisie des coordonnées d'un contact (Kristjansson et al., 2004).

Le système d'EI extrait des segments de texte à partir du texte non structuré et remplit les champs correspondants. Les aspects les plus importants de l'interface utilisateur sont les suivants.

— Elle affiche une aide visuelle qui permet à l'utilisateur de vérifier rapidement la justesse des champs extraits. Chaque type d'information est surligné par une couleur distincte (par exemple, les informations téléphoniques sont surlignées en bleu et les adresses électroniques en jaune).
— Elle permet une correction rapide. Les segments de texte peuvent, par exemple, être facilement regroupés et traités par blocs.
— Elle attire l'attention de l'utilisateur sur les champs qui ont un faible score de confiance. De plus, dans la partie droite, des alternatives peuvent être

surlignées dans le texte.

Pour estimer la confiance d'un champ extrait par le CRF, les auteurs utilisent une technique nommée *Constrained Forward Backward* (Culotta & McCallum, 2004). Cet algorithme calcule la probabilité pour que chaque séquence vérifie un ensemble de contraintes. Chaque contrainte peut être soit positive soit négative. Dans le cadre d'un remplissage interactif d'un formulaire, les contraintes correspondent à un champ extrait automatiquement. Les contraintes positives spécifient les mots étiquetés à l'intérieur du champ et les contraintes négatives spécifient les frontières du champ.

Pour évaluer la performance de leur système d'EI interactif, les auteurs proposent une mesure appelée ENUA (*Expected Number of User Actions*) qui représente le nombre des actions utilisateur attendues. Ils montrent que cette mesure peut être diminuée d'environ 14% par rapport à un remplissage manuel de tous les champs grâce à la technique de propagation de corrections. Ils montrent également que les prédictions de scores de confiance faibles peuvent augmenter l'efficacité d'un annotateur humain et réduire l'erreur d'environ 56%.

La plupart des travaux sur l'apprentissage actif se sont souvent concentrés sur l'amélioration des algorithmes et des mesures pour la sélection de l'exemple suivant à étiqueter par l'utilisateur. Ils ont, cependant, souvent négligé le temps mis entre les itérations de l'apprentissage actif, ce qui provoque des temps d'attente longs entre les interactions pour l'utilisateur. (Probst & Ghani, 2007) s'intéressent à l'aspect interactif de l'apprentissage actif. Ils proposent un outil interactif qui a pour but de rendre le processus d'extraction plus efficace et moins coûteux. Cet outil se compose de deux parties : la première partie consiste en un classifieur semi supervisé (non interactif) (CoEM avec Naive Bayes) qui permet d'extraire un ensemble de paires attribut-valeur à partir de données de descriptions de produits sportifs (Probst & Ghani, 2007). La deuxième partie consiste en un outil dont se sert l'utilisateur pour fournir un *feedback* et améliorer la précision de l'extraction. Nous nous intéressons dans cette section à la deuxième partie du système.

Quand CoEM est utilisé comme le classifieur semi-supervisé (R. Jones, Ghani, Mitchell, & Riloff, 2003 ; Muslea, Minton, & Knoblock, 2002) dans la phase d'apprentissage actif, la précision d'extraction obtenue est bonne mais le temps d'attente pour l'utilisateur est extrêmement long. Pour rendre la phase d'apprentissage actif plus rapide sans dégrader la précision, les auteurs proposent une approximation rapide de CoEM avec Naive Bayes qui peut prendre en compte le retour de l'utilisateur presque instantanément et qui peut fonctionner avec n'importe quelle stratégie d'apprentissage actif. Les résultats expérimentaux sur des données de produits sportifs montrent que l'algorithme proposé atteint des performances (précision, rappel et F-mesure) comparables à celles obtenues par l'algorithme original (CoEM avec Naive Bayes) mais tout en étant plus rapide. Le système d'EI obtenu est pratique et minimise le temps d'attente de l'utilisateur. Les auteurs pensent que la combinaison de l'apprentissage actif et de l'apprentissage semi-supervisé peut avoir d' énormes implications pratiques mais le problème du temps d'attente long peut être un obstacle dans plusieurs applications réelles. Ils considèrent leur travail comme une étape vers la construction de systèmes d'EI pratiques.

3.4 Systèmes d'extraction d'information dotés d'interfaces de visualisation

Plusieurs travaux sur les techniques de visualisation d'information interactives comme dans (Heer, Card, & Landay, 2005) ont motivé l'apparition des systèmes d'EI dotés d'interfaces de visualisation et d'investigation des informations extraites.

Le système d'EI dynamique interactif IDEX (Eichler et al., 2008) intègre par exemple une interface nommée IdexVisor qui permet une exploration interactive de l'espace d'extraction. IdexVisor permet à l'utilisateur d'accéder à différentes visualisations des données extraites et de naviguer à travers l'espace des données de manière flexible et dynamique. La figure 3.3 montre les composants principaux du système IDEX. Ce dernier est composé de deux parties principales : IdexExtractor responsable de l'EI et IdexVisor qui constitue l'interface graphique de l'utilisateur.

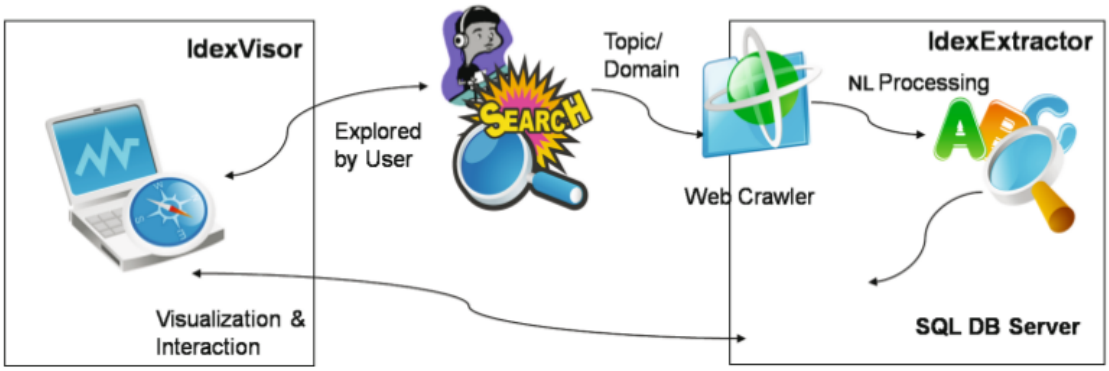


Figure 3.3 – Architecture du système IDEX (Eichler et al., 2008).

Le processus d'EI dans IDEX commence par l'envoi d'informations pertinentes à propos d'un sujet ou d'un domaine sous forme de requêtes utilisateur sur le web. Les documents retrouvés sont traités par les composants principaux d'EI qui réalisent une extraction d'entités nommées et une identification et un clustering des relations d'intérêt. L'information extraite ainsi que son contexte textuel et linguistique sont stockés dans différentes tables SQL qui sont maintenues par un serveur SQLDB. Les tables constituent une entrée pour le système IdexVisor qui crée dynamiquement différentes représentations visuelles des données dans les tables pour permettre une recherche flexible et une exploration des entités par l'utilisateur.

Nous nous intéressons dans cette section au système IdexVisor. En utilisant l'interface IdexVisor, l'utilisateur peut accéder à différentes visualisations des données extraites et naviguer à travers l'espace des données de manière flexible et dynamique. IdexVisor est une application indépendante. Elle peut être configurée dynamiquement au regard de la structure du modèle de la base de données en utilisant une configuration déclarative basée sur XML. L'interface est placée entre l'utilisateur et le serveur de base de données MySQL. IdexVisor suit l'approche MVC (*Model View Controller*) illustrée dans la figure 3.4.

Les résultats d'IdexExtractor sont disponibles sous forme de tables. Chaque table représente un aspect des données extraites. La configuration de l'interface spécifie

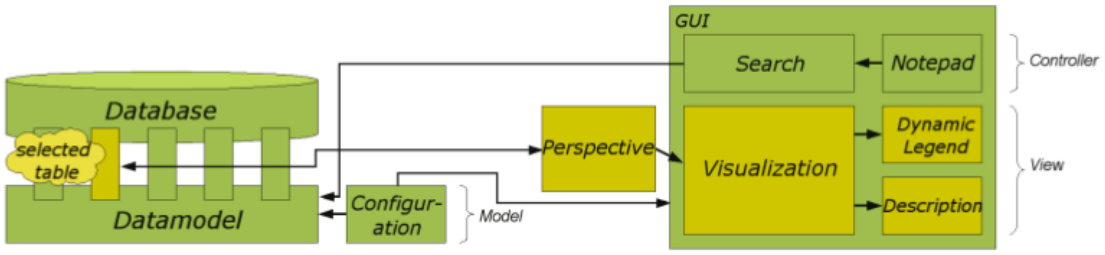


FIGURE 3.4 – Architecture d'IdexVisor (Eichler et al., 2008).

des méta-informations à propos de la partie des données qui doit être utilisée. Grâce à la vue (*View*), les données sélectionnées ainsi que les informations additionnelles peuvent être visualisées. En utilisant le contrôleur (*Controller*), l'utilisateur peut naviguer à travers les données sélectionnées. Il peut chercher des informations spécifiques ou formuler des requêtes au moteur de recherche. La visualisation offre un ensemble de perspectives qui offrent différents points de vue sur des sous ensembles de données accessibles à travers des onglets séparés dans l'interface (*GUI*). IdexVisor a été évalué par 7 utilisateurs. Ils reportent que la commutation entre les perspectives est lourde et que les bénéfices ne sont pas évidents.

I4E (Interactive Investigation of Iterative Information Extraction) (Sarma et al., 2010) est une autre approche développée pour une investigation interactive post-extraction pour les systèmes d'EI interactifs. Cette investigation repose sur 3 phases importantes : l'explication des résultats d'extraction, le diagnostic des composants erronnés potentiels et la réparation des résultats extraits (en réparant leurs composants). L'efficacité de I4E a été démontrée à travers une évaluation expérimentale détaillée sur 6 ensembles de données réelles obtenues à partir d'un corpus web de 500 millions de documents. Les algorithmes de I4E ont permis d'identifier et de réparer un système d'extraction avec un retour utilisateur minimal.

3.5 Systèmes d'aide au développement d'extracteurs d'information à base de règles

Dans le milieu industriel, il y a un besoin croissant de développer des systèmes d'EI à base de règles (Chiticariu et al., 2013). En effet, les systèmes à base d'apprentissage statistique pourraient permettre d'obtenir de meilleures performances mais le besoin de comprendre et d'interpréter les résultats obtenus pousse aujourd'hui vers un retour aux systèmes à base de règles. Pour réduire le coût de mise en place de tels systèmes, on doit permettre à des développeurs avec des connaissances générales en informatique de manipuler de tels systèmes. Des travaux sur l'amélioration des processus de création de règles dans les systèmes d'EI se sont concentrés sur l'assistance de l'utilisateur dans l'utilisation des techniques d'apprentissage artificiel comme la pré-génération d'expressions régulières (Brauer, Rieger, Mocan, & Barczynski, 2011) ou la suggestion de patrons (Li, Chu, Blohm, Zhu, & Ho, 2011). Pour améliorer la convivialité, des systèmes qui proposent des environnements pour gui-

der l'utilisateur dans le développement d'extracteurs d'information ont vu le jour. Nous pouvons citer parmi ces systèmes WizIE (Li et al., 2012) et Propminer (Akbik et al., 2013).

WizIE (Li et al., 2012) est un environnement de développement d'applications d'EI qui a pour but de permettre aux développeurs disposant de peu ou ne disposant pas de connaissances linguistiques d'écrire des règles d'EI de bonne qualité. WizIE permet de guider les développeurs d'extracteurs d'information, étape par étape, à travers un processus de développement qui s'appuie sur des bonnes pratiques synthétisées à partir des expériences de développeurs experts. WizIE permet également de réduire l'effort manuel requis dans le développement des tâches d'EI clés en offrant une investigation automatique des résultats et une fonctionnalité de découverte de règles. Le processus de développement dans WizIE se compose de 5 phases comme le montre la figure 3.5.

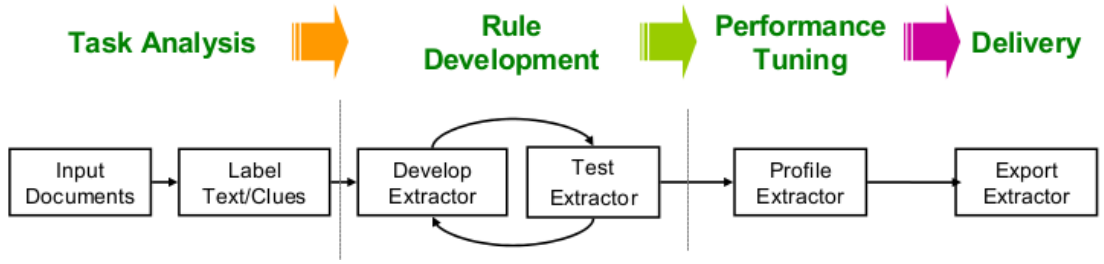


Figure 3.5 – Processus de développement d'un extracteur d'information dans WizIE (Li et al., 2012).

Analyse de la tâche : WizIE demande explicitement à l'utilisateur de sélectionner et d'examiner manuellement un petit nombre de documents exemples, d'identifier et d'étiqueter les fragments d'intérêt dans ces documents et de détecter les indices qui permettent d'identifier de tels fragments. La définition et le contexte des tâches d'EI sont capturés par une structure d'arbre appelée « plan d'extraction ». Les nœuds feuilles dans un plan d'extraction correspondent à des tâches d'EI atomiques alors que les autres nœuds correspondent à des tâches de plus haut niveau qui peuvent s'appuyer sur une ou plusieurs tâches atomiques.

Développement des règles : une fois les tâches d'EI définies, WizIE guide le développeur dans l'écriture des règles en s'appuyant sur des bonnes pratiques. La figure 3.6 montre une capture d'écran de la phase de développement de règles. Le panneau de la tâche d'EI à gauche fournit des informations et des conseils pour le développement de règles, alors que le panneau du plan d'extraction à droite guide le développement des règles pour chaque tâche d'EI.

Comme le montre la figure 3.6, les types de règles associées à chaque nœud peuvent être de 3 catégories : les attributs de base, la génération de candidats et le filtrage et la consolidation. Cette catégorisation est basée sur les meilleures pratiques de développement de règles (Chiticariu, Li, Raghavan, & Reiss, 2010). Le

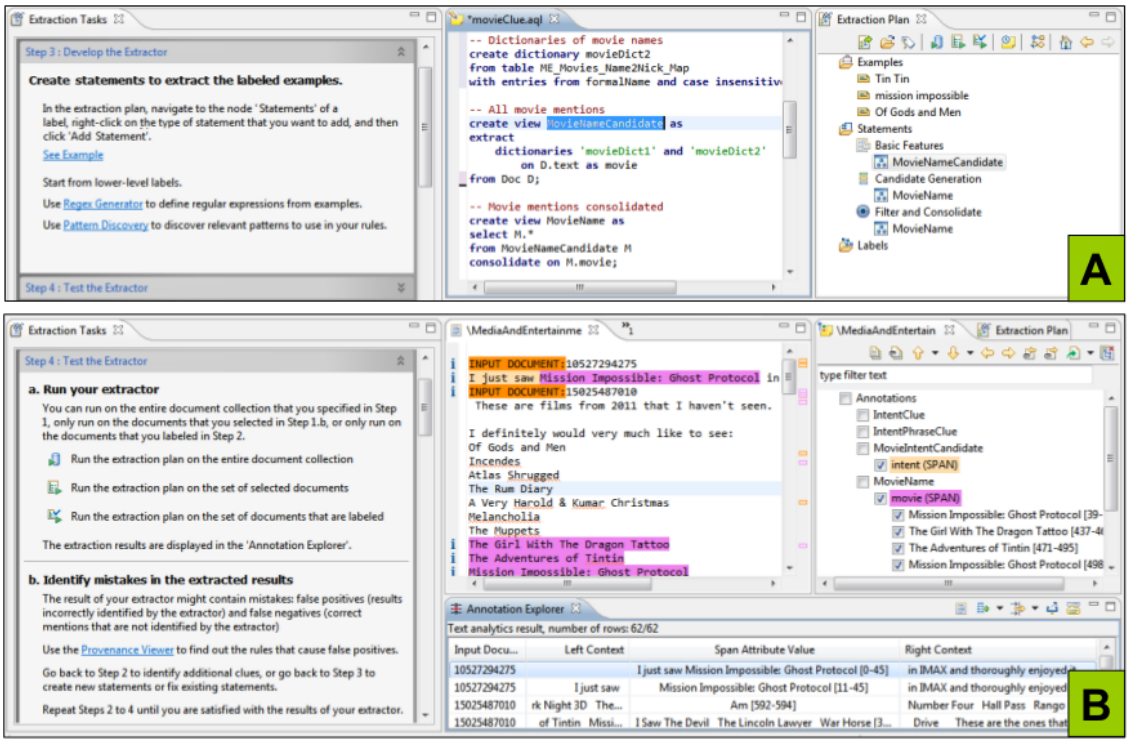


FIGURE 3.6 – Phase de développement des règles dans WizIE (Li et al., 2012) : (A) développement et (B) test.

développeur peut créer des règles soit directement dans l'éditeur de règles soit à travers l'assistant « Create statement » accessible à partir du nœud « Statements » de chaque label dans le plan d'extraction. Une fois que le développeur a complété une itération de développement de règles, WizIE le guide dans le test et le raffinage de l'extracteur comme le montre la figure 3.6 (B).

WizIE fournit également une suite d'outils sophistiqués pour l'investigation automatique des résultats et la découverte de règles, ce qui le différencie des environnements de développement d'applications d'EI conventionnels. En effet, quand l'utilisateur clique sur un résultat extrait, le « provenance viewer » fournit une explication complète de comment ce résultat a été produit par l'extracteur, sous forme d'un graphe qui montre la séquence des règles ainsi que les fragments de texte responsables de ce résultat. Ces explications permettent au développeur, par exemple, de comprendre pourquoi un faux positif est généré par le système et d'identifier les règles problématiques qui peuvent être affinées pour corriger l'erreur. Les indices contextuels négatifs peuvent être utiles pour la création de règles qui filtrent les faux positifs. Les indices positifs, en revanche, sont utiles pour la création de règles qui séparent les correspondances ambigües et les correspondances précises. Le composant de découverte de patrons de WizIE facilite la découverte automatique des indices en cherchant dans les données disponibles des patrons communs dans des contextes spécifiques (Li, Chu, et al., 2011).

WizIE permet également la découverte de patrons d'expressions régulières. Le générateur d'expressions régulières prend comme entrée des exemples de mentions

et suggère des expressions régulières qui capturent ces exemples en variant des plus spécifiques (grande précision) aux plus générales (grande couverture).

Réglage de la performance : une fois le développeur satisfait de la qualité de l'extracteur, WizIE le guide dans la mesure et le réglage de la performance d'exécution de l'extracteur pour préparer son déploiement dans un environnement de production. Le composant de profilage (profiler) observe l'exécution de l'extracteur sur une collection de données sur une période de temps et enregistre le pourcentage de temps mis pour exécuter chaque règle. Après le profilage, WizIE affiche les 25 règles et opérations qui consomment le plus de temps ainsi que le débit (quantité de données traitée par unité de temps). En s'appuyant sur ces informations, le développeur peut régler à la main les parties critiques de l'extracteur, relancer le composant de profilage et valider une augmentation du débit. Le développeur répète le processus jusqu'à ce qu'il soit content des performances de l'extracteur.

Livraison et déploiement : une fois satisfait de la qualité des résultats et de la performance d'exécution, le développeur est guidé par WizIE à travers un processus d'exportation de l'extracteur sous forme d'exécutable compilé. L'exécutable généré peut être imbriqué dans une application en utilisant une interface API Java.

WizIE a été évalué par 14 participants. Les résultats montrent que WizIE constitue un pas vers le développement d'extracteurs par des développeurs novices en extraction d'information.

Propminer (Akbik et al., 2013) est un outil et un *workflow* conçu pour permettre à des utilisateurs initiés d'explorer interactivement l'effet et l'expressivité de la création de règles d'EI à partir d'arbres d'analyse syntaxique. Propminer implémente un *workflow* constitué de 5 étapes.

Annoter : les utilisateurs commencent par construire une phrase type pour un type d'information désiré. Cette phrase constitue un exemple qui exprime la relation cible. Dans cette phrase, l'utilisateur annote les mots qui appartiennent à la relation en leur associant les rôles sujet, prédicat et objet. La figure 3.7 montre la vue « sentence view » de Propminer avec une phrase exemple saisie et annotée dans la partie supérieure et la phrase analysée syntaxiquement dans le panneau du centre.

Générer : Propminer génère une règle à partir d'une phrase annotée en déterminant l'arbre minimal qui connecte tous les mots étiquetés comme sujet, prédicat et objet dans l'arbre de dépendances. La règle est composée de cet arbre minimal plus les contraintes exprimées dans les étiquettes morpho-syntaxiques et les valeurs lexicales des mots impliqués.

Les règles sont formulées comme des requêtes sur une base de données dans laquelle sont stockées les phrases analysées syntaxiquement sous formes de graphes : les nœuds représentent les mots et les arcs représentent les dépendances. L'étiquette morpho-syntaxique et la valeur lexicale du mot sont stockées dans chaque nœud comme attributs.

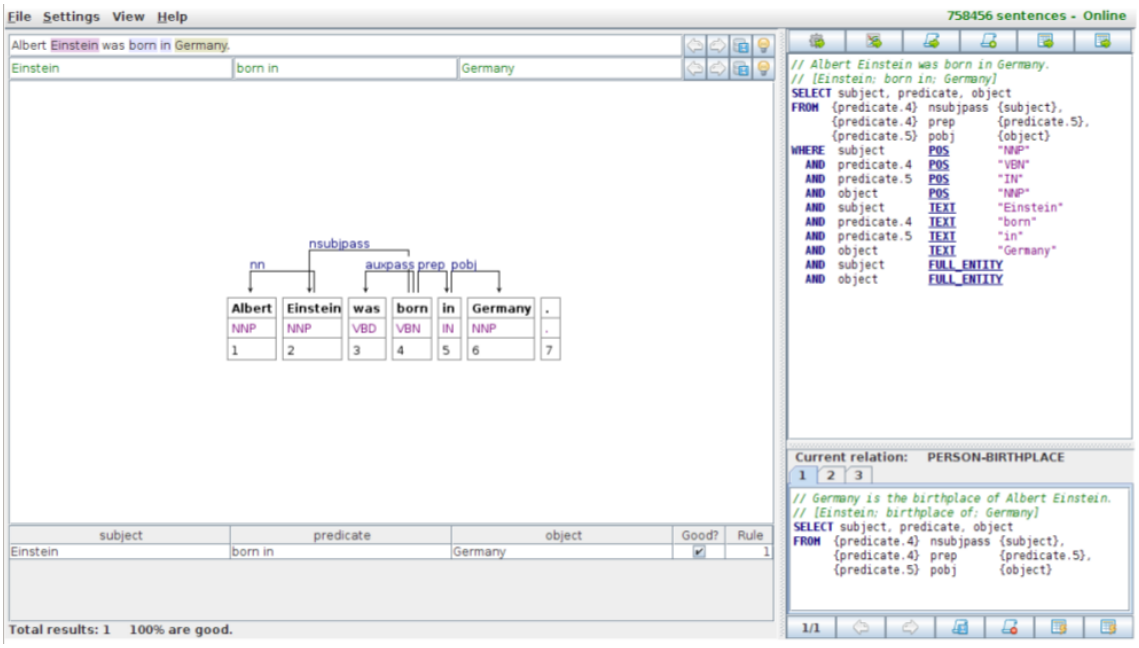


FIGURE 3.7 – La vue « sentence view » de Propminer (Akbik et al., 2013) où les deux premières étapes du *workflow* sont exécutées.

Une règle Propminer (requête) se compose essentiellement de 3 parties : une clause SELECT qui détermine les champs à retourner par la requête, une clause FROM qui exprime le chemin du sous graphe qui doit être couvert par la règle et qui spécifie quels nœuds du sous graphe correspondent aux champs de la clause SELECT et une clause WHERE où des restrictions sur les attributs des mots du sous arbre sont définies.

Généraliser : la règle générée dans la deuxième étape est spécifique à la phrase annotée. Elle n'est capable de trouver que les instances de relations semblables à celles de la phrase type. L'utilisateur généralise dans cette étape la règle à l'aide de suggestions faites par Propminer. La généralisation des règles se fait essentiellement en supprimant ou modifiant les contraintes sur les attributs des mots dans la clause WHERE.

Évaluer : chaque règle créée par l'utilisateur est évaluée dans la vue *corpus view* de Propminer représentée dans la figure 3.8. Cette vue montre un échantillon des résultats d'extraction de la règle dans une table.

L'utilisateur peut naviguer dans la table et consulter dans chaque ligne les informations extraites ainsi que la phrase à partir de laquelle les informations ont été extraites. Après chaque modification, les résultats d'extraction de l'état courant de la règle sont affichés pour aider l'utilisateur. Si l'utilisateur juge que l'information extraite est bonne, il peut marquer le fait comme correct.

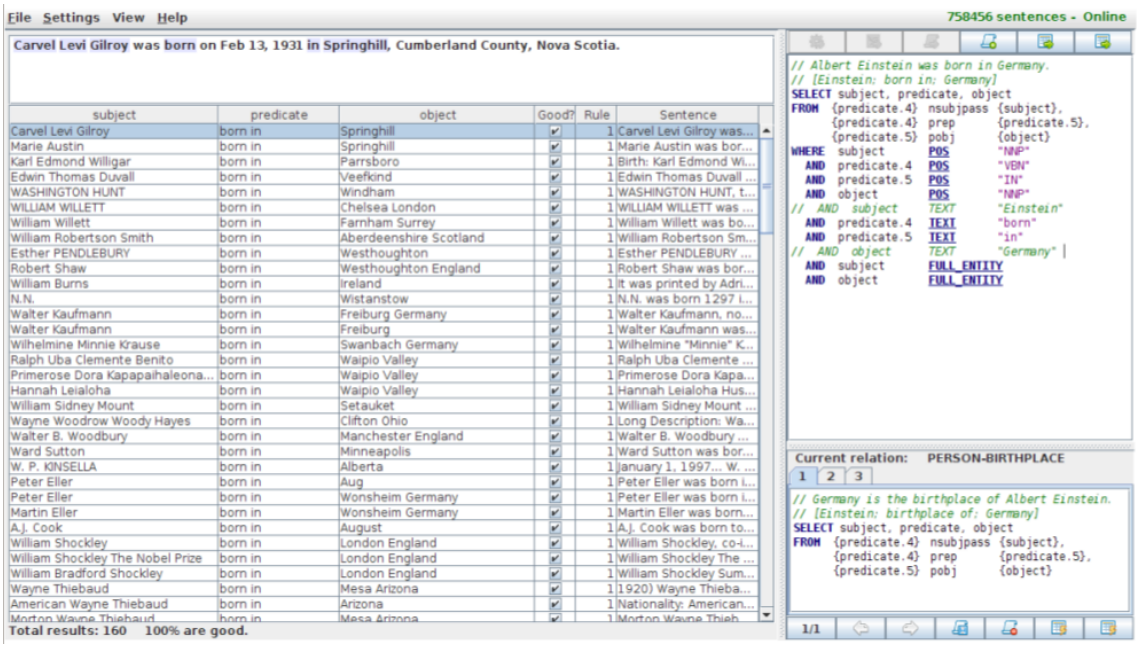


Figure 3.8 – La vue *corpus view* de Propminer (Akbik et al., 2013) où les règles d'extraction sont modifiées et évaluées.

Stocker : si l'utilisateur est satisfait de la règle d'extraction, il peut l'attribuer à une relation et la stocker dans l'ensemble des règles. Il peut répéter le processus avec une autre phrase pour trouver davantage de patrons pour la relation cible. En répétant le *workflow*, l'ensemble des règles se construit progressivement tout comme l'ensemble des résultats d'évaluation. Les résultats d'évaluation sont utilisés pour aider l'utilisateur à trouver des nouvelles phrases qui peuvent être pertinentes par rapport à la relation cible. Pour éviter les conflits avec des relations existantes, l'ensemble total des règles est appliqué à chaque phrase faisant l'objet d'un *workflow*.

Les évaluateurs du système Propminer trouvent que la nature interactive de l'outil aide à comprendre les règles d'EI.

3.6 Discussion

D'après Kristjansson et al. (2004), l'extraction d'information interactive crée de nouveaux besoins dans les systèmes d'EI. Pour faciliter l'expérience des utilisateurs, un système d'EI interactif doit afficher les champs dont l'indice de confiance est faible et prendre en considération de manière optimale les corrections de l'utilisateur.

Plusieurs approches d'EI statistiques sont compatibles avec ce paradigme. Les classifieurs à base d'Entropie Maximale permettent l'introduction d'attributs arbitraires et d'estimer la confiance dans les décisions. Les Champs Aléatoires Conditionnels qui représentent une généralisation à la fois des modèles à Entropie Maximale et des modèles de Markov Cachés capturent les dépendances entre les labels, peuvent estimer la confiance dans ces labels et disposent d'un chemin naturel pour propager de manière optimale les corrections de l'utilisateur.

Les approches d'ingénierie les plus communes en EI consistent à construire un ensemble d'expressions régulières qui permettent d'extraire les champs cibles. Kristjansson et al. (2004) pensent que les expressions régulières ne sont pas adaptées au paradigme d'extraction d'information interactive car elles ne peuvent ni estimer la confiance ni incorporer de manière naturelle les annotations et les corrections de l'utilisateur. Des travaux à l'instar de ceux menés par S. Soderland et al. (1999), Thompson et al. (1999), T. Wu et Pottenger (2005) montrent cependant qu'il est tout à fait possible d'introduire la notion de confiance dans des systèmes d'EI à base de règles sans pour autant utiliser les méthodes classiques d'estimation de confiance qui proviennent de l'état de l'art de l'apprentissage actif. De nos jours, il y a même une tendance à revenir à ces systèmes à base de règles surtout dans le milieu industriel car les règles sont beaucoup plus compréhensibles pour l'utilisateur que les sorties d'un système d' apprentissage statistique. Des systèmes d'EI interactifs à base de règles ont donc vu le jour. Le système I^2E^2 (Cardie & Pierce, 1998) s'occupe seulement d'entraîner de manière interactive l'algorithme d'apprentissage de règles. Le système WizIE (Li et al., 2012) dispose d'une interface interactive qui guide l'utilisateur étape par étape dans la construction du système d'EI mais ne dispose pas d'un module d'apprentissage automatique, ce qui le rend coûteux en termes d'effort humain. Le système Propminer (Akbik et al., 2013) définit un *workflow* plus complet qui permet à un utilisateur initié de développer un extracteur d'information de bout en bout mais ne dispose pas d'un module qui facilite le choix des exemples à annoter par l'utilisateur.

Nous cherchons, dans ce travail, à mettre en place une approche générique d'apprentissage interactif de règles d'EI qui réunit les avantages de plusieurs systèmes d'EI interactifs comme l'entraînement interactif du système, l'utilisation d'une inetrface de visualisation des résultats, l'utilisation d'un langage de règles expressif pour pouvoir obtenir des règles compréhensibles et la prise en compte des annotations et des corrections de l'utilisateur. Une description plus détaillée des propriétés clés qui doivent caractériser le système que nous cherchons à mettre en place figure dans le chapitre 5.

Conclusion

Les systèmes d'extraction d'information interactifs ont vu le jour d'une part pour réduire le coût de l'écriture manuelle de règles ou l'annotation d'exemples d'apprentissage et d'autre part pour permettre à l'utilisateur d'investiguer les erreurs faites par le système d'EI et de les corriger. Nous avons essayé de définir, dans ce chapitre, ce qu'est l'EI interactive et nous avons distingué 4 catégories de systèmes d'EI interactifs. Les systèmes entraînés de manière interactive permettent une annotation interactive et progressive des exemples d'apprentissage. Les systèmes à base d'apprentissage actif permettent d'estimer les scores de confiance dans les champs extraits et de souligner à l'utilisateur les champs dont le score de confiance est le plus bas afin de les annoter ou les corriger. Les systèmes dotés d'une interface de visualisation ou d'investigation permettent une vérification visuelle de l'exactitude des champs extraits et une correction des erreurs. Les systèmes d'aide au développement d'extracteurs d'information à base de règles définissent des *workflows* plus

ou moins complets qui guident l'utilisateur étape par étape depuis l'annotation des exemples jusqu'à la livraison et le déploiement du système. Le système que nous cherchons à mettre en place, dans ce travail, est un combinaison de ces 4 types de systèmes interactifs. Les propriétés qui doivent le caractériser sont détaillées dans le chapitre 5. Nous nous intéressons dans le chapitre 4 à l'environnement technique qui nous permettra de mettre en place notre approche.

Chapitre 4

Outillage

Dans une tâche d'extraction d'information (EI), l'environnement technique joue un rôle très important dans l'optimalité des résultats. Dans les systèmes d'EI à base de règles, par exemple, le langage dans lequel sont écrites ou inférées les règles, l'algorithme utilisé pour inférer les règles s'il s'agit d'un apprentissage de règles ainsi que les annotations sur lesquelles les règles s'appuient sont des facteurs qui peuvent avoir un impact important sur les performances de ces dernières.

L'approche que nous mettons en place dans ce travail étant basée sur des règles d'EI écrites manuellement et sur des règles également inférées de manière automatique, nous présentons, dans ce chapitre, nos choix concernant la plateforme utilisée pour prétraiter les corpus, le langage de règles que nous utilisons pour écrire les règles d'EI et l'algorithme d'apprentissage de règles dont nous nous servons pour inférer les règles.

4.1 Gestion de l'information non structurée

L'information non structurée représente une source d'information disponible, large et qui ne cesse d'évoluer. Très vite, la quantité d'informations disponibles sur le réseau est devenue très importante, mais son hétérogéïnité et son manque de structuration ont rendu l'accès à cette information très difficile.

Une application ou une plateforme de gestion d'informations non structurées (Unstructured Information Management, UIM) est généralement une application capable d'analyser de grands volumes d'informations non structurées (texte, audio, vidéo, images) afin de découvrir, d'organiser et de fournir aux clients finaux des connaissances exploitables stockées dans des formats structurés (template, formulaire, etc.) pour des traitements ultérieurs (indexation dans des applications de recherche d'information par exemple).

Pour analyser du contenu non structuré, les applications UIM utilisent différentes technologies comme le traitement du langage naturel, la recherche d'information, l'apprentissage artificiel, les ontologies, le raisonnement automatique, etc. Plusieurs outils s'appuyant sur ces technologies ont été développés de manière indépendante en utilisant différentes techniques et interfaces. Cependant, l'intégration de ces outils pour construire le pont entre le monde du non structuré et le monde du structuré est souvent très coûteuse.

Es-salihe et Bond (2006) ont étudié et comparé 3 frameworks qui regroupent plusieurs technologies de traitement d'informations non structurées : UIMA (*Unstructured Information Management Architecture*) Ferrucci et Lally (2004), GATE (*General Architecture of Text Engineering*) Cunningham (2002) et OpenNLP [1] (*Open Natural Language Processing*). Ils ont conclu que les trois projets sont forts intéressants et que chacun des trois a ses avantages et ses inconvénients. OpenNLP ne dispose pas d'une architecture générale mais peut être vu comme un ensemble d'implémentations de techniques de traitement automatique de langage naturel qui peuvent être intégrées dans d'autres applications. UIMA et GATE, d'un autre côté, peuvent être considérés comme des projets complémentaires : GATE a été développé dans un milieu de recherche académique et contient donc beaucoup d'implémentations de techniques et d'algorithmes robustes tandis que UIMA provient d'un milieu plutôt industriel et répond plus au besoin de déploiement distribué et à large échelle. UIMA présente également l'avantage de travailler sur toute source de documents (texte, audio, vidéo, image, etc.).

Bank et Schierle (2012) donnent un aperçu sur les architectures de gestion du langage naturel les plus courantes (TIPSTER (Grishman, 1996), Ellogon (Petasis, Karkaletsis, Paliouras, Androutsopoulos, & Spyropoulos, 2002), GATE et Heart of Gold (Schäfer, 2006)) ainsi que leurs avantages et leurs inconvénients et les opposent à la première architecture standardisée UIMA. Ils concluent qu'actuellement UIMA peut être considérée comme l'architecture la plus évoluée et la plus complète et encouragent la communauté scientifique à étendre la norme OASIS avec les normes linguistiques pour permettre un meilleur échange entre les chercheurs, ce qui augmenterait la réutilisabilité et la qualité du code.

Dans ce travail, UIMA semble la meilleure alternative car il s'agit d'un projet plus maintenu, bien documenté (des manuels détaillés), soutenu par une communauté très active (une multitude de blogs et de listes de diffusion où on peut poser ses questions et recevoir une réponse rapide) et qui ne cesse d'être amélioré et mis à jour du fait de l'augmentation du nombre de ses adhérents. UIMA présente également un avantage majeur : l'utilisation d'un *Type system* qui définit les types utilisés pour annoter le texte ainsi que leurs propriétés. Ceci assure une interopérabilité entre les différents composants d'une chaîne d'annotation.

4.2 Apache UIMA : un standard OASIS

UIMA est un framework de traitement des données non structurées d'abord lancé par IBM et dont l'architecture a été normalisée par l'OASIS le 19 Mars 2009. Le code source d'une implémentation de référence de ce framework a été mis à disposition d'abord sur SourceForge, ensuite sur le site Internet de la fondation Apache [2]. Nous présentons, dans cette section, les notions de base concernant le framework Apache UIMA que nous avons tirées de la documentation fournie par la fondation Apache [3].

L'objectif d'UIMA est de décrire les étapes de traitement d'un document non

1. `http://opennlp.apache.org/`
2. `http://uima.apache.org/`
3. `https://uima.apache.org/documentation.html`

structuré (texte, image, vidéo, etc.), pour en extraire de façon automatique des informations structurées. Ce framework qui prend en compte de nombreuses problématiques de façon native (réutilisation de composants, déploiement distribué, la gestion des erreurs, etc.) permet de spécifier les interfaces de composants, les structures de données, les patrons de conception et la démarche de développement pour créer, décrire, détecter, agréger et déployer des capacités d'analyse multimodale.

UIMA fournit un environnement d'exécution dans lequel les développeurs peuvent intégrer leurs implémentations de composants UIMA pour construire et déployer des applications, ainsi qu'un kit de développement qui inclut une implémentation Java de l'architecture et du framework UIMA. Ce kit inclut aussi un ensemble de plugins Eclipse qui facilitent le développement avec UIMA.

Dans les sections qui suivent, nous définissons les éléments de base caractérisant le framework UIMA.

4.2.1 Analysis Engines (AEs)

L'architecture UIMA s'appuie sur des composants de base appelés *Analysis Engines* (AEs). Les AEs permettent d'analyser un document et d'en extraire des attributs sous forme de méta-données décrivant soit le document en entier soit des régions particulières du document. Ces attributs descriptifs sont appelés des *Analysis Results* (ARs).

Dans un document textuel, le mot *span* est utilisé pour désigner une séquence de caractères. Si nous considérons, par exemple, le document textuel identifié par le nom *BTID-10080* contenant le *span* "human gastrointestinal tract" à la position 50, un AE chargé de détecter les habitats de bactéries dans le texte devrait produire comme AR l'énoncé suivant :

Le span qui s'étend de la position 50 à la position 78 dans le document BTID-10080 représente un habitat de bactéries.

Le terme *habitat de bactéries* désigne ce qu'on appelle en UIMA un type. Les types caractérisent, en effet, les différents types d'ARs que peuvent produire les AEs.

Les algorithmes qui permettent d'analyser les documents et d'enregistrer les ARs sont implémentés dans des composants appelés annotateurs. Les annotateurs sont encapsulés dans les AEs pour ajouter l'infrastructure nécessaire à leur composition et leur déploiement dans le framework UIMA.

Les composants de base UIMA dont les AEs et les annotateurs sont réutilisés pour créer n'importe quel composant. Ils se composent essentiellement de deux parties.

- La partie déclarative contient des méta-données qui décrivent le composant, son identité, sa structure et son comportement. Elle est représentée par un fichier XML.
- La partie code contient l'implémentation de l'algorithme encapsulé par le composant dans un langage de programmation comme Java.

On appelle un AE primitif (AE) un AE qui encapsule un seul annotateur et donc qui effectue une tâche unitaire comme la segmentation en mots, la lemmatisation, etc. Un AE complexe ou agrégat (AAE) est un AE qui contient une collection d'AEs primitifs ou agrégats.

4.2.2 Common Analysis Structure (CAS)

Un *Common analysis Structure* (CAS) est défini au sein de l'architecture UIMA pour permettre aux annotateurs de représenter et de partager leurs ARs. Il s'agit donc du point central des échanges entre les composants. Le CAS contient le document original, ses méta-données (annotations) et des interfaces qui permettent d'accéder aux données.

Un CAS est une structure de données pour représenter des objets, des propriétés et des valeurs. L'organisation du CAS est définie par un *Type System* (TS). UIMA fournit des types de base pour constituer le TS. Ces types de base peuvent être étendus pour aboutir à un TS plus riche. Un TS est spécifique à un domaine ou à une application, et les types dans un TS peuvent être organisés sous la forme d'une taxonomie.

4.2.3 Développement d'applications avec UIMA

Plusieurs applications de gestion d'informations non structurées sont amenées à analyser des collections entières de documents et à exploiter les résultats d'analyse de différentes manières, mais une application de base utilisant UIMA doit généralement suivre un schéma de fonctionnement décrit par les étapes suivantes :

1. la connexion à une source physique de documents ;
2. la sélection d'un document à analyser ;
3. l'initialisation du CAS avec le document sélectionné ;
4. l'envoi du CAS à un AE sélectionné ;
5. le traitement du CAS résultant ;
6. retour à l'étape 2 jusqu'à ce que la collection de documents soit traitée en entier ;
7. procéder à n'importe quel traitement final après l'analyse de tous les documents.

La figure 4.1 montre le fonctionnement général d'une application utilisant UIMA. Pour assurer ce fonctionnement, l'architecture UIMA définit d'autres composants de base dont les principaux sont les suivants.

Le *CAS initialyzer* – Il est propre à un format de documents sources et permet de préparer et d'ajouter des documents au CAS

Le *Collection Reader* – Il se connecte à une source de documents et itère sur la collection pour initialiser les objets CAS avant l'analyse

Le *CAS Consumer* – Les CAS Consumers interviennent à la fin de la chaîne de traitement pour traiter les résultats d'analyse stockés dans le CAS final et les rendre exploitables par d'autres applications comme l'indexation des résultats pour un moteur de recherche ou l'alimentation d'une base de données par exemple.

Le *Collection Processing Engine* (CPE) – Il s'agit d'un composant agrégat qui décrit une chaîne de traitement en partant d'un Collection Reader, en

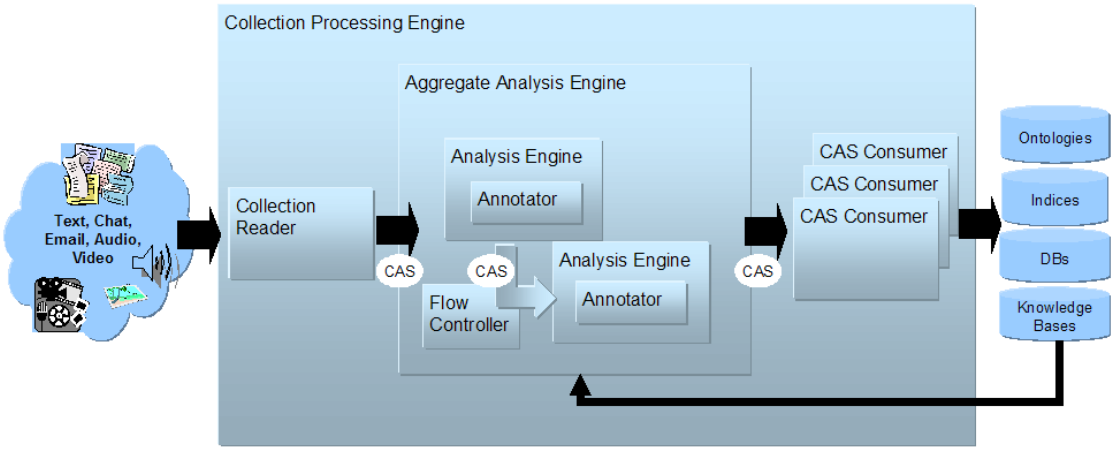


Figure 4.1 – Fonctionnement général d'une application utilisant UIMA. Schéma tiré de la documentation Apache UIMA.

passant par un ensemble d'AEs et en finissant par les CAS Consumers. Ce composant est décrit par un fichier XML (un descripteur) déclaratif qui pointe vers les différents composants de la chaîne et spécifie le flux qui circule entre eux.

Le *Collection Processing Manager* (CPM) – Ce composant se charge de déployer et d'exécuter les descripteurs XML des CPEs. Il permet de gérer les objets CAS, de reprendre en cas d'incident, de créer les fichiers journaux et de gérer les performances.

Pour plus de détails, se référer au tutoriel complet (*UIMA Tutorial Developers' guides*) [4].

4.3 Apache UIMA Ruta

Nous donnons, dans cette section, des notions de base concernant le langage Ruta que nous avons tirées de la documentation fournie par la fondation Apache [5].

4.3.1 Le langage Ruta : Concepts de base

Ruta est un langage de règles impératif étendu avec des éléments de script. Il a été adopté par la fondation Apache UIMA pour soutenir le développement d'applications d'analyse de texte dans UIMA.

Une règle Ruta définit un patron d'annotations avec des conditions supplémentaires. Si ce patron s'applique, alors les actions de la règle sont effectuées sur les annotations couvertes.

Comme l'écriture des règles (voir section 4.3.2) est une tâche fastidieuse et source d'erreurs manuelles, un *UIMA Ruta Workbench* a été développé pour faciliter l'écri-

4. `uima.apache.org/d/uimaj-2.6.0/tutorials_and_users_guides.pdf`
5. `https://uima.apache.org/d/ruta-current/tools.ruta.book.html`

ture de règles en fournissant autant d'outillage que possible. Cela comprend, par exemple, la vérification de la syntaxe et l'auto-complétion, ce qui rend le développement moins sujet aux erreurs. Parfois, il est aussi nécessaire de déboguer les règles parce qu'elles ne se comportent pas comme prévu. Dans ce cas, la *explanation perspective* offre une vue qui détaille le processus d'application des règles. Certains outils comme la *Query view* peuvent également utiliser les règles Ruta comme des requêtes pour interroger des documents annotés.

UIMA Ruta s'intègre parfaitement avec Apache UIMA. En effet, les règles Ruta sont appliquées à l'aide d'un AE générique (*Ruta Engine*) (voir section 4.3.3) et par conséquent les scripts de règles Ruta peuvent facilement être ajoutés aux chaînes de traitement UIMA. UIMA Ruta permet également d'importer et d'utiliser d'autres composants UIMA comme les AEs et les TSs. Les règles UIMA Ruta peuvent se référer à n'importe quel type défini dans un TS importé, et l'*UIMA Ruta Workbench* génère un TS contenant tous les types qui ont été définis dans un script de règles Ruta.

4.3.2 Écriture des règles Ruta

Avant d'être appliquées sur un document, respectivement sur un CAS, les règles Ruta sont toujours regroupées dans un fichier de script. Toutefois, un fichier de script Ruta contient non seulement des règles, mais aussi d'autres déclarations (voir figure 4.2).

```
PACKAGE ruta;
TYPESYSTEM ruta.BBTypeSystemDescriptorRLP;

Token{FEATURE("lemma","alkaline")} Token{FEATURE("lemma","environment")-> MARKONCE(MBTO00000032,1,2)} ;
Token{REGEXP("alluvial")} Token{REGEXP("gravel")} Token{FEATURE("lemma","aquifer")-> MARKONCE(MBTO00000077,1,3)} ;
Term{FEATURE("termLemma","sulfide-oxidizing")} Term{FEATURE("termLemma","bioreactor")-> MARKONCE(MBTO00001719,1,2)} ;
```

FIGURE 4.2 – Exemple d'un script Ruta.

Chaque fichier de script commence, tout d'abord, par une déclaration de *package*, suivie par une liste d'importations optionnelles, suivie par la partie commune qui constitue le corps du script et qui contient les règles et les déclarations de types ou de blocs.

Annotations de base

Le système UIMA Ruta utilise un tokenizer évolué pour créer dans un premier temps un ensemble d'annotations de base. La figure 4.3 montre la hiérarchie formée par ces annotations (types).

Le type `ALL` est la racine de la hiérarchie. Il forme, avec les types d'annotation `ANY`, `WS`, `W`, `PM` et `SENTENCEEND`, un ensemble de types abstraits. Cela signifie qu'ils ne sont pas réellement créés par l'analyseur lexical. Les feuilles de la hiérarchie sont créées par l'analyseur lexical. Chaque feuille correspond à un type propre, mais aussi hérite de l'un des types d'annotation abstraits figurant plus haut dans la hiérarchie.

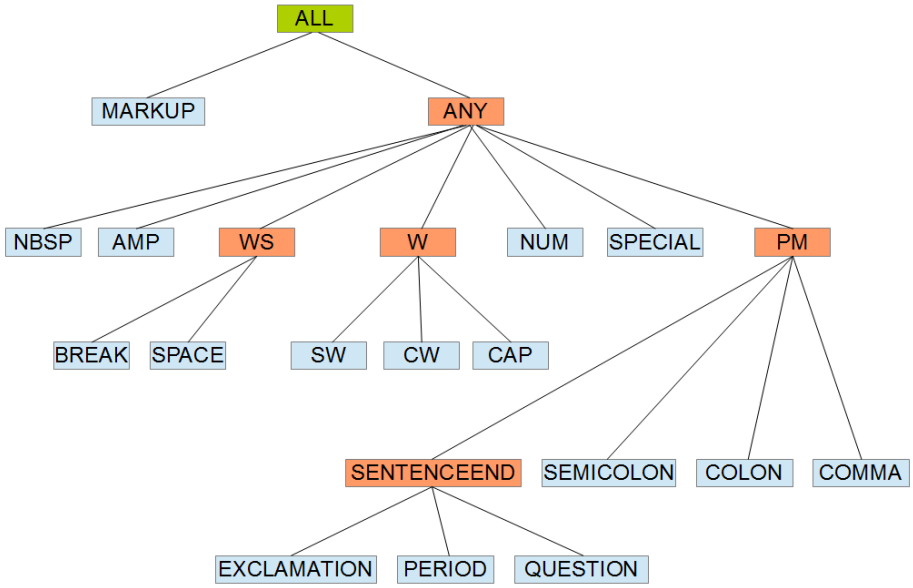


Figure 4.3 – La hiérarchie des annotations de base dans le langage Ruta. Schéma tiré de la documentation Apache UIMA Ruta.

Chaque unité de texte dans un document d'entrée appartient à exactement un de ces types d'annotations feuilles. Les différents types de la hiérarchie sont décrits plus en détails dans la table 4.1.

Ces annotations sont créées, par défaut, dans tous les documents concernés par l'application d'un script Ruta. Toutefois, l'utilisateur peut choisir d'ignorer ces annotations en paramétrant son *Ruta Engine* convenablement (voir section 4.3.3) et se contenter d'utiliser ses propres types pour inférer des règles.

Syntaxe des règles

Une règle Ruta est composée d'une séquence d'éléments de règles. Un élément de règle est essentiellement constitué de quatre parties.

- **Une condition de correspondance** : il s'agit généralement d'un type d'annotation par lequel l'élément de règle est mis en correspondance avec le fragment de texte couvert.
- **Un quantificateur optionnel** : il spécifie combien de fois l'élément de règle peut s'appliquer et s'il doit obligatoirement s'appliquer.
- **Une liste de conditions** : elle spécifie des contraintes supplémentaires que le texte ou les annotations identifiées doivent remplir.
- **Une liste d'actions** : elle définit les conséquences de la règle. Elle crée souvent de nouvelles annotations ou modifie des annotations existantes.

Voici un exemple de règle Ruta : étant donné un *Type System* UIMA qui contient le type `SPACE` (espace) et `Token` (avec une propriété `lemma` qui contient la forme lemmatisée du mot en question), le règle suivante peut être utilisée pour reconnaître

Annotation	**Père**	**Description**
ALL	-	Le père de tous les autres types
ANY	ALL	Toutes les annotations à part celles dont le type est MARKUP
W	ANY	Tous les mots
PM	ANY	Tous les signes de ponctuation
WS	ANY	Tous les caractères blancs
SENTENCEEND	PM	Tous les signes de ponctuation qui indiquent une fin de phrase
MARKUP	ALL	Les éléments HTML et XML
NBSP	ANY	Espace insécable
AMP	ANY	Caractère typographique ou symbôle &
BREAK	WS	Caractère de retour à la ligne
SPACE	WS	Espace
COLON	PM	Deux-points (:)
COMMA	PM	Virgule
PERIOD	SENTENCEEND	Point
EXCLAMATION	SENTENCEEND	Point d'exclamation
SEMICOLON	PM	Point-virgule (;)
QUESTION	SENTENCEEND	Point d'interrogation
SW	W	Mot en miniscules
CW	W	Mot commençant par une majuscule
CAP	W	Mot en majuscules
NUM	ANY	Séquence de chiffres
SPECIAL	ANY	Tout autre symbôle

TABLE 4.1 – Description des annotations de base dans le langage Ruta.

le terme *human body* :

```
Token{REGEXP("human")} SPACE Token{FEATURE("lemma","body")
-> MARK(BacteriaHabitat,1,2,3)};
```

Cette règle peut être lue de la manière suivante : si on trouve un `Token` dont l'expression est *human* suivi par un espace suivi par un `Token` dont le lemme est *body* alors on crée une annotation appelée `BacteriaHabitat` qui couvre les expressions identifiées par les trois éléments de la règle.

Le langage Ruta est d'une grande expressivité et offre beaucoup d'actions possibles (autres que la création de nouvelles annotations) comme la suppression d'annotations, l'attribution d'un score à une annotation, le filtrage d'annotations, etc. Si on spécifie, par exemple, dans notre script Ruta qu'on veut filtrer l'annotation `SPACE`, la règle suivante qui a une expression plus simple permettrait d'identifier les mêmes expressions couvertes par la règle précédente.

```
Token{REGEXP("human")} Token{FEATURE("lemma","body")
-> MARK(BacteriaHabitat,1,2)};
```

Il est important de mentionner que, sauf indication contraire, une règle UIMA Ruta commence normalement par chercher une correspondance au premier élément de la règle. Le premier élément cherche les positions possibles pour sa condition de correspondance, ensuite avise l'élément de règle suivant pour continuer le processus d'appariement. C'est pour cette raison que l'écriture de règles qui contiennent en premier élément un quantificateur optionnel est déconseillée et revient à ignorer l'attribut optionnel du quantificateur.

Une description formelle et détaillée de la syntaxe du langage Ruta figure dans la documentation fournie par Apache UIMA Ruta disponible à l'adresse `http://uima.apache.org/ruta.html`.

4.3.3 Application des règles Ruta : Ruta Engine

Ruta Engine est le plus important AE dans le langage UIMA Ruta dans la mesure où il est responsable de l'application des règles UIMA Ruta sur un CAS donné. Il s'agit d'un AE générique qui doit être configuré en utilisant les paramètres de configuration résumés dans la table 4.2.

Ruta Engine peut être configuré et lancé soit en utilisant des interfaces UIMA soit directement à partir d'un code en Java par exemple. Cette deuxième alternative peut être plus intéressante dans la mesure où l'application d'un script Ruta devient plus automatique et plus transparente.

4.4 Apache UIMA Ruta TextRuler

Le système TextRuler (Kluegl, Atzmueller, Hermann, & Puppe, 2009) est un système de développement semi-automatique d'applications d'EI à base de règles s'appuyant sur le langage Ruta.

Paramètre	**Description**
mainScript	Nom (contenant l'espace de noms complet) du script qui sera interprété et exécuté par l'AE
scriptEncoding	Encodage de tous les fichiers de script UIMA Ruta
scriptPaths	Liste des chemins absolus des répertoires qui contiennent les fichiers de script nécessaires comme le mainScript
descriptorPaths	Liste des chemins absolus des répertoires qui contiennent les fichiers de descripteurs nécessaires comme les TSs
resourcePaths	Liste des chemins absolus des répertoires qui contiennent les fichiers de ressources nécessaires comme la liste des mots
additionalScripts	Liste des noms (contenant l'espace de noms complet) des scripts supplémentaires qui peuvent être appelés
additionalEngines	Liste des noms (contenant l'espace de noms complet) des AEs supplémentaires qui peuvent être appelés par les règles Ruta
additionalUimafitEngines	Liste des noms (contenant l'espace de noms complet) des uimaFIT AEs supplémentaires qui peuvent être appelés par les règles Ruta
additionalEngineLoaders	Liste des noms des classes qui sont en mesure d'effectuer des tâches supplémentaires lors du chargement d'AE externes
additionalExtensions	Liste des classes pour les extensions supplémentaires du langage UIMA Ruta comme les conditions de propriété
reloadScript	Option pour initialiser le script de règles chaque fois que l'AE traite un CAS
seeders	Liste des noms des classes qui fournissent des annotations supplémentaires avant l'exécution des règles
defaultFilteredTypes	Liste des noms complets des types d'annotation qui sont invisibles par défaut (filtrés)
removeBasics	Option pour supprimer toutes les annotations de base de Ruta après l'exécution du script de règles
strictImports	Option pour limiter la résolution des noms de types courts à ceux figurant dans les TSs déclarés
dynamicAnchoring	Option pour permettre au processus d'appariement de règles de commencer à n'importe quel élément de règle
lowMemoryProfile	Option pour réduire la consommation de mémoire lors du traitement d'un grand CAS
simpleGreedyForComposed	Option pour activer un processus d'inférence différent pour les éléments des règles composés

debug	Option pour ajouter des informations de débogage au CAS
debugWithMatches	Option pour ajouter des informations concernant les correspondances des règles au CAS
debugOnlyFor	Liste d'identifiants de règles. Si fournie, alors les informations de débogage sont créées uniquement pour ces règles
profile	Option pour ajouter des informations de profiling au CAS
statistics	Option pour ajouter les statistiques des conditions et des actions au CAS
createdBy	Option pour ajouter des informations supplémentaires, quelle règle a créé une annotation

Table 4.2 – Description des paramètres de configuration de Ruta Engine.

4.4.1 Algorithmes d'apprentissage de règles

Le système TextRuler correspondant à la version 2.0.0 de Ruta (anciennement appelé TextMarker) contient les implémentations des algorithmes $(LP)^2$ (Ciravegna, 2001, 2003) ($(LP)^2{}_R$), WHISK (S. Soderland et al., 1999) ($WHISK_R$), RAPIER (Califf & Mooney, 2003) ($RAPIER_R$), BWI (Freitag & Kushmerick, 2000) (BWI_R) et WIEN (Kushmerick, Weld, & Doorenbos, 1997) ($WIEN_R$) adaptés au langage Ruta.

À partir de la version 2.0.1 de Ruta, TextRuler ne contient plus les implémentations des algorithmes RAPIER, BWI et WIEN qui ont laissé la place aux algorithmes TraBal (Eckstein, Kluegl, & Puppe, 2011) et KEP.

$(LP)^2{}_R$

Cet algorithme implémente les idées publiées dans (Ciravegna, 2003) mais s'appuie sur le langage Ruta.

$(LP)^2{}_R$ apprend des règles distinctes pour le début et la fin d'un segment de texte qui représente un exemple d'apprentissage. Ces règles sont ensuite combinées dans le but d'identifier l'annotation cible. La stratégie d'apprentissage utilisée par cet algorithme couvrant est une stratégie ascendante. Il commence par créer une instance spécifique d'une graine avec une fenêtre de contexte de w *tokens* à gauche et à droite de la limite cible (limite gauche ou droite de l'annotation cible) et recherche ensuite la meilleure généralisation possible. Des règles contextuelles supplémentaires sont induites dans le but d'identifier les limites manquantes. L'implémentation actuelle de l'algorithme n'inclut pas les règles de correction prévues dans l'algorithme $(LP)^2$ de base. TextRuler propose deux versions de cet algorithme : $(LP)^2{}_R$ (version naïve) est une implémentation simple de $(LP)^2$ avec une expressivité limitée concernant les règles Ruta résultantes. $(LP)^2$ textsubscriptR (version optimisée) est une version améliorée avec une approche de programmation dynamique qui fournit souvent de meilleurs résultats. Les paramètres qui peuvent être fixés par l'utilisateur sont :

— la taille de la fenêtre de contexte (à gauche et à droite) ;
— la taille de la liste des meilleures règles ;
— le nombre minimum d'exemples positifs qui doivent être couverts par une règle ;
— le taux d'erreur maximum ;
— la taille de la liste des règles contextuelles.

$\mathbf{WHISK_R}$

Cet algorithme implémente les idées publiées dans (S. Soderland et al., 1999) mais s'appuie sur le langage Ruta.

WHISK_R utilise une méthode *multi-slot* pour apprendre des règles et traite trois types de documents textuels : structurés, semi structurés et non structurés. Les règles *multi-slot* ou *single-slot* qu'il apprend sont similaires à des expressions régulières. Cependant, l'implémentation actuelle de l'algorithme n'inclut que les règles *single-slot*. Cet algorithme couvrant descendant commence avec la règle la plus générale et la spécialise en ajoutant des termes jusqu'à ce qu'elle ne fasse plus d'erreurs sur l'ensemble d'apprentissage. TextRuler propose deux versions de cet algorithme : WHISK_R (token) est une implémentation naïve qui s'appuie sur les *tokens* annotés par le langage Ruta. WHISK_R (générique) est une implémentation améliorée capable d'utiliser les annotations disponibles dans le corpus (celles ajoutées par un utilisateur par exemple). Les paramètres qui peuvent être fixés par l'utilisateur sont :
— la taille de la fenêtre de contexte (à gauche et à droite) ;
— le taux d'erreur maximum.

TraBal

Cet algorithme est implémentation des idées publiées dans (Eckstein et al., 2011).

Trabal induit des règles qui tentent de corriger des erreurs d'annotation en s'appuyant sur deux ensembles de documents : un ensemble de documents *gold standard* et un ensemble de documents annotés avec le même texte qui contiennent éventuellement des annotations erronées, pour lesquelles des règles de correction doivent être apprises. L'algorithme compare, tout d'abord, les deux ensembles de documents et identifie les erreurs présentes. Ensuite, des règles sont induites et étendues pour chaque erreur. Ce processus peut être itéré afin d'éliminer progressivement toutes les erreurs. Les paramètres qui peuvent être fixés par l'utilisateur sont :
— le nombre d'itérations de l'algorithme ;
— le nombre de règles de base à inférer pour un exemple ;
— le nombre de règles optimisées à inférer pour un exemple ;
— le nombre maximum d'itérations pour optimiser les règles ;
— le taux d'erreur maximum.

KEP

Le nom de l'algorithme KEP (patrons d'ingénierie de connaissances) est dérivé de l'idée que les humains utilisent différents patrons d'ingénierie pour écrire des règles d'annotation. Cet algorithme implémente des méthodes simples d'induction

de règles pour certains patrons, telles que la détection de limite ou la restriction de la fenêtre basée sur les annotations. Les résultats sont ensuite combinés de manière à tirer profit de la combinaison de différents types de règles induites. Comme les règles simples sont construites selon la façon dont les humains écrivent les règles d'annotation, l'ensemble des règles résultant devrait ressembler à un ensemble de règles écrites manuellement. En outre, en exploitant la synergie des patrons, des solutions pour certaines annotations sont beaucoup plus simples. Les paramètres qui peuvent être fixés par l'utilisateur sont :

— le nombre maximum des règles d'expansion ;
— le nombre maximum des règles d'annotation.

4.4.2 Interface TextRuler

La figure 4.4 représente la vue TextRuler dans la perspective UIMA Ruta d'un projet Apache UIMA Ruta (pour plus de détails concernant la création d'un projet Apache UIMA Ruta dans Eclipse, se référer à la documentation fournie par Apache).

L'interface TextRuler est composée de 4 parties.

— La barre d'outils : contient des boutons pour lancer (bouton vert) et arrêter (bouton rouge) le processus d'apprentissage et un bouton (bleu) qui permet de configurer les différents algorithmes d'induction de règles.
— La partie supérieure de la vue : contient des champs textuels pour définir l'ensemble des documents utilisés. *Training Data* pointe vers le chemin absolu du répertoire contenant les documents gold standard (les documents d'apprentissage). *Additional Data* pointe vers le chemin absolu du répertoire contenant des documents supplémentaires qui peuvent être utilisés par les algorithmes d'apprentissage. Pour l'instant, seul l'algorithme TraBal a besoin de ces documents. *Test Data* n'est pas encore disponible. Enfin, *Preprocess Script* pointe vers le chemin absolu du script UIMA Ruta contenant tous les types nécessaires et qui peut être appliqué sur les documents avant le lancement de l'algorithme pour ajouter des annotations utiles pour l'apprentissage. La phase du prétraitement peut néanmoins être ignorée.
— La partie centrale de la vue : *Information Types* contient la liste des annotations (types) cibles qui doivent être induits et *Filtered Feature Types* contient la liste des types à filtrer dans l'induction des règles.
— La partie inférieure de la vue : contient la liste des algorithmes d'induction de règles disponibles. Les algorithmes cochés sont tous lancés quand le bouton de lancement dans la barre d'outils est pressé.
— La partie droite de la vue : Quand les algorithmes d'induction de règles sont lancés, leurs résultats (l'ensemble des règles apprises) sont affichés dans la partie droite de la vue où un onglet portant le nom de l'algorithme suivi du mot *Results* est créé pour chaque algorithme.

Un exemple d'utilisation de TextRuler est disponible dans la documentation Apache UIMA Ruta.

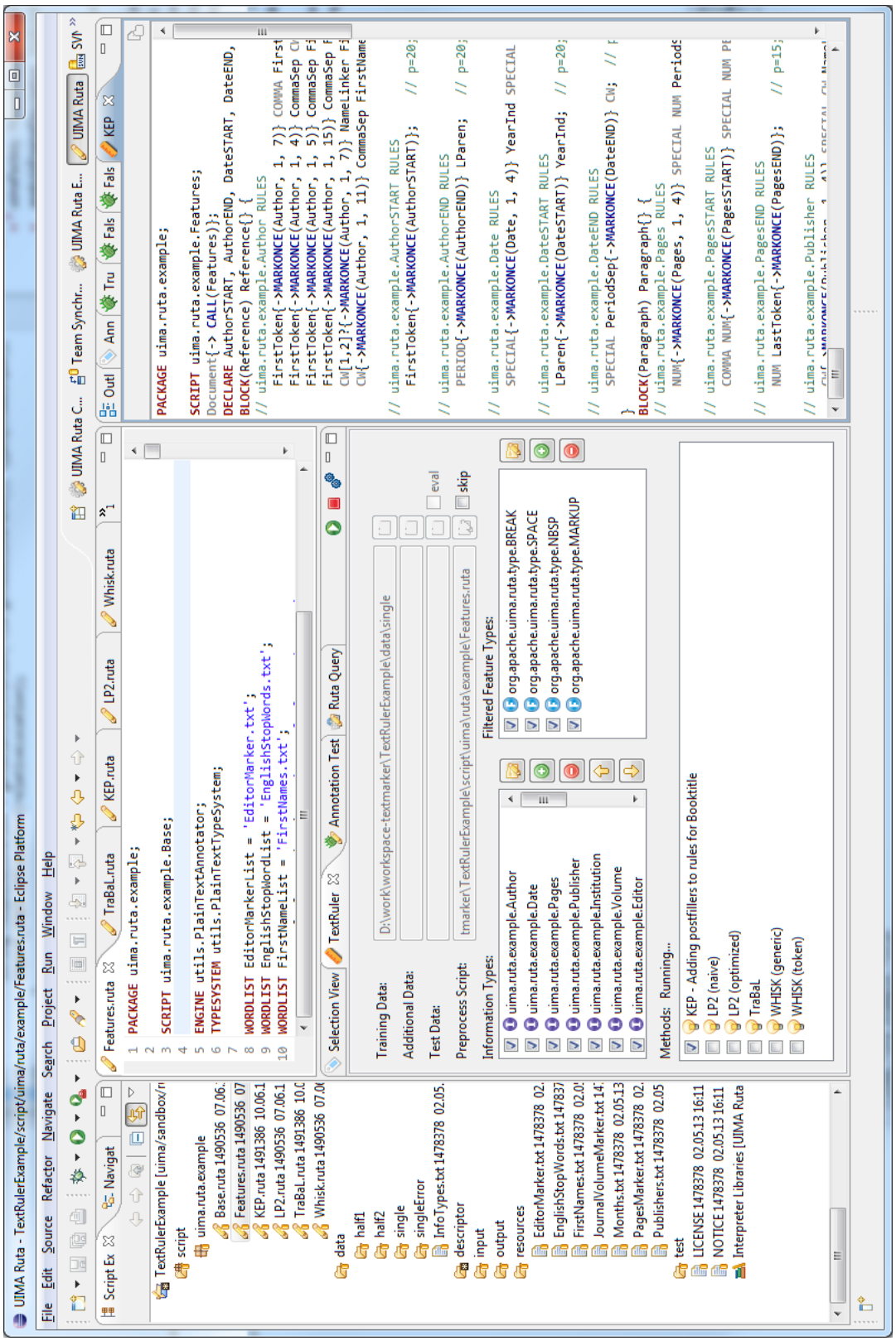


FIGURE 4.4 – L'interface de TextRuler. Schéma tiré de la documentation Apache UIMA Ruta.

Conclusion

Nous avons présenté, dans ce chapitre, nos choix concernant la plateforme de gestion de contenus non structurés et le langage de règles d'EI à utiliser. La plateforme UIMA semble convenir le mieux pour annoter le texte de départ car non seulement elle offre plus de fonctionnalités que ses concurrentes (standardisation, parallélisation, gestion des ressources, etc.) mais aussi parce que c'est un projet actif qui continue à être maintenu et amélioré. Le langage Ruta est un langage de règles d'une grande expressivité qui s'appuie sur UIMA et qui a été jugé par plusieurs utilisateurs plus complet que le langage JAPE, un des langages les plus connus et intégré à la plateforme GATE. Le système TextRuler est un système de développement d'applications d'EI à base de règles s'appuyant sur le langage Ruta. Il s'agit de l'un des premiers systèmes à fournir aux utilisateurs des algorithmes d'apprentissage de règles qui utilisent des langages de règles communs et familiers aux ingénieurs de connaissance. Même si les implémentations courantes de ces algorithmes ne sont pas optimisées et parfois incomplètes, elles peuvent être améliorées et étendues de manière à devenir intéressantes. L'algorithme WHISK_{R}, par exemple, semble être un bon candidat dans un système interactif d'apprentissage de règles car les règles qu'il produit sont simples, compréhensibles et facilement extensibles par un être humain. L'ajout de fonctionnalités, comme la prise en compte des propriétés des annotations, à cet algorithme permettrait d'améliorer ses performances.

Nous expliquons de manière détaillée, dans le chapitre suivant, les problématiques posées dans ce travail, ce qui permettra de guider davantage nos choix concernant l'environnement technique.

Chapitre 5

Problématique : interaction dans l'apprentissage de règles d'extraction d'information

Comme nous l'avons mentionné dans le chapitre 2, il existe deux principales familles d'approches d'extraction d'information (EI) : les approches à base de règles écrites manuellement et les approches à base d'apprentissage artificiel. Les deux familles d'approches nécessitent un effort manuel non négligeable que ce soit pour écrire les règles ou pour annoter les exemples d'apprentissage. Quelques systèmes interactifs ont été proposés (voir chapitre 3) pour permettre à l'utilisateur d'intervenir dans des processus d'EI automatiques. Néanmoins, cette intervention est souvent limitée à une sélection d'exemples d'apprentissage ou à une correction de quelques erreurs.

Nous pensons qu'un système interactif qui combine les deux approches d'EI permettrait de réduire considérablement l'effort manuel requis. Ce système doit, en revanche, assurer un certain nombre d'objectifs et avoir un certain nombre de propriétés clés.

Après l'explicitation des différents objectifs du système que nous voulons mettre en place et la définition du profil de l'utilisateur visé dans la première section de ce chapitre, nous détaillons dans la deuxième section les différentes propriétés clés que doit avoir ce système pour lever un certain nombre de verrous. La troisième section est réservée à la présentation des corpus que nous utilisons tout au long de ce travail pour valider notre approche.

5.1 Placer l'utilisateur au centre du système

Un système interactif d'apprentissage de règles d'EI est un système qui nécessite une coopération de l'utilisateur et de l'algorithme d'apprentissage dans le but de réduire au maximum l'effort humain requis dans l'écriture manuelle de règles et dans l'annotation des exemples d'apprentissage. Pour ce faire, ce système doit permettre :

— une communication aisée entre l'utilisateur et le module d'apprentissage de règles ;

— une liberté pour l'utilisateur de choisir l'action à effectuer pour faire évoluer le système entre l'annotation d'exemples d'apprentissage, l'écriture de règles d'EI et la modification de règles existantes ;
— une exploration facile des exemples ;
— une aide pour identifier les bons exemples à annoter ;
— un test aisé des règles écrites ou inférées automatiquement et une navigation facile dans les exemples qu'elles couvrent.

Nous ne ciblons pas, dans ce travail, un profil particulier d'utilisateur. Nous voulons que notre système puisse être utilisé par différents profils d'utilisateurs. Cependant le domaine ciblé peut jouer beaucoup sur le profil d'utilisateur qui peut utiliser le système.

Pour des domaines qui nécessitent des connaissances métier (extraction d'informations à propos de gènes ...), la présence d'un expert de domaine est nécessaire. Cet expert peut soit utiliser le système seul soit coopérer avec un ingénieur de connaissances ou un informaticien. Dans le premier cas, il peut se contenter d'annoter des exemples et de valider/invalider les exemples couverts par des règles inférées. Il peut également être assisté dans l'annotation des exemples : le système peut contenir un module d'apprentissage actif qui propose à l'expert des exemples à annoter à chaque itération. Dans le deuxième cas, l'ingénieur de connaissances se charge de traduire en règles les connaissances métier fournies par un expert (annotation d'exemples). Il peut également modifier/ajuster des règles inférées en s'appuyant sur les statuts donnés par l'expert de domaine aux exemples couverts par les règles inférées.

Pour des domaines qui ne nécessitent, en revanche, pas ou peu de connaissances métier (extraction d'opinions à propos d'un produit, extraction d'informations à propos de conférences, extraction d'informations à propos de contacts mail, ...), un utilisateur qui a des connaissances générales en informatique (capable de comprendre la syntaxe des règles) serait capable de travailler à la fois sur les règles et l'annotation des exemples. Il s'agit du profil d'utilisateur qui tirera le plus de bénéfices d'un système interactif d'EI. C'est donc à ce profil d'utilisateur que nous nous référons dans ce travail.

5.2 Propriétés d'un système interactif d'apprentissage de règles d'extraction d'information

Pour assurer ses objectifs, un système interactif d'apprentissage de règles d'EI doit se caractériser par des propriétés clés.

5.2.1 Compréhensibilité des règles

Contrairement à la plupart des techniques d'apprentissage connexionnistes telles que les réseaux de neurones qui sont considérés comme des techniques difficilement interprétables car la connaissance acquise est dissimulée dans un grand nombre de connexions et n'est pas transparente pour l'utilisateur, les techniques d'apprentissage symboliques comme l'induction de règles sont généralement considérées comme

des techniques compréhensibles car les connaissances apprises sont souvent exprimées sous forme de règles de production facilement interprétables par l'utilisateur (Z.-H. Zhou & Jiang, 2003). Heller, Veltink, Rijkhoff, Rutten, et Andrews (1993) comparent une technique d'induction de règles à une technique basée sur les réseaux de neurones sur une tâche de reconstruction des modèles d'activation des muscles lors de la marche normale à partir de données cinématiques. Ils remarquent que les règles produites par l'apprentissage inductif sont à la fois explicites et compréhensibles alors que celles provenant du réseau de neurones sont implicites et pas facilement interprétables par un humain.

Dans les projets de fouille de données temps réel, les données sont souvent imprécises et peuvent contenir des erreurs ou des valeurs manquantes, ce qui rend impossible la création de modèles avec des performances suffisantes pour les systèmes entièrement automatisés. Dans ces cas, les prédictions doivent être analysées et ajustées manuellement. Étant donné que l'analyse de ces prédictions nécessite des modèles compréhensibles, les modèles opaques ne pourraient pas être analysés. Pour contourner cet inconvénient, les chercheurs ont développé des techniques d'extraction de règles qui tentent d'extraire des règles compréhensibles à partir de modèles opaques, tout en conservant une précision suffisante (Huysmans, Baesens, & Vanthienen, 2006).

La compréhensibilité des règles est un critère important quand on cherche à évaluer des techniques d'extraction ou d'apprentissage de règles. Cependant, étant donné la nature subjective de ce critère, il est difficile de le décrire de manière expérimentale ou de le mesurer indépendamment de l'utilisateur.

Dans les techniques d'extraction de règles, la compréhensibilité des règles est souvent évaluée au regard de la taille des règles sachant que la taille d'une règle peut être calculée de différentes manières. Sönströd, Johansson, et König (2007) pensent, en revanche, que la compréhensibilité des règles doit être décomposée en deux mesures.

Interprétabilité : le modèle doit exprimer les conditions de la même manière que l'humain a tendance à les exprimer.

Concision : le modèle doit classer autant d'instances que possible avec peu de conditions.

Dans les techniques d'apprentissage ou d'induction de règles, la compréhensibilité des règles n'a pas été définie de manière claire car les règles symboliques sont réputées être compréhensibles par nature. Nous pensons, cependant, que cette compréhensibilité est très liée à l'expressivité et l'intuitivité du langage dans lequel elles sont exprimées.

Pour être compréhensibles, nous pensons que les règles d'EI doivent :

Avoir une expression symbolique – Contrairement à l'apprentissage statistique, l'apprentissage symbolique a l'avantage de produire généralement une sortie lisible par un être humain.

Être assez courtes – Pour ne pas avoir une expression trop complexe, les règles doivent avoir une expression courte pour pouvoir être facilement lues par un être humain.

Être en nombre réduit – Plus l'ensemble des règles dont on dispose est réduit, plus il est facile pour l'utilisateur de l'explorer.

Utiliser des termes intuitifs – Les éléments constituant les règles doivent être écrits dans des termes similaires à ceux qu'aurait utilisés l'utilisateur. Les règles exprimées à l'aide de vocabulaires de la logique, par exemple, sont des règles faciles à interpréter par l'utilisateur.

5.2.2 Généricité/généralité des règles

La compréhensibilité et la généricité/généralité des règles sont deux critères étroitement liés dans la mesure où la généricité implique :

— Des expressions de règles plus courtes – Plus les règles sont génériques, moins elles contiennent des conditions spécifiques et plus elles sont courtes.
— Un ensemble de règles réduit – Plus les règles sont génériques, plus elles sont capables de généraliser sur l'ensemble d'exemples d'apprentissage et donc moins de règles sont nécessaires pour couvrir cet ensemble.

Dans les systèmes d'EI à base de règles, tous les travaux s'accordent sur le fait que des règles trop spécifiques qui ont une faible couverture posent un problème dans la mesure où elles entraînent un risque de surapprentissge. Pour éviter ce problème, ces règles doivent être généralisées, c'est à dire avoir une expression plus générique et couvrir plus d'exemples. La généricité des règles permet, en effet, d'améliorer les performances en augmentant la couverture des règles et par conséquent de diminuer le nombre total des règles, un facteur très important dans un système interactif où l'utilisateur a besoin d'examiner l'ensemble des règles.

Prenons les trois patrons suivants écrits dans le langage Ruta :

```
1-Token{REGEXP("base")} Token{REGEXP("de")} Token{REGEXP("données")};
2-Token{FEATURE("lemma","base")} Token{FEATURE("lemma","de")}
Token{FEATURE("lemma","donnée")};
3-Term{{FEATURE("lemma","base de données")};
```

Le premier patron permet de reconnaître uniquement le terme « base de données » mais pas ses variations (Base de données, bases de données, etc.). Le second est plus générique car il reconnaît toutes les variations du terme. Le troisième patron est encore plus générique car il permet de détecter des termes comme « BD » lemmatisés « base de données » par un extracteur de termes.

En effet, pour être générique, une règle doit pouvoir reposer sur des annotations elles mêmes génériques, ce qui implique que le texte d'entrée doit être bien préparé avant qu'on puisse construire et apprendre des règles dessus. Il faut cependant noter que la généricité peut impliquer un temps d'apprentissage plus long dans la mesure où il faut tester plus de règles (des règles s'appuyant sur les différentes annotations linguistiques disponibles dans le texte par exemple). Il faut donc trouver le bon compromis entre la généricité et le temps d'apprentissage.

5.2.3 Stabilité des règles

Dans un système interactif d'apprentissage de règles, l'utilisateur examine les règles inférées par l'algorithme d'apprentissage. Il peut se rendre compte que ces règles ne couvrent pas certains exemples qui lui semblent évidents ou au contraire font quelques erreurs. Dans ces cas, il est amené soit à écrire des règles qui ont pour but de chercher des exemples omis, soit à corriger quelques règles existantes. L'algorithme d'apprentissage prendra en compte ces modifications dans l'itération suivante et produira un nouvel ensemble de règles. Toutefois, ce nouvel ensemble peut ne pas contenir les règles que l'utilisateur a écrites ou modifiées. Ceci peut constituer un problème pour l'utilisateur qui aimerait que l'algorithme d'apprentissage garde ses règles parce qu'il les trouve satisfaisantes et compréhensibles. L'algorithme d'apprentissage doit donc prendre en compte les règles écrites ou modifiées par l'utilisateur et essayer de les étendre sans pour autant modifier complètement leurs expressions. Ceci permettrait à l'utilisateur de suivre l'évolution de ses règles au fil des itérations et de garder le contrôle sur leur complexité.

Cet enjeu dépend beaucoup de l'algorithme d'apprentissage utilisé car certains algorithmes sont stables par nature. En effet, lorsqu'on leur fournit de nouveaux exemples, ces algorithmes peuvent procéder de deux manières.

- **Si les exemples fournis ressemblent à des exemples déjà couverts :** l'algorithme essaie de modifier des règles existantes pour prendre en compte les nouveaux exemples. L'expression des nouvelles règles ne diffère généralement pas beaucoup de celle des anciennes.
- **Si les exemples fournis ne ressemblent pas aux exemples déjà couverts :** l'algorithme infère de nouvelles règles pour couvrir les nouveaux exemples.

Il est cependant à noter que la stabilité des règles n'assure pas forcément les meilleures performances car les règles écrites par l'utilisateur peuvent être non optimales, très spécifiques et faire beaucoup d'erreurs sur de nouvelles données. L'utilsateur est donc parfois amené à choisir entre stabilité et performances.

5.2.4 Sélection d'exemples

Dans un système interactif d'apprentissage de règles, le but étant de réduire au maximum le nombre d'exemples à annoter manuellement, un processus de sélection d'exemples pertinents semble indispensable pour permettre à l'utilisateur d'annoter moins d'exemples. La mise en place de la sélection d'exemples peut se faire à deux niveaux pour répondre à deux besoins différents.

Réduire l'espace de travail : l'utilisateur reconnaît des exemples correspondant à des mots clés pertinents. Au lieu de parcourir tout le corpus à la recherche de ces exemples, il doit pouvoir écrire des règles prospectives qui vont lui permettre de les trouver. Les exemples couverts par ces règles constituent un espace de travail réduit pour l'utilisateur. Il peut sélectionner parmi ces exemples couverts ceux qui l'intéressent et les annoter.

Favoriser une convergence rapide de l'algorithme d'apprentissage : pour améliorer le rappel de son système, l'utilisateur doit fournir plus d'exemples

à l'algorithme d'apprentissage. Néanmoins, ces exemples ne sont pas faciles à trouver et l'utilisateur peut avoir du mal à les repérer. Il doit donc pouvoir accéder facilement aux exemples les plus pertinents dans le texte non couverts par les règles dont il dispose en vue de les étiqueter. L'apprentissage actif est une solution qui a été proposée par plusieurs études dans la littérature pour résoudre le problème de sélection d'exemples. Muslea et al. (2000) ; Probst et Ghani (2007) ; Thompson et al. (1999) ont montré que l'apprentissage actif permet de réduire de manière significative le nombre d'exemples annotés manuellement. Finn et Kushmerick (2003) ont étudié plusieurs stratégies d'apprentissage actif qui peuvent être appliquées dans des tâches d'EI et ont montré que plusieurs d'entre elles ont donné des résultats nettement meilleurs qu'une stratégie de sélection aléatoire d'exemples.

5.2.5 Temps d'apprentissage

Même si les systèmes d'EI à base de règles sont considérés comme des systèmes lents, il n'y a jamais eu véritablement d'évaluation expérimentale de ces systèmes en termes de temps. Dans un système interactif, le facteur temps est très important car pour pouvoir interagir avec l'algorithme d'apprentissage, l'utilisateur ne doit pas attendre longtemps avant d'avoir une réponse de l'algorithme. Autrement dit, l'apprentissage de règles doit se faire de manière assez rapide. Il est difficile d'assurer cette rapidité car elle dépend de beaucoup de facteurs comme :

Le temps d'exécution des règles – Le temps que met le moteur d'application des règles pour tester une règle sur le texte diffère d'un langage de règles à un autre et dépend beaucoup de la manière avec laquelle est implémenté le langage de règles.

La taille du corpus – Plus la taille du corpus est grande et plus le test des règles sur le corpus est lent.

Le nombre total des règles à évaluer dans l'espace de recherche de règles – pour pouvoir sélectionner les règles les plus performantes, l'algorithme d'apprentissage de règles commence par construire un espace de recherche de règles qui contient un certain nombre de règles à évaluer et comparer entre elles. Le nombre de ces règles dépend de trois facteurs.

- L'expression des exemples d'apprentissage : plus les exemples sont différents les uns des autres et plus il est difficile de trouver des règles capables de généraliser sur ces exemples. Il faut donc plus de règles pour couvrir des exemples qui s'expriment de manières différentes.
- La richesse des annotations dans le texte : pour pouvoir sélectionner les meilleures règles, l'algorithme d'apprentissage doit tester toutes les règles possibles. Autrement dit, il doit tester les règles qui s'appuient sur les différentes annotations disponibles dans le texte. Par conséquent, plus il y a d'annotations dans le texte et plus le nombre de règles à tester est élevé.
- La taille du contexte à considérer pour apprendre une règle : pour pouvoir désambiguïser certains exemples ou réduire le taux d'erreur de certaines règles, l'algorithme d'apprentissage de règles utilise des éléments de

contexte dans l'expression des règles. Plus la taille de ce contexte (fenêtre de mots) est grande (par exemple les 5 mots qui précèdent et suivent l'exemple en question) et plus il y a des règles à tester.

En optimisant l'un ou plusieurs de ces facteurs, l'utilisateur peut réduire le temps d'apprentissage de règles. Nous ne devons cependant pas perdre de vue que les mesures à prendre pour réduire ce temps ne doivent pas dégrader les performances. Dans le cas contraire, l'utilisateur doit trouver le bon compromis entre performances et temps d'apprentissage.

5.3 Corpus de travail

Pour évaluer et valider notre approche interactive d'apprentissage de règles d'EI, nous utilisons essentiellement deux corpus : le corpus Bacteria Biotope (BB) de BioNLP-ST 2013 et le corpus SyntSem.

5.3.1 Corpus BB BioNLP-ST 2013

Le corpus BB BioNLP est un ensemble de documents qui décrivent de manière générale des espèces de bactéries. Chaque document est centré autour d'une espèce, un genre ou une famille de bactéries ; il contient des informations générales sur leur classification, écologie et intérêt pour les activités humaines. Ces documents ont été extraits à partir de plus de 20 sites Web publics.

Parmi les 2040 documents constituant le corpus, 85 ont été sélectionnés au hasard pour le challenge BioNLP-ST 2013. Les documents ont été annotés en mode double aveugle par les bioinformaticiens de l'équipe Bibliome du laboratoire MIG à l'Institut National de Recherche Agronomique (INRA) en utilisant l'éditeur AlvisAE (Papazian, Bossy, & Nédellec, 2012).

Le corpus que nous considérons dans ce travail est le corpus fourni dans le cadre de la sous-tâche 1 du BB BioNLP-ST (Bossy, Golik, Ratkovic, Bessières, & Nédellec, 2013) dont le but est de détecter, dans le texte, les habitats de bactéries et leur associer une ou plusieurs catégories à partir de l'ontologie OntoBiotope [1] fournie pour la tâche. Ce corpus est composé de 3 ensembles de données : un ensemble d'entraînement (52 documents), un ensemble de développement (26 documents) et un ensemble de test (27 documents). Concernant les ensembles d'enraînement et de développement, des documents contenant des annotations manuelles d'habitats de bactéries et des catégories qui leur sont associées ont été fournis. Chaque habitat est au moins associé à une catégorie. Les annotations de référence n'ont, en revanche, pas été fournies pour les données de test. La seule manière d'évaluer les prédictions d'un algorithme sur ces dernières est de faire une soumission sur l' adressse web du service d'évaluation de la tâche [2].

La table 5.1 donne une vue plus détaillée sur les données d'entraînement (E) et de développement (D) fournies par la tâche.

1. `http://bibliome.jouy.inra.fr/MEM-OntoBiotope/OntoBiotope_BioNLP-ST13.obo`
2. `http://genome.jouy.inra.fr/~rbossy/cgi-bin/bionlp-eval/BB.cgi`

	E	**D**	**E∪D**
Nombre de catégories	333	274	**491**
Nombre d'habitats	934	611	1545
Nombre d'annotations	948	626	1574
Nombre de catégories avec 1 Annotation	182	179	272
Nombre de catégories avec 2 Annotations	66	41	86
Nombre de catégories avec ≥ 3 Annotations	85	54	**133**
Nombre de catégories dans l'ontologie OntoBiotope : **1756**			

TABLE 5.1 – Aperçu sur les données fournies pour la sous-tâche 1 du BB BioNLP-ST 2013. E représente les données d'entraînement et D les données de développement.

La table 5.1 montre que parmi les 1756 catégories qui figurent dans l'ontologie OntoBiotope, uniquement 333 d'entre elles apparaîssent dans les données d'entraînement. Il existe 934 habitats de bactéries dans les données d'entraînement correspondant à 948 annotations qui réfèrent chacune à au moins une catégorie dans l'ontologie OntoBiotope (un habitat est parfois rattaché à plus qu'une catégorie). Parmi les 333 catégories figurant dans les données d'entraînement, 182 possèdent une seule occurrence (annotation), 66 possèdent deux occurrences et 85 possèdent plus que deux occurrences.

En examinant de près la table 5.1, nous pouvons déduire que l'échantillon de données fourni est trop petit pour conduire à un bon rappel pour un algorithme d'apprentissage s'appuyant uniquement sur cet échantillon :

— 158 ($491 - 333$) des 274 catégories (58%) figurant dans les données de développement n'apparaîssent pas dans les données d'entraînement.
— Les catégories qui apparaîssent dans les données d'entraînement représentent 19% (333/1756) du nombre de catégories figurant dans l'ontologie OntoBiotope. Celles qui apparaîssent dans les données de développement représentent 16% (274/1756) et enfin, celles qui apparaîssent dans l'union des données d'entraînement et de développement représentent 28% (491/1756)).
— Il est clairement difficile pour un algorithme d'apprentissage d'apprendre (généraliser) à partir d'un seul exemple. Ceci est le cas pour 55% (272/491) des catégories si on considère à la fois les données d'entraînement et de développement.
— Si nous considérons qu'il faut au moins 3 exemples pour pouvoir appliquer un algorithme d'apprentissage, uniquement 27% (133/491) des catégories présentes dans les données d'entraînement ou de développement sont concernées. Cela veut dire que la couverture de l'ontologie est inférieure à 8% (133/1756).

5.3.2 Corpus SyntSem

Le deuxième corpus sur lequel nous avons travaillé est un corpus qui a été constitué dans le cadre du projet SyntSem. Il s'agit d'un corpus de plus de 6 millions de mots composé de textes variés qui représentent un large échantillon de la langue française issus de 5 thèmes : journaux, sciences humaines, périodiques, textes littéraires

et textes institutionnels européens. Ce corpus est étiqueté morpho-syntaxiquement, lemmatisé et comporte un étiquetage syntaxique et sémantique peu profond. L'étiquetage sémantique concerne essentiellement 60 mots polysémiques (20 noms, 20 verbes et 20 adjectifs). Les occurrences de ces mots dans le corpus (53 796) ont été étiquetés en se basant sur un dictionnaire distributionnel Reymond (2001).

La table 5.2 décrit les 20 noms ploysémiques figurant dans le corpus.

Nom	**Nombre de lexies**	**Nombre total d'exemples**	**Nombre d'exemples de la lexie majoritaire**
Barrage	5	92	70
Chef	11	1133	861
Communication	13	1703	691
Compagnie	12	412	294
Concentration	6	246	111
Constitution	6	422	211
Degré	18	507	297
Détention	2	112	81
Économie	10	930	457
Formation	9	1528	579
Lancement	5	138	110
Observation	3	572	492
Organe	6	366	140
Passage	19	600	222
Pied	62	960	361
Restauration	5	104	45
Solution	4	880	821
Station	8	266	85
Suspension	5	110	68
Vol	10	278	112

Table 5.2 – Statistiques sur les noms ambigus dans le corpus SyntSem.

Dans le cadre de ce travail, nous utiliserons ce corpus pour extraire le sens ou la lexie majoritaire de quelques noms en les considérant comme des concepts à part à annoter.

Conclusion

Convaincus par les multiples avantages d'une approche interactive d'apprentissage de règles dans le cadre d'une tâche d'EI, nous avons présenté dans ce chapitre les différents objectifs ainsi que les différentes propriétes qui doivent caractériser une telle approche et les corpus que nous avons considérés dans ce travail pour la valider.

Un système interactif qui s'appuie sur cette approche doit permettre une communication aisée entre le module d'apprentissage de règles et l'utilisateur en facilitant à ce dernier le travail à la fois sur les exemples d'apprentissage et sur les règles

d'EI. Sachant que les domaines ciblés dans ce travail ne nécessitent pas forcément des connaissances métier, un utilisateur informaticien peut interagir avec le module d'apprentissage de règles à travers deux moyens : les règles d'EI et les exemples d'apprentissage. Ceci n'empêche pas que le système interactif peut être utilisé dans domaines qui nécessitent des connaissances métier mais une coopération avec un expert de domaine est nécessaire dans ce cas.

Pour pouvoir interpréter le résultat d'un apprentissage de règles, l'utilisateur doit pouvoir comprendre les règles inférées. Ces règles doivent également être peu nombreuses et donc avoir des expressions assez génériques pour pouvoir être facilement consultées par l'utilisateur. Pour pouvoir suivre l'évolution de l'ensemble des règles construit au fur et à mesure que le système avance dans la tâche d'EI dont il se charge, les règles inférées ou écrites ne doivent pas changer complètement d'une itération à une autre et assurer une certaine stabilité à l'ensemble total des règles.

Concernant les exemples d'apprentissage, le système interactif d'apprentissage de règles doit contenir un module de sélection d'exemples. Ce module permet à l'utilisateur à la fois de chercher des exemples à travers l'écriture de règles prospectives et d'annoter des exemples proposés par un module intelligent qui permettrait de réduire au maximum le nombre d'exemples à annoter et favoriser ainsi une convergence rapide de l'apprentissage.

Pour mener à bien une tâche d'EI, l'utilisateur ne doit pas attendre longtemps la réponse de l'algorithme d'apprentissage à chaque fois qu'il le lance. Le temps d'apprentissage peut être réduit de diffrentes manières à condition de ne pas dégrader les performances du système.

Les deux corpus présentés dans ce chapitre permettront d'évaluer les différents modules proposés dans ce travail.

Nous présentons dans le chapitre suivant nos différentes propositions pour mettre en place l'approche interactive d'apprentissage de règles.

Chapitre 6

Proposition : une approche interactive d'apprentissage de règles d'extraction d'information

Nous proposons dans ce chapitre une méthodologie permettant une extraction interactive d'informations dans des documents textuels. La particularité de l'approche présentée est de combiner une approche d'extraction d'information (EI) à base de règles écrites manuellement à une approche à base d'apprentissage artificiel, permettant ainsi d'assister l'utilisateur à travers un processus interactif et itératif où il est possible de travailler de manière duale sur les règles d'EI ainsi que sur des exemples d'apprentissage.

6.1 Processus hybride, interactif et itératif

Comme nous l'avons mentionné dans le chapitre 2, il existe deux principales familles d'approches d'EI à base de règles : les approches à base de règles écrites manuellement et les approches à base d'apprentissage artificiel. Les deux familles d'approches nécessitent un effort manuel non négligeable que ce soit pour écrire les règles ou pour annoter les exemples d'apprentissage. Afin de réduire au maximum cet effort manuel, nous proposons une méthode hybride, interactive et itérative qui permet à l'utilisateur ou à l'ingénieur des connaissances de communiquer avec le module d'apprentissage de règles dans le but d'arriver plus rapidement à effectuer la tâche d'EI cible. La figure 6.1 illustre cette méthode.

Le caractère hybride de l'approche que nous proposons est dû à la combinaison des deux approches d'EI à base de règles. Cette combinaison peut être intéressante dans la mesure où elle donne à l'utilisateur la possibilité de choisir l'opération qu'il juge nécessaire entre l'annotation d'exemples d'apprentissage et l'écriture de règles. Néanmoins, le choix de l'opération à effectuer ne peut être judicieux que s'il prend en compte certains critères comme la réponse du module d'apprentissage. L'intérêt de l'hybridité de la méthode ne peut, par conséquent, avoir du sens que dans un processus interactif où l'utilisateur peut communiquer avec le module d'apprentissage afin de choisir la bonne opération à effectuer. Cette communication peut également nécessiter plusieurs interactions pour mener à bien la tâche à effectuer,

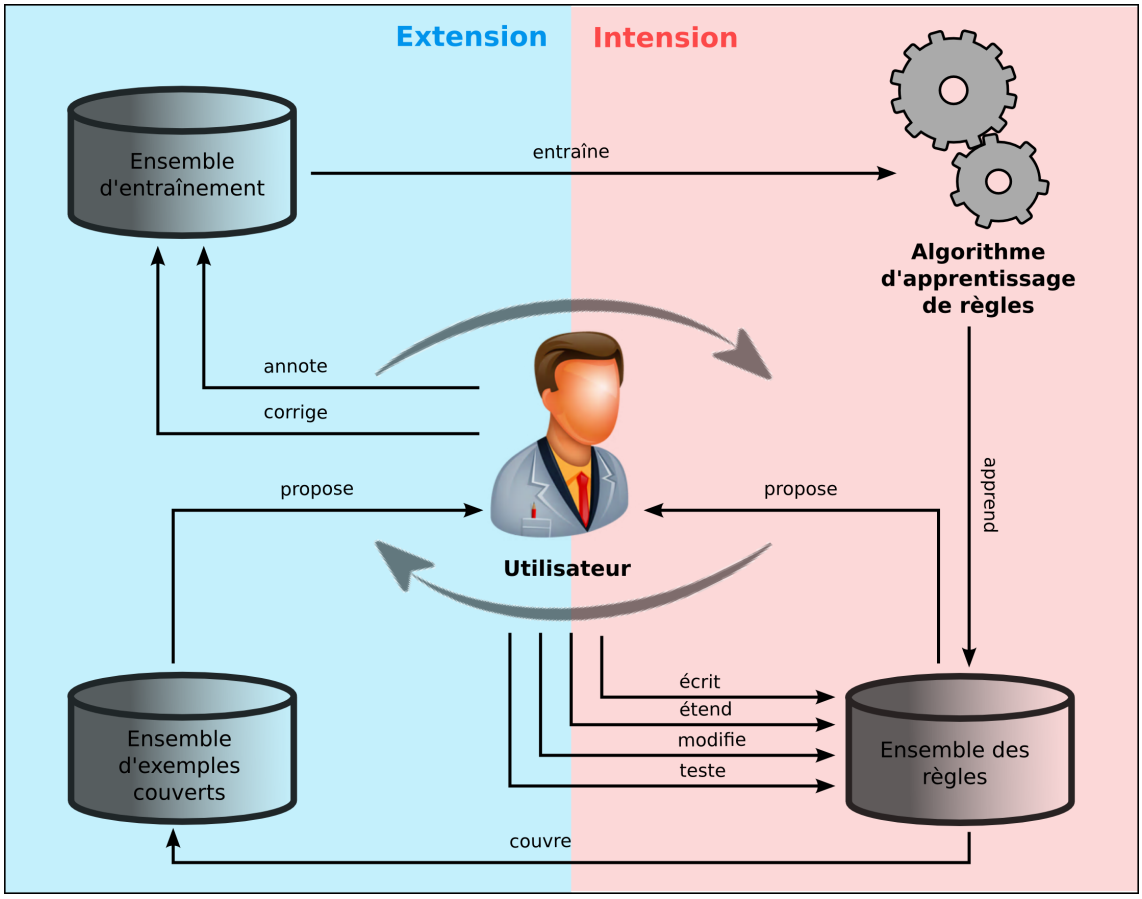


Figure 6.1 – Processus hybride, interactif et itératif d'apprentissage de règles d'extraction d'information.

d'où la nécessité de l'itérativité de l'approche.

Nous appelons interaction l'intervention de l'utilisateur pour effectuer l'une des opérations élémentaires suivantes :

— écrire des règles pour chercher des exemples dans le texte ;
— écrire des règles ou modifier des règles existantes (écrites ou inférées) dans l'ensemble des règles ;
— tester des règles ;
— annoter des exemples.

Ces opérations élémentaires s'inscrivent dans des scénarios plus complexes que nous appelons itérations et qui ont tous pour but de fournir un ensemble d'exemples d'apprentissage au module d'apprentissage permettant d'inférer de bonnes règles. Une itération peut donc contenir plusieurs interactions mais contient exactement une application du module d'apprentissage. Elle commence par un apprentissage automatique de règles à partir de l'ensemble d'entraînement à disposition et finit par un remaniement de l'ensemble d'entraînement pour préparer l'itération suivante. La première itération constitue une exception dans la mesure où elle commence par la préparation d'un premier ensemble d'entraînement.

Il est à noter que même si l'utilisateur peut introduire de nouvelles règles ou modifier des règles inférées par le module d'apprentissage, et à moins d'être sûr que les règles écrites ou modifiées ne font pas d'erreurs et couvrent les exemples sans ambiguité, il est préférable d'écrire manuellement des règles uniquement dans le but de chercher des exemples qui seront utilisés pour inférer un nouvel ensemble de règles plus cohérent. En effet, l'utilisateur ne peut pas garantir les performances des règles qu'il écrit manuellement mais peut valider ou invalider les exemples qu'elles couvrent et donc fournir un nouvel ensemble d'exemples d'apprentissage.

Voici 4 scénarios possibles qui peuvent se dérouler dans le cadre d'une itération du processus proposé.

Scénario 1 : ce scénario peut se dérouler dans le cadre d'une première itération du processus global. L'utilisateur commence par annoter un ensemble d'exemples d'apprentissage qui sera utilisé par le module d'apprentissage pour inférer un ensemble de règles et le proposer à l'utilisateur. Ce scénario est décrit par la figure 6.2.

Scénario 2 : l'utilisateur examine les exemples couverts par les règles inférées par le module d'apprentissage, étend l'ensemble d'entraînement en conséquence (en validant et invalidant des exemples couverts) et relance l'apprentissage sur l'ensemble total des exemples d'apprentissage. Ce scénario est décrit par la figure 6.3.

Scénario 3 : l'utilisateur écrit des règles prospectives pour chercher des exemples en s'appuyant sur des mots clés pertinents. Il les teste sur son corpus, annote les exemples qu'elles couvrent et relance le module d'apprentissage sur l'ensemble total des exemples d'apprentissage. Ce scénario est décrit par la figure 6.4.

Scénario 4 : en examinant les règles inférées par le module d'apprentissage, l'utilisateur se rend compte que certaines d'entre elles sont trop spécifiques ou trop générales. Il les modifie, examine les exemples qu'elles couvrent, met à jour son ensemble d'entraînement et relance le module d'apprentissage. Ce scénario est décrit par la figure 6.5.

Pour pouvoir interagir facilement avec le module d'apprentissage, l'utilisateur doit avoir une vue constante sur les ensembles suivants.

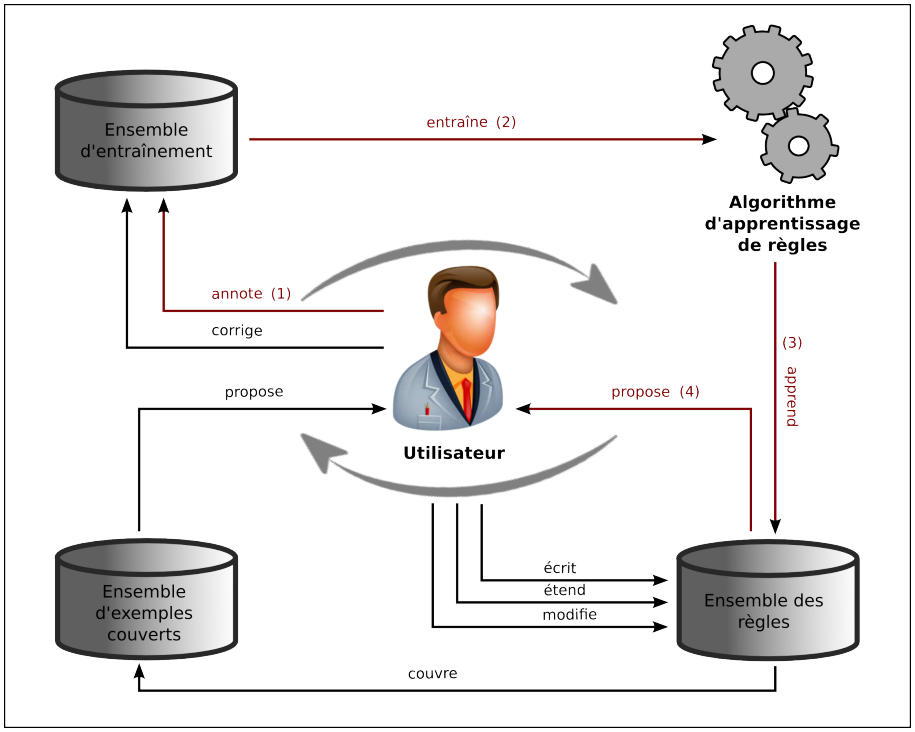


Figure 6.2 – Scénario 1 : première itération.

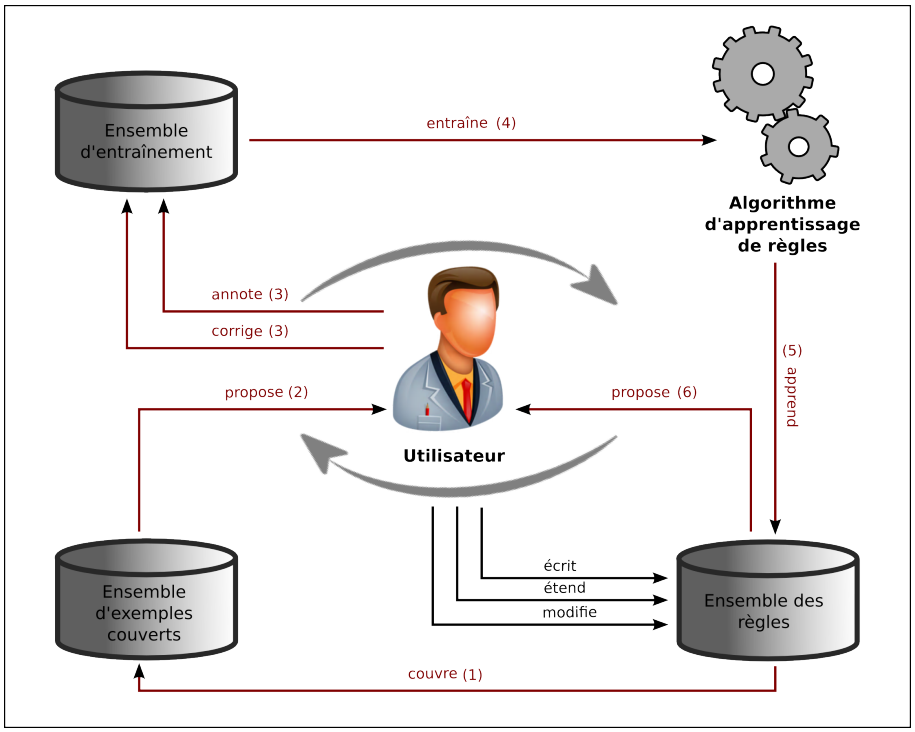


Figure 6.3 – Scénario 2 : annotation des exemples couverts.

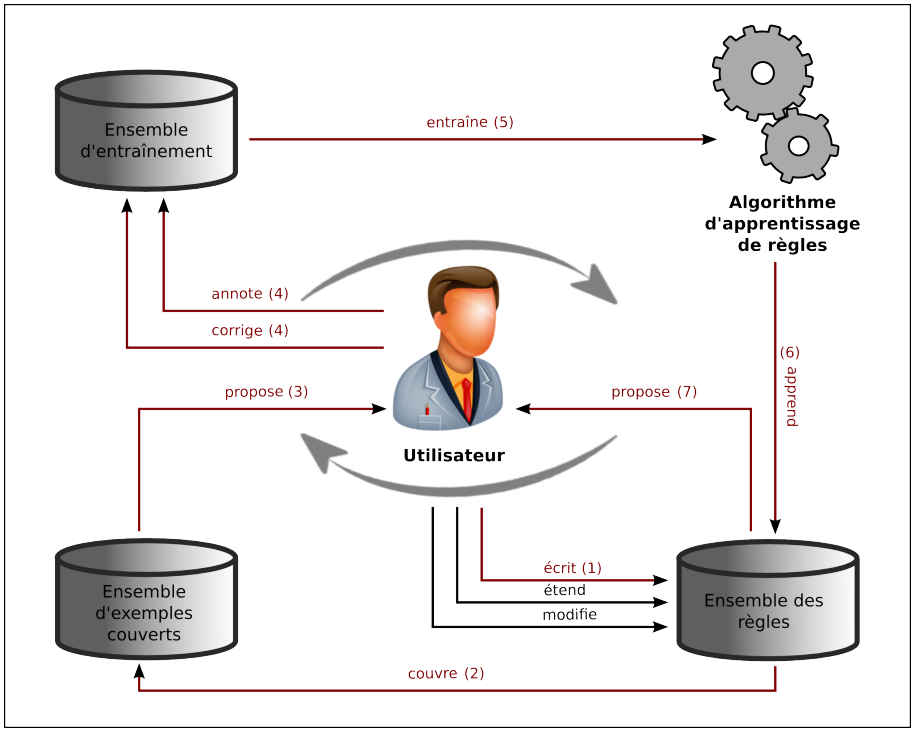


Figure 6.4 – Scénario 3 : prospection et annotation de nouveaux exemples.

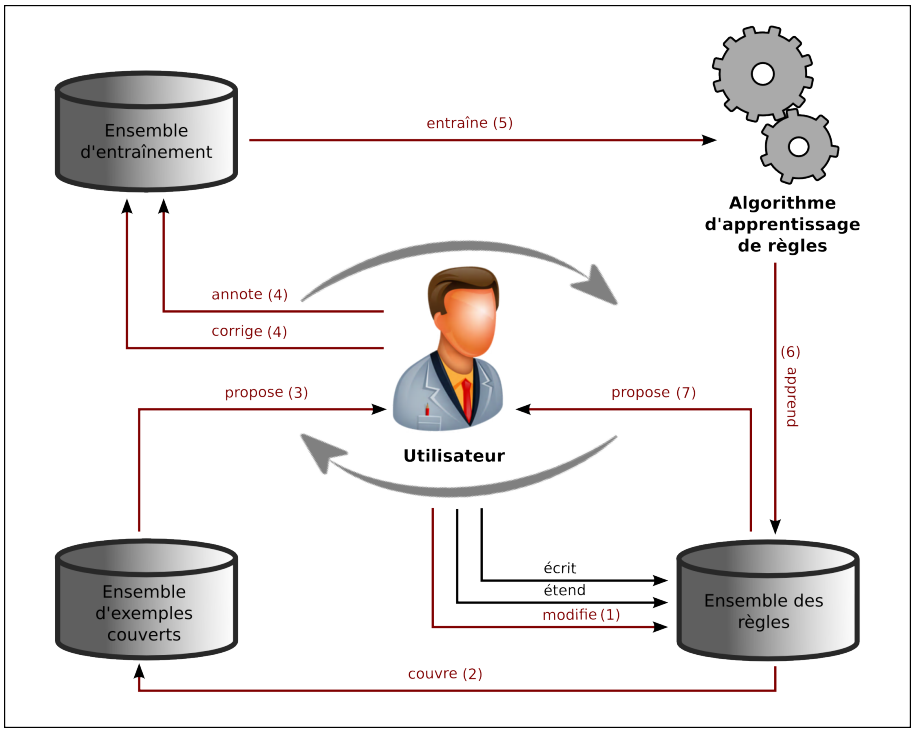


Figure 6.5 – Scénario 4 : travail sur les règles.

— L'ensemble des exemples couverts par l'ensemble des règles : l'utilisateur doit avoir une idée sur la couverture de ses règles pour pouvoir corriger quelques erreurs ou chercher des exemples manquants.
— L'ensemble des exemples annotés non couverts par l'ensemble des règles : l'ensemble des règles inférées par le module d'apprentissage peuvent ne pas couvrir tous les exemples annotés positivement par l'utilisateur. Ce dernier doit avoir une idée sur ces exemples pour pouvoir soit écrire des règles qui les couvrent soit annoter davantage d'exemples qui leur ressemblent. À terme, cet ensemble ne doit contenir que des exemples négatifs.
— L'ensemble des règles écrites et inférées : l'utilisateur doit pouvoir examiner la totalité de l'ensemble des règles à une itération donnée pour pouvoir le modifier ou l'étendre.

La figure 6.6 propose un exemple d'interface permettant de faciliter le processus itératif et interactif.

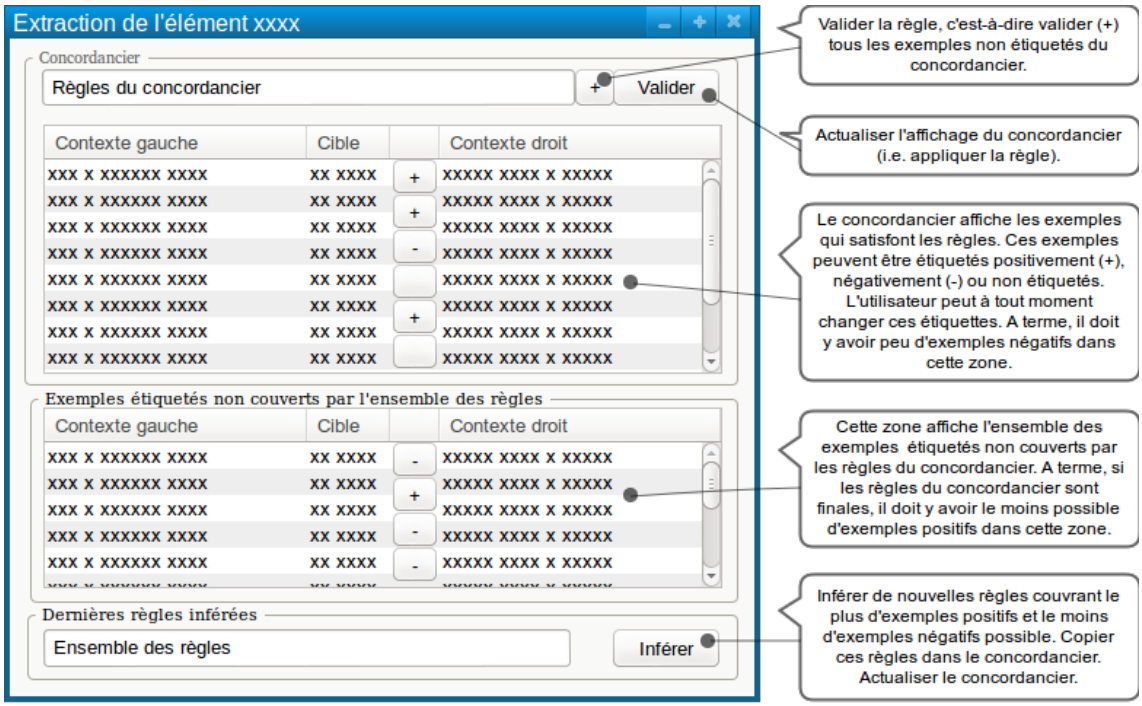


Figure 6.6 – Interface de travail pour la production de règles d'EI de manière interactive et itérative.

Cette interface permet à l'utilisateur d'agir à la fois sur l'intension (les règles elles-mêmes) et sur l'extension (les exemples décrits par les règles) afin d'aboutir à un ensemble de règles d'EI satisfaisant. Ainsi, à chaque itération, l'utilisateur ou l'ingénieur des connaissances, conformément au schéma du processus proposé (voir figure 6.1), a la possibilité :

— d'écrire ou de modifier manuellement les règles d'EI ;
— d'étiqueter positivement (+) ou négativement (-) des portions de textes qui constituent respectivement une bonne illustration de l'élément que l'on cherche

à décrire par les règles d'EI (étiquette +), ou au contraire un contre-exemple (étiquette -), un exemple non étiqueté étant un exemple dont le statut est indéterminé.

A chaque itération, un nouvel ensemble de règles est automatiquement inféré en utilisant des techniques d'apprentissage à partir d'exemples.

Pour pouvoir mettre en place le processus hybride, interactif et itératif proposé, nous devons d'abord résoudre les points suivants :

— Pour pouvoir communiquer avec le module d'apprentissage de règles, l'utilisateur doit pouvoir comprendre les règles inférées par ce dernier. Ces règles ne doivent également pas être trop nombreuses pour que l'utilisateur puisse les consulter dans un temps raisonnable.

— Dans un système interactif d'apprentissage de règles, les exemples d'apprentissage sont annotés de manière progressive. À une itération donnée, les règles construites ne doivent donc pas considérer tous les exemples non encore annotés comme négatifs car certains peuvent être positifs.

— Une certaine stabilité doit être assurée à l'ensemble des règles. En effet, les règles écrites par l'utilisateur ou inférées par le module d'apprentissage de règles ne doivent pas subir de grosses modifications d'une itération à une autre afin de pouvoir suivre l'évolution de l'ensemble des règles.

— Dans un système interactif d'apprentissage de règles, l'utilisateur est amené à annoter les exemples d'apprentissage de manière progressive. Ces exemples sont parfois évidents à trouver et parfois le sont beaucoup moins. La mise en place d'un module qui aide l'utilisateur à trouver les bons exemples à annoter s'avère indispensable dans un système où on cherche à réduire au maximum l'effort humain tout en cherchant à tirer profit de son expertise.

Pour résoudre ces contraintes, nous avons proposé des méthodes et des stratégies qui tiennent compte des propriétés clés d'un système interactif énoncées dans la section 5.2.

6.2 Stratégies proposées pour la compréhensibilité et la généricité des règles

Nous pensons que des règles d'EI exprimées dans un langage de règles accessible et s'appuyant sur des annotations textuelles riches et génériques permettent de construire des règles compréhensibles et génériques.

6.2.1 Une chaîne d'annotation du texte

Les règles d'EI sont d'autant plus générales et concises qu'elles peuvent s'appuyer sur des niveaux d'annotation complexes (étiquetage morpho-syntaxique, entités nommées, termes, analyse syntaxique, etc.). C'est la raison pour laquelle il est important de traiter préalablement le corpus en utilisant une plateforme d'annotation linguistique avant de travailler sur les règles d'EI.

La chaîne d'annotation du texte que nous proposons dans ce travail (voir figure 6.7) se compose de 3 modules :

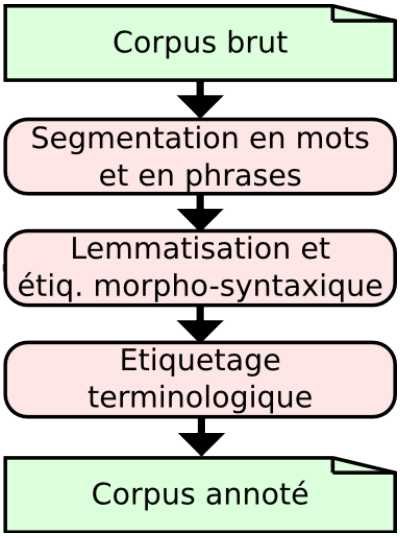


Figure 6.7 – Chaine d'annotation du texte proposée.

— un module de segmentation en mots et en phrases qui permet de découper le texte en mots et phrases sachant que le mot est la plus petite unité linguistique considérée dans le texte ;
— un module de lemmatisation et d'étiquetage morpho-syntaxique qui repose sur le module précédent et permet d'attribuer à chaque mot du texte un lemme (la forme canonique du mot) et une étiquette morpho-syntaxique qui contient des informations sur les parties du discours (nom, verbe, adjectif, adverbe ...) ;
— un module d'étiquetage terminologique qui permet d'identifier des termes candidats dans le texte en s'appuyant sur un extracteur de termes.

Une fois le texte annoté, nous pouvons faire en sorte que les règles d'EI construites ou inférées à partir du texte se basent dans leur expression sur les différentes annotations posées (lemme, étiquette morpho-syntaxique, terme, etc.). Cependant, la compréhensibilité de ces règles dépend beaucoup de l'expressivité du langage utilisé pour écrire ces règles ou les inférer et de sa manière d'exploiter les différentes annotations posées dans le texte.

6.2.2 Un langage de règles expressif

Une étude des différents langages de règles utilisés dans la littérature (voir chapitre 2) a révélé que les langages les plus récents JAPE et Ruta sont parmi les plus génériques, les plus expressifs et les plus complets. Selon les expériences de certains utilisateurs, Ruta est plus complet que JAPE. Une description plus détaillée de ce langage figure dans la section 4.3.

Nous avons choisi d'utiliser, dans ce travail, le langage Ruta et nous montrons dans cette section comment écrire des règles dans ce langage.

L'expression des règles dans le langage Ruta dépend des mentions des annotations avec lesquelles le texte est annoté. Appliquons la chaine d'annotation décrite dans la section précédente sur notre texte d'entrée. Nous supposons que les annotations

résultantes correspondent aux mentions suivantes.

— `Token` : il s'agit du type de base. Il est attribué à chaque mot du texte et possède deux propriétés : le lemme (`lemma`) et l'étiquette morpho-syntaxique (`postag`).

— `Term` : ce type ou cette annotation est attribuée à chaque segment de texte reconnu comme étant un terme par un extracteur de termes. Elle possède les mêmes propriétés qu'un `Token`, à savoir un lemme (`termLemma`) et une étiquette morpho-syntaxique (`termPostag`).

Tenant compte de toutes ces informations et en supposant que tous les caractères blancs sont filtrés, les règles suivantes peuvent être utilisées pour reconnaître, par exemple, le terme *human body* :

```
1-Token{REGEXP("human")} Token{REGEXP("body")}->MARKONCE(HumanBody,1,2)};
2-Token{FEATURE("lemma","human")} Token{FEATURE("lemma","body")
 ->MARKONCE(HumanBody,1,2)};
3-Term{REGEXP("human body")} ->MARKONCE(HumanBody)};
4-Term{{FEATURE("termLemma","human body")} ->MARKONCE(HumanBody)};
```

La première règle se lit de la manière suivante : si on trouve un mot dont l'expression exacte est *human* suivi d'un mot dont l'expression exacte est *body*, on crée une annotation de type `HumanBody` qui couvre les deux mots.

La deuxième règle est plus générique que la première dans la mesure où elle permet de reconnaître toutes les variations du terme *human body* (*human body*, *Human body*, *human bodies*, *Human bodies*).

La troisième règle n'est pas plus générique que la première règles car elle permet de reconnaître les mêmes expressions (on suppose bien évidemment que toutes les expressions *human body* sont reconnues comme étant des termes). Cependant, l'exécution de cette règle est plus rapide car étant donné que les termes sont moins nombreux que les mots dans un texte, un parcours des termes à la recherche d'un terme spécifique est plus rapide qu'un parcours de tous les mots du texte. Les termes peuvent aussi contenir des unités composées comme le terme *human body*. Au lieu de faire un parcours à la recherche de chaque mot composant le terme, un seul parcours des termes peut être fait à la recherche de l'expression complexe du terme.

La quatrième règle est la plus générique car elle réunit les avantages de la deuxième et de la troisième règle et permet, en plus, de reconnaître, par exemple, des acronymes du terme *human body* reconnus par l'extracteur de termes.

Indépendamment de la généricité, les quatre règles sont claires, lisibles et compréhensibles par un utilisateur. En effet, pour pouvoir interpréter ces règles, il suffit de connaître des notions de base sur la syntaxe du langage Ruta et bien évidemment les mentions des annotations avec lesquelles le texte est annoté.

6.3 Choix d'un algorithme d'apprentissage de règles : WHISK basé sur Ruta ($WHISK_R$)

Dans un système d'EI à base de règles, les règles d'EI sont généralement écrites par un expert du domaine. Ces règles peuvent être également apprises en utilisant

des algorithmes d'apprentissage à base de règles. Toutefois, nous avons remarqué que les langages utilisés par les humains pour écrire leurs règles ne sont souvent pas les mêmes que ceux utilisés par les algorithmes d'apprentissage. Ceci constitue un handicap quand il s'agit de mixer les deux modes de production de règles dans un seul système : soit l'humain doit comprendre le langage utilisé par l'algorithme d'apprentissage et ce n'est pas la moindre des tâches car ces langages sont souvent mal documentés, soit c'est à l'algorithme d'apprentissage d'utiliser un langage de règles standard et des efforts ont commencé à être déployés dans ce sens.

Le système TextRuler (Kluegl, Atzmueller, Hermann, & Puppe, 2009), par exemple, est un système de développement semi-automatique d'applications d'EI à base de règles qui contient des implémentations de certains algorithmes comme $(LP)^2$ (Ciravegna, 2001, 2003), WHISK (S. Soderland et al., 1999), RAPIER (Califf & Mooney, 2003), BWI (Freitag & Kushmerick, 2000) et WIEN (Kushmerick et al., 1997) adaptés au langage Ruta. Même si les implémentations, dans leur état actuel, contiennent des erreurs, sont non optimisées et parfois incomplètes, elles peuvent être étendues de manière à devenir intéressantes.

Kluegl, Atzmueller, Hermann, et Puppe (2009) ont évalué les trois algorithmes $(LP)^2{}_R$, $WHISK_R$ et $RAPIER_R$ sur une tâche d'extraction des gros titres pour les diagnostics, les thérapies et les examens dans des lettres de décharges médicales. Ces algorithmes ont été comparés sur la base des critères suivants.

La compréhensibilité des règles – La compréhensibilité des règles apprises est essentielle pour l'introspection, la sélection et l'écriture de nouvelles règles.

L'extensibilité des règles – L'ingénieur de la connaissance doit être en mesure d'étendre et d'optimiser les règles proposées en adaptant et en généralisant les éléments de langage et leurs propriétés utilisées. Un transfert des règles à d'autres domaines et la possibilité de les améliorer par un humain sont notés.

L'intégrabilité des règles – Les règles apprises doivent pouvoir s'intégrer dans l'ensemble des règles existant. Ce critère mesure la simplicité de l'intégration et la transférabilité des règles.

L'utilisation des propriétés – L'utilisation des propriétés attribuées aux différentes unités lexicales est d'un grand intérêt quand on cherche à produire des règles performantes.

Le résultat et les performances – Le temps d'apprentissage et le nombre des règles apprises doivent être raisonnables. La vitesse d'extraction et la précision des règles dans le domaine cible sont également notées.

Les résultats de cette comparaison figurent dans la table 6.1.

En regardant les trois premiers critères, nous pouvons remarquer que $WHISK_R$ dépasse les deux autres algorithmes. Autrement dit, les règles qu'il produit sont beaucoup plus compréhensibles et extensibles que celles produites par les deux autres algorithmes. Cependant, le fait qu'il n'exploite pas les propriétés des annotations fait qu'il obtient des résultats qui ne sont pas bons comparés à ceux obtenus par $(LP)^2{}_R$ qui exploite les propriétés des annotations. En examinant de près l'implémentation de l'algorithme $WHISK_R$, nous avons découvert que cette dernière peut être améliorée de manière à permettre d'exploiter les propriétés des annotations. D'un autre

	$(LP)^2_R$	$WHISK_R$	$RAPIER_R$
Compréhensibilité	6	**8**	6
Extensibilité	6	**7**	5
Intégrabilité	6	**8**	7
Utilisation des propriétés	9	2	2
Résultat, performances	9	3	1
Total	36	28	21

Table 6.1 – Evaluation qualitative de $(LP)^2_R$, $WHISK_R$ et $RAPIER_R$ dans (Kluegl, Atzmueller, Hermann, & Puppe, 2009). (1=faible, 10=très bien).

coté, $(LP)^2_R$ infère trois types de règles : les règles d'annotation, les règles contextuelles et les règles de correction. Les règles d'annotation sont elles-mêmes divisées en règles de délimitation gauche et des règles de délimitation droite et elles sont totalement indépendantes. Ceci constitue un inconvénient majeur dans un système qui interagit avec l'utilisateur car ce dernier devrait manipuler tous ces types de règles. $WHISK_R$ (WHISK basé sur Ruta) produit, en revanche, un seul type de règles, ce qui en fait un bon candidat dans un système interactif d'apprentissage de règles d'EI.

Pour toutes ces raisons, nous avons choisi de travailler avec l'algorithme $WHISK_R$.

Une description détaillée de l'algorithme WHISK et de son implémentation dans TextRuler ($WHISK_R$) figurent respectivement dans les sections 2.6 et 4.4.1. Nous utiliserons, dans ce travail, notre propre version étendue de cet algorithme détaillée dans la section 7.3.2.

6.4 Stratégies proposées pour la construction de l'ensemble des règles

6.4.1 Réduction de l'espace de recherche de règles : réduction de la fenêtre de mots autour du terme cible à la phrase

Afin de sélectionner la règle la plus performante, un algorithme d'apprentissage de règles doit d'abord tester et comparer un ensemble de règles qui constitue son espace de recherche de règles. Nous avons remarqué que la plupart des algorithmes d'apprentissage de règles d'EI comme $(LP)^2$ et WHISK considèrent une fenêtre de mots autour du terme cible comme contexte d'apprentissage. Ce contexte est fixé à l'avance par l'utilisateur. Pour l'algorithme $WHISK_R$ par exemple, ce contexte est fixé par défaut à 5. Autrement dit, l'algorithme considère les 5 mots qui précèdent et les 5 mots qui suivent le terme cible lors de la création de l'ensemble des règles à tester.

Prenons un exemple concret. Considérons l'extrait de texte de la figure 6.8.

En supposant que la règle la plus générale retenue par l'algorithme $WHISK_R$ étendu avec la lemmatisation et l'étiquetage morpho-syntaxique (voir section 7.3.2)

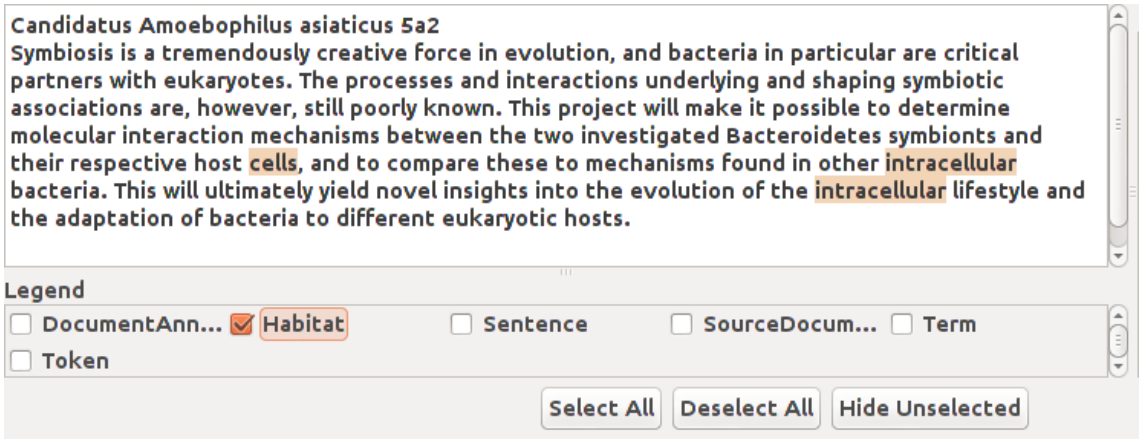


Figure 6.8 – Extrait du corpus BB BioNLP-ST 2013.

est

```
Token{->MARKONCE(Habitat)};
```

les règles testées qui étendent cette règle pour essayer d'extraire le premier terme *intracellular* sont représentées dans les figures 6.9, 6.10 et 6.11. Pour des questions de simplicité, nous ne considérons pas dans cet exemple les règles qui s'appuient sur les termes mais uniquement celles qui s'appuient sur les Tokens. L'ensemble des règles testées peut être divisé en trois sous ensembles : l'ensemble des règles qui décrivent la cible (figure 6.9), l'ensemble des règles qui décrivent le contexte gauche de la cible (figure 6.10) et l'ensemble des règles qui décrivent le contexte droit de la cible (figure 6.11).

```
- Token{REGEXP("intracellular")->MARKONCE(Habitat)};
- Token{FEATURE("lemma", "intracellular")->MARKONCE(Habitat)};
- Token{FEATURE("postag", "JJ")->MARKONCE(Habitat)};
```

Figure 6.9 – Règles qui décrivent la cible.

Supposons que a est le nombre d'attributs ou propriétés (expression exacte, lemme, étiquette morpho-syntaxique) à considérer pour un mot du texte et c est le nombre de mots à considérer à gauche et à droite du terme cible. Dans notre exemple, a vaut 3 car nous évaluons à la fois les règles qui s'appuient sur les expressions exactes des mots, celles qui s'appuient sur leurs lemmes et celles qui s'appuient sur leurs étiquettes morpho-syntaxiques et c vaut 5 car l'algorithme WHISK_R considère, par défaut, 5 mots de part et d'autre du terme cible.

Le terme cible étant composé d'un seul mot dans notre exemple, il y a donc 3 (a) règles qui le décrivent. Concernant les règles de contexte gauche et les règles de contexte droit, en plus des règles les plus génériques (au nombre de 2), il y a a règles pour chaque mot de contexte (c mots). Il y a donc $2 + a * c = 17$ règles à tester pour chacun des contextes gauche et droit.

```
- Token # Token{->MARKONCE(Habitat)};
- Token{REGEXP("to")} # Token{->MARKONCE(Habitat)};
- Token{FEATURE("lemma", "to")} # Token{->MARKONCE(Habitat)};
- Token{FEATURE("postag", "TO")} # Token{->MARKONCE(Habitat)};
- Token{REGEXP("mechanisms")} # Token{->MARKONCE(Habitat)};
- Token{FEATURE("lemma", "mechanism")} # Token{->MARKONCE(Habitat)};
- Token{FEATURE("postag", "NN")} # Token{->MARKONCE(Habitat)};
- Token{REGEXP("find")} # Token{->MARKONCE(Habitat)};
- Token{FEATURE("lemma", "find")} # Token{->MARKONCE(Habitat)};
- Token{FEATURE("postag", "VBP")} # Token{->MARKONCE(Habitat)};
- Token{REGEXP("in")} # Token{->MARKONCE(Habitat)};
- Token{FEATURE("lemma", "in")} # Token{->MARKONCE(Habitat)};
- Token{FEATURE("postag", "IN")} # Token{->MARKONCE(Habitat)};
- Token Token{->MARKONCE(Habitat)};
- Token{REGEXP("other")} Token{->MARKONCE(Habitat)};
- Token{FEATURE("lemma", "other")} Token{->MARKONCE(Habitat)};
- Token{FEATURE("postag", "JJ")} Token{->MARKONCE(Habitat)};
```

Figure 6.10 – Règles qui décrivent le contexte gauche de la cible.

```
- Token{->MARKONCE(Habitat)} Token;
- Token{->MARKONCE(Habitat)} Token{REGEXP("bacteria")};
- Token{->MARKONCE(Habitat)} Token{FEATURE("lemma", "bacterium")};
- Token{->MARKONCE(Habitat)} Token{FEATURE("postag", "NN")};
- Token{->MARKONCE(Habitat)} # Token;
- Token{->MARKONCE(Habitat)} # Token{REGEXP(".")};
- Token{->MARKONCE(Habitat)} # Token{FEATURE("lemma", ".")};
- Token{->MARKONCE(Habitat)} # Token{FEATURE("postag", ".")};
- Token{->MARKONCE(Habitat)} # Token{REGEXP("This")};
- Token{->MARKONCE(Habitat)} # Token{FEATURE("lemma", "this")};
- Token{->MARKONCE(Habitat)} # Token{FEATURE("postag", "DT")};
- Token{->MARKONCE(Habitat)} # Token{REGEXP("will")};
- Token{->MARKONCE(Habitat)} # Token{FEATURE("lemma", "will")};
- Token{->MARKONCE(Habitat)} # Token{FEATURE("postag", "MD")};
- Token{->MARKONCE(Habitat)} # Token{REGEXP("ultimately")};
- Token{->MARKONCE(Habitat)} # Token{FEATURE("lemma", "ultimate")};
- Token{->MARKONCE(Habitat)} # Token{FEATURE("postag", "RB")};
```

Figure 6.11 – Règles qui décrivent le contexte droit de la cible.

Lors de la construction de son espace de règles, l'algorithme WHISK_R considère une fenêtre de mots autour du terme cible indépendamment des limites de la phrase. La question que nous nous sommes donc posée est : est-il vraiment utile de sortir des limites de la phrase ? Plusieurs travaux dans la littérature se sont intéressés à la taille optimale de la fenêtre de contexte à considérer quand on cherche à désambiguïser un mot polysémique ou à mesurer une similarité distributionnelle. Plusieurs d'entre eux encouragent l'utilisation de contextes locaux courts (1 à 3 mots de part et d'autre du terme ou mot cible) (Choueka & Lusignan, 1985 ; Yarowsky, 1993) et certains recommandent de ne pas sortir de la phrase (Audibert, 2003). Nous trouvons plus prudent d'arrêter le contexte aux frontières de la phrase car les corpus de travail peuvent être construits de manière à supprimer ou déplacer certaines parties. Dans ce cas, sortir des limites des phrases peut impliquer l'utilisation d'informations de voisinage entre des phrases qui ne sont pas voisines à la base.

Nous décidons, par conséquent, d'utiliser les mots de la phrase apparaissant de part et d'autre du terme cible comme contexte à condition de ne pas dépasser 5 mots de chaque côté. Quand les phrases sont longues, l'algorithme d'apprentissage aura au pire le même nombre de règles à tester. Sur notre exemple, l'ensemble des règles de contexte droit se réduirait à l'ensemble présenté dans la figure 6.12.

```
- Token{->MARKONCE(Habitat)} Token;
- Token{->MARKONCE(Habitat)} Token{REGEXP("bacteria")};
- Token{->MARKONCE(Habitat)} Token{FEATURE("lemma", "bacterium")};
- Token{->MARKONCE(Habitat)} Token{FEATURE("postag", "NN")};
- Token{->MARKONCE(Habitat)} # Token;
- Token{->MARKONCE(Habitat)} # Token{REGEXP(".")};
- Token{->MARKONCE(Habitat)} # Token{FEATURE("lemma", ".")};
- Token{->MARKONCE(Habitat)} # Token{FEATURE("postag", ".")};
```

Figure 6.12 – Règles qui décrivent le contexte droit de la cible sans sortir des limites de la phrase.

3 mots ont été supprimés du contexte droit du terme cible. Par conséquent, 9 $(3 * a)$ règles ont été supprimées en total.

L'expérience décrite dans la section 8.3 montre que la réduction de la fenêtre de contexte autour du terme cible à la phrase permet de réduire le temps d'apprentissage.

6.4.2 Apprentissage sur un corpus réduit

La construction de l'ensemble des règles pour une tâche d'EI dans un système interactif d'apprentissage de règles est un travail collaboratif entre l'utilisateur et le module d'apprentissage. Les règles que l'utilisateur introduit dans l'ensemble des règles sont généralement des règles qu'il a pris soin de tester sur tout le corpus. Les règles inférées de manière automatique, en revanche, ignorent le statut des exemples non encore annotés par l'utilisateur à une itération donnée et posent, par conséquent, beaucoup de problèmes :

— Elles sont très spécifiques et tendent à faire du surapprentissage. En effet, les règles générales faisant partie de l'espace de recherche de règles de l'algorithme d'apprentissage font beaucoup d'erreurs quand elles sont évaluées sur l'ensemble du corpus car elles considèrent comme négatifs tous les exemples positifs non annotés par l'utilisateur. Elles sont donc spécifiées jusqu'à coller aux exemples d'apprentissage. Nous remarquons également que parfois même les versions spécifiques des règles font beaucoup d'erreurs et ne peuvent être gardées, ce qui fait que certains exemples annotés par l'utilisateur se retrouvent sans règles qui les couvrent.
— Elles ralentissent le travail d'annotation d'exemples par l'utilisateur car elles ont un faible rappel. En effet, ces règles couvrent généralement uniquement les exemples à partir desquels elles ont été construites. Elles ne couvrent donc pas de nouveaux exemples que l'utilisateur peut consulter et annoter.
— Elles perturbent l'utilisateur car elles sont nombreuses et sont vouées à être remplacées par un petit nombre de règles plus génériques dans l'ensemble final des règles quand l'utilisateur aurait annoté beaucoup plus d'exemples.

Prenons un exemple. Le terme « human » dans le corpus d'apprentissage BB BioNLP-ST 2013 (dans le cadre d'une validation croisée 10 fois, il s'agit de 9/10 du corpus (ensemble d'entraînement et ensemble de développement)) est considéré comme un habitat de bactéries dans 95 cas sur 98. Pour couvrir ces 95 exemples positifs, l'algorithme WHISK$_{\mathrm{R}}$ étendu avec la lemmatisation et l'étiquetage morpho-syntaxique (voir section 7.3.2) construit la règle générique :

```
Token{FEATURE("lemma", "human")->MARKONCE(Habitat)};
```

Supposons qu'à une itération donnée du processus interactif et itératif d'apprentissage de règles d'EI, l'algorithme d'apprentissage ne dispose, pour le terme *human*, que des exemples annotés dans l'extrait de corpus représenté dans la figure 6.13.

Les règles que l'algorithme infère sont les suivantes :

```
1-Token{FEATURE("lemma","pneumonia")} Token{FEATURE("lemma","infection")}
  # Token{FEATURE("lemma", "human")->MARKONCE(Habitat)};
2-Token{FEATURE("lemma","important")} Token{FEATURE("lemma","human")
  ->MARKONCE(Habitat)};
```

Ces règles couvrent uniquement les deux derniers exemples. Nous obtenons donc deux règles spécifiques qui couvrent exactement 2 exemples à la place d'une règle générique qui couvre 95 exemples. L'algorithme d'apprentissage n'arrive pas à trouver des règles fiables qui couvrent les trois autres exemples qui se retrouvent donc non couverts.

Pour résoudre ce problème, nous proposons que l'évaluation des règles faisant partie de l'espace de recherche de règles de l'algorithme d'apprentissage se fasse uniquement sur les phrases contenant des exemples annotés positivement par l'utilisateur. Nous suggérons donc que l'apprentissage se fasse, en quelque sorte, sur un corpus réduit composé uniquement de ces phrases. Si ces phrases contiennent des exemples positifs non encore annotés par l'utilisateur, ils seront ignorés.

La réduction du contexte autour du terme cible à la phrase s'accorde bien avec le fait d'apprendre sur un corpus réduit dans la mesure où, dans ce dernier, les phrases

Figure 6.13 – Extrait du corpus BB BioNLP-ST 2013.

qui avoisinent la phrase qui contient le terme cible ne l'avoisinent pas forcément dans le corpus d'origine.

Reprenons l'exemple précédent et essayons, cette fois, d'apprendre uniquement sur les phrases qui contiennent des exemples annotés positivement par l'utilisateur. Voici les règles construites par l'algorithme d'apprentissage pour couvrir les exemples de l'extrait textuel représenté dans la figure 6.8.

```
Token{FEATURE("lemma", "human")->MARKONCE(Habitat)};
```

En apprenant sur un corpus réduit composé uniquement des phrases contenant des exemples annotés positivement par l'utilisateur, l'algorithme a inféré la même règle que celle obtenue en apprenant sur l'ensemble du corpus quand tous les exemples sont annotés.

6.4.3 Effet sur les performances et le temps d'apprentissage

La réduction de l'espace de recherche de règles et par la suite l'apprentissage sur un corpus réduit permettent certes de ne pas considérer tous les exemples non encore annotés par l'utilisateur comme négatifs, mais quel effet ont ces stratégies sur les performances et sur le temps d'apprentissage ?

Comme nous l'avons mentionné dans la section 5.2.5, le temps d'apprentissage est un facteur important dans un système interactif où l'utilisateur a besoin d'une réponse relativement rapide. Ce facteur dépend, en effet, de trois paramètres :

— la rapidité du moteur d'exécution des règles ;
— la taille de l'espace de recherche de règles de l'algorithme d'apprentissage de règles ;
— la taille du corpus d'apprentissage.

L'exécution des règles étant non transparente à l'utilisateur, pour réduire le temps d'apprentissage, il suffit de réduire l'espace de recherche des règles de l'algorithme d'apprentissage de règles ou de réduire le corpus d'apprentissage. Pour un algorithme d'apprentissage de règles d'EI, la taille de l'espace de recherche des règles pour l'extraction d'un terme cible dépend de deux facteurs :

— le nombre des annotations et des propriétés des annotations à considérer dans le texte ;
— la taille du contexte d'apprentissage.

Pour le premier facteur, une étude expérimentale qui a pour but de comparer les différentes combinaisons de traits linguistiques (expression exacte, lemme, étiquette morpho-syntaxique) afin de déterminer les combinaisons qui mènent vers l'ensemble de règles le plus performant figure dans la section 8.2.

Des travaux sur l'apprentissage de règles (Ciravegna, 2003 ; Freitag & Kushmerick, 2000) montrent que le temps d'apprentissage augmente de manière exponentielle avec l'augmentation de la taille de la fenêtre de contexte. Si nous nous basons sur ces constatations, nous pouvons dire que la réduction du contexte autour du terme cible à la phrase permettrait de réduire le temps d'apprentissage. D'un autre coté, l'apprentissage sur un corpus réduit implique la réduction du corpus d'apprentissage (l'un des facteurs impactant le temps d'apprentissage) et par conséquent la réduction du temps d'apprentissage.

Nous pensons que l'apprentissage sur un corpus réduit permettra, d'un côté, de baisser légèrement les valeurs de précision des règles et, d'un autre côté, d'augmenter légèrement leurs valeurs de rappel. En effet, les règles devraient être moins précises car elles sont testées uniquement sur un échantillon du corpus où elles ne font pas beaucoup d'erreurs. Elles devraient, en revanche, avoir un meilleur rappel car si les règles les plus génériques ne font pas beaucoup d'erreurs sur l'échantillon consulté du corpus, ce sont les règles que l'algorithme d'apprentissage gardera.

Une étude expérimentale qui évalue la différence entre les trois versions de l'algorithme d'apprentissage figure dans la section 8.3. Elle compare l'ancienne version de l'algorithme d'apprentissage, celle qui considère la phrase comme contexte d'apprentissage et celle qui apprend sur un corpus réduit, en termes de précision, rappel, F-mesure et temps d'apprentissage. Elle montre que les deux dernières versions, et surtout celle qui permet d'apprendre sur un corpus réduit, réduisent de manière significative le temps d'apprentissage sans dégrader la F-mesure.

6.5 Stratégies proposées pour mettre en place la sélection d'exemples

Comme nous l'avons expliqué dans la section 5.2.4, la sélection d'exemples est nécessaire dans un système interactif d'apprentissage de règles d'EI où l'utilisateur a besoin d'annoter des exemples d'apprentissage de manière progressive. Cette sélection d'exemples intervient à deux niveaux du système interactif pour répondre à deux besoins différents :

— chercher des exemples en s'appuyant sur des mots clés dont dispose l'utilisateur ;
— chercher des exemples pertinents à annoter de manière à favoriser une convergence rapide de l'algorithme d'apprentissage de règles.

Pour chacun de ces besoins, une solution peut être adoptée :

— un concordancier pour écrire des règles prospectives ;
— un module d'apprentissage actif pour une sélection intelligente d'exemples.

6.5.1 Un concordancier pour écrire des règles prospectives

Un concordancier est un outil qui permet à un utilisateur d'écrire des règles dans un langage de règles quelconque et d'afficher les exemples qu'elles couvrent ainsi que leurs contextes gauche et droite (par exemple les 100 caractères qui précèdent et qui suivent l'exemple couvert).

Dans le cadre d'un système interactif d'apprentissage de règles d'EI, un concordancier est d'une grande utilité dans la mesure où il permet à l'utilisateur d'écrire des règles prospectives. Une règle prospective est, en effet, une règle qui a uniquement pour but de chercher des exemples dans le texte en se basant sur des mots clés dont dispose l'utilisateur. Il ne s'agit pas d'une règle qui sera gardée dans l'ensemble final des règles mais plutôt d'un moyen pour l'utilisateur de réduire son espace de travail. L'affichage des exemples couverts par une règle prospective permet à l'utilisateur de naviguer dans ces exemples, d'en distinguer les bons des mauvais et de

les annoter. Les règles prospectives permettent à l'utilisateur non seulement de sélectionner des exemples dans le texte mais aussi de le faire de manière rapide car au lieu de parcourir tout le corpus à la recherche de mots clés, des règles prospectives qui s'appuient sur ces mots clés permettent de présenter à l'utilisateur un espace de travail réduit qui représente la couverture de ces règles.

6.5.2 Un module d'apprentissage actif

Dans un système interactif d'apprentissage de règles d'EI, l'utilisateur est amené à fournir, de manière progressive, des exemples d'apprentissage à l'algorithme d'induction de règles afin d'améliorer les performances de son système. Cependant, ce travail d'annotation peut s'avérer fastidieux surtout si les exemples à annoter ne sont pas intuitifs. L'apprentissage actif permet, selon plusieurs études, de réduire le nombre total des exemples que l'utilisateur est amené à annoter pour atteindre les performances optimales du système en proposant des exemples sélectionnés de manière intelligente qui permettrait une convergence plus rapide de l'algorithme d'apprentissage.

Plusieurs méthodes qui permettent de mettre en place l'apprentissage actif en EI existent dans la littérature (voir section 3.3.2). Nous proposons, dans ce travail, deux modules d'apprentissage actif génériques indépendants de tout algorithme d'apprentissage de règles (IAL4Sets (section 6.5.2) et IAL3Sets (section 6.5.2)) qui étendent le module proposé par les concepteurs de l'algorithme WHISK pour rendre ce dernier interactif (S. Soderland et al., 1999) (voir section 2.6). Il est à noter que le module interactif de WHISK n'a pas été implémenté dans l'algorithme WHISK_R. Sachant que ce module est indépendant de WHISK et peut fonctionner avec d'autres algorithmes d'apprentissage de règles, nous l'avons implémenté et utilisé comme Baseline pour évaluer l'apport de notre module d'apprentissage actif dans la section 8.4.

Comme nous l'avons mentionné dans la section 2.6, l'algorithme WHISK alterne le processus d'annotation d'exemples avec le processus d'apprentissage. À chaque itération, il propose à l'utilisateur un ensemble d'instances à annoter et induit par la suite un ensemble de règles à partir de l'ensemble d'apprentissage étendu. Les règles de WHISK n'utilisent ni des niveaux de confiance ni des schémas de vote pour sélectionner des instances à annoter. Le système propose des exemples à annoter à l'utilisateur parmi 3 ensembles d'exemples. Nous rappelons ces 3 ensembles et nous leur attribuons les notations suivantes :

- Ensemble CnA – c'est l'ensemble des exemples couverts par des règles existantes non encore annotés par l'utilisateur.
- Ensemble *NM* – c'est l'ensemble des exemples couverts par des généralisations minimales de règles existantes (*near misses*).
- Ensemble R – c'est le reste des exemples non couverts et non annotés par l'utilisateur. Dans notre cas l'ensemble R est constitué de tous les mots du corpus et de tous les termes extraits par un extracteur de termes à partir du corpus non couverts et non annotés.

La définition de l'ensemble *NM* telle qu'elle a été faite dans le module interactif de l'algorithme WHISK n'est pas claire dans la mesure où elle ne spécifie pas ce que

c'est une généralisation minimale d'une règle dans le langage de WHISK. De plus, l'algorithme WHISK_R que nous avons choisi d'étendre par la suite n'utilise pas le même langage que WHISK mais plutôt le langage Ruta. Une définition claire de ce que c'est qu'une généralisation minimale d'une règle dans le langage Ruta s'avère donc nécessaire. Avant donc de présenter les modules d'apprentissage actif proposés, nous tenons à expliquer comment l'ensemble *NM* est construit.

Construction de l'ensemble *NM*

nous commençons par poser les définitions suivantes :

— Nous appelons item chaque condition de la règle y compris la condition de correspondance. Prenons la règle suivante comme exemple :
```
Token{FEATURE("lemma","human")->MARKONCE(Habitat)} #
Token{FEATURE("postag","VBZ")} # Token{FEATURE("postag","CC")};
```
Cette règle contient 5 items :
```
1-Token{FEATURE("lemma","human")};
2-#;
3-Token{FEATURE("postag","VBZ")};
4-#;
5-Token{FEATURE("postag","CC")}.
```

— Nous qualifions un item de spécifique si une condition spécifique est attribuée à son annotation correspondante et de générique si aucune condition n'est attribuée à son annotation correspondante. Si nous prenons la même règle précédente, les items 1, 3 et 5 sont spécifiques car des conditions (`lemma`, `postag`, `postag`) sont attribuées à leur annotation (`Token`). Les items 2 et 4 (#) sont des jokers qui peuvent représenter n'importe quel nombre d'éléments, ils font une catégorie à part. Prenons une deuxième règle comme exemple :
```
Token Token{REGEXP("tsetse")->MARKONCE(Habitat, 2, 3)}
Token{FEATURE("lemma","fly")} Token;
```

Dans cette règle, les items 1 et 4 sont génériques car aucune condition n'est attribuée à leur annotation (`Token`) alors que les items 2 et 3 sont spécifiques car des conditions (`REGEXP` (expression exacte), `lemma`) sont attribuées à leur annotation (`Token`).

— Nous appelons *filler* d'une règle l'ensemble des items qui correspondent à la cible à extraire. Prenons par exemple les deux règles suivantes :
```
1-Token{FEATURE("lemma","tick")->MARKONCE(Habitat)};
2-Token Token{FEATURE("lemma","tsetse")->MARKONCE(Habitat, 2, 3)}
  Token{FEATURE("lemma","fly")} Token;
```

Dans la première règle, le *filler* est composé d'un seul item :
```
Token{FEATURE("lemma","tick")}
```
Dans la deuxième règle, le *filler* contient, en revanche, deux items et il est le suivant :
```
Token{FEATURE("lemma","tsetse")} Token{FEATURE("lemma","fly")}
```
car l'action de la règle
```
->MARKONCE(Habitat, 2, 3)
```

porte sur ces deux items.

— Nous appelons contexte d'une règle l'ensemble total des items la constituant dépourvu des items composant le *filler*. Pour les deux règles précédentes, la taille du contexte est égale à 0 pour la première règle et 2 pour la deuxième règle.

La généralisation d'un item spécifique se fait en supprimant toute condition attribuée à son annotation. Prenons, par exemple, les trois items suivants :

```
1-Token{FEATURE("lemma","tick")}
2-Token{REGEXP("tick")}
3-Token{FEATURE("postag","NNS")}
```

Leur généralisation correspond à l'item `Token`.

Pour qu'une règle admette une ou plusieurs généralisations minimales, elle doit satisfaire un ensemble de conditions parmi les 3 ensembles de conditions suivants.

— Le *filler* est composé d'un seul item qui est spécifique et il y a au moins un item spécifique dans son contexte : l'idée ici est de garder le *filler* intact et de généraliser en généralisant à chaque fois l'un des items spécifiques du contexte.

— Le *filler* est composé d'un seul item qui est générique et il y a au moins 3 items spécifiques dans son contexte : il s'agit ici de généraliser en généralisant à chaque fois l'un des items spécifiques du contexte.

— le *filler* contient plus qu'un item et la règle contient au moins 3 items spécifiques (y compris ceux composant le *filler*) : les généralisations peuvent se faire en généralisant à chaque fois l'un des items spécifiques de la règle.

Pour plus de détails, se référer à l'algorithme de construction des généralisations minimales des règles de la figure 6.14.

Pour mieux comprendre l'algorithme présenté, appuyons nous sur quelques exemples. Considérons l'ensemble des règles suivant :

```
1- Token{FEATURE("lemma","intracellular")->MARKONCE(Habitat)};
2- Token{FEATURE("lemma","food")->MARKONCE(Habitat)}
   # Token{FEATURE("lemma", "contaminated")};
3- Token{->MARKONCE(Habitat)} Token{REGEXP("rhizosphere")};
4- Token{FEATURE("lemma","when")} Token{->MARKONCE(Habitat)}
   Token{FEATURE("postag","VBD")};
5- Token{FEATURE("lemma","soil")->MARKONCE(Habitat,1,2)}
   Token{FEATURE("lemma","environment")} Token;
6- Token{FEATURE("postag","JJ")->MARKONCE(Habitat,1,2)}
   Token{FEATURE("lemma","fly")} Token;
7- Token{FEATURE("postag","IN")} Token{FEATURE("lemma","the")}
   Token{->MARKONCE(Habitat,3,4)} Token{FEATURE("lemma","cell")} Token
   # Token{FEATURE("lemma","male")};
```

Dans la première règle, la taille du contexte est nulle donc la règle n'admet aucune généralisation minimale. Dans la deuxième règle, le contexte n'est pas vide et le *filler* est spécifique et composé d'un seul item. Comme le contexte contient un item spécifique, cette règle admet une généralisation minimale qui est la suivante :

```
ConstruireReglesNearMisses (ReglesExistantes,ReglesNearMisses)
  Pour règle r dans ReglesExistantes faire
  |  compteur = 0
  |  Si taille(filler(r)) = 1 faire
  |  | Si filler(r) est spécifique et filler(r) != # faire
  |  | | Pour item it dans contexte(r) faire
  |  | | | Si it est spécifique
  |  | | | | generaliserRegle(r,r2,it)
  |  | | | | ajouter(r2,ReglesNearMisses)
  |  | | | Fin si
  |  | | Fin pour
  |  | Sinon
  |  | | Si taille(contexte) >= 3 faire
  |  | | | Pour item it dans contexte(r) faire
  |  | | | | Si it est spécifique
  |  | | | | | compteur <-- compteur + 1
  |  | | | | Fin si
  |  | | | Fin pour
  |  | | | Si compteur >=3
  |  | | | | Pour item it dans contexte(r) faire
  |  | | | | | Si it est spécifique
  |  | | | | | | generaliserRegle(r,r2,it)
  |  | | | | | | ajouter(r2,ReglesNearMisses)
  |  | | | | | Fin si
  |  | | | | Fin pour
  |  | | | Fin si
  |  | | Fin si
  |  | Fin si
  |  Sinon
  |  | Pour item it dans filler(r) faire
  |  | | Si it est spécifique
  |  | | | compteur <-- compteur + 1
  |  | | Fin si
  |  | Fin pour
  |  | Pour item it dans contexte(r) faire
  |  | | Si it est spécifique
  |  | | | compteur <-- compteur + 1
  |  | | Fin si
  |  | Fin pour
  |  | Si compteur >=3
  |  | | Pour item it dans filler(r) faire
  |  | | | Si it est spécifique
  |  | | | | generaliserRegle(r,r2,it)
  |  | | | | ajouter(r2,ReglesNearMisses)
  |  | | | Fin si
  |  | | Fin pour
  |  | | Pour item it dans contexte(r) faire
  |  | | | Si it est spécifique
  |  | | | | generaliserRegle(r,r2,it)
  |  | | | | ajouter(r2,ReglesNearMisses)
  |  | | | Fin si
  |  | | Fin pour
  |  | Fin si
  |  Fin si
  Fin pour
```

```
Avec

generaliserRegle(r,r2,it)
 it2 <-- generaliserItem(it)
 r2 <-- construireRegle(r,it,it2) //remplacer it par it2 dans la règle
```

Figure 6.14 – Algorithme de construction des généralisations minimales de règles.

```
Token{FEATURE("lemma","food")->MARKONCE(Habitat)} # Token;
```

Cette règle peut bien évidemment ramener des exemples négatifs mais surtout des exemples positifs car elle dispose d'une information centrale qui est le lemme de la cible à extraire. Les erreurs que la règle peut faire sont donc uniquement des termes qui ont le même lemme que la cible mais dont le sens est autre. Ces termes sont intéressants à regarder car ils peuvent constituer des bons contre exemples.

Dans les troisième et quatrième règles, le *filler* est générique et composé d'un seul item et la taille du contexte est inférieure à 3 donc les deux règles n'admettent pas de généralisations minimales. En effet, si nous ne posons pas de contraintes sur la taille du contexte et sur le nombre d'items spécifiques, la quatrième règle, par exemple, admettrait les deux généralisations minimales suivantes :

```
1-Token Token{->MARKONCE(Habitat)} Token{FEATURE("postag","VBD")};
2-Token{FEATURE("lemma","when")} Token{->MARKONCE(Habitat)} Token;
```

Ces règles sont trop générales et peuvent couvrir des termes qui n'ont aucun lien avec le terme cible.

Dans les cinquième et sixième règles, le *filler* est composé de deux items et le nombre des items spécifiques dans l'ensemble contexte-*filler* est inférieur à 3 donc les deux règles n'admettent pas de généralisations minimales. En effet, sans la contrainte sur le nombre d'items spécifiques dans l'ensemble contexte-*filler*, la cinquième règle aurait admis comme généralisations minimales :

```
1-Token{->MARKONCE(Habitat,1,2)} Token{FEATURE("lemma","environment")}
  Token;
2-Token{FEATURE("lemma","soil")->MARKONCE(Habitat,1,2)} Token Token;
```

et la sixième aurait admis comme généralisations minimales :

```
1-Token{->MARKONCE(Habitat,1,2)} Token{FEATURE("lemma","fly")} Token;
2-Token{FEATURE("postag","JJ")->MARKONCE(Habitat,1,2)} Token Token;
```

Avec une contrainte sur un seul item, même si l'item fait partie de l'expression de la cible, ces règles restent trop générales et ramènent beaucoup d'exemples négatifs.

Concernant la dernière règle, le *filler* est composé de deux items et le nombre d'items spécifiques dans l'ensemble *filler*-contexte est 4 donc elle admet les généralisations minimales suivantes :

```
1-Token Token{FEATURE("lemma","the")} Token{->MARKONCE(Habitat,3,4)}
  Token{FEATURE("lemma","cell")} Token # Token{FEATURE("lemma","male")};
2-Token{FEATURE("postag","IN")} Token Token{->MARKONCE(Habitat,3,4)}
  Token{FEATURE("lemma","cell")} Token # Token{FEATURE("lemma","male")};
3-Token{FEATURE("postag","IN")} Token{FEATURE("lemma","the")}
  Token{->MARKONCE(Habitat,3,4)} Token Token #
  Token{FEATURE("lemma","male")};
4-Token{FEATURE("postag","IN")} Token{FEATURE("lemma","the")}
  Token{->MARKONCE(Habitat,3,4)} Token{FEATURE("lemma","cell")} Token
  # Token;
```

Avec 3 items spécifiques, ces règles restent assez spécifiques et ne font pas beaucoup d'erreurs.

Après la construction des généralisations minimales des règles existantes, l'ensemble *NM* (*near misses*) peut être construit. Il s'agit, en effet, des exemples couverts par des généralisations minimales de règles existantes non couverts par des règles existantes.

IAL4Sets

Nous avons établi, dans un travail préalable (Bannour, Audibert, & Nazarenko, 2011), une étude de mesures de similarité entre des termes tels qu'ils peuvent être extraits par un extracteur de termes, en s'appuyant sur l'hypothèse distributionnelle selon laquelle des termes sémantiquement proches tendent à apparaître dans des contextes similaires. Les mesures de similarités étudiées combinent des fonctions de poids et des mesures de similarité élémentaires. Nous avons redéfini, dans un premier temps, la notion d'analyse distributionnelle habituellement appliquée sur des mots pour la combiner à une analyse terminologique et prendre en compte les unités composées. Cela a impliqué une adaptation des fonctions de poids et de la notion de contexte. Nous avons présenté ensuite une étude méthodique de différentes combinaisons de fonctions de poids et de mesures de similarité. Certaines ont produit des résultats convaincants. La combinaison poids-mesure qui a donné les meilleurs résultats est `TTEST-COSINUS`.

En s'appuyant sur les résultats de cette étude et sur l'intuition que les termes sémantiquement proches des termes couverts par des règles existantes jugés comme positifs peuvent être des termes synonymes ou liés (dénotent un concept fils, père ou frère), nous avons eu l'idée de proposer à l'utilisateur ce genre de termes à annoter. Nous avons également remarqué que lorsque les exemples d'apprentissage ont des expressions très différentes, si l'ensemble de départ annoté par l'utilisateur pour lancer la première itération de l'algorithme d'apprentissage contient toutes ces expressions, les deux premiers ensembles parmi lesquels WHISK propose des exemples à annoter à l'utilisateur (CnA et *NM*) suffisent pour retrouver rapidement les exemples non annotés. Cependant, si les exemples de départ ne contiennent qu'une partie des expressions possibles pour un exemple d'apprentissage, il faut attendre que ces exemples soient proposés dans le troisième ensemble de WHISK (R). Cet ensemble contient un très grand nombre d'exemples, ce qui mène à un temps plus long pour atteindre les performances optimales du système. D'où l'idée de proposer cet ensemble

supplémentaire qui contient des exemples proches des exemples positifs couverts sans obligatoirement avoir les mêmes expressions, ce qui peut accélérer l'extraction des exemples dont les expressions ne font pas partie de celles annotées au départ. Nous appelons cet ensemble TSEP (Termes les plus Similaires aux Exemples Positifs couverts).

Pour expliquer la méthode de construction de l'ensemble TSEP, nous posons les notations suivantes.

- `termesPosCouverts` : il s'agit de la liste des termes couverts par des règles existantes jugés comme positifs et reconnus par l'extracteur de termes en tant que tels.
- `nbTermesPosCouverts` : il s'agit de la taille de la liste `termesPosCouverts`.
- `similairesTermePosCouvert(t)` : il s'agit de la liste complète des termes extraits par l'extracteur des termes ordonnés par similarité décroissante par rapport au terme t.
- `termesAProposer` : il s'agit de la liste des termes sélectionnés pour être proposés à l'utilisateur pour annotation dans l'ensemble TSEP.
- `nbTermesAProposer` : il s'agit du nombre de termes à proposer à l'utilisateur dans l'ensemble TSEP dans le cadre d'une itération.

Pour construire l'ensemble TSEP, nous procédons comme suit :

- une matrice de similarité est construite en calculant la similarité entre chaque paire de termes extraits par un extracteur de termes (les valeurs de similarité sont calculées en utilisant la mesure `TTEST-COSINUS`)
- pour chaque terme couvert par des règles existantes annoté positivement t, la liste des termes extraits par un extracteur de termes est ordonnée par valeurs de similarité décroissantes par rapport au terme t (construction de la liste `similairesTermePosCouvert(t)`)
- si `nbTermesPosCouverts` >= `nbTermesAProposer`
  - sélectionner au hasard `nbTermesAProposer` termes parmi les termes couverts annotés positivement
  - pour chaque terme couvert annoté positivement sélectionné t, sélectionner le premier terme de la liste `similairesTermePosCouvert(t)` (le terme le plus similaire au terme t) et l'ajouter à l'ensemble TSEP

  Sinon
  - pour chaque terme t couvert annoté positivement, sélectionner à partir de `similairesTermePosCouvert(t)` les (`nbTermesAProposer` / `nbTermesPosCouverts`) premiers termes et les ajouter à l'ensemble TSEP
  - s'il reste des termes à proposer à l'utilisateur, les sélectionner à partir de la liste `similairesTermePosCouvert(t)` d'un terme t couvert annoté positivement choisi au hasard et les ajouter à l'ensemble TSEP

L'algorithme présenté dans la figure 6.15 donne plus de détails sur la méthode de construction de l'ensemble TSEP.

Après la proposition des termes sélectionnés à l'utilisateur, les listes des termes similaires aux termes positifs couverts sont mises à jour en supprimant tous les termes qui ont été proposés.

Après avoir explicité comment l'ensemble TSEP est construit, nous expliquons

```
selectionnerTermes(termesPosCouverts,termesAProposer,nbTermesAProposer)
   Si nbTermesAProposer < nbTermesPosCouverts faire
   | choisir au hasard nbTermesAProposer termes à partir de
   | termesPosCouverts
   | Pour chaque terme t choisi faire
   | | plusSimilaire(t,t2,similairesTermePosCouvert(t),termesAProposer)
   | | ajouter (t2,termesAPoposer)
   | Fin pour
   Sinon
   | n <-- nbTermesAProposer/nbTermesPosCouverts
   | Pour chaque terme t choisi faire
   | | nPlusSimilaires(t,tList,n,similairesTermePosCouvert(t)
   | |                  ,termesAProposer)
   | | ajouter (tList,termesAProposer)
   | Fin pour
   Fin si.

Avec

 plusSimilaire(t,t2,similairesTermePosCouvert(t),termesAProposer)
   compteur <-- 0
   trouve <-- false
   Tant que compteur < taille(similairesTermePosCouvert(t)) et non(trouve)
   faire
   | Si termesAProposer ne contient pas
   | similairesTermePosCouvert(t)[compteur] faire
   | | t2 <-- similairesTermePosCouvert(t)[compteur]
   | Sinon
   | | compteur <-- compteur + 1
   | Fin si
   Fin tant que.

et

 nPlusSimilaires(t,tList,n,similairesTermePosCouvert(t),termesAProposer)
   compt <-- 0
   compt2 <-- 0
   Tant que compt<taille(similairesTermePosCouvert(t)) et compt2<n faire
   | Si termesAProposer ne contient pas
   | similairesTermePosCouvert(t)[compteur] faire
   | | ajouter(similairesTermePosCouvert(t)[compteur],tList)
   | | compt2 <-- compt2 + 1
   | Fin si
   | compt <-- compt + 1
   Fin tant que.
```

Figure 6.15 – Algorithme de construction de l'ensemble TSEP.

le principe du module d'apprentissage actif proposé IAL4Sets. Ce module permet, en effet, de proposer des exemples à annoter à l'utilisateur équitablement à partir de 4 ensembles : CnA, *NM*, TSEP et R (voir l'algorithme plus en détail dans la figure 6.16). Les ensembles CnA, *NM* et R sont les mêmes ensembles proposés dans le module interactif de l'algorithme WHISK.

```
Annotation de k exemples par l'utilisateur
 Déroulement de l'algorithme d'apprentissage des règles à partir des
 exemples annotés
 Test des règles inférées sur le corpus de test
 Calcul des performances (précision, rappel et F-mesure)
 Pour i de 1 à n faire
 |  Proposer à l'utilisateur k/4 exemples de l'ensemble CnA
 |  Proposer à l'utilisateur k/4 exemples de l'ensemble NM
 |  Proposer à l'utilisateur k/4 exemples de l'ensemble TSEP
 |  Proposer à l'utilisateur k/4 exemples de l'ensemble R
 |  Annotation des k exemples par l'utilisateur
 |  Ajout des k exemples annotés à l'ensemble total des exemples annotés
 |  Déroulement de l'algorithme d'apprentissage des règles sur
 |   l'ensemble total des exemples annotés
 |  Test des règles inférées sur le corpus de test
 |  Calcul des performances (précision, rappel et F-mesure)
 Fin pour.

Avec

- n est le nombre d'itérations nécessaires pour atteindre des
  performances qui satisfont l'utilisateur;
- k est le nombre d'exemples proposés à l'utilisateur à chaque itération.
  Il est fixé par l'utilisateur.
```

Figure 6.16 – Algorithme du module d'apprentissage actif IAL4Sets.

IAL3Sets

En examinant de manière approfondie la liste des termes extraits par un extracteur de termes sur le corpus BB BioNLP-ST 2013, nous avons remarqué que la plupart des exemples positifs annotés sur ce corpus sont reconnus en tant que termes. En se basant sur ces observations, nous avons eu l'idée de proposer une deuxième version d'un module d'apprentissage actif qui exclut l'ensemble R. Cet algorithme que nous nommons IAL3Sets permet donc de proposer à l'utilisateur équitablement des exemples à annoter à partir des 3 ensembles CnA, *NM* et TSEP (voir l'algorithme présenté dans la figure 6.17).

Certes cet algorithme n'est pas complet dans le sens où il existe une minorité d'exemples positifs que l'extracteur de termes ne reconnait pas (ces exemples ne seront jamais retrouvés) mais nous pensons qu'il permettra une convergence plus rapide de l'apprentissage.

```
Annotation de k exemples par l'utilisateur
 Déroulement de l'algorithme d'apprentissage de règles à partir des
 exemples annotés
 Test des règles inférées sur le corpus de test
 Calcul des performances (précision, rappel et F-mesure)
 Pour i de 1 à n faire
 |  Proposer à l'utilisateur k/3 exemples de l'ensemble CnA
 |  Proposer à l'utilisateur k/3 exemples de l'ensemble NM
 |  Proposer à l'utilisateur k/3 exemples de l'ensemble TSEP
 |  Annotation des k exemples par l'utilisateur
 |  Ajout des k exemples annotés à l'ensemble total des exemples annotés
 |  Déroulement de l'algorithme d'apprentissage de règles sur
 |   l'ensemble total des exemples annotés
 |  Test des règles inférées sur le corpus de test
 |  Calcul des performances (précision, rappel et F-mesure)
 Fin pour.
```

Figure 6.17 – Algorithme du module d'apprentissage actif IAL3Sets.

Une évaluation détaillée des deux modules d'apprentissage actif que nous avons proposés (IAL4Sets et IAL3Sets) figure dans la section 8.4.

Conclusion

Nous avons proposé, dans ce chapitre, une approche interactive d'apprentissage de règles d'EI. Cette approche repose sur un processus hybride qui combine l'écriture manuelle des règles d'EI et l'annotation d'exemples d'apprentissage pour inférer de manière automatique des règles, ce qui donne à l'utilisateur la liberté de choisir l'opération qu'il juge la plus convenable à un instant donné. Ce processus est interactif pour permettre une communication entre l'utilisateur et le module d'apprentissage automatique de règles dans le but de réduire au maximum l'effort manuel requis pour mener à bien une tâche d'EI. Il est également itératif pour permettre une construction progressive de l'ensemble des règles d'EI, ce qui offre à l'utilisateur une meilleure compréhension du système lui facilitant la prise de la bonne décision concernant l'opération à effectuer à chaque itération.

Pour pouvoir être mise en place, l'approche que nous proposons doit satisfaire un certain nombre de contraintes et respecter certaines propriétés clés que nous avons déjà énumérées dans le chapitre précédent. Nous avons donc proposé des stratégies et des méthodes qui permettent d'assurer ces contraintes.

Pour la compréhensibilité et la généricité des règles, nous avons proposé d'utiliser le langage de règles Ruta qui est un langage expressif, générique et accessible et de prétraiter le corpus de travail avec une chaîne d'annotation qui a pour rôle d'étiqueter le texte avec des annotations génériques (lemme, étiquette morpho-syntaxique, terme) sur lesquelles s'appuie l'expression des règles dans le langage Ruta.

Pour plus de prudence et parce que plusieurs travaux encouragent l'utilisation

de fenêtres de contexte courtes, nous avons proposé de réduire l'espace de recherche des règles pour l'algorithme d'apprentissage en ne considérant que les mots faisant partie de la phrase qui contient l'exemple cible.

Pour ne pas considérer comme négatifs les exemples positifs non encore annotés par l'utilisateur dans un système où l'annotation des exemples se fait de manière progressive, nous avons proposé d'apprendre, à chaque itération, sur un corpus réduit composé uniquement des phrases contenant des exemples annotés positivement par l'utilisateur.

Enfin, pour permettre une sélection aisée des exemples d'apprentissage par l'utilisateur, nous avons proposé deux solutions pour deux buts différents. La première solution consiste à mettre en place un concordancier pour écrire des règles prospectives qui ont pour but de permettre à l'utilisateur de chercher des exemples en s'appuyant sur des mots clés dont il dispose et de réduire ainsi son espace de travail aux exemples couverts par ses règles prospectives lui facilitant l'évaluation de ces exemples et leur annotation. La deuxième solution, quant à elle, consiste à mettre en place un module d'apprentissage actif qui a pour but de proposer à l'utilisateur de manière progressive des exemples sélectionnés d'une manière intelligente qui favorise une convergence plus rapide de l'algorithme d'apprentissage vers la solution optimale. Les deux modules d'apprentisssage actif que nous avons proposés (IAL4Sets et IAL3Sets) étendent le module interactif de base de l'algorithme WHISK. Étant donné que nous avons choisi de nous appuyer sur l'algorithme WHISK_{R} pour la mise en place et l'évaluation de notre approche, nous y avons introduit le module d'apprentissage actif de base conçu pour l'algorithme WHISK qui sera utilisé comme Baseline dans l'évaluation de nos modules d'apprentissage actif.

Le développement ainsi que l'évaluation des différentes méthodes et stratégies que nous proposons dans le cadre de l'approche conçue dans ce chapitre sont détaillés dans les chapitres 7 et 8 qui suivent.

Chapitre 7

IRIES : un système interactif d'apprentissage de règles d'extraction d'information

Nous avons proposé dans le chapitre précédent une approche interactive d'apprentissage de règles d'extraction d'information (EI) ainsi que des méthodes et des stratégies qui permettent d'assurer les propriétés qui doivent caractériser une telle approche. Nous décrivons dans ce chapitre le système IRIES qui met en œuvre cette approche ainsi que les étapes techniques qui ont permis de le mettre en place.

7.1 Architecture du système IRIES

Comme le montre la figure 7.1, le système IRIES (*Interactive Rule based Information Extraction System*) que nous avons mis en place est composé de 3 modules :

- **Un concordancier** – il permet à l'utilisateur d'écrire des règles prospectives pour chercher des exemples particuliers dans le texte, tester des règles inférées par le module d'apprentissage ou modifiées par l'utilisateur et d'annoter les exemples couverts par les règles testées. L'annotation d'exemples dans le concordancier agit sur l'ensemble d'exemples d'apprentissage en l'étendant avec de nouveaux exemples ou en modifiant le statut de certains exemples.
- **Un module d'apprentissage de règles d'EI** – ce module encapsule un alogrithme d'apprentissage de règles et permet d'inférer des règles d'EI ou d'annotation à partir d'exemples d'apprentissage pour un concept cible donné. Les règles inférées sont consultées par l'utilisateur et peuvent être modifiées par ce dernier et testées dans le concordancier.
- **Un module d'apprentissage actif** – ce module permet de proposer à l'utilisateur, de manière itérative, des exemples à annoter. Ces exemples sont sélectionnés d'une manière intelligente afin de permettre une convergence rapide du système vers les performances optimales. L'utilisateur répond en annotant les exemples proposés, ce qui permet au module d'apprentissage actif de mettre à jour l'ensemble des exemples d'apprentissage et d'appeler le module d'apprentissage de règles.

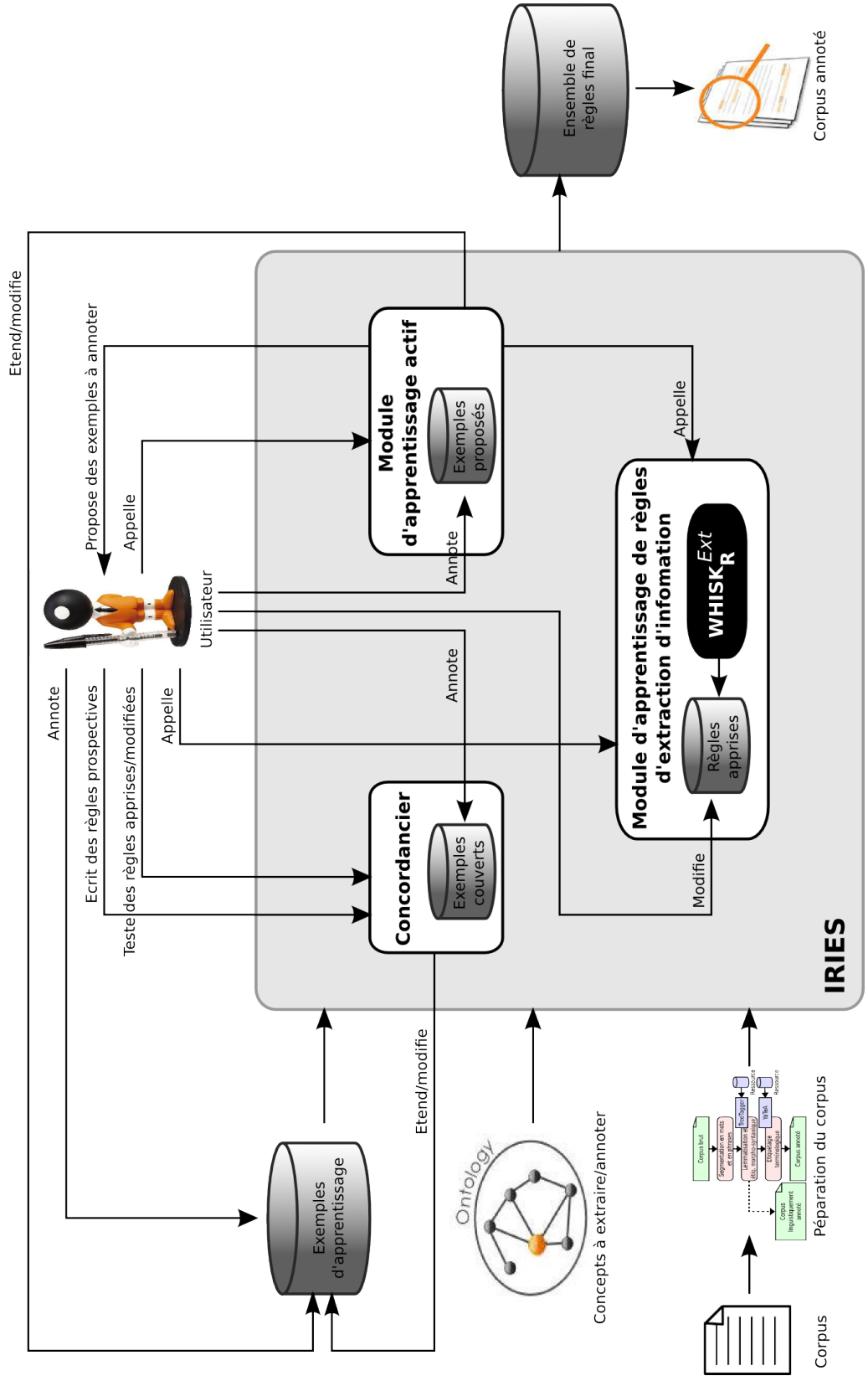


Figure 7.1 – Architecture du système IRIES.

Il est à noter que l'appel au module d'apprentissage de règles peut s'effectuer soit en appelant le module d'apprentissage actif qui fait lui même un appel au module d'apprentissage de règles soit en appelant directement le module d'apprentissage de règles si l'utilisateur se passe de l'aide du module actif à un moment donné.

L'ensemble de règles final contient des règles apprises, des règles écrites par l'utilisateur et jugées bonnes et des règles inférées modifiées par l'utilisateur dont la version modifiée s'avère plus performante. Appliquées sur le corpus d'entrée, les règles finales permettent d'annoter dans le texte les segments de texte qui décrivent les concepts cibles. Les concepts cibles à annoter peuvent provenir d'une ontologie ou tout simplement être des concepts communs.

Afin de produire des règles compréhensibles, génériques et performantes, IRIES prend en entrée un corpus prétraité. Nous présentons en détails dans la section suivante la chaine de traitements appliquée au texte initial pour le préparer.

7.2 Préparation des corpus : une chaine d'annotation basée sur UIMA

La chaine d'annotation décrite dans le chapitre précédent permet d'étiqueter le texte d'entrée avec des annotations qui apparaissent dans l'expression des règles écrites ou inférées. Plus ces annotations sont intuitives, plus les règles sont compréhensibles et plus elles sont génériques, plus les règles sont génériques.

Pour mettre en place cette chaine d'annotation, nous choisissons d'utiliser le framework UIMA (Ferrucci & Lally, 2004) (voir section 4.1.

Pour découper le texte en mots et en phrases, nous mettons en place un algorithme qui s'appuie sur les patrons du langage Java (`package java.util.regex`). Concernant la lemmatisation, l'étiquetage morpho-syntaxique et l'étiquetage terminologique, nous développons des modules qui encapsulent des outils externes.

Prenons, par exemple, nos deux corpus de travail. Pour le corpus BB BioNLP-ST 2013, les outils externes utilisés sont :

— Bioc (Liu, Christiansen, Baumgartner, & Verspoor, 2012 ; Smith, Rindflesch, & Wilbur, 2004) – Il permet à la fois la segmentation du texte en *tokens*, la lemmatisation et l'étiquetage morpho-syntaxique.

— BioYaTeA (Golik, Bossy, Ratkovic, & Claire, 2013) – Cet outil permet l'extraction des termes dans un texte. Il s'agit d'une version étendue de l'extracteur de termes YaTeA (Hamon & Aubin, 2006) adaptée au domaine biomédical.

Concernant le corpus SyntSem, les outils externes utilisés sont :

— TreeTagger (Schmid, 1994) – Cet outil permet la lemmatisation et l'étiquetage morpho-syntaxique des mots dans un texte

— YaTeA (Hamon & Aubin, 2006) – Cet outil permet d'identifier des termes candidats dans le texte en se basant sur une désambiguisation endogène.

Pour définir l'ensemble des types utilisés pour annoter le texte, nous avons hésité entre les deux *Types Systems* schématisés dans les figures 7.2 et 7.3.

Dans le *Type System* schématisé dans la figure 7.3, les unités linguistiques sont concentrées dans les types `Token` attribué à chaque mot du texte et `Sentence` attri-

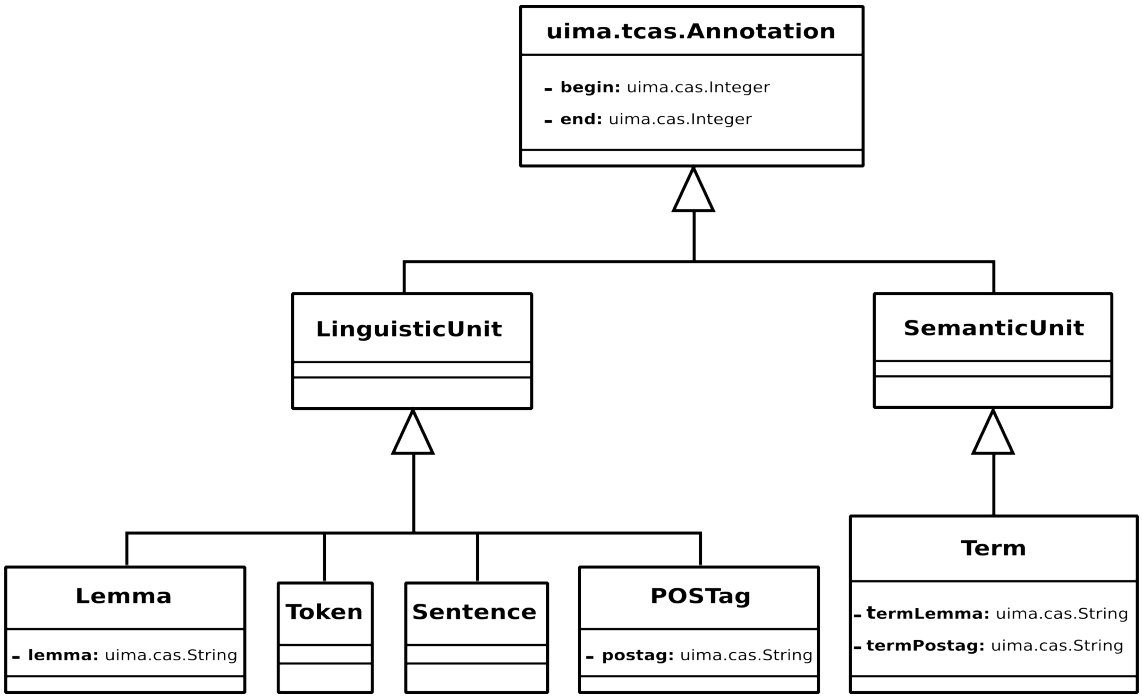


Figure 7.2 – *Type System* UIMA non retenu.

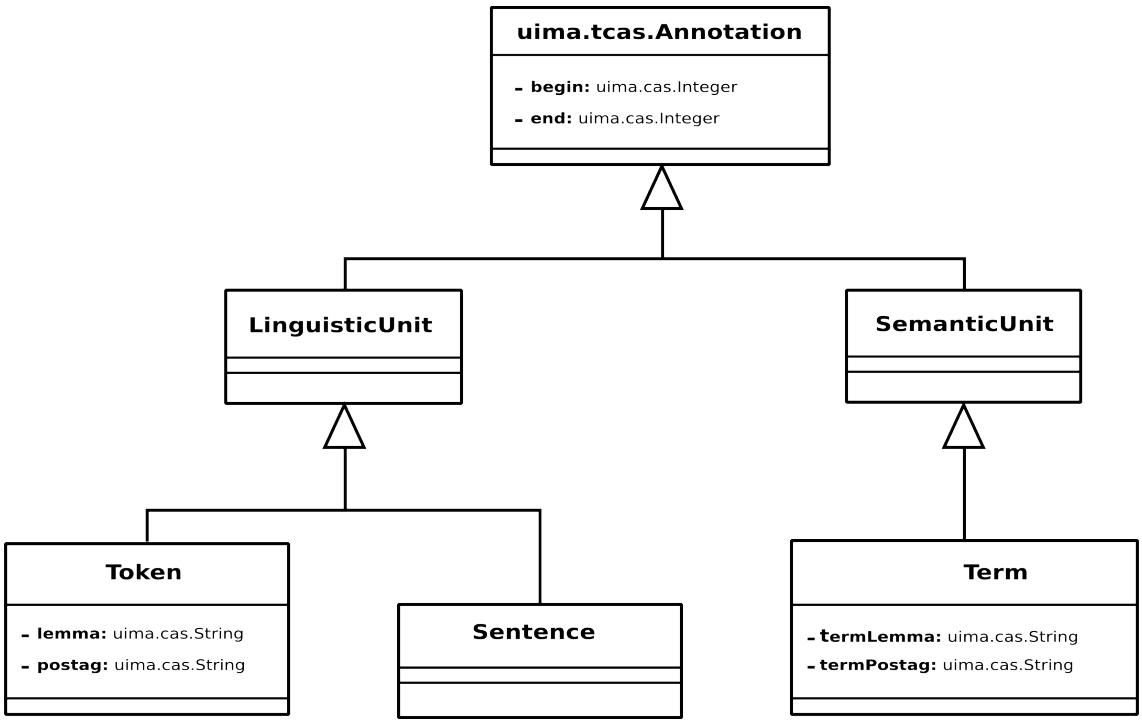


Figure 7.3 – *Type System* UIMA utilisé.

bué à chaque phrase du texte. Nous caractérisons le type `Token` par deux propriétés supplémentaires : `lemma` et `postag`. `lemma` contient la forme lemmatisée et `postag` l'étiquette morpho-syntaxique attribuées par un outil externe (TreeTagger ou Bioc) au `Token` en question. Dans le *Type System* schématisé dans la figure 7.2, en revanche, à part le type `Sentence`, il existe 3 types linguistiques qui délimitent la même unité qui est le mot : `Token`, `Lemma`, et `POSTag`. `Lemma` et `POSTag` sont caractérisés respectivement par les propriétés `lemma` et `postag` qui contiennent la forme lemmatisée et l'étiquette morpho-syntaxique attribuées par le module de lemmatisation et d'étiquetage morpho-syntaxique au `Token` en question.

Nous considérons, en revanche, dans les deux cas le terme (`Term`) comme une unité sémantique et non linguistique et nous la caractérisons par les mêmes propriétés qu'un `Token` dans le *Type System* schématisé dans la figure 7.3 (`termLemma` et `termPostag`) dont les valeurs sont fournies par l'extracteur de termes.

Comme nous l'avons déjà mentionné dans le chapitre précédent, l'expression des règles d'EI dépend beaucoup de comment sont exprimées les annotations dans le texte. Prenons un exemple qui explique la différence d'expression des règles dans le langage Ruta en utilisant à chaque fois l'un des *Type Systems* proposés. Supposons que nous voulions écrire une règle qui traduit la phrase suivante : si on trouve un nom suivi d'un mot dont l'expression est « de » suivi d'un mot lemmatisé « donnée », on annote le tout avec l'annotation « BaseDeDonnées ».

En s'appuyant sur le *Type System* schématisé dans la figure 7.2, la règle s'écrit comme suit :

```
POSTag{FEATURE("postag","NN")} Token{REGEXP("de")}
Lemma{FEATURE("lemma","donnée") ->MARKONCE(BaseDeDonnées,1,3)};
```

Alors qu'en s'appuyant sur le *Type System* schématisé dans la figure 7.3, la règle s'écrit comme suit :

```
Token{FEATURE("postag","NN")} Token{REGEXP("de")}
Token{FEATURE("lemma","donnée") ->MARKONCE(BaseDeDonnées,1,3)};
```

La deuxième règle est plus intuitive car au final les termes que la règle cherche à couvrir sont des Tokens avec à chaque fois une contrainte spécifique. La première règle aurait un sens si les annotations `POSTag` et `Lemma` étaient des annotations indépendantes de l'annotation `Token` et avaient des offsets différents de ceux délimitant les annotations de type `Token` mais ce n'est pas le cas. En effet, les 3 types `Token`, `Lemma` et `POSTag` couvrent exactement les mêmes offsets et désignent tous les mots du texte d'où l'intérêt d'avoir un seul type pour désigner un mot (`Token`) et des propriétés intuitives pour le caractériser (`lemma` et `postag` comme propriétés de `Token`). Nous optons, par conséquent, pour l'utilisation du deuxième *Type System* (figure 7.3).

La figure 7.4 explique en détails la mise en place dans la plateforme UIMA de la chaine d'annotation que nous proposons avec les différentes communications entre les modules.

Dans la plateforme UIMA, un module ou une classe (*Annotator*) ne peut pas être exécuté directement mais à travers un AE qui l'appelle.

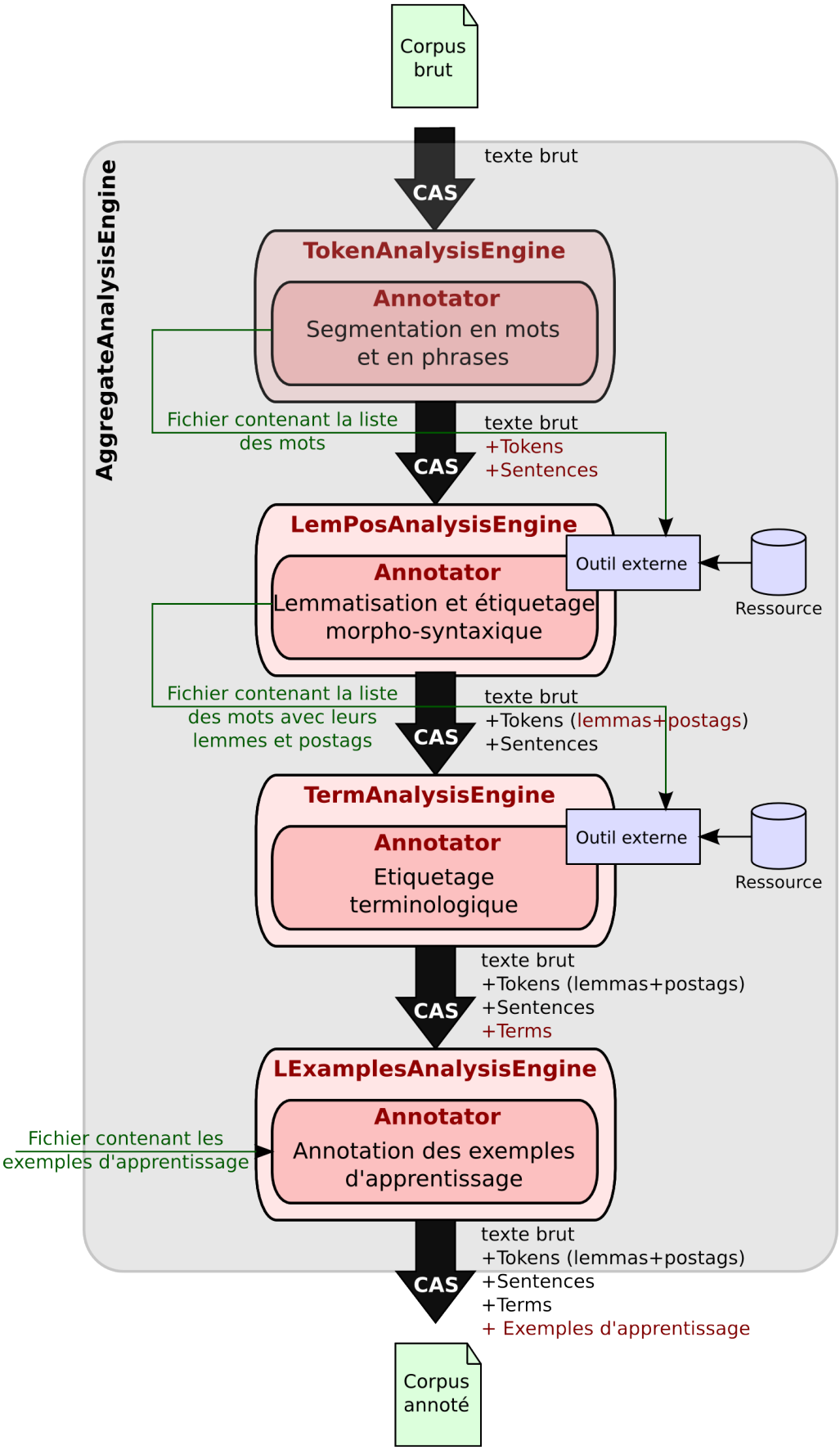


Figure 7.4 – Mise en place de la chaine d'annotation proposée dans la plateforme UIMA.

Au départ, le CAS contient uniquement le texte initial. Nous faisons appel au *TokenAnalysisEngine* appelant à son tour le module de segmentation en mots et en phrases qui alimente le CAS avec les annotations de type `Token` et de type *Sentence* (phrases). Étant appelé en exécutant le moteur *LemPosAnalysisEngine*, le module de lemmatisation et d'étiquetage morpho-syntaxique prend en entrée les annotations de type `Token` récupérées du CAS, fait un appel à l'outil externe qui prend en entrée le fichier des mots généré par le module de segmentation en mots et en phrases et génère un fichier contenant la même liste des mots en ajoutant leurs informations sur les parties de discours et leurs lemmes, ensuite enrichit les annotations de type `Token` dans le CAS avec les propriétés `postag` et `lemma`. Le module d'étiquetage terminologique appelé par *TermAnalysisEngine* encapsule un outil externe qui prend en entrée le fichier généré par le module de lemmatisation et d'étiquetage morpho-syntaxique, génère une liste de termes candidats et alimente le CAS avec des annotations de type `Term`.

Pour pouvoir apprendre des règles d'EI ou d'annotation sur le corpus, il faut également annoter des exemples d'apprentissage. Ces exemples sont déportés et fournis dans des documents à part. Cependant pour pouvoir exploiter ces exemples, ils doivent être traduits en annotations UIMA intégrées au corpus car le langage Ruta dans lequel les règles sont inférées s'appuie sur des annotations UIMA. Pour ce faire, nous développons un module qui permet de transformer les exemples d'apprentissage déportés en annotations UIMA. Ce module appelé par le moteur *LExamplesAnalysisEngine* alimente le CAS avec les exemples d'apprentissage.

L'appel de l'*AggregateAnalysisEngine* qui aggrège tous les AEs pemet l'exécution de toute la chaine d'annotation.

Après la préparation du corpus de travail, nous passons au développement des différents modules composant le système IRIES.

7.3 Développement du système IRIES

Le système IRIES étant composé de 3 modules de base, nous détaillons, dans cette section la mise en place de ces modules en tenant compte des différentes méthodes et stratégies proposées dans le chapitre 6.

7.3.1 Réutilisation du système TextRuler

IRIES est, en effet, une extension du système TextRuler (Kluegl, Atzmueller, Hermann, & Puppe, 2009), un framework de développement semi-automatique d'applications d'EI basées sur des règles Ruta. TextRuler partage le même processus itératif décrit par notre méthode (voir figure 7.5) mais l'aspect interactif est absent.

Textruler est un framework générique dans le sens où il contient des implémentations de plusieurs algorithmes d'apprentissage de règles adaptés au langage Ruta et l'utilisateur peut choisir d'utiliser n'importe lequel d'entre eux, voire plusieurs en parallèle. Il est également facile à étendre dans le sens où l'utilisateur peut implémenter ses propres algorithmes en se basant sur des classes génériques fournies dans le système TextRuler et les rajouter à ce dernier. Une description plus détaillée du système TextRuler figure dans le chapitre 4.

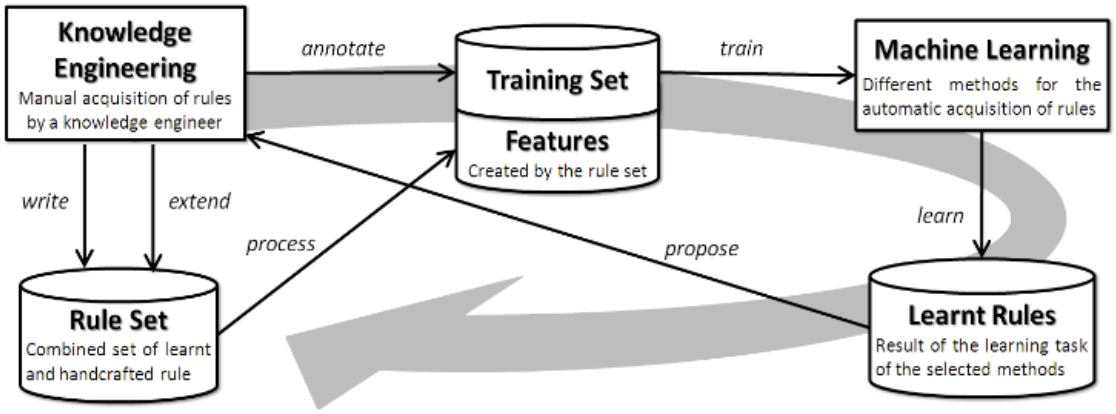


Figure 7.5 – Modèle de processus de TextRuler (Kluegl, Atzmueller, Hermann, & Puppe, 2009).

Pour mettre en place notre approche interactive d'apprentissage de règles d'EI, nous apportons des modifications et des extensions au système TextRuler dont les principales classes (implémentées en Java) sont représentées dans la figure 7.6. Sont exclues du diagramme schématisé dans la figure 7.6 les classes graphiques, les classes de communication entre les classes de base et les classes graphiques et les classes représentant les autres algorithmes d'apprentissage de règles implémentés dans TextRuler (autres que WHISK) pour des raisons de simplification.

Les classes schématisées en bleu sont des classes que nous modifions pour nos propres besoins. Elles seront détaillées ainsi que les classes que nous rajoutons dans les sections qui suivent. La classe Whisk dans TextRuler correspond à l'implémentation de WHISK_R.

7.3.2 $\text{WHISK}_\text{R}^{Ext}$: une extension de WHISK_R

Le module d'apprentissage de règles d'EI contenu dans le système IRIES encapsule l'algorithme $\text{WHISK}_\text{R}^{Ext}$. $\text{WHISK}_\text{R}^{Ext}$ est, en effet, une extension de l'algorithme WHISK_R avec lequel nous avons choisi de travailler. Nous développons cette extension pour deux raisons :

— étendre les règles de WHISK_R pour prendre en compte les informations linguistiques attribuées aux Tokens ;
— modifier WHISK_R pour tenir compte de certaines contraintes nécessaires pour la mise en place d'un système interactif.

Il est à noter que les modifications que nous apportons à l'algorithme WHISK_R peuvent être faites sur n'importe quel algorithme d'apprentissage de règles. D'ailleurs, on peut tout à fait créer une classe générique qui représente un algorithme d'apprentissage de règles respectant les contraintes que doit satisfaire un système interactif de laquelle hérite n'importe quel algorithme d'apprentissage de règles utilisé (la classe *TextRulerBasicLearner* par exemple dans le diagramme de la figure 7.6).

La figure 7.7 représente les classes de TextRuler que nous modifions pour obtenir l'algorithme $\text{WHISK}_\text{R}^{Ext}$. Les attributs et fonctions coloriés en rouge sont des

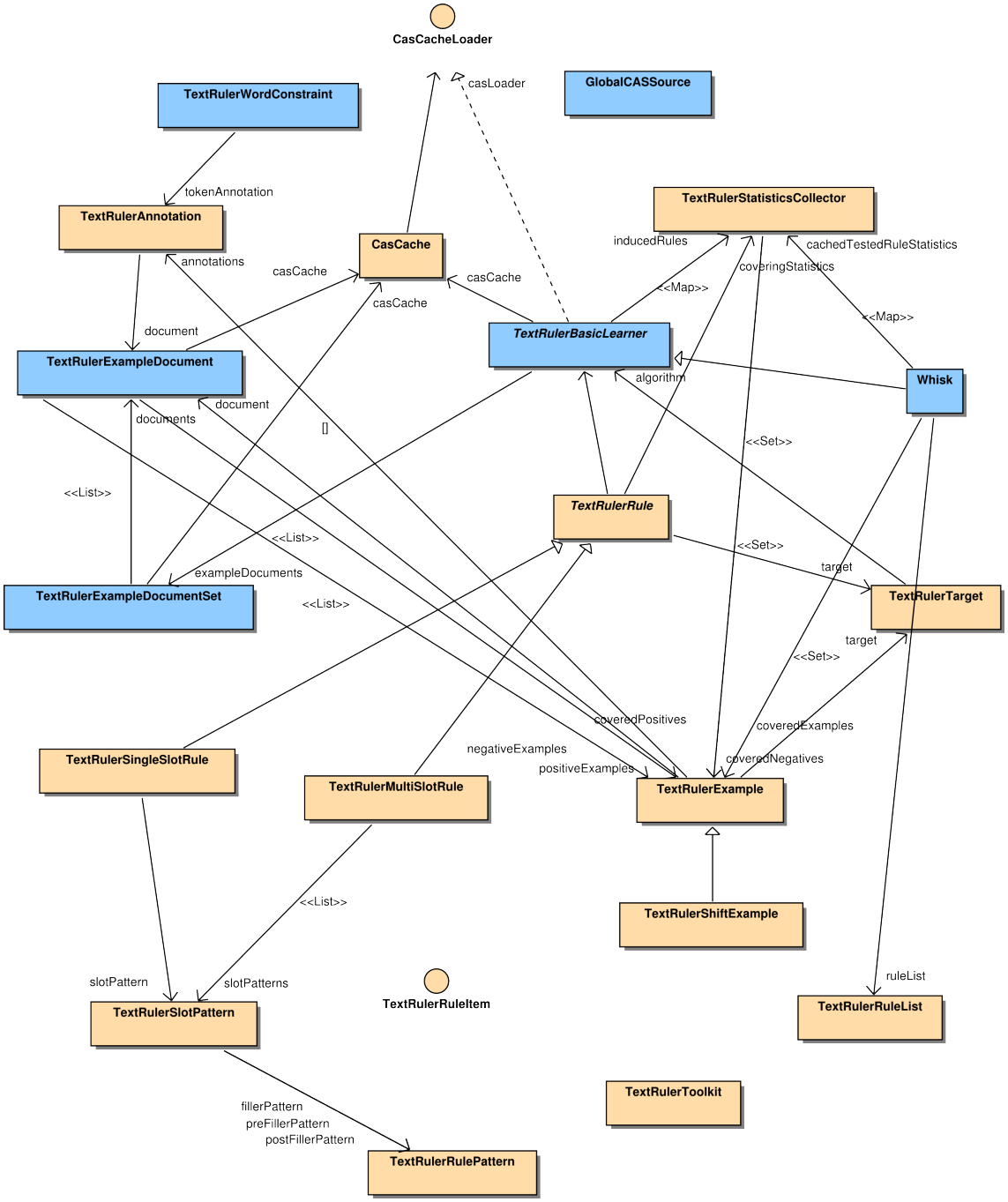


Figure 7.6 – Diagramme des classes principales de TextRuler.

attributs et fonctions que nous modifions et ceux coloriés en vert sont des attributs et fonctions que nous ajoutons.

La version initiale de WHISK_{R} souffre d'un inconvénient majeur : les règles qu'il apprend considèrent uniquement l'expression exacte des mots. Voici un exemple de règle inférée par cette version initiale de WHISK_{R} :

```
Token{REGEXP("human")-->MARKONCE(BacteriaHabitat)};
```

Cette règle permet d'annoter chaque mot du texte dont l'expression est *human* avec le concept « Habitat de bactéries ». Elle n'est, cependant, pas capable d'identifier les expressions `humans`, `Human` ou encore `Humans` qui devraient également être annotées.

Les règles apprises par la version initiale de WHISK_{R} étant limitées aux expressions exactes des mots, nous étendons l'implémentation de l'algorithme de manière à rendre les règles capables de s'appuyer sur les étiquettes linguistiques des mots (lemmes et étiquettes morpho-syntaxiques) et donc d'exploiter la richesse du langage Ruta.

Nous appelons la version initiale de WHISK_{R} $\text{WHISK}_{\text{Mot}}$ puisqu'elle s'appuie uniquement sur l'expression exacte des mots et nous distinguons les versions suivantes de notre algorithme ${\text{WHISK}_{\text{R}}}^{Ext}$.

- $\text{WHISK}_{\text{Lem}}$: il s'agit d'une version de WHISK_{R} qui s'appuie sur les formes lemmatisées des mots
- $\text{WHISK}_{\text{Pos}}$: il s'agit d'une version de WHISK_{R} qui s'appuie sur les étiquettes morpho-syntaxiques des mots
- $\text{WHISK}_{\text{MotLem}}$: il s'agit d'une version de WHISK_{R} qui s'appuie sur expressions exactes et les formes lemmatisées des mots
- $\text{WHISK}_{\text{MotPos}}$: il s'agit d'une version de WHISK_{R} qui s'appuie sur les expressions exactes et les étiquettes morpho-syntaxiques des mots
- $\text{WHISK}_{\text{LemPos}}$: il s'agit d'une version de WHISK_{R} qui s'appuie sur les formes lemmatisées et les étiquettes morpho-syntaxiques des mots
- $\text{WHISK}_{\text{MotLemPos}}$: il s'agit d'une version de WHISK_{R} qui s'appuie sur les expressions exactes, les formes lemmatisées et les étiquettes morpho-syntaxiques des mots.

Un étude expérimentale qui figure dans la section 8.2 permet d'évaluer et de comparer entre elles ces différentes versions étendues de WHISK_{R} dans le but d'expliciter l'effet d'un prétraitement linguistique sur les performances d'un algorithme d'apprentissage de règles d'EI et particulièrement l'algorithme $\text{WHISK}_{\text{Mot}}$.

Les attributs et les classes que nous ajoutons à l'implémentation de base sont les suivants.

- *CONSIDERED_FEATURES* dans la classe *Whisk* : cet attribut contient la liste des propriétés dont l'algorithme d'apprentissage doit tenir compte quand il parcourt les annotations. Dans notre cas, nous considérons les propriétés `lemma` et `postag` de l'annotation `Token`.
- *hideFeature* et *activeFeature* dans la classe *WhiskRuleItem* : ces attributs permettent de vérifier si une certaine propriété est activée (apparaît dans les expressions des règles) ou non activée.

Figure 7.7 – Principales classes modifiées dans TextRuler.

— *isHideFeature()*, *setHideFeature()*, *activateFeature()* et *deactivateFeature()* dans la classe *WhiskRuleItem* : ces fonctions permettent de récupérer ou de modifier l'état d'une propriété et d'activer ou désactiver une propriété.
— *getActivatedFeatures()* dans la classe *WhiskRuleItem* : cette fonction permet de récupérer la liste des propriétés qui peuvent apparaître dans l'expression des règles.

La prise en compte des informations linguistiques n'est pas la seule extension que nous apportons à l'algorithme WHISK_R. Afin de ne pas discriminer les exemples non encore annotés par l'utilisateur à un stade donné de l'apprentissage interactif de règles, nous avons proposé dans la section 6.4 d'apprendre sur un corpus réduit composé uniquement des phrases annotées par l'utilisateur, ce qui a impliqué la réduction du contexte à examiner lors de l'apprentissage d'un exemple cible à la phrase. Afin de mettre en place ces stratégies, nous apportons d'autres modifications sur la version initiale de WHISK_R. Concernant les versions de $\text{WHISK}_\text{R}{}^{Ext}$ qui réduisent le contexte d'apprentissage à la phrase et qui apprennent sur un corpus réduit, nous adoptons respectivement les notations $\text{WHISK}\_{}^{P}$ et $\text{WHISK}\_{}^{CR}$ avec :

$\_ \in \{Mot, Lem, Pos, MotLem, MotPos, LemPos, MotLemPos\}$ est la combinaison d'étiquettes linguistiques utilisées.

Une étude expérimentale qui figure dans la section 8.3 permet d'évaluer le gain en temps d'apprentissage que réalisent ces nouvelles versions étendues de WHISK_R par rapport aux versions qui apprennent sur la totalité du corpus d'apprentissage sans pour autant dégrader les performances.

Le reste des attributs et fonctions coloriés en vert dans la figure 7.7 et qui ne concernent pas l'ajout des traits linguistiques sont des attributs et des fonctions que nous implémentons pour mettre en place ces stratégies d'apprentissage sur un corpus réduit et de modification de la taille de fenêtre de contexte d'apprentissage. La fonction *createTrainCAS()*, par exemple, permet de créer le CAS d'apprentissage qui est une version réduite du CAS correspondant aux phrases consultées par l'utilisateur. La fonction *testRulesOnDocumentSetRestricted()*, elle, permet de tester les règles construites dans l'espace de recherche de règles de l'algorithme d'apprentissage uniquement sur les phrases contenant des exemples annotés positivement par l'utilisateur.

Les attributs et les fonction coloriés en rouge dans la figure 7.7 sont des attributs et des fonctions que nous modifions pour prendre en compte les nouveaux attributs et fonctions ajoutés.

7.3.3 Mise en œuvre de la sélection d'exemples

Pour mettre en œuvre la sélection d'exemples dans un système interactif d'apprentissage de règles d'EI, nous avons proposé dans la section 6.5 la mise en place d'un concordancier et d'un module d'apprentissage actif. Ces deux modules sont les deux principaux composants de l'architecture du système IRIES décrite dans la section 7.1.

RMatcher : un concordancier basé sur Ruta

RMatcher (*Rule Matcher*) est le concordancier que nous implémentons pour permettre à l'utilisateur d'écrire des règles prospectives dans le langage Ruta.

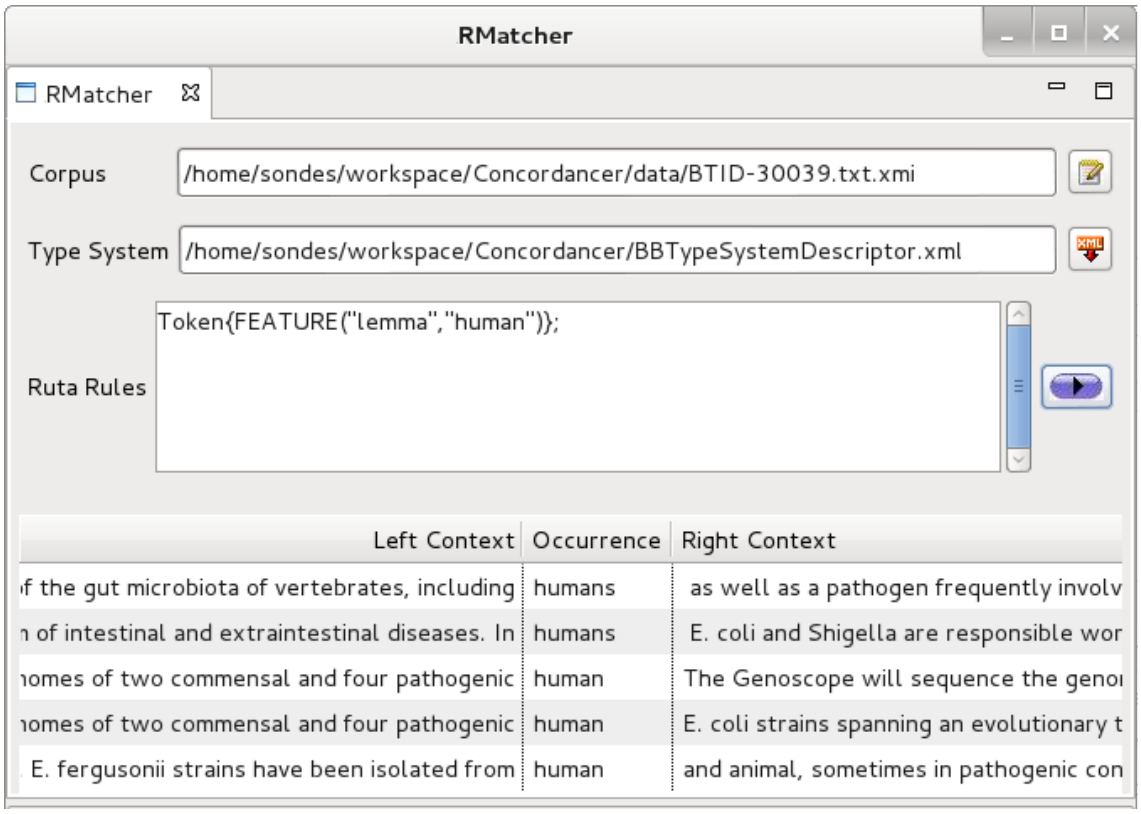


Figure 7.8 – RMatcher : un concordancier basé sur Ruta.

Comme le montre la figure 7.8, RMatcher prend en entrée un corpus préparé avec la plateforme UIMA (Training Data) et un *Type system* (Type System) qui contient les types avec lesquels le corpus est annoté et dont se sert l'utilisateur pour écrire ses règles (Ruta rules).

Après l'écriture de ses règles dans le langage Ruta, l'utilisateur appuie sur le bouton en face des règles (bouton violet) pour les appliquer sur le corpus. RMatcher applique les règles sur le corpus et affiche les termes couverts par ces règles ainsi que leurs contextes gauche et droit.

Les règles de l'utilisateur ne contiennent pas d'action car elles ont pour but d'afficher les exemples couverts et non de les annoter.

La figure 7.9 montre le diagramme de classes de RMatcher.

Ce diagramme est composé des classes suivantes.

— *DataProcess* : cette classe permet de récupérer le corpus d'entrée et ses annotations.

— *FinalCas* : cette classe permet d'alimenter le CAS sur lequel sont appliquées les règles de l'utilisateur avec les annotations du corpus d'entrée et les annotations de base de Ruta.

— *PatternHandler* : cette classe permet d'appliquer les règles de l'utilisateur sur le corpus d'entrée.

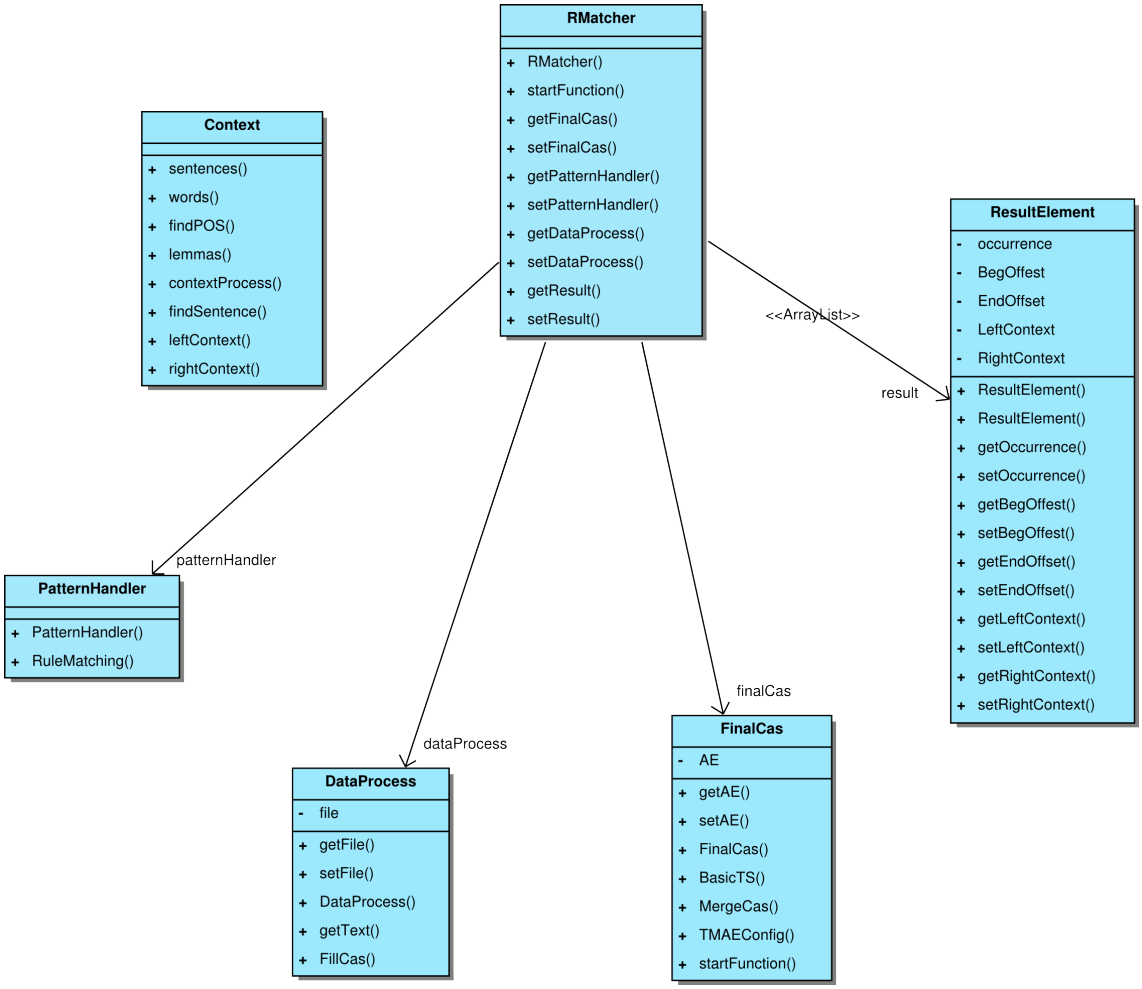


Figure 7.9 – Classes implémentées pour mettre en place RMatcher.

— *Context* : cette classe permet de définir le contexte d'un exemple couvert par une règle.
— *ResultElement* : cette classe permet de déterminer, pour un exemple couvert par une règle, ses offsets de début et de fin et son contexte gauche et droit.
— *RMatcher* : c'est la classe qui permet de faire le lien entre toutes les autres classes et d'exécuter le traitement à faire à partir du moment où l'utilisateur appuie sur le bouton de lancement.

Développement d'un module d'apprentissage actif

Nous avons proposé dans la section 6.5.2 deux modules d'apprentissage actif (IAL4Sets et IAL3Sets) qui étendent le module Baseline correspondant au module interactif conçu pour l'algorithme WHISK de base.

Nous implémentons ces trois modules dont les classes sont schématisées dans la figure 7.10.

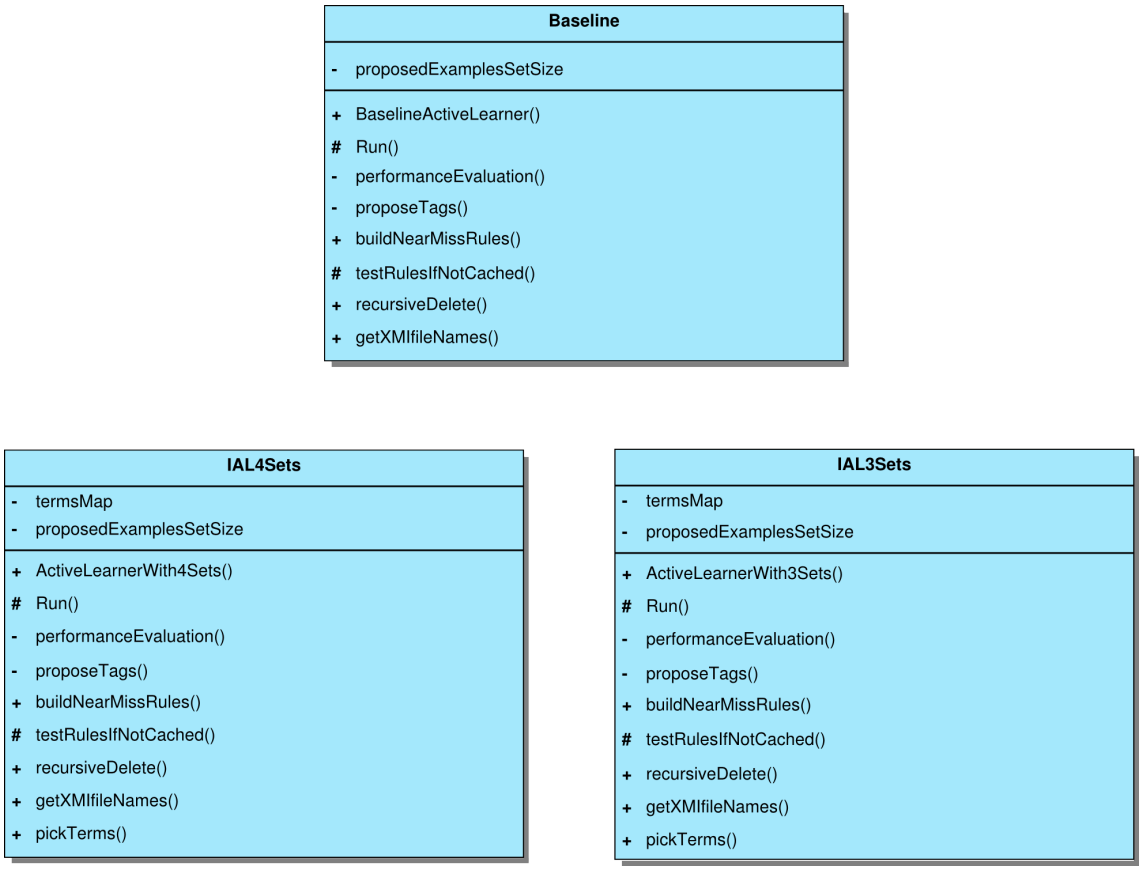


Figure 7.10 – Classes implémentées pour mettre en place le module d'apprentissage actif.

Commentons les principaux attributs et fonctions de ces classes. L'attribut *proposedExamplesSetSize* correspond au nombre d'exemples à proposer à l'utilisateur à chaque itération pour annotation. La fonction *proposeTags()* permet de proposer des exemples à partir des ensembles concernés pour chaque module d'apprentissage

actif (CnA, *NM*, TSEP et R pour IAL4Sets par exemple). La fonction *buildNear-MissRules()* permet de construire les règles qui couvrent les *Near Misses* des règles existantes (l'ensemble *NM*). La fonction *Run()* permet de lancer l'algorithme d'apprentissage. La fonction *pickTerms()* dans les classes IAL4Sets et IAL3Sets permet de sélectionner les exemples à proposer de l'ensemble TSEP. L'attribut *termsMap* dans ces mêmes classes représente une *map* qui contient, pour chaque terme extrait par un extracteur de termes, la liste des termes ordonnée par ordre décroissant de similarité par rapport au terme en question.

7.4 Interface du système IRIES

L'interface du système IRIES que nous proposons est présentée dans la figure 7.11.

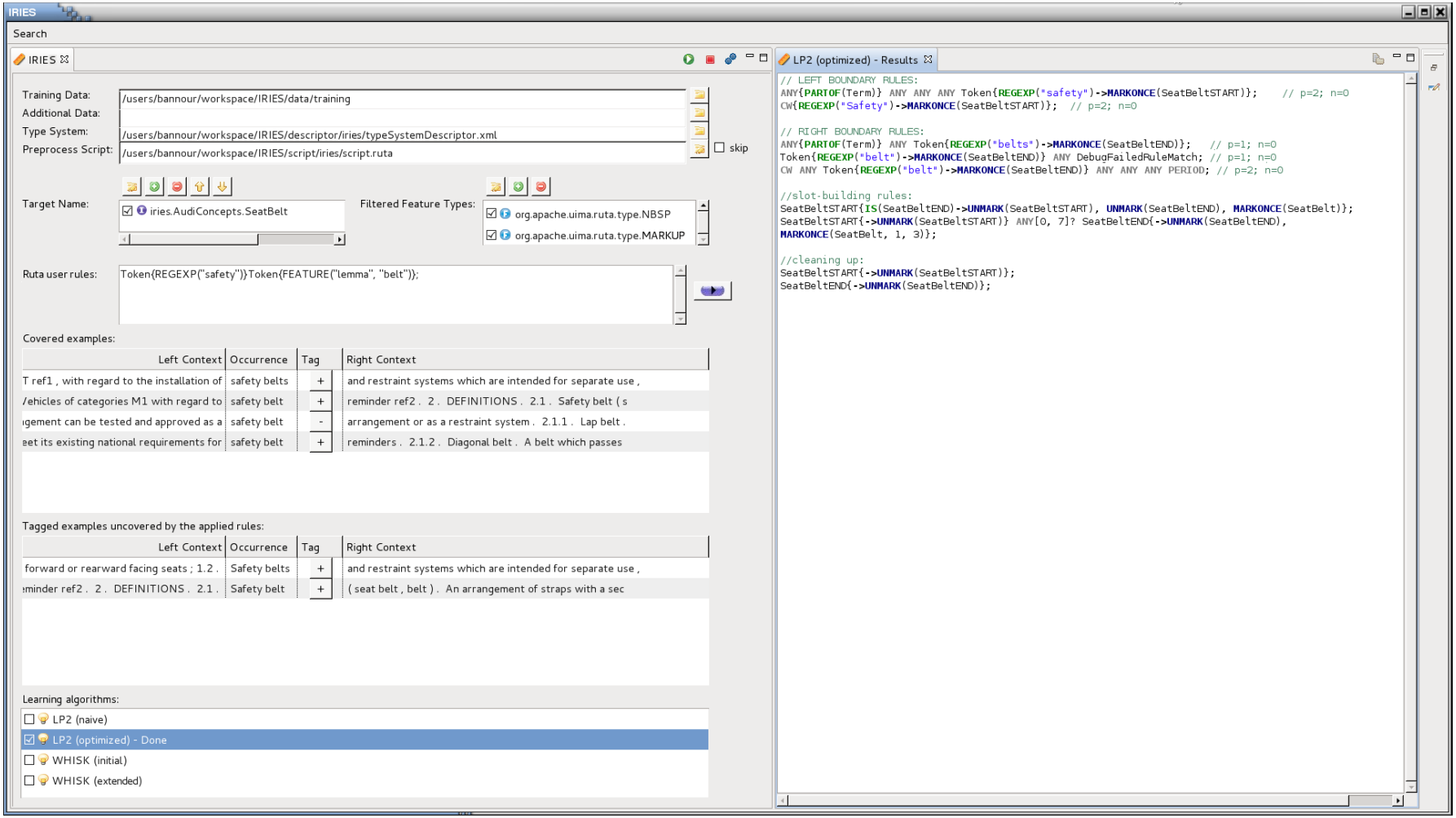


Figure 7.11 – Interface du système IRIES.

Cette interface est composée de 5 parties :

— **Les données d'entrée** – Elles comprennent le corpus d'apprentissage préparé avec la plateforme UIMA (.xmi), le *Type System* qui contient les types avec lesquels le corpus est annoté et le script Ruta de prétraitement. Le script de prétraitement peut être vide comme il peut contenir des règles que l'utilisateur a écrites à l'avance pour annoter des exemples en s'appuyant sur des mots clés dont il dispose.

— **Les concepts à extraire et les types d'annotations à filtrer** –Les types d'annotations à filtrer sont les types à ne pas considérer dans les expressions des règles (les espaces blancs et les balises html par exemple). Les concepts à extraire sont traités l'un après l'autre.

— **Le concordancier** – Pour intégrer le concordancier dans l'interface du système IRIES, nous ajoutons un champ devant les exemples couverts par les règles testées (Tag) pour donner la possibilité à l'utilisateur de les annoter. Une vue sur les exemples annotés non couverts par les règles est également ajoutée au concordancier.

— **Les algorithmes d'apprentissage à tester** – IRIES peut fonctionner avec n'importe quel algorithme d'apprentissage de règles basé sur Ruta. Pour choisir l'algorithme à utiliser pour l'apprentissage de règles, il suffit de le cocher. Pour paramétrer les différents algorithmes avant de les tester, il faut appuyer sur le bouton représenté par des engrenages bleus.

— **Les résultats d'apprentissage** sont affichés sur la partie droite de l'interface.

Les algorithmes d'apprentissage actif que nous avons implémentés dans ce travail (IAL4Sets et IAL3Sets) ne sont pas encore intégrés dans l'interface graphique d'IRIES. Pour l'instant, les seuls exemples que l'utilisateur peut annoter à travers l'interface d'IRIES sont les exemples couverts par des règles existantes (inférées ou écrites) et les exemples couverts par des règles prospectives.

Conclusion

Nous avons présenté dans le chapitre précédent les différents modules proposés pour mettre en place notre approche interactive d'EI. Nous décrivons dans ce chapitre le système IRIES qui met en œuvre les différents modules proposés. IRIES prend en entrée des concepts à extraire et un corpus préparé et fournit un ensemble de règles qui permettront d'annoter les concepts cibles dans le corpus. Pour préparer le corpus, nous avons mis en place une chaîne d'annotation basée sur UIMA tout en prenant le soin de définir un *Type System* qui assure une bonne expressivité et intuitivité des règles. IRIES est composé de 3 modules : un concordancier (RMatcher) qui permet à l'utilisateur d'écrire des règles prospectives dans le langage Ruta, un algorithme d'apprentissage de règles ($\text{WHISK}_\text{R}^{Ext}$) pour inférer les règles d'EI et un module d'apprentissage actif (IAL4Sets ou IAL3Sets) qui propose à l'utilisateur des exemples pertinents à annoter. Nous avons présenté dans ce chapitre les détails d'implémentation de ces modules. L'interface du système IRIES étend celle de TextRuler mais n'est pas encore complètement achevée car le module d'apprentissage actif n'a pas encore été intégré dans cette dernière. Nous présentons dans le chapitre qui suit

une étude expérimentale qui permet d'évaluer les différents modules proposés.

Chapitre 8

Expérimentations

Nous présentons dans ce chapitre une étude expérimentale qui a pour but l'évaluation des différentes propositions détaillées dans les chapitres 6 et 7. Nous n'avons pas fait de propositions pour assurer la stabilité des règles dans ce travail. Nous présentons, en revanche, une étude expérimentale dans ce chapitre qui permet de le justifier.

8.1 Fonctions de mesure de performances utilisées

Pour évaluer quantitativement certaines de nos contributions, nous utilisons des fonctions de mesure de performances classiques très utilisées dans le domaine d'extraction d'information (EI) qui sont la précision, le rappel et la F-mesure (voir section 2.4.2). Nous rappelons ces fonctions et nous les exprimons dans ces termes :

Précision (P) – Cette fonction permet de détermnier la proportion des exemples positifs couverts parmi l'ensemble total des exemples couverts par l'ensemble des règles produites par notre algorithme d'apprentissage de règles

$$P = \frac{vp}{vp + fp}$$

avec :
— vp= nombre d'exemples positifs couverts ;
— fp= nombre de mauvais exemples couverts.

Rappel (R) – Cette fonction permet de déterminer la proportion des exemples positifs couverts parmi l'ensemble total des exemples positifs qui doivent être couverts par l'ensemble de règles produites par notre algorithme d'apprentissage de règles

$$R = \frac{vp}{vp + fn}$$

avec :
— fn= nombre d'exemples positifs non couverts

F-mesure (F) – Cette fonction permet de trouver un bon compromis entre le rappel et la précision

$$F = \frac{2 \times P \times R}{P + R}$$

Les fonctions R_n et F_n sont des fonctions que nous avons dérivées de R et F et dont les expressions sont les suivantes :

$$R_n = \frac{vp}{vp + fn_n} \qquad F_n = \frac{2 \times P \times R_n}{P + R_n}$$

avec fn_n= nombre d'exemples positifs non couverts et dont la fréquence est au moins égale à n.

Ces fonctions sont utiles pour évaluer l'apprentissage des concepts qui sont représentés par des exemples qui ont une certaine fréquence. En effet, dans quelques corpus, on ne peut apprendre certains concepts que lorsqu'ils disposent d'un nombre minimum d'exemples. Il est donc important de pouvoir évaluer l'apprentissage uniquement sur les concepts qui possèdent suffisamment d'exemples d'apprentissage.

8.2 Rôle des informations linguistiques dans les performances des règles

Les langages de règles récents comme JAPE ou Ruta donnent à l'utilisateur la possibilité d'exploiter la richesse des annotations qui peuvent être disponibles dans les textes pour exprimer ses règles. Ces annotations sont des informations qui peuvent être de nature linguistique comme la lemmatisation et l'étiquetage morpho-syntaxique ou de nature sémantique comme l'étiquetage terminologique et l'étiquetage des entités nommées. Elles permettraient, selon plusieurs chercheurs, d'obtenir des règles d'EI plus expressives, plus génériques et plus performantes.

Après avoir montré dans le chapitre 6 l'intérêt de ces informations dans l'expressivité et la généricité des règles, nous nous intéressons dans cette section à leur effet sur les performances des règles d'EI. Les informations que nous considérons sont linguistiques et comprennent la lemmatisation et l'étiquetage morpho-syntaxique.

8.2.1 Cas d'usage 1 : apprentissage des habitats de bactéries

Dans le cadre de la sous-tâche 1 du BB BioNLP-ST 2013 (Bossy et al., 2013) dont le but est de détecter, dans le texte, les habitats de bactéries et leur associer une ou plusieurs catégories à partir de l'ontologie OntoBiotope [1] fournie pour la tâche, nous avons tenté d'apprendre directement les catégories d'habitats de bactéries à partir du texte en utilisant différentes versions de notre algorithme $\text{WHISK}_\text{R}^{Ext}$. Deux expériences ont été réalisées dans ce contexte.

Expérience 1

Nous avons entraîné, dans cette expérience, quelques versions de notre algorithme $\text{WHISK}_\text{R}^{Ext}$ (WHISK_Lem, WHISK_Pos, $\text{WHISK}_\text{MotLem}$, $\text{WHISK}_\text{MotPos}$, $\text{WHISK}_\text{LemPos}$ et $\text{WHISK}_\text{MotLemPos}$) sur les catégories qui disposent de 3 exemples ou plus, c'est

1. `http://bibliome.jouy.inra.fr/MEM-OntoBiotope/OntoBiotope_BioNLP-ST13.obo`

à dire sur 634 exemples parmi les 948 exemples des données d'entraînement. Ces catégories comptent pour 308 occurrences parmi les 626 annotations de référence dans les données de développement. Par conséquent, la valeur de rappel maximale (colonne R dans la table 8.1) que notre algorithme peut atteindre ne peut pas dépasser 49%.

Les versions de $\text{WHISK}_\text{R}{}^{Ext}$ qui utilisent un contexte d'apprentissage réduit à la phrase et celles qui permettent d'apprendre sur un corpus réduit ne sont pas considérées dans cette section. Elles font l'objet d'une autre expérience qui figure dans la section 8.3.

La table 8.1 montre les résultats donnés par les différentes versions de $\text{WHISK}_\text{R}{}^{Ext}$ utilisées (sur les données de développement) en considérant différentes combinaisons de traits linguistiques (Expression exacte des mots (Mot), lemme (Lem), postag (Pos)), comparés à ceux donnés par la version initiale de WHISK_R (première ligne). La colonne vp correspond au nombre d'annotations de catégories d'habitats prédites correctement dans le corpus de développement (si un habitat est prédit avec la mauvaise catégorie, il est considéré comme une mauvaise prédiction). La colonne Nb de règles représente le nombre total de règles d'EI construites sur le corpus d'entraînement.

Version de $\text{WHISK}_\text{R}{}^{Ext}$	**Nb de règles**	**vp**	**P**	**R**	**F**	**R_3**	**F_3**
$\text{WHISK}_{\text{Mot}}$ (initial)	291	160	78%	26%	38%	52%	62%
$\text{WHISK}_{\text{Lem}}$	273	156	76%	25%	37%	51%	61%
$\text{WHISK}_{\text{Pos}}$	83	0	0%	0%	0%	0%	0%
$\text{WHISK}_{\text{MotLem}}$	269	174	75%	28%	41%	56%	64%
$\text{WHISK}_{\text{MotPos}}$	277	164	78%	26%	38%	53%	63%
$\text{WHISK}_{\text{LemPos}}$	**266**	**178**	**78%**	**28%**	**42%**	**58%**	**66%**
$\text{WHISK}_{\text{MotLemPos}}$	267	180	77%	29%	42%	58%	66%

Table 8.1 – Performances de certaines versions de $\text{WHISK}_\text{R}{}^{Ext}$ sur les données de développement. vp correspond au nombre d'annotations de catégories d'habitats prédites correctement dans le corpus de développement. P, R et F correspondent respectivement aux valeurs de précision, rappel et F-mesure sur les données de développement et R_3 et F_3 correspondent aux valeurs de rappel et F-mesure pour les catégories sur lesquelles les différentes versions de $\text{WHISK}_\text{R}{}^{Ext}$ utilisées ont été entraînées.

Il est important de mentionner que P, R et F sont utiles pour avoir une idée sur le comportement de l'algorithme sur d'autres données de test. Cependant, P, R_3 et F_3 permettent de mieux évaluer les performanaces des différentes versions de l'algorithme. En effet, comme P, R_3 et F_3 concernent uniquement les catégories sur lesquelles l'algorithme a été entraîné (catégories qui sont représentées par au moins 3 exemples), la valeur maximale que R_3 peut atteindre est 1 et non 0.49 comme pour R.

Nous pouvons remarquer à partir de la table 8.1 que l'utilisation de $\text{WHISK}_{\text{Lem}}$ n'améliore pas les performances. Au contraire, elle dégrade les résultats (61% de F-mesure contre 62% pour $\text{WHISK}_{\text{Mot}}$). En effet, l'apprentissage des catégories d'ha-

bitats de bactéries de manière indépendante implique que chaque catégorie dispose de peu d'exemples d'apprentissage (3 exemples pour la plupart d'entre elles). Ceci explique l'incapacité de $\text{WHISK}_{\text{Lem}}$ de généraliser sur ces exemples.

Prenons un exemple concret : pour la catégorie `MBTO:00001188` représentée par 5 exemples (*leaves*, *leaves*, *citrus leaf*, *leaf*, *leaf*) dans les données d'entraînement, $\text{WHISK}_{\text{Mot}}$ obtient 67% de F-mesure et donne les règles suivantes :

```
1-Token{REGEXP("leaves")->MARKONCE(MBTO00001188)};
2-Token{->MARKONCE(MBTO00001188, 1, 2)} Token Token{REGEXP("miner")};
3-Token{REGEXP("oryzae")} # Token{REGEXP("leaf")->MARKONCE(MBTO00001188)};
```

$\text{WHISK}_{\text{Lem}}$ donne, en revanche, uniquement la règle

```
Token{->MARKONCE(MBTO00001188,1,2)} Token Token{FEATURE("lemma","miner")};
```

Cette règle couvre uniquement l'exemple *citrus leaf*. En effet, en essayant de couvrir les 4 autres exemples, $\text{WHISK}_{\text{Lem}}$ trouve la règle suivante :

```
Token{FEATURE("lemma","leaf")->MARKONCE(MBTO00001188)};
```

Cette règle couvre bien les 4 autres exemples mais couvre aussi le mot *leaf* dans l'exemple *citrus leaf*, ce qui est une erreur. L'algorithme essaie donc d'étendre cette règle.

Voici la trace de l'extension de cette règle.

```
----------------------------
BEST EXTENSION IS: Token{FEATURE("lemma", "leaf")->MARKONCE(Habitat)};
Laplacian: 0.3333333333333333    ; p=4; n=1
Testing 13 rules on training set...
Token # Token{FEATURE("lemma", "leaf")->MARKONCE(Habitat)};     p=4; n=1
Token{FEATURE("lemma", "citrus")} # Token{FEATURE("lemma", "leaf")->MARKONCE(Habitat)}; p=1; n=1
Token{FEATURE("lemma", ".")} # Token{FEATURE("lemma", "leaf")->MARKONCE(Habitat)};      p=2; n=1
Token{FEATURE("lemma", "the")} # Token{FEATURE("lemma", "leaf")->MARKONCE(Habitat)};    p=3; n=1
Token{FEATURE("lemma", "canker")} # Token{FEATURE("lemma", "leaf")->MARKONCE(Habitat)}; p=1; n=0
Token{FEATURE("lemma", "fruit")} # Token{FEATURE("lemma", "leaf")->MARKONCE(Habitat)};  p=2; n=0
Token{FEATURE("lemma", "leaf")->MARKONCE(Habitat)} Token;       p=4; n=1
Token{FEATURE("lemma", "leaf")->MARKONCE(Habitat)} Token{FEATURE("lemma", "and")};      p=1; n=0
Token{FEATURE("lemma", "leaf")->MARKONCE(Habitat)} # Token;     p=4; n=1
Token{FEATURE("lemma", "leaf")->MARKONCE(Habitat)} # Token{FEATURE("lemma", "stem")};   p=1; n=0
Token{FEATURE("lemma", "leaf")->MARKONCE(Habitat)} # Token{FEATURE("lemma", ",")};      p=4; n=1
Token{FEATURE("lemma", "leaf")->MARKONCE(Habitat)} # Token{FEATURE("lemma", "lead")};   p=1; n=0
Token{FEATURE("lemma", "leaf")->MARKONCE(Habitat)} # Token{FEATURE("lemma", "to")};     p=2; n=0
----------------------------
BEST EXTENSION IS: Token # Token{FEATURE("lemma", "leaf")->MARKONCE(Habitat)};
Laplacian: 0.3333333333333333    ; p=4; n=1
----------------------------
```

L'algorithme trouve 6 extensions possibles qui ne font pas d'erreurs ($n = 0$) sur les données d'entraînement mais ne les garde pas car elles sont plus mauvaises que la règle de départ selon la fonction de préférence de l'algorithme (Laplacien). La meilleure extension que l'algorithme retient est donc :

```
Token # Token{FEATURE("lemma","leaf")->MARKONCE(MBTO00001188)};
```

Comme cette règle a les mêmes performances que la règle de départ qui ne peut être acceptée à cause de son score qui dépasse la limite autorisée (Laplacien>0.1), elle n'est pas gardée.

Par conséquent, l'algorithme ne garde aucune règle pour les 4 exemples dont le lemme est *leaf*. L'unique règle qu'il garde (pour le 5^{eme} exemple) n'a aucune correspondance dans les données de développement. $\text{WHISK}_{\text{Lem}}$ obtient donc 0% de F-mesure pour la catégorie `MBTO:00001188`.

Pour d'autres catégories qui disposent de plus d'exemples d'entraînement, l'algorithme $\text{WHISK}_{\text{Lem}}$ obtient de meilleurs résultats (pour `MBTO:00001402` (74 exemples d'apprentissage), il obtient 94% de F-mesure contre 91% pour $\text{WHISK}_{\text{Mot}}$).

Les règles résultant de $\text{WHISK}_{\text{Pos}}$ sont souvent trop générales. Elles font, par conséquent, beaucoup d'erreurs. En essayant de les étendre, elles deviennent trop spécifiques et font du surapprentissage. Ce sont les raisons pour lesquelles ces règles obtiennent de très mauvais résultats. Les règles basées uniquement sur des informations d'étiquetage morpho-syntaxique ne peuvent en aucun cas être utilisées dans nos tâches d'EI.

Nous pouvons également remarquer à partir de la table 8.1 que le fait de combiner deux traits linguistiques ou plus améliore les résultats. Les meilleures performances ont été atteintes par $\text{WHISK}_{\text{LemPos}}$ qui a augmenté la F-mesure de 4 points. Dans le cadre de notre participation à la la sous-tâche 1 du BB BioNLP-ST 2013 (Bossy et al., 2013), nous avons mis en place une approche d'annotation sémantique au regard d'ontologies qui combine une méthode de projection d'une ontologie et un algorithme d'apprentissage de règles d'EI (Bannour, Audibert, & Soldano, 2013). Cette approche dans laquelle nous avons utilisé l'algorithme $\text{WHISK}_{\text{Mot}}$ comme algorithme d'apprentissage de règles nous a permis d'être classés premiers sur cette tâche [2]. Le fait de remplacer $\text{WHISK}_{\text{Mot}}$ par $\text{WHISK}_{\text{LemPos}}$ dans cette approche nous a permis d'améliorer davantage les résultats en diminuant la mesure SER sur laquelle les participants ont été évalués de 4.5 points.

Si nous prenons le même exemple précédent, nous remarquons que le fait d'ajouter les étiquettes morpho-syntaxiques a permis à $\text{WHISK}_{\text{LemPos}}$ d'obtenir 71% de F-mesure (contre 0% pour $\text{WHISK}_{\text{Lem}}$). Les règles apprises par $\text{WHISK}_{\text{LemPos}}$ sont les suivantes :

```
1-Token{FEATURE("postag","VBP")} # Token{FEATURE("lemma","leaf")
  ->MARKONCE(MBTO00001188)};
2-Token{FEATURE("postag","VBG")} # Token{FEATURE("lemma","leaf")
  ->MARKONCE(MBTO00001188)};
3-Token{->MARKONCE(MBTO00001188,1,2)} Token Token{FEATURE("lemma","miner")};
```

Les deux premières règles sont, en effet, deux extensions de l'ancienne règle non prise :

```
Token{FEATURE("lemma","leaf")->MARKONCE(MBTO00001188)};
```

Elles ont été gardées car leurs scores (selon la fonction de préférence de l'algorithme) sont meilleurs que la règle initiale.

2. `http://2013.bionlp-st.org/tasks/bacteria-biotopes`

Nous pouvons aussi remarquer à partir de la table 8.1 que la lemmatisation permet d'obtenir des règles génériques et donc de réduire le nombre total des règles inférées par l'algorithme d'apprentissage de règles. $\text{WHISK}_{\text{Lem}}$ induit 18 règles de moins que $\text{WHISK}_{\text{Mot}}$ qui représente la version la plus précise (78% de précision) et aussi celle qui induit le nombre le plus élevé de règles. Pour la même précision que $\text{WHISK}_{\text{Mot}}$, $\text{WHISK}_{\text{LemPos}}$ infère moins de règles (25 règles de moins que $\text{WHISK}_{\text{Mot}}$). Nous notons également que le fait de combiner les étiquettes morpho-syntaxiques à une autre information linguistique permet d'améliorer la généricité des règles et réduit par conséquent le nombre total de règles induites. $\text{WHISK}_{\text{MotPos}}$ infère 14 règles de moins que $\text{WHISK}_{\text{Mot}}$ et $\text{WHISK}_{\text{LemPos}}$ infère 7 règles de moins que $\text{WHISK}_{\text{Lem}}$.

La figure 8.1 montre les valeurs de précision et de rappel obtenues par l'algorithme $\text{WHISK}_{\text{LemPos}}$ en fonction du nombre d'exemples d'entraînement.

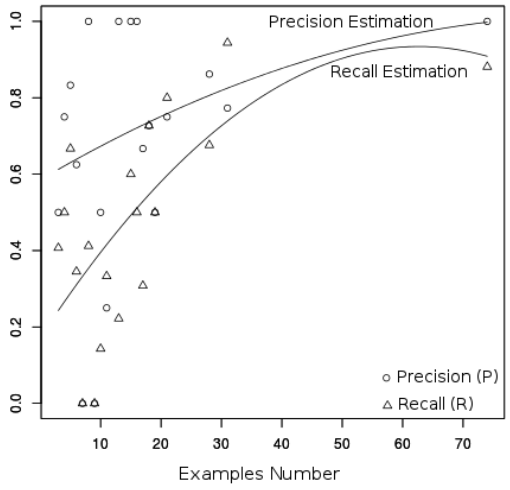


FIGURE 8.1 – Précision et rappel de $\text{WHISK}_{\text{LemPos}}$ en fontion du nombre d'exemples d'entraînement.

Ces valeurs sont très bruitées. Afin de pouvoir schématiser leurs courbes d'estimation, nous avons harmonisé les données en utilisant la méthode statistique LOESS (Cleveland, 1979).

Nous pouvons observer à partir des courbes que $\text{WHISK}_{\text{LemPos}}$ atteint de bonnes valeurs de précision et de rappel en dessus de 40 exemples. Malheureusement, les données d'entraînement contiennent une seule catégorie avec plus de 40 exemples et le nombre moyen d'exemples pour les catégories qui disposent de plus de 3 exemples est 7.5. Ceci explique, en effet, pourquoi le rappel (R_3) n'est pas élevé dans la table 8.1.

Cette première expérience a, en effet, été réalisée en suivant une méthode de validation simple étant donné que l'apprentissage s'est fait sur une partie (>60%) du corpus total (corpus d'entraînement) et le test sur le reste du corpus (corpus de développement).

Pour confirmer ces résultats, nous avons réalisé une deuxième expérimentation.

Expérience 2

Contrairement à la première expérience où il s'agissait d'apprendre les catégories d'habitats de bactéries une par une, cette deuxième expérience vise l'apprentissage du concept générique *Habitat* en considérant uniquement les habitats de bactéries dont les catégories sont représentées par au moins 20 exemples dans le texte (471 parmi 948 exemples) aussi bien dans l'apprentissage que dans le test (d'où les mentions R_{20} et F_{20} pour le rappel et la F-mesure dans la table 8.2). Sont évalués dans cette deuxième expérience les versions de $\text{WHISK}_\text{R}^{Ext}$ suivantes : WHISK_Mot, WHISK_Lem, $\text{WHISK}_\text{LemPos}$ et $\text{WHISK}_\text{MotLemPos}$.

Pour pouvoir apprendre sur la totalité des exemples d'apprentissage, nous avons utilisé un processus de validation croisée 10 fois. La validation croisée 10 fois consiste à diviser le copus total en 10 échantillons, sélectionner un des 10 échantillons comme ensemble de test et utiliser les 9 échantillons qui restent comme ensemble d'apprentissage. Cette opération est répétée 10 fois pour que chaque échantillon soit utilisé une fois comme ensemble de test. Les performances sont calculées en faisant la moyenne des performances obtenues lors des 10 tests.

La table 8.2 représente les performances moyennes résultant de cette deuxième expérience réalisée.

Version de $\text{WHISK}_\text{R}^{Ext}$	**Nombre de règles**	**vp**	**P**	**R_{20}**	**F_{20}**
WHISK_Mot (initial)	97.5	29.2	85%	62%	71%
WHISK_Lem	96.1	30	85%	64%	73%
$\text{WHISK}_\text{LemPos}$	95.2	30	85%	64%	73%
$\text{WHISK}_\text{MotLemPos}$	96.9	31.6	84%	67%	75%

Table 8.2 – Performances de WHISK_Mot, WHISK_Lem, $\text{WHISK}_\text{LemPos}$ et $\text{WHISK}_\text{MotLemPos}$. vp correspond au nombre d'annotations d'habitats prédites correctement dans le corpus de développement. P correspond à la valeur de précision sur les données de développement et R_{20} et F_{20} correspondent respectivement aux valeurs de rappel et F-mesure pour les catégories sur lesquelles les différentes versions de $\text{WHISK}_\text{R}^{Ext}$ utilisées ont été entraînées.

Nous remarquons d'après cette expérience que le problème de WHISK_Lem rencontré lors de la première expérience a été résolu dans la mesure où on dispose d'au moins 20 exemples pour chaque catégorie d'habitat de bactéries considérée dans l'ensemble d'apprentissage. Il obtient les mêmes performances que $\text{WHISK}_\text{LemPos}$. La différence entre les performances de WHISK_Mot et celles de $\text{WHISK}_\text{LemPos}$, en revanche, n'est plus aussi claire que dans la première expérience. Même si F_{20} de $\text{WHISK}_\text{LemPos}$ reste un peu supérieure à celle de WHISK_Mot, nous ne pouvons pas dire que $\text{WHISK}_\text{LemPos}$ est meilleur que WHISK_Mot. Un *t-test* qui montre que la différence de F-mesure entre les deux algorithmes à un degré de confiance égal à 90% n'est pas significative. En effet, pour les catégories bien représentées dans le corpus (disposant d'au moins 20 exemples), toutes les versions de $\text{WHISK}_\text{R}^{Ext}$ fonctionnent correctement, ce qui explique leurs performances proches. Elles induisent également presque le même nombre de règles.

8.2.2 Cas d'usage 2 : désambiguïsation lexicale

La désambiguïsation lexicale est la tâche qui consiste à choisir la bonne significaion d'un mot polysémique dans un contexte donné.

Nous considérons, dans ce travail, que la désambiguïsation lexicale peut être une tâche particulière d'EI. En effet, il s'agit de repérer, dans le texte, un seul mot mais qui peut avoir plusieurs sens. Pour un mot donné, nous cherchons à annoter uniquement le sens (lexie) majoritaire. Ceci constitue un défi pour un algorithme d'EI car plusieurs contre exemples s'expriment de la même manière que les exemples d'apprentissage, ce qui rend la tâche d'apprentissage plus délicate.

La table 8.3 donne quelques statistiques sur les deux mots du corpus SyntSem sur lesquels portent nos expérimentations.

Mot	**Nombre de lexies**	**Nombre total d'exemples**	**Nombre d'exemples de la lexie majoritaire**
Compagnie	12	412	294
Organe	6	366	140

TABLE 8.3 – Statistiques sur les mots à extraire.

— **Compagnie** : il s'agit d'annoter dans le texte 294 parmi 412 occurrences du mot « compagnie » dont le sens est décrit dans la figure 8.2.

FIGURE 8.2 – Lexie majoritaire du mot « compagnie ».

— **Organe** : il s'agit d'annoter dans le texte 140 parmi 366 occurrences du mot « organe » dont le sens est décrit dans la figure 8.3.

Nous réalisons donc deux expérimentations : la première a comme but l'apprentissage de la lexie majoritaire du mot « compagnie » et la deuxième vise l'apprentissage de la lexie majoritaire du mot « organe ». Les deux expérimentations sont effectuées en suivant un processus de validation croisée 10 fois pour les versions suivantes de $\text{WHISK}_{\text{R}}{}^{Ext}$: $\text{WHISK}_{\text{Mot}}$ (initial), $\text{WHISK}_{\text{Lem}}$, $\text{WHISK}_{\text{Pos}}$, $\text{WHISK}_{\text{MotLem}}$, $\text{WHISK}_{\text{MotPos}}$, $\text{WHISK}_{\text{LemPos}}$ et $\text{WHISK}_{\text{MotLemPos}}$. Les tables 8.4 et 8.5 présentent les performances moyennes obtenues respectivement dans la première et la deuxième expérimentation.

Dans la première expérimentation où il s'agit de reconnaître la lexie majoritaire du mot « compagnie » représentée par 71% du nombre d'exemples qui représentent

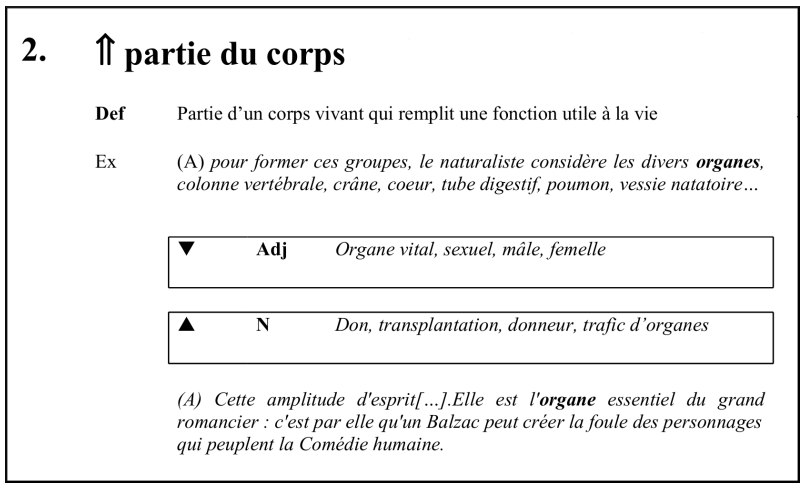


Figure 8.3 – Lexie majoritaire du mot « organe ».

Version de WHISK$_R^{Ext}$	**Nombre de règles**	**vp**	**P**	**R**	**F**
WHISK$_{Mot}$ (initial)	41.1	25.2	**89%**	85%	86%
WHISK$_{Lem}$	40	26.9	79%	91%	83%
WHISK$_{Pos}$	125	2.9	25%	1%	14%
WHISK$_{MotLem}$	34.1	26.7	83%	91%	86%
WHISK$_{MotPos}$	32	26	88%	89%	87%
WHISK$_{LemPos}$	**27.6**	**28.1**	84%	**96%**	**88%**
WHISK$_{MotLemPos}$	27.9	27.7	85%	94%	88%

Table 8.4 – Performances moyennes des versions de WHISK$_R^{Ext}$ utilisées dans la tâche d'apprentissage de la lexie majoritaire du mot « compagnie ». vp correspond au nombre moyen d'annotations prédites correctement dans le corpus de test. P, R et F correspondent respectivement aux valeurs moyennes de précision, rappel et F-mesure sur les données de test.

Version de WHISK$_R^{Ext}$	**Nombre de règles**	**vp**	**P**	**R**	**F**
WHISK$_{Mot}$ (initial)	58.4	6.1	71%	53%	50%
WHISK$_{Lem}$	51.8	6.2	**72%**	54%	52%
WHISK$_{Pos}$	71.5	0.5	11%	4%	5%
WHISK$_{MotLem}$	54.1	7.5	67%	63%	54%
WHISK$_{MotPos}$	56.4	6.7	66%	53%	50%
WHISK$_{LemPos}$	**51.1**	**7.8**	69%	66%	60%
WHISK$_{MotLemPos}$	55.8	7.2	66%	**75%**	**63%**

Table 8.5 – Performances moyennes des versions de WHISK$_R^{Ext}$ utilisées dans la tâche d'apprentissage de la lexie majoritaire du mot « organe ». vp correspond au nombre moyen d'annotations prédites correctement dans le corpus de test. P, R et F correspondent respectivement aux valeurs moyennes de précision, rappel et F-mesure sur les données de test.

le concept en général, conformément aux résultats des expérimentations sur l'apprentissage des habitats de bactéries, les différentes versions étendues de WHISK_{R} se comportent de la même manière et $\text{WHISK}_{\text{LemPos}}$ obtient la meilleure F-mesure (à égalité avec $\text{WHISK}_{\text{MotLemPos}}$) même si elle ne dépasse celle de $\text{WHISK}_{\text{Mot}}$ que de deux points en F-mesure. Pour vérifier si ces résultats sont significatifs, nous avons calculé les scores *t-test* relatifs à la F-mesure (*paired t-test*) pour pouvoir comparer deux à deux entre elles les différentes versions de ${\text{WHISK}_{\text{R}}}^{Ext}$ testées.

Le *t-test* révèle, dans la première expérimentation, les assertions suivantes :

— $\text{WHISK}_{\text{Pos}}$ est moins performant que toutes les autres versions de ${\text{WHISK}_{\text{R}}}^{Ext}$ testées avec un niveau de confiance égal à 100% ;
— $\text{WHISK}_{\text{Mot}}$ est plus performant que $\text{WHISK}_{\text{Lem}}$ avec un niveau de confiance égal à 90% ;
— $\text{WHISK}_{\text{LemPos}}$ est plus performant que $\text{WHISK}_{\text{Lem}}$ avec un niveau de confiance égal à 90%.

Dans la deuxième expérimentation où il s'agit de reconnaître la lexie majoritaire du mot « organe » représentée par 38% du nombre d'exemples qui représentent le concept en général, en revanche, les différents algorithmes testés peinent à distinguer la lexie majoritaire du mot « organe » des autres lexies, ce qui explique les valeurs relativement faibles de F-mesure (63% au meilleur des cas obtenue par $\text{WHISK}_{\text{MotLemPos}}$). Même si $\text{WHISK}_{\text{LemPos}}$ n'obtient pas les meilleures performances dans cette expérimentation, il dépasse largement $\text{WHISK}_{\text{Mot}}$ (10 points de plus en F-mesure).

Le *t-test* relatif à la F-mesure réalisé également pour cette deuxième expérimentation, révèle les assertions suivantes :

— $\text{WHISK}_{\text{Pos}}$ est moins performant que toutes les autres versions de ${\text{WHISK}_{\text{R}}}^{Ext}$ testées avec un niveau de confiance égal à 100% ;
— à part $\text{WHISK}_{\text{LemPos}}$, $\text{WHISK}_{\text{MotLemPos}}$ est meilleur que toutes les autres versions de ${\text{WHISK}_{\text{R}}}^{Ext}$ testées avec un niveau de confiance supérieur à 90%.

Concernant le nombre de règles inférées, $\text{WHISK}_{\text{LemPos}}$ induit le moins de règles respectivement dans les expérimentations 1 et 2 avec respectivement environ 14 règles et 7 règles de moins que $\text{WHISK}_{\text{Mot}}$.

À part l'élimination de $\text{WHISK}_{\text{Pos}}$, le calcul des scores *t-test* dans les deux expérimentations ne donne pas les mêmes conclusions.

Nous pouvons remarquer d'après les expérimentations réalisées dans cette section que :

— Plus on ajoute des traits linguistiques aux règles, plus elles sont générales (meilleur rappel) et plus elles sont performantes par conséquent (meilleure F-mesure).
— $\text{WHISK}_{\text{LemPos}}$ a toujours un rappel nettement supérieur à celui de $\text{WHISK}_{\text{Mot}}$ à valeurs de précision proches, ce qui fait de lui la bonne version de ${\text{WHISK}_{\text{R}}}^{Ext}$ à utiliser quand on cherche à privilégier le rappel.
— $\text{WHISK}_{\text{Mot}}$ obtient les meilleures valeurs de précision, ce qui fait de lui la bonne version de ${\text{WHISK}_{\text{R}}}^{Ext}$ quand on cherche à privilégier la précision. Ceci est compréhensible car $\text{WHISK}_{\text{Mot}}$ cherche les formes exactes des mots à extraire, ce qui diminue le risque d'erreur.

8.3 Réduction de la fenêtre de contexte à la phrase et apprentissage sur un corpus réduit : évaluation du gain en temps d'apprentissage

De nos jours, on parle moins du facteur temps dans l'évaluation des algorithmes d'apprentissage car le matériel informatique évolue tellement rapidement (loi de Moore) que les problèmes de lenteur diagnostiqués aujourd'hui ne le seront plus dans quelques années. Cependant, il est important de mentionner que les algorithmes d'apprentissage de règles peuvent être lents. Cette lenteur est souvent liée au langage dans lequel sont apprises les règles et aux mécanismes d'application/test de ces règles.

Le facteur temps d'apprentissage est très important dans notre travail car il s'agit d'un système interactif où l'utilisateur ne devrait pas attendre longtemps la réponse du module d'apprentissage de règles. Comme la rapidité de l'apprentissage de règles dépend surtout du temps mis pour tester les règles (la construction des règles est relativement rapide), il est important que le langage de règles soit optimisé. Le langage Ruta, étant un langage récent, possède encore une grande marge de progression. Les deux dernières versions du langage (2.2.1 et 2.2.0) sont d'ailleurs 3 à 17 fois plus rapides que les premières versions.

Les stratégies de réduction de la fenêtre d'apprentissage à la phrase et d'apprentissage sur un corpus réduit que nous avons proposées dans le chapitre 6 pour construire nos règles d'EI permettent, en effet, de réduire davantage et significativement le temps d'apprentissage de règles sans dégrader les performances.

Pour évaluer ce gain en temps d'apprentissage, nous avons réalisé les trois expériences suivantes.

Expérience 1 : elle consiste à apprendre, dans le corpus BB BioNLP-ST 2013, les habitats de bactéries restreints à ceux dont les catégories sont représentées par au moins 20 exemples (471 exemples en total) en utilisant un processus de validation croisée 10 fois. Pour construire les règles, nous forçons l'algorithme d'apprentissage de règles à utiliser une fenêtre d'apprentissage égale à 5 (fenêtre d'apprentisage par défaut pour l'algorithme WHISK_R). Autrement dit, pour construire son espace de recherche de règles, l'algorithme d'apprentissage peut explorer jusqu'à 5 mots avant et 5 mots après le terme cible à extraire.

Expérience 2 : elle consiste à reproduire l'expérience 1 mais en ne gardant dans le fenêtre d'apprentissage que les mots contenus dans la phrase si la fenêtre dépasse la phrase.

Expérience 3 : elle consiste à reproduire l'expérience 2 mais au lieu d'apprendre sur tout le corpus d'apprentissage, on apprend uniquement sur les phrases qui contiennent des exemples positifs. Autrement dit, l'algorithme d'apprentissage évalue les règles construites dans son espace de recherche de règles uniquement sur les phrases qui contiennent des exemples positifs. En effet, dans une itération donnée du processus d'apprentissage interactif de règles d'EI, l'utilisateur peut disposer d'un petit ensemble d'apprentissage et ne veut pas que l'algorithme d'apprentissage discrimine le reste des exemples en

les considérant tous comme négatifs alors qu'ils peuvent contenir des positifs. Dans ce cas, l'une des solutions que nous avons proposées dans le chapitre 6 est l'apprentissage sur un corpus réduit qui comprend uniquement les phrases vues par l'utilisateur. Pour évaluer l'apport de cette stratégie, nous nous positionnons, dans cette expérience, dans le cas où l'utilisateur a annoté toutes les phrases qui contiennent des exemples positifs.

Nous avons réalisé ces 3 expériences en utilisant les quatre algorithmes $\text{WHISK}_{\text{Mot}}$, $\text{WHISK}_{\text{Lem}}$,$\text{WHISK}_{\text{LemPos}}$ et $\text{WHISK}_{\text{MotLemPos}}$, leurs versions s'appuyant sur un contexte d'apprentissage réduit à la phrase et leurs versions permettant d'apprendre sur un corpus réduit. Les résultats en termes de temps d'apprentissage, précision, rappel et F-mesure sont notés dans les tables 8.6, 8.7 et 8.8. Notons que les valeurs présentées dans ces tables sont des valeurs moyennes (moyenne des 10 tests réalisés lors du processus de valisation croisée 10 fois).

Version de $\text{WHISK}_{\text{R}}^{Ext}$	**Temps d'apprentissage**	**P**	**R_{20}**	**F_{20}**
$\text{WHISK}_{\text{Mot}}$ (initial)	05h43m56s	85%	64%	72%
$\text{WHISK}_{\text{Lem}}$	05h36m17s	85%	65%	73%
$\text{WHISK}_{\text{LemPos}}$	05h03m58s	85%	65%	73%
$\text{WHISK}_{\text{MotLemPos}}$	07h51m34s	84%	67%	75%

TABLE 8.6 – Temps d'apprentissage moyen et performances moyennes de $\text{WHISK}_{\text{Mot}}$, $\text{WHISK}_{\text{Lem}}$, $\text{WHISK}_{\text{LemPos}}$ et $_{\text{MotLemPos}}$ dans l'expérience 1. P correspond à la valeur moyenne de précision sur le corpus de test et R_{20} et F_{20} correspondent respectivement aux valeurs moyennes de rappel et de F-mesure pour les catégories sur lesquelles les différentes versions de $\text{WHISK}_{\text{R}}^{Ext}$ utilisées ont été entraînées.

Version de $\text{WHISK}_{\text{R}}^{Ext}$	**Temps d'apprentissage**	**P**	**R_{20}**	**F_{20}**
$\text{WHISK}_{\text{Mot}}^{P}$ (initial)	05h03m36s	85%	62%	71%
$\text{WHISK}_{\text{Lem}}^{P}$	04h21m39s	85%	64%	73%
$\text{WHISK}_{\text{LemPos}}^{P}$	04h29m28s	85%	64%	73%
$\text{WHISK}_{\text{MotLemPos}}^{P}$	06h40m40s	84%	67%	75%

TABLE 8.7 – Temps d'apprentissage moyen et performances moyennes de $\text{WHISK}_{\text{Mot}}^{P}$, $\text{WHISK}_{\text{Lem}}^{P}$, $\text{WHISK}_{\text{LemPos}}^{P}$ et $_{\text{MotLemPos}}^{P}$ dans l'expérience 2. P correspond à la valeur moyenne de précision sur le corpus de test et R_{20} et F_{20} correspondent respectivement aux valeurs moyennes de rappel et de F-mesure pour les catégories sur lesquelles les différentes versions de $\text{WHISK}_{\text{R}}^{Ext}$ utilisées ont été entraînées.

Nous observons un léger gain dans le temps d'apprentissage entre l'expérience 1 (table 8.6) et l'expérience 2 (table 8.7). Ce gain est évalué à 15% en moyenne (voir colonne GTA dans la table 8.9) et s'accompagne d'un maintien des mêmes performances. En effet, même si les valeurs de rappel baissent légèrement, ce qui est tout à fait compréhensible étant donné qu'on réduit l'espace de recherche de règles

Version de WHISK$_{\mathbf{R}}^{Ext}$	**Temps d'apprentissage**	**P**	**$\mathbf{R_{20}}$**	**$\mathbf{F_{20}}$**
WHISK$_{\text{Mot}}^{CR}$ (initial)	00h49m05s	79%	68%	72%
WHISK$_{\text{Lem}}^{CR}$	00h45m46s	78%	70%	73%
WHISK$_{\text{LemPos}}^{CR}$	00h57m09s	78%	70%	73%
WHISK$_{\text{MotLemPos}}^{CR}$	01h28m33s	76%	71%	73%

Table 8.8 – Temps d'apprentissage moyen et performances moyennes de WHISK$_{\text{Mot}}^{CR}$, WHISK$_{\text{Lem}}^{CR}$, WHISK$_{\text{LemPos}}^{CR}$ et $_{\text{MotLemPos}}^{CR}$ dans l'expérience 3. P correspond à la valeur moyenne de précision sur le corpus de test et R_{20} et F_{20} correspondent respectivement aux valeurs moyennes de rappel et de F-mesure pour les catégories sur lesquelles les différentes versions de WHISK$_{\text{R}}^{Ext}$ utilisées ont été entraînées.

en réduisant la fenêtre d'apprentissage, cette faible baisse n'affecte pas la F-mesure qui reste pratiquement inchangée.

Version de WHISK$_{\mathbf{R}}^{Ext}$	**GTA/expérience 1**
WHISK$_{\text{Mot}}^{P}$ (initial)	11.73%
WHISK$_{\text{Lem}}^{P}$	22.19%
WHISK$_{\text{LemPos}}^{P}$	11.35%
WHISK$_{\text{MotLemPos}}^{P}$	15.03%
Moyenne	15.07%

Table 8.9 – Gain en temps d'apprentissage (GTA) dans l'expérience 2 par rapport à l'expérience 1 pour les algorithmes WHISK$_{\text{Mot}}^{P}$, WHISK$_{\text{Lem}}^{P}$, WHISK$_{\text{LemPos}}^{P}$ et WHISK$_{\text{MotLemPos}}^{P}$.

Concernant le corpus BB BioNLP-ST 2013 qui contient 1466 phrases dont 340 contiennent des exemples concernés par les expériences 1, 2 et 3, nous pouvons dire qu'en apprenant, dans l'expérience 3, sur uniquement 23% des phrases du corpus, nous avons pu gagner environ 84% en temps d'apprentissage par rapport à l'expérience 1, ce qui constitue un gain considérable d'autant que la F-mesure reste inchangée. Cette expérience a permis, en effet, de rééquilibrer les valeurs de précision et de rappel sans perdre en F-mesure. La précision a baissé, en moyenne, d'environ 7 points. Ceci s'explique par le fait qu'en évaluant ses règles sur un échantillon du corpus, l'algorithme d'apprentissage ignore les exemples négatifs que ces dernières peuvent couvrir, d'où le manque de précision de ces règles. Le rappel, en revanche, s'est élevé, en moyenne, de 6 points. En effet, comme WHISK commence par les règles les plus générales pour les spécifier par la suite, les règles construites sur un petit échantillon du corpus ont tendance a être génériques car elles font peu d'erreurs sur cet échantillon.

Version de WHISK$_{\mathbf{R}}{}^{Ext}$	**GTA/expérience 1**	**GTA/expérience 2**
WHISK$_{\text{Mot}}{}^{CR}$ (initial)	85.73%	83.83%
WHISK$_{\text{Lem}}{}^{CR}$	86.39%	82.51%
WHISK$_{\text{LemPos}}{}^{CR}$	81.20%	78.79%
WHISK$_{\text{MotLemPos}}{}^{CR}$	81.22%	77.90%
Moyenne	83.63%	80.76%

TABLE 8.10 – Gain en temps d'apprentissage (GTA) dans l'expérience 3 par rapport aux expériences 1 et 2 pour les algorithmes WHISK$_{\text{Mot}}{}^{CR}$, WHISK$_{\text{Lem}}{}^{CR}$, WHISK$_{\text{LemPos}}{}^{CR}$ et WHISK$_{\text{LemPos}}{}^{CR}$.

8.4 Apports des modules d'apprentissage actif proposés

L'apprentissage actif est le pilier de notre approche interactive car pour réduire au maximum l'effort humain dans une telle approche, l'utilisateur doit être guidé notamment dans le choix des exemples à annoter pour ne pas avoir à annoter tout le corpus. Nous avons donc proposé, dans le chapitre 6, deux modules d'apprentissage actif (IAL4Sets et IAL3Sets) qui étendent le module interactif proposé dans la conception de base de l'algorithme WHISK.

8.4.1 Comparaison des modules IAL4Sets, IAL3Sets et Baseline

Pour évaluer l'apport de nos modules par rapport à celui de WHISK (nous avons implémenté le module interactif de base dans WHISK$_{\text{R}}$ et l'avons considéré comme Baseline), nous avons réalisé une expérience qui consiste à apprendre, dans le corpus BB BioNLP-ST 2013, les habitats de bactéries restreints à ceux dont les catégories sont représentées par au moins 20 exemples (471 exemples en total) en utilisant un processus de validation croisée 10 fois. L'apprentissage se fait de manière progressive et itérative. À chaque itération, les actions suivantes sont exécutées :

- Apprentissage sur les phrases contenant des exemples annotés positivement (corpus réduit) ;
- Application des règles inférées sur le corpus d'apprentissage pour construire l'ensemble des exemples à annoter. Ces exemples sont ensuite proposés à l'utilisateur (oracle) ;
- Annotation des exemples proposés et mise à jour des phrases contenant des exemples annotés positivement ;

Ces actions sont répétées autant de fois que l'utilisateur le souhaite. À chaque itération, les performances d'apprentissage sont calculées sur le corpus de test. L'utilisateur met généralement fin au processus quand les performances deviennent stationnaires.

Pour construire les règles, nous paramétrons l'algorithme d'apprentissage de règles de manière à utiliser une fenêtre d'apprentissage égale à 5 (fenêtre d'appren-

tisage par défaut pour l'algorithme WHISK_R) mais en ne gardant dans le fenêtre d'apprentissage que les mots contenus dans la phrase si la fenêtre dépasse la phrase.

Nous menons cette expérience sur ${\text{WHISK}_\text{Mot}}^{CR}$ et ${\text{WHISK}_\text{LemPos}}^{CR}$. Chacun de ces algorithmes a été testé à la fois avec nos modules d'apprentissage actif (IAL4Sets et IAL3Sets) et celui de base (Baseline). Pour pouvoir comparer nos modules d'apprentissage actif à celui de base, nous les obligeons, à chaque test élémentaire (parmi les 10 tests du processus de validation croisée 10 fois), à démarrer avec le même ensemble d'apprentissage de départ, et ceci pour tous les algorithmes d'apprentissage testés. Les 3 modules d'apprentissage actif ont été testés sur 200 itérations en proposant, à chaque itération, à l'utilisateur 25 exemples à annoter. Ceci correspond au total à 5000 exemples proposés à l'utilisateur à partir d'un corpus d'entrée qui compte 29466 mots.

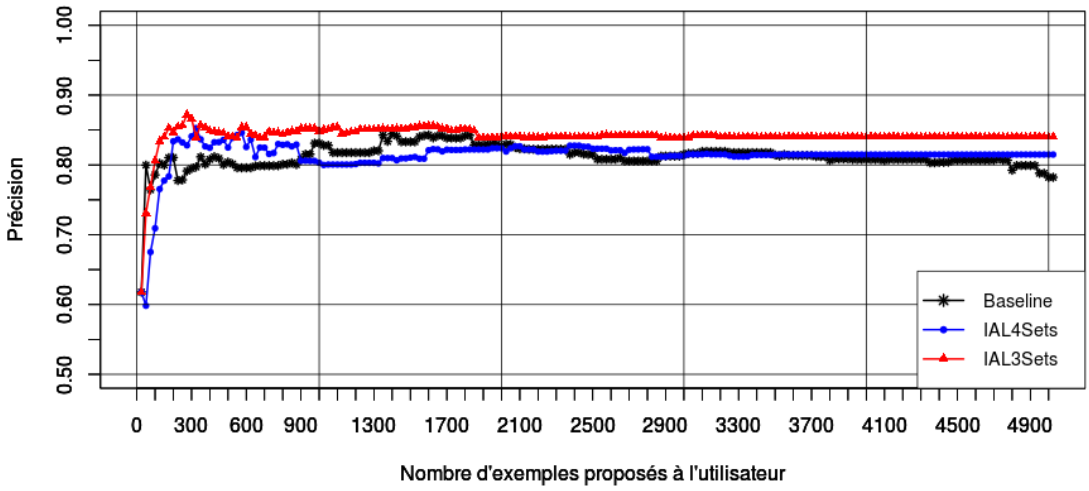
À partir des résultats de cette expérience, nous avons réalisé, pour chaque algorithme d'apprentissage de règles testé, deux figures :

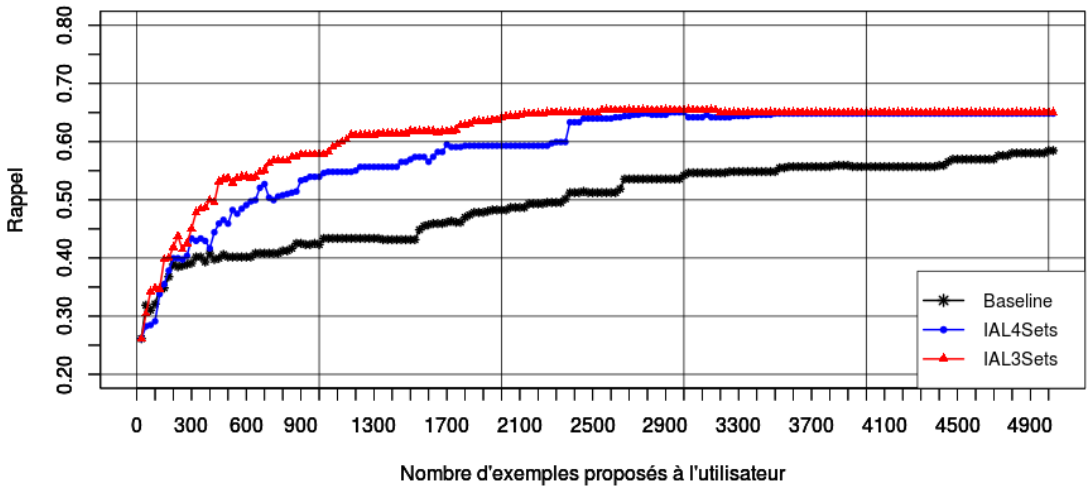
— une figure qui représente les performances (précision, rappel et F-mesure) de l'algorithme d'apprentissage concerné en fonction du nombre d'exemples à annoter proposés à l'utilisateur, à chaque fois, par l'un des modules d'apprentissage actif (Baseline, IAL4Sets et IAL3Sets) (voir figures 8.4 et 8.5) ;

— une figure qui présente le nombre d'exemples positifs annotés en fonction du nombre d'exemples proposés à l'utilisateur, à chaque fois, par l'un des modules d'apprentissage actif (Baseline, IAL4Sets et IAL3Sets) (voir figures 8.6 et 8.7).

Nous remarquons d'après les courbes 8.4 et 8.5 que quelque soit la version de ${\text{WHISK}_\text{R}}^{Ext}$ utilisée, nos deux modules d'apprentissage actif obtiennent des performances meilleures que le module Baseline. Tout en ayant des valeurs de précision presque égales, les modules IAL3Sets et IAL4Sets obtiennent des valeurs de rappel significativement plus élevées que celles du module Baseline entrainant des valeurs plus élevées de F-mesure également. Considérons, par exemple, la figure 8.5 qui présente les performances obtenues par l'algorithme ${\text{WHISK}_\text{LemPos}}^{CR}$. Au bout de 5000 exemples proposés à l'utilisateur, le module d'apprentissage actif Baseline obtient une valeur de rappel égale à 0.66, une valeur que notre module IAL3Sets atteint uniquement au bout de 2925 exemples proposés à l'utilisateur. Ceci montre que IAL3Sets converge de manière significativement plus rapide que le Baseline. Au bout de 5000 exemples proposés à l'utilisateur, IAL3Sets atteint une valeur de rappel égale à 0.71.

Ces constatations se confirment dans les figures 8.6 et 8.7. Par exemple, dans la figure 8.7 qui représente le nombre d'exemples positifs couverts par l'algorithme d'apprentissage $\text{WHISK}_\text{LemPos}$ en fonction des exemples proposés par les modules d'apprentissage actif Baseline, IAL3Sets et IAL4Sets, Baseline arrive à couvrir, en moyenne, 320 exemples positifs au bout de 5000 exemples proposés à l'utilisateur. Ce nombre d'exemples positifs est couvert par le module IAL3Sets seulement au bout de 2850 exemples proposés à l'utilisateur. En proposant à l'utilisateur 5000 exemples, IAL3Sets couvre, en revanche, en moyenne 349 exemples, ce qui correspond à environ 82% des exemples qui peuvent être couverts (424 exemples peuvent être couverts dans l'ensemble d'apprentissage). Ce nombre d'exemples positifs couverts dans l'ensemble d'apprentissage permet d'obtenir, en moyenne, une valeur de





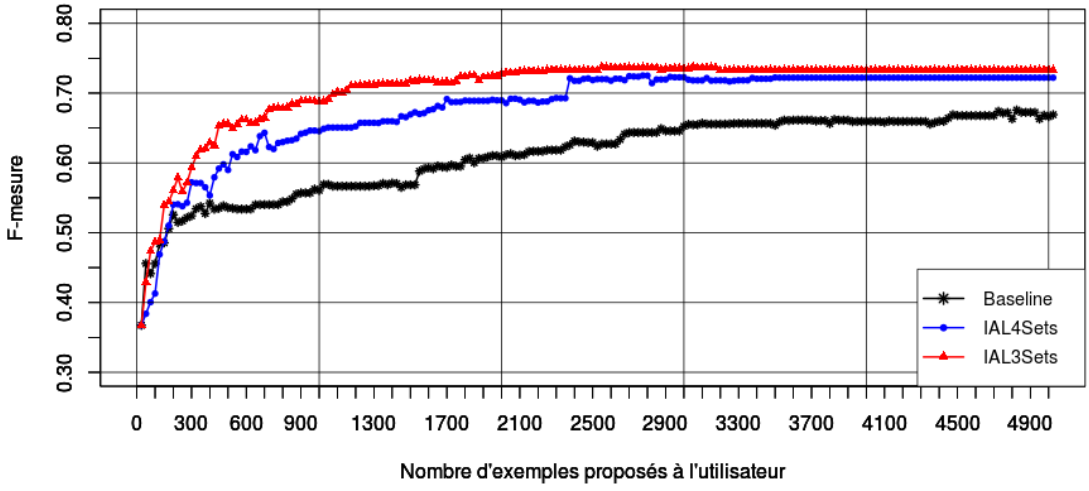


FIGURE 8.4 – Précision, rappel et F-mesure de $\text{WHISK}_{\text{Mot}}{}^{CR}$ en fonction du nombre d'exemples proposés à l'utilisateur par les modules d'apprentissage actif Baseline, IAL4Sets et IAL3Sets.

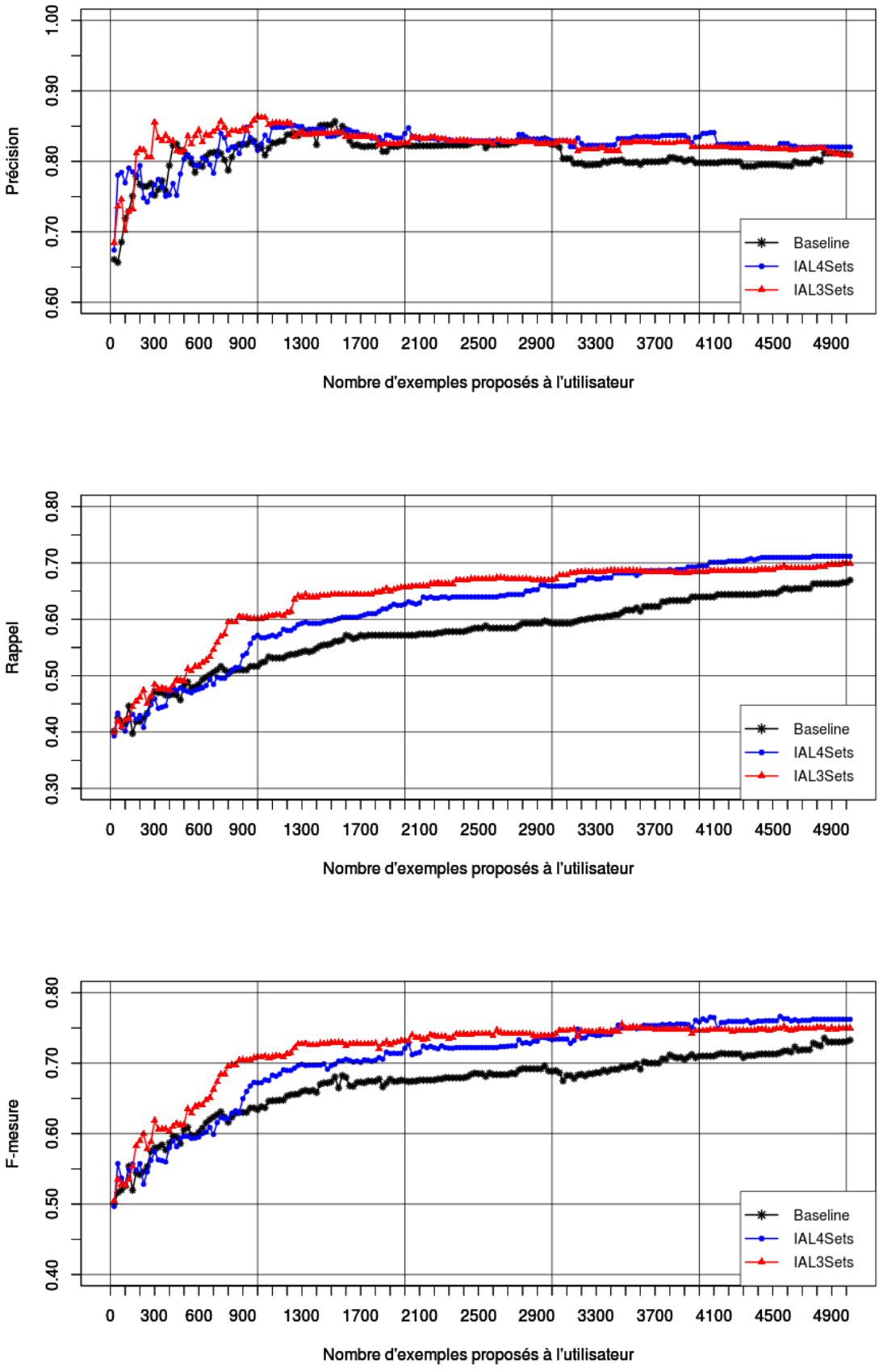


Figure 8.5 – Précision, rappel et F-mesure de $\text{WHISK}_{\text{LemPos}}{}^{CR}$ en fonction du nombre d'exemples proposés à l'utilisateur par les modules d'apprentissage actif Baseline, IAL4Sets et IAL3Sets.

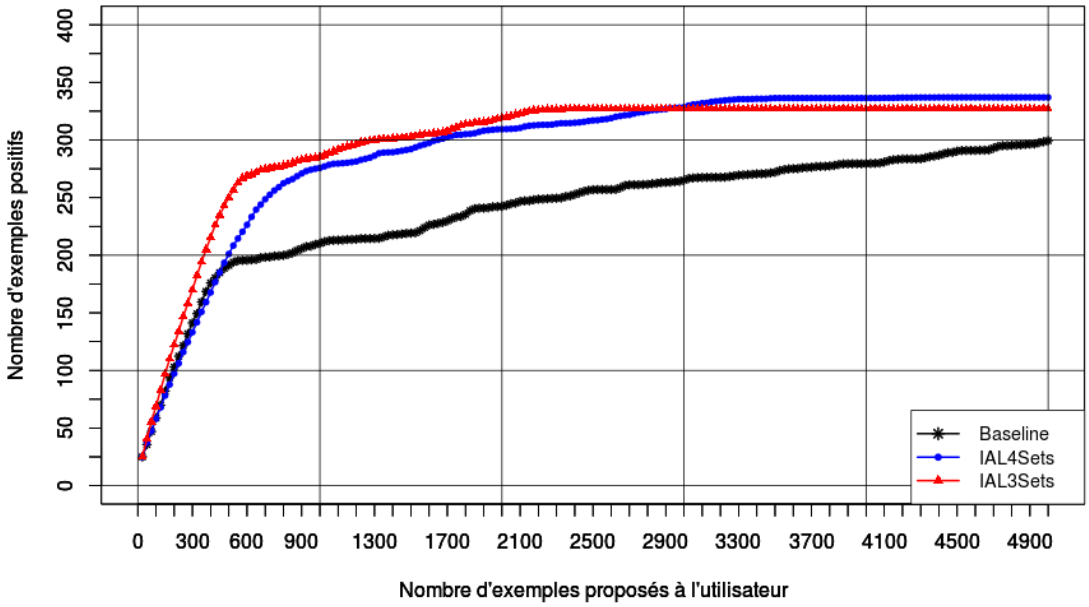


FIGURE 8.6 – Nombre d'exemples positifs couverts par $\text{WHISK}_{\text{Mot}}{}^{CR}$ en fonction du nombre d'exemples proposés à l'utilisateur par les modules d'apprentissage actif Baseline, IAL4Sets et IAL3Sets.

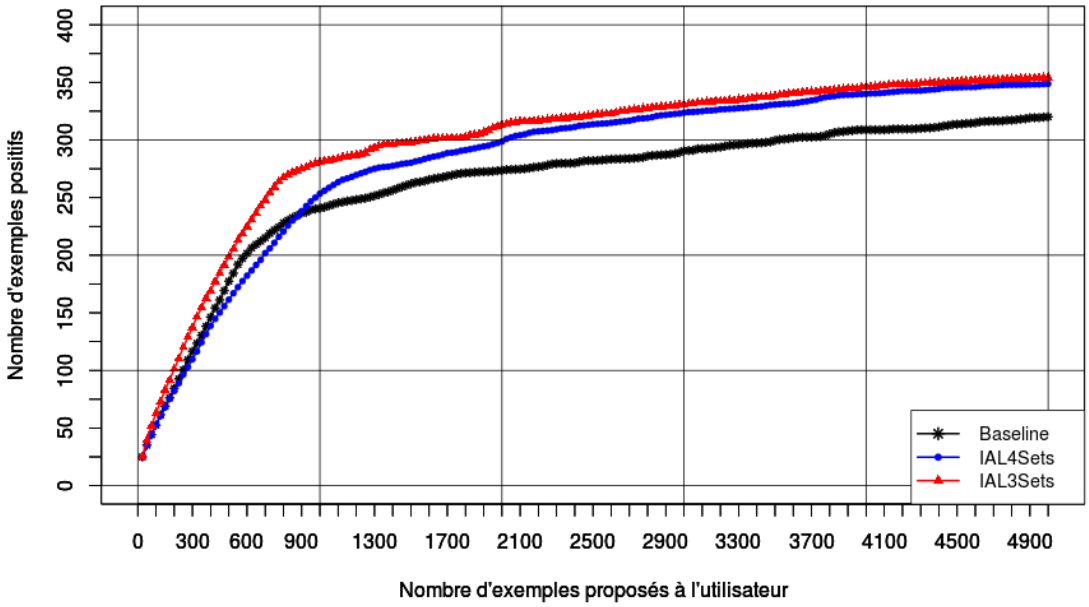


FIGURE 8.7 – Nombre d'exemples positifs couverts par $\text{WHISK}_{\text{LemPos}}{}^{CR}$ en fonction du nombre d'exemples proposés à l'utilisateur par les modules d'apprentissage actif Baseline, IAL4Sets et IAL3Sets.

rappel égale à 0.71, une valeur de précision égale à 0.82 et une valeur de F-mesure égale à 0.76 (voir figure 8.5) sur l'ensemble de test.

Sachant que les valeurs de précision, rappel et F-mesure moyennes obtenues par $\text{WHISK}_{\text{LemPos}}$ dans les expériences 2 et 3 décrites dans la section précédente sont les suivantes.

— **Expérience 2 :** précision = 0.85 ; rappel = 0.64 ; F-mesure = 0.73
— **Expérience 3 :** précision = 0.78 ; rappel = 0.70 ; F-mesure = 0.73

nous pouvons remarquer que $\text{WHISK}_{\text{LemPos}}{}^{CR}$ utilisé dans le cadre de l'apprentissage actif assuré par le module IAL4Sets où il ne s'agit ni d'apprendre sur tout le corpus d'apprentissage (expérience 2) ni d'apprendre uniquement sur les phrases qui contiennent des exemples positifs (expérience 3) mais plutôt d'apprendre sur les phrases qui contiennent des exemples annotés par l'utilisateur au fur et à mesure des itérations, obtient une F-mesure meilleure que celles obtenues dans les expériences 2 et 3. Il obtient, en effet, une valeur de précision proche de la meilleure des deux expériences 2 et 3 et une valeur de rappel supérieure à celles obtenues dans les deux expériences. Ce phénomène est aussi vrai pour tous les autres algorithmes et quelque soit le module d'apprentissage actif utilisé. Ceci montre que le fait d'apprendre sur un corpus réduit, permet non seulement, de réduire le temps d'apprentissage mais également d'améliorer les performances : les règles apprises sont plus précises que celles obtenues en apprenant uniquement sur les phrases contenant des exemples positifs (expérience 3) car les phrases sur lesquelles ces règles ont été évaluées sont supérieures, en nombre, à celles contenant des exemples positifs (phrases consultées par l'utilisateur au fur et à mesure des itérations pour annoter les exemples proposés par le module d'apprentissage actif) et plus génériques que celles obtenues en apprenant sur le corpus en entier (expérience 2) car l'algorithme WHISK est un algorithme descendant qui commence par construire la règle la plus générale pour la spécifier par la suite pour corriger les éventuelles erreurs de la règle. Comme les règles construites sur un échantillon du corpus font moins d'erreurs, les règles gardées sur cet échantillon sont plus génériques que celles qui auraient été gardées pour tout le corpus.

Nous remarquons également à partir des figures 8.4, 8.5, 8.6 et 8.7 que les courbes corespondant au module IAL3Sets proposé montent plus rapidement au cours des premières itérations et finissent par rejoindre voir descendre sous celles qui correspondent au module IAL4Sets. Si nous regardons, par exemple, la figure 8.7, nous pouvons noter, par exemple, qu'au bout de 1125 exemples proposés à l'utilisateur, IAL3Sets couvre en moyenne 280 exemples positifs alors que IAL4Sets couvre 265 exemples à ce stade. IAL4Sets n'atteint les 280 exemples positifs couverts qu'au bout de 1500 exemples annotés par l'utilisateur. Au bout d'environ 2150 exemples annotés par l'utilisateur, les deux modules couvrent, à 2 ou 3 exemples près, le même nombre d'exemples positifs. Ces résultats peuvent s'expliquer par les faits suivants :

— La suppression de l'ensemble R qui représente le reste des exemples du corpus permet de proposer plus d'exemples des 3 autres ensembles, à savoir l'ensemble des exemples couverts par des règles existantes non annotés par l'utilisateur (CnA), l'ensemble des *near misses* des règles existantes (NM) et l'ensemble des exemples les plus proches sémantiquement des exemples couverts par des règles existantes (TSEP). Ces trois ensembles permettent de

retrouver d'une manière rapide les exemples les plus évidents et le fait d'augmenter le nombre d'exemples à annoter parmi ces trois ensembles permet une augmentation plus rapide des performances.

— Une fois les exemples les plus triviaux couverts, les exemples les plus rares sont difficiles à trouver dans les trois premiers ensembles. Le fait de supprimer le quatrième ensemble (l'ensemble R), peut entraîner le fait que ces exemples ne soient jamais retrouvés. Ceci peut expliquer les performances les moins bonnes obtenues par IAL3Sets comparées à celles obtenues par IAL4Sets à partir d'un certain seuil.

Nous retenons de l'expérimentation menée dans cette section que notre module d'apprentissage IAL4Sets obtient les meilleures performances quelque soit la version de $\text{WHISK}_{\text{R}}{}^{Ext}$ utilisée. Ceci est dû à l'ajout de l'ensemble TSEP aux ensembles parmi lesquels le module d'apprentissage actif propose des exemples à annoter à l'utilisateur.

8.4.2 Rôle de l'ensemble TSEP dans l'amélioration de l'apprentissage actif

L'ensemble TSEP que nous avons introduit dans nos modules d'apprentissage actif permet d'ajouter un volet sémantique au module Baseline dont la recherche d'exemples se limite à ceux qui ressemblent aux exemples couverts (CnA et *NM*) sinon au hasard (R). L'ensemble TSEP permet de récupérer des exemples qui ne ressemblent pas aux exemples couverts mais qui y sont liés par une mesure de similarité distributionnelle que nous avons pris le soin de concevoir, de mettre en place et de tester sur différents scénarios.

Prenons un exemple concret qui illustre le rôle de l'ensemble TSEP dans la convergence rapide de l'apprentissage actif.

Considérons l'un des tests de l'expérience menée dans la sous section précédente. L'algorithme considéré dans ce test est $\text{WHISK}_{\text{LemPos}}{}^{CR}$ et les deux modules d'apprentissage actif à comparer sont Baseline et IAL4Sets. Pour pouvoir interpréter les résultats de ce test, nous appelons nEPC le nombre d'exemples positifs couverts dans l'ensemble d'apprentissage (9 ensembles sur 10 dans le cadre d'une validation croisée 10 fois (424 exemples) dans lesquels l'annotation d'exemples se fait de manière incrémentale).

Nous rappelons les notations suivantes.

— CnA : ce sont les exemples couverts par les règles existantes et non encore annotés par l'utilisateur.

— *NM* : ce sont les *near misses* des règles existantes.

— TSEP : ce sont les termes similaires aux exemples positifs couverts par les règles existantes.

— R : il s'agit du reste des exemples non annotés par l'utilisateur.

Nous démarrons l'itération 0 du test avec l'ensemble d'exemples décrit dans la table 8.11. Ces exemples ont été choisis de manière aléatoire parmi l'ensemble total des exemples d'apprentissage.

Nous pouvons remarquer d'après ces exemples que le concept Habitat de bactéries s'exprime de manières très différentes dans le corpus. Les performances initiales

Exemples positifs fournis à l'algorithme d'apprentissage au départ			
person		plants	intracellular
Human		plant	intestinal tract
Human		opportunistic feeders	plants
soil		cells	microorganisms
micro-organism		foodborn	cell
micro-organism		human	human
humans		humans	human
tsetse		soil	cells
intestinal			
	Performances sur le corpus de test		
nEPC	**précision**	**rappel**	**F-mesure**
25	48%	28%	35%

Table 8.11 – Itération 0 du test. nEPC représente le nombre d'exemples positifs couverts dans l'ensemble d'apprentissage.

atteintes par $\text{WHISK}_{\text{LemPos}}{}^{CR}$ à partir des 25 exemples sélectionnés correspondent à une valeur de précision égale à 48%, une valeur de rappel égale à 28% et une valeur de F-mesure égale à 35%.

Dans les itérations suivantes, l'intervention de l'utilisateur pour annoter les exemples proposés par l'un ou l'autre des modules d'apprentissage actif Baseline et IAL4Sets est simulée. Si un exemple proposé figure dans l'ensemble total des exemples positifs dont nous disposons à l'avance, il est annoté comme positif sinon il est annoté comme négatif.

Exemples à annoter proposés à l'utilisateur							
Baseline				IAL4Sets			
CnA	***NM***	**R**		**CnA**	***NM***	**TSEP**	**R**
cells	plants	~~B.~~		soil	~~cell~~	intestinal tract	~~)~~
Humans	plant	~~,~~		human	plant	~~tsetse~~	~~and~~
~~photographs~~	plant	~~is~~		~~cells~~	plant	microorganisms	~~are~~
humans	plants	~~dioxygenases~~		~~species~~	cells	Human	~~of~~
~~soil~~	plant	~~therefore~~		~~cells~~	human	~~wall~~	~~as~~
humans	plant	~~respiratory~~		~~are~~	plant	intracellular	~~the~~
cell	humans	~~virulence~~				~~soil~~	
~~similar~~	plant	~~will~~					
~~cell~~							
Performances sur le corpus de test				Performances sur le corpus de test			
nEPC	**précision**	**rappel**	**F-mesure**	**nEPC**	**précision**	**rappel**	**F-mesure**
38	40%	32%	36%	36	67%	42%	52%

Table 8.12 – Itération 1 du test. nEPC représente le nombre d'exemples positifs couverts dans l'ensemble d'apprentissage.

Dans la première itération du test (voir table 8.12), nous remarquons que, pour les deux modules, à part les exemples proposés dans l'ensemble R qui sont tous annotés comme négatifs, les autres exemples sont tous pertinents même s'ils sont annotés comme négatifs. En effet, ces derniers sont des exemples qui ont presque les mêmes expressions que les exemples de départ. Parmi ces exemples, certains sont annotés comme négatifs car ils sont soit des homographes qui ont d'autres sens soit des mots qui s'emploient dans des contextes semblables mais qui sont différents. Ces exemples sont généralement utiles à l'amélioration de la précision des règles existantes. Pour le module IAL4Sets par exemple, la précision passe de 28% à 67%. Le module IAL4Sets obtient des performances meilleures que le module Baseline sur cette première itération même si Baseline couvre plus d'exemples positifs sur le

corpus d'apprentissage (38 exemples contre 36 exemples pour le module IAL4Sets). Ceci est dû essentiellement au manque de précision de certaines règles construites dans le cadre de Baseline qui font beaucoup d'erreurs (précision égale à 40% sur le corpus de test). En effet, nous remarquons que les exemples positifs introduits dans la première itération par le module Baseline concernent uniquement les formes *cell*, *human*, et *plant*. Les règles construites sur les autres formes se basent majoritairement sur un seul exemple (exemple de départ) d'où le problème de surapprentissage surtout que les règles sont évaluées uniquement sur la portion de corpus consultée par l'utilisateur (très petite aux premières itérations). Les exemples positifs introduits, en revanche, par le module IAL4Sets dans la première itération sont plus variés et concernent plus de formes (soil, human, plant, cell, intestinal tract, microoganism, intracellular) d'où la meilleure précision et le meilleur rappel. Cette variété d'exemples est due essentiellement à leur provenance de trois sources d'exemples pertinents différentes (cnA, *NM* et TSEP). TSEP fournit des exemples très pertinents aux premières itérations car les termes qui ont les mêmes formes que les exemples positifs couverts se retrouvent en haut des listes des termes les plus similaires à ces derniers. Le groupe d'exemples *NM* fournit des exemples majoritairement pertinents car des généralisations minimales de règles qui font du surapprentissage permettent de récupérer des exemples pertinents.

Exemples à annoter proposés à l'utilisateur							
Baseline				**IAL4Sets**			
CnA		***NM***	**R**	**CnA**	***NM***	**TSEP**	**R**
~~cell~~		soil	~~about~~	humans	soil	~~cell wall~~	~~from~~
~~the~~		humans	~~chromosone~~	humans	human	~~Burkhorderia~~	~~it~~
~~cell~~		~~soil~~	~~of~~	plant	~~cell~~	~~plant~~	~~a~~
~~sulfide-rich~~		plant	~~fifth~~	intracellular	~~cells~~	human	~~which~~
human		plant	~~bodies~~	~~intracellularly~~	~~cell~~	intestinal	~~,~~
~~efficient~~		soil	~~proton~~	intracellular	~~soil~~	tsetse fly	~~while~~
humans		~~cells~~	~~.~~			~~target~~	
~~opportunistic pathogens~~		~~soil~~	~~and~~				
~~cell~~							
	Performances sur le corpus de test				**Performances sur le corpus de test**		
nEPC	**précision**	**rappel**	**F-mesure**	**nEPC**	**précision**	**rappel**	**F-mesure**
45	37%	28%	32%	46	77%	51%	61%

TABLE 8.13 – Itération 2 du test. nEPC représente le nombre d'exemples positifs couverts dans l'ensemble d'apprentissage.

Dans l'itération 2 du test (voir table 8.13), nous notons les mêmes phénomènes observés dans la première itération. Toutefois, nous remarquons que le module IAL4Sets commence à proposer, dans l'ensemble TSEP, des exemples qui diffèrent des exemples de départ comme le terme *tsetse fly* récupéré dans la liste des termes les plus similaires à l'exemple positif *tsetse*. Ce terme n'a pas été trouvé par le module Baseline au bout de 200 itérations, ce qui confirme nos intuitions par rapport à l'ajout de l'ensemble TSEP. Nous remarquons également que l'écart en performances se creuse davantage entre les deux modules d'apprentissage actif (61% de F-mesure pour notre module IAL4Sets contre 32% pour le module Baseline) car la qualité des exemples proposés par IAL4Sets est meilleure que celle des exemples proposés par Baseline et permet de produire des règles de meilleure qualité qui permettent à leur tour de couvrir des exemples plus pertinents.

Exemples à annoter proposés à l'utilisateur								
Baseline					IAL4Sets			
CnA	***NM***	**R**			**CnA**	***NM***	**TSEP**	**R**
		~~leading~~	~~are~~	~~.~~		~~diseases~~	~~effect~~	~~(~~
		~~for~~	~~The~~	~~Chlorobi~~		~~sugar~~	~~Maudlin~~	~~these~~
		~~an~~	~~serogroups~~	~~environment~~		intracellular	~~breakdown~~	~~of~~
		~~/~~	~~implicated~~	~~91E135~~		~~data~~	~~limited ability~~	~~the~~
		~~is~~	~~of~~	~~/~~		~~reactions~~	blood	~~since~~
		~~recurrentis~~	~~evolution~~	~~a~~		~~genes~~	~~catalase~~	~~of~~
		~~the~~	~~this~~	~~contaminating~~		~~presence~~	~~extra~~	~~C.~~
		~~to~~	~~reduction~~	~~in~~		~~cell~~	blood	~~by~~
		~~scolopes~~					~~year~~	
	Performances sur le corpus de test					Performances sur le corpus de test		
nEPC		**précision**	**rappel**	**F-mesure**	**nEPC**	**précision**	**rappel**	**F-mesure**
234		96%	49%	65%	239	76%	66%	70%

Table 8.14 – Itération 46 du test. nEPC représente le nombre d'exemples positifs couverts dans l'ensemble d'apprentissage.

Exemples à annoter proposés à l'utilisateur								
Baseline					IAL4Sets			
CnA	***NM***	**R**			**CnA**	***NM***	**TSEP**	**R**
		~~strains~~	~~GIT~~	~~those~~	blood	~~systems~~	blood	~~.~~
		~~organism~~	~~industry~~	~~thaliana~~	blood	~~mycobacteria~~	~~recent~~	~~or~~
		~~in~~	~~Detecting~~	~~.~~	~~blood~~	~~auspices~~	~~type III~~	~~into~~
		~~of~~	~~has~~	~~comparison~~	~~blood~~	~~sequences~~	~~severe~~	~~(~~
		~~I~~	~~-:~~	~~cyanobacterial~~	blood	~~Bordetella~~	~~polychlorinatred biphenyl~~	~~of~~
		~~and~~	~~order~~	~~BPDOs~~			~~extra-intestinal~~	~~as~~
		~~of~~	~~Marenda~~	~~times~~			~~first true insect endosymbiont~~	~~of~~
		~~of~~	~~involved~~	~~a~~				
		~~being~~						
	Performances sur le corpus de test					Performances sur le corpus de test		
nEPC		**précision**	**rappel**	**F-mesure**	**nEPC**	**précision**	**rappel**	**F-mesure**
234		96%	49%	65%	243	76%	66%	70%

Table 8.15 – Itération 47 du test. nEPC représente le nombre d'exemples positifs couverts dans l'ensemble d'apprentissage.

Dans des itérations plus éloignées comme les itérations 46 (voir table 8.14) et 47 (voir table 8.15), nous remarquons que le module Baseline a épuisé tous les exemples qu'il peut proposer dans le cadre des ensembles CnA et *NM* et compte désormais uniquement sur des exemples tirés au hasard dans l'ensemble R. En effet, tous les exemples couverts par les règles existantes dans l'ensemble d'entraînement, ont été évalués d'où la très bonne précision des règles. Toutefois, la valeur du rappel sur le corpus de test est faible (49%) car Baseline peine à trouver les exemples qui diffèrent des exemples de départ et qui ne peuvent être proposés que dans l'ensemble R. IAL4Sets, en revanche, continue à proposer ce genre d'exemples comme *blood* dans l'itération 46 qui permettent de construire de nouvelles règles qui couvrent de nouveaux exemples (ensemble CnA dans l'itération 47), ce qui explique la meilleure valeur de rappel (66%). L'exemple *blood* récupéré dans la liste des termes les plus proches sémantiquement du terme *tsetse fly* n'a toujours pas été retrouvé par le module Baseline après 200 itérations. C'est également le cas pour l'exemple *soil environments* récupéré dans la liste des termes les plus proches du terme *microorganism* (voir table 8.16).

Exemples à annoter proposés à l'utilisateur								
Baseline					**IAL4Sets**			
CnA	***NM***	**R**			**CnA**	***NM***	**TSEP**	**R**
		~~.~~	~~toward~~	~~species~~		animal	~~causaive agents~~	~~the~~
		~~it~~	~~causes~~	~~,~~		animals	~~harmless stomach~~	~~,~~
		~~alkaline~~	~~reactions~~	~~,~~		~~DNA~~	soil environments	~~contains~~
		~~capacity~~	~~T.~~	~~.~~		~~genomes~~	~~sequence~~	~~penetrating~~
		~~to~~	~~discovered~~	~~such~~		~~establishes~~	~~acids~~	~~commonly~~
		~~penicillin~~	~~homologues~~	~~of~~		~~al.~~	~~cell culture system~~	~~to~~
		~~as~~	~~mats~~	~~the~~		~~resistance~~	~~species resistance~~	~~for~~
		~~.~~	~~sp.~~	~~Burkhordelia~~		~~strains~~	~~free~~	~~are~~
		~~This~~						
	Performances sur le corpus de test					**Performances sur le corpus de test**		
nEPC		**précision**	**rappel**	**F-mesure**	**nEPC**	**précision**	**rappel**	**F-mesure**
242		96%	49%	65%	260	74%	68%	71%

TABLE 8.16 – Itération 62 du test. nEPC représente le nombre d'exemples positifs couverts dans l'ensemble d'apprentissage.

Au bout de 114 itérations (2850 exemples proposés), la tendance ne change pas (voir table 8.17). Autrement dit, l'ensemble TSEP continue à proposer des exemples pertinents (microbial) que le module Baseline ne retrouve pas car il compte uniquement sur le hasard en comptant sur l'ensemble R qui contient beaucoup d'exemples et par conséquent les chances de tirer un exemple positif peu fréquent sont très faibles.

En examinant en détail l'ensemble des 200 itérations du test réalisé, nous résumons les statistiques qui en ressortent dans la table 8.18.

Au bout de 200 itérations, le module Baseline couvre 290 exemples positifs (68% des exemples positifs qui doivent être couverts) dans l'ensemble d'apprentissage qui compte 424 exemmples contre 322 exemples (76% des exemples positifs qui doivent être couverts) pour le module IAL4Sets. Sans compter les 25 exemples positifs de

Exemples à annoter proposés à l'utilisateur								
Baseline					**IAL4Sets**			
CnA	***NM***	**R**			**CnA**	***NM***	**TSEP**	**R**
		~~.~~	~~is~~	~~economic~~		~~nitrogen~~	~~possible~~	~~be~~
		~~It~~	~~is~~	~~,~~		~~antibiotics~~	~~IV secretion system~~	~~proteolytic~~
		~~W.~~	~~abscessus~~	~~however~~		~~sugar~~	~~representative~~	~~shown~~
		~~to~~	~~disease~~	~~)~~		~~structure~~	~~major~~	~~in~~
		~~water~~	~~and~~	~~A~~		~~Similarity~~	~~AB type toxin~~	~~S.~~
		~~orale~~	~~curvus~~	~~also~~		~~attraction~~	microbial	~~,~~
		~~delivery~~	~~transport~~	~~,~~		~~contrary~~	~~accompanying symptoms~~	~~the~~
		~~group~~	~~strain~~	~~for~~		~~products~~	~~indeed~~	~~veins~~
		~~,~~					~~Strain ATCC 35469T~~	
	Performances sur le corpus de test					**Performances sur le corpus de test**		
nEPC		**précision**	**rappel**	**F-mesure**	**nEPC**	**précision**	**rappel**	**F-mesure**
252		96%	51%	67%	282	74%	68%	71%

Table 8.17 – Itération 114 du test. nEPC représente le nombre d'exemples positifs couverts dans l'ensemble d'apprentissage.

	Baseline			**IAL4Sets**			
	CnA	***NM***	**R**	**CnA**	***NM***	**TSEP**	**R**
Exemples positifs couverts dans l'ensemble d'entraînement	129	122	14	186	74	37	0
Total	**265**			**297**			

Table 8.18 – Répartition des exemples positifs couverts dans l'ensemble d'entraînement dans le cadre du test réalisé.

départ, ils couvrent respectivement 265 et 297 exemples positifs. Les 32 exemples non couverts par le module Baseline (8% des exemples positifs qui doivent être couverts) sont essentiellement des exemples proposés par l'ensemble TSEP dans le module IAL4Sets comme nous l'avons illustré à travers les différentes itérations du test analysées auparavant.

En regardant en détail la répartition des exemples positifs couverts dans l'ensemble d'apprentissage, nous constatons que :

— pour le module Baseline, 49% des exemples sont retrouvés par l'ensemble CnA, 46% des exemples sont retrouvés par l'ensemble *NM* et 5% des exemples sont retrouvés par l'ensemble R ;

— pour le module IAL4Sets, 63% des exemples sont retrouvés par l'ensemble CnA, 25% des exemples sont retrouvés par l'ensemble *NM*, 12% des exemples sont retrouvés par l'ensemble TSEP et 0% des exemples sont retrouvés par l'ensemble R.

À première vue, l'ensemble R paraît ne pas contribuer de manière significative dans les résultats obtenus par le module Baseline et pareillement les ensembles R et TSEP pour le module IAL4Sets. Seulement, nous savons qu'un nouvel exemple reconnu dans l'ensemble R ou TSEP permet de construire une nouvelle règle qui couvre de nouveaux exemples ou dont les généralisations minimales couvrent de nouveaux exemples. Autrement dit, certains des exemples retrouvés dans les ensembles CnA ou *NM* sont en réalité retrouvés par les ensembles R ou TSEP. La comptabilisation de ces exemples nous a permis de dégager de nouvelles statistiques résumées dans la table 8.19.

	Baseline			IAL4Sets			
	CnA	***NM***	**R**	**CnA**	***NM***	**TSEP**	**R**
Exemples positifs couverts dans l'ensemble d'apprentissage	92	115	58	149	57	91	0
Total	**265**			**297**			

TABLE 8.19 – Vraie répartition des exemples positifs couverts dans l'ensemble d'apprentissage dans le cadre du test réalisé.

La répartition des exemples positifs couverts dans l'ensemble d'apprentissage devient donc :

— pour le module Baseline, 35% des exemples sont retrouvés par l'ensemble CnA, 43% des exemples sont retrouvés par l'ensemble *NM* et 22% des exemples sont retrouvés par l'ensemble R ;

— pour le module IAL4Sets, 50% des exemples sont retrouvés par l'ensemble CnA, 19% des exemples sont retrouvés par l'ensemble *NM*, 31% des exemples sont retrouvés par l'ensemble TSEP et 0% des exemples sont retrouvés par l'ensemble R.

Nous pouvons constater, d'après les nouveaux chiffres, que l'ensemble R joue un rôle très important dans le module Baseline. Il a permis, dans le cadre du test réalisé, de retrouver plus que le cinquième (22%) de l'ensemble total des exemples positifs couverts. Les exemples retrouvés par l'ensemble R n'auraient pas pu être retrouvés par les ensembles CnA et *NM* car depuis la 25*eme* itération, le module Baseline compte uniquement sur les exemples proposés dans l'ensemble R. Pour le module IAL4Sets, l'ensemble TSEP joue un rôle important dans la mesure où il permet de retrouver 31% des exemples positifs couverts alors que l'ensemble R ne permet de trouver aucun exemple. En effet, la plupart des exemples positifs qui doivent normalement figurer dans l'ensemble R se retrouvent dans l'ensemble TSEP car ils sont proches d'exemples positifs déjà couverts d'où la suppression de ces exemples de l'ensemble R. La différence avec le module Baseline réside dans le fait que TSEP permet de retrouver rapidement les exemples non retrouvés dans CnA et *NM* alors que l'ensemble R dans Baseline s'appuie sur un tirage aléatoire d'exemples qui peut donc nécéssiter la consultation de tous les mots du corpus pour retrouver les exemples non retrouvés par CnA et *NM*.

8.5 Etude expérimentale de la stabilité des règles

Comme nous l'avons déjà mentionné dans les chapitres 5 et 6, la stabilité des règles est un critère très important à considérer dans un système interactif d'apprentissage de règles d'EI afin d'éviter :

— Un changement complet de l'ensemble des règles produites par le module d'apprentissage d'une itération à une autre. Un tel changement peut déconcerter l'utilisateur et lui faire perdre le fil de l'évolution de l'ensemble des règles.

— La non prise en compte des règles écrites par l'utilisateur par le module d'apprentissage. L'utilisateur juge que certaines règles qu'il écrit sont pertinentes et ne veut pas les voir disparaître dans les itérations qui suivent.

Pour étudier les deux points cités, nous avons réalisés deux expériences qui sont détaillées dans les deux sous sections qui suivent.

8.5.1 Expérience 1 : ajout d'exemples d'apprentissage

Cette expérience a pour but de vérifier si l'ajout d'exemples d'apprentissage change ou non complètement l'ancien ensemble de règles.

Description

Il s'agit toujours d'apprendre, en utilisant l'algorithme $\text{WHISK}_{\text{LemPos}}$, les habitats de bactéries dont les catégories possèdent 20 ou plus d'instances dans le corpus BB BioNLP-ST 2013. Seulement, pour des raisons de simplification, nous avons sélectionné 3 textes (voir figures 8.8, 8.9 et 8.10) sur lequels nous avons réalisé les 6 tests suivants.

test 1 : apprendre seulement sur Texte 1

test 2 : apprendre seulement sur Texte 2

test 3 : apprendre seulement sur Texte 3

test 1-2 : apprendre sur Texte 1 et Texte 2

test 1-3 : apprendre sur Texte 1 et Texte 3

test 2-3 : apprendre sur Texte 2 et Texte 3

Ces tests ont comme but de vérifier si en apprenant sur une combinaison de corpus d'apprentissage, les règles résultant de l'apprentissage sur chacun des corpus seul se conservent ou pas. Les termes surlignés dans les textes utilisés correspondent à des exemples d'apprentissage positifs.

Résultats

Les ensembles de règles produits par $\text{WHISK}_{\text{LemPos}}$ dans le cadre de ces tests figurent respectivement dans les figures 8.11, 8.12, 8.13, 8.14, 8.15 et 8.16.

Si nous regardons Texte 1 et Texte 2, nous pouvons remarquer que les seuls exemples dans Texte 2 qui ressemblent à ceux de Texte 1 sont les exemples dont le lemme est *animal*. Tous les autres exemples représentent de nouvelles formes. Pour assurer la stabilité, par exemple, de l'ensemble des règles produites en apprenant sur Texte 1, nous devons voir des nouvelles règles apparaître pour couvrir les nouvelles formes d'exemples introduites et des modifications mineures sur certaines règles pour couvrir des exemples ressemblant à ceux couverts par ces règles.

En examinant les ensembles de règles résultant de l'apprentissage sur Texte 1 (figure 8.11), sur Texte 2 (figure 8.12) et sur Texte 1 et Texte 2 (figure 8.14), nous pouvons constater la stabilité a été assurée car l'ensemble de règles résultant de l'apprentissage sur Texte 1 et Texte 2 est tout simplement une combinaison des deux ensembles résultant de l'apprentissage respectivement sur Texte 1 et Texte 2 avec la mise en facteur de la règle qui permet de couvrir les exemples qui se ressemblent :

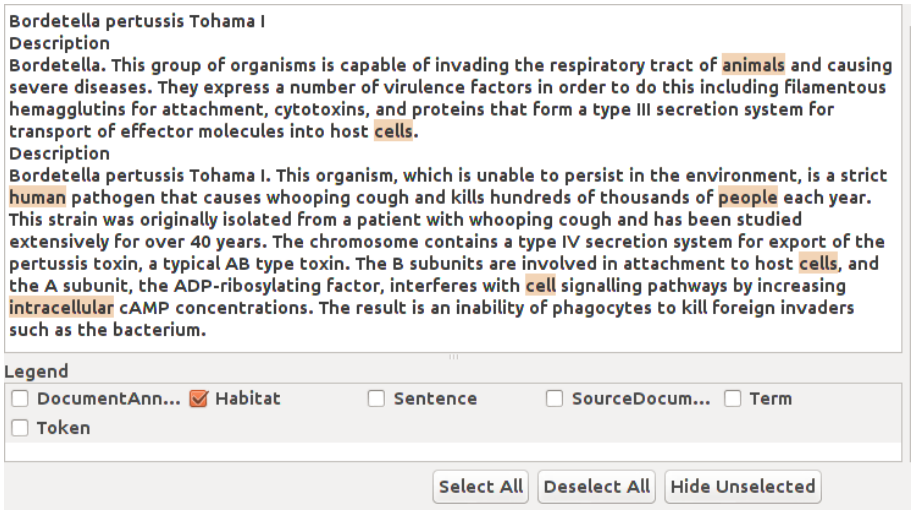


FIGURE 8.8 – Texte 1.

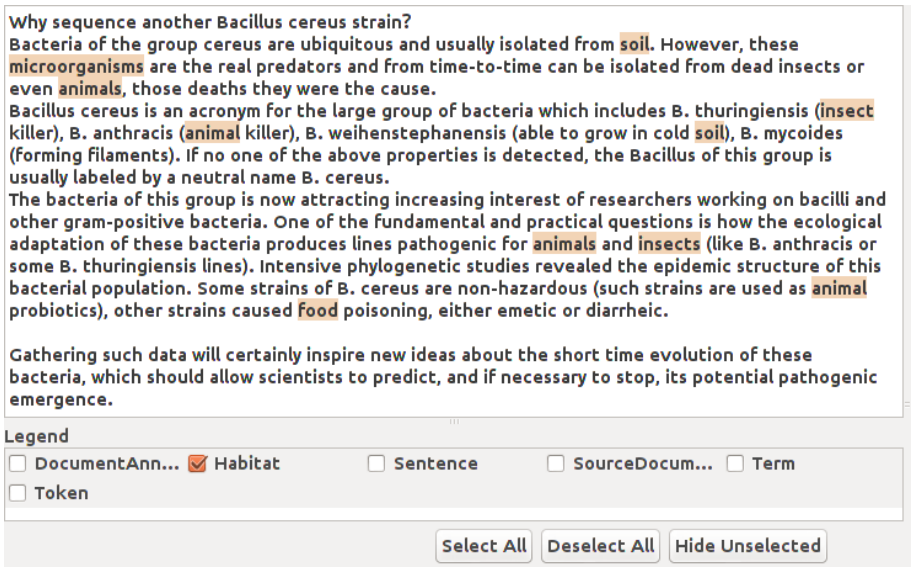


FIGURE 8.9 – Texte 2.

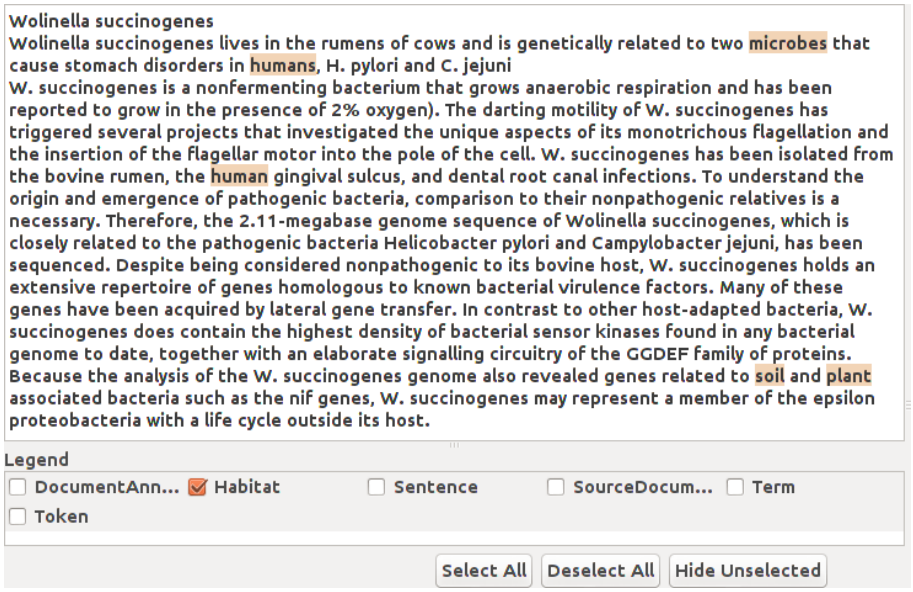


FIGURE 8.10 – Texte 3.

```
Token{FEATURE("lemma", "animal")->MARKONCE(Habitat)};   // p=1; n=0
Token{FEATURE("lemma", "cell")->MARKONCE(Habitat)}; // p=3; n=0
Token{FEATURE("lemma", "human")->MARKONCE(Habitat)};    // p=1; n=0
Token{FEATURE("lemma", "people")->MARKONCE(Habitat)};   // p=1; n=0
Token{FEATURE("lemma", "intracellular")->MARKONCE(Habitat)};    // p=1; n=0
```

Figure 8.11 – Règles produites par $\text{WHISK}_{\text{LemPos}}$ en apprenant sur Texte 1.

```
Token{FEATURE("lemma", "soil")->MARKONCE(Habitat)}; // p=2; n=0
Token{FEATURE("lemma", "microorganism")->MARKONCE(Habitat)};    // p=1; n=0
Token{FEATURE("lemma", "animal")->MARKONCE(Habitat)};   // p=4; n=0
Token{->MARKONCE(Habitat)} Token{FEATURE("lemma", "killer")};   // p=2; n=0
Token{FEATURE("lemma", "animal")} # Token{FEATURE("lemma", "insect")->MARKONCE(Habitat)};   // p=2; n=0
Token{FEATURE("lemma", "food")->MARKONCE(Habitat)}; // p=1; n=0
```

Figure 8.12 – Règles produites par $\text{WHISK}_{\text{LemPos}}$ en apprenant sur Texte 2.

```
Token{FEATURE("lemma", "microbe")->MARKONCE(Habitat)};  // p=1; n=0
Token{FEATURE("lemma", "human")->MARKONCE(Habitat)};    // p=2; n=0
Token{FEATURE("lemma", "soil")->MARKONCE(Habitat)}; // p=1; n=0
Token{FEATURE("lemma", "plant")->MARKONCE(Habitat)};    // p=1; n=0
```

Figure 8.13 – Règles produites par $\text{WHISK}_{\text{LemPos}}$ en apprenant sur Texte 3.

```
Token{FEATURE("lemma", "soil")->MARKONCE(Habitat)}; // p=2; n=0
Token{FEATURE("lemma", "microorganism")->MARKONCE(Habitat)};    // p=1; n=0
Token{FEATURE("lemma", "animal")->MARKONCE(Habitat)};   // p=5; n=0
Token{->MARKONCE(Habitat)} Token{FEATURE("lemma", "killer")};   // p=2; n=0
Token{FEATURE("lemma", "animal")} # Token{FEATURE("lemma", "insect")->MARKONCE(Habitat)};   // p=2; n=0
Token{FEATURE("lemma", "food")->MARKONCE(Habitat)}; // p=1; n=0
Token{FEATURE("lemma", "cell")->MARKONCE(Habitat)}; // p=3; n=0
Token{FEATURE("lemma", "human")->MARKONCE(Habitat)};    // p=1; n=0
Token{FEATURE("lemma", "people")->MARKONCE(Habitat)};   // p=1; n=0
Token{FEATURE("lemma", "intracellular")->MARKONCE(Habitat)};    // p=1; n=0
```

Figure 8.14 – Règles produites par $\text{WHISK}_{\text{LemPos}}$ en apprenant sur Texte 1 et Texte 2.

```
Token{FEATURE("lemma", "animal")->MARKONCE(Habitat)};   // p=1; n=0
Token{FEATURE("lemma", "host")} Token{FEATURE("lemma", "cell")->MARKONCE(Habitat)}; // p=2; n=0
Token{FEATURE("lemma", "human")->MARKONCE(Habitat)};    // p=3; n=0
Token{FEATURE("lemma", "people")->MARKONCE(Habitat)};   // p=1; n=0
Token{FEATURE("lemma", "cell")->MARKONCE(Habitat)} # Token{FEATURE("lemma", "pathway")}; // p=3; n=0
Token{FEATURE("lemma", "intracellular")->MARKONCE(Habitat)};    // p=1; n=0
Token{FEATURE("lemma", "microbe")->MARKONCE(Habitat)};  // p=1; n=0
Token{FEATURE("lemma", "soil")->MARKONCE(Habitat)}; // p=1; n=0
Token{FEATURE("lemma", "plant")->MARKONCE(Habitat)};    // p=1; n=0
```

Figure 8.15 – Règles produites par $\text{WHISK}_{\text{LemPos}}$ en apprenant sur Texte 1 et Texte 3.

```
Token{FEATURE("lemma", "soil")->MARKONCE(Habitat)}; // p=3; n=0
Token{FEATURE("lemma", "microorganism")->MARKONCE(Habitat)};    // p=1; n=0
Token{FEATURE("lemma", "animal")->MARKONCE(Habitat)};   // p=4; n=0
Token{->MARKONCE(Habitat)} Token{FEATURE("lemma", "killer")};   // p=2; n=0
Token{FEATURE("lemma", "animal")} # Token{FEATURE("lemma", "insect")->MARKONCE(Habitat)};   // p=2; n=0
Token{FEATURE("lemma", "food")->MARKONCE(Habitat)}; // p=1; n=0
Token{FEATURE("lemma", "microbe")->MARKONCE(Habitat)};  // p=1; n=0
Token{FEATURE("lemma", "human")->MARKONCE(Habitat)};    // p=2; n=0
Token{FEATURE("lemma", "plant")->MARKONCE(Habitat)};    // p=1; n=0
```

Figure 8.16 – Règles produites par $\text{WHISK}_{\text{LemPos}}$ en apprenant sur Texte 2 et Texte 3.

```
Token{FEATURE("lemma", "animal")->MARKONCE(Habitat)};
```

Cette règle n'a pas été modifiée car elle permet de couvrir les nouveaux exemples introduits sans faire d'erreurs.

Pour l'apprentissage sur Texte 1 (voir figure 8.11), sur Texte 3 (voir figure 8.13) et sur Texte 1 et Texte 3 (voir figure 8.15), nous avons le même scénario avec une petite différence. La règle qui couvre les trois exemples dont le lemme est *cell* dans Texte 1 et qui est la suivante :

```
Token{FEATURE("lemma", "cell")->MARKONCE(Habitat)};
```

couvre un exemple négatif dans Texte 3. En apprenant donc sur Texte 1 et Texte 3, cette règle a été remplacée par les deux règles suivantes :

```
1-Token{FEATURE("lemma","host")} Token{FEATURE("lemma","cell")
  ->MARKONCE(Habitat)};
2-Token{FEATURE("lemma", "cell")->MARKONCE(Habitat)}
  # Token{FEATURE("lemma", "pathway")};
```

Ces deux règles sont des spécifications de l'ancienne règle. Elles gardent le même contenu et y ajoutent un élément de contexte gauche pour la première et un élément de contexte droite pour la deuxième. Autrement dit, il n'y a pas de modification majeure dans l'ensemble de règles. Nous pouvons donc conclure que la stabilité a aussi été assurée pour ces tests.

Pour les règles résultant de l'apprentissage sur Texte 2 (figure 8.12), sur Texte 3 (figure 8.13) et sur Texte 2 et Texte 3 (figure 8.16), la stabilité est assurée exactement de la même manière que pour les tests d'apprentissage sur Texte 1, sur Texte 2 et sur Texte 1 et Texte 2.

8.5.2 Expérience 2 : écriture de règles

Dans un système interactif d'apprentissage de règles d'EI, l'utilisateur est amené à écrire des règles pour deux buts différents :

- Chercher des exemples dans le texte. On parle, dans ce cas, de règles prospectives.
- Enrichir ou modifier l'ensemble de règles. Les règles écrites, dans ce cas, par l'utilisateur sont jugées assez pertinentes par ce dernier et souhaite que le module d'apprentissage les conserve si elles font leurs preuves ou les améliore sans pour autant les supprimer si elles font des erreurs.

Nous nous intéressons bien évidemment, dans cette section, au deuxième type de règles.

Description

Le test que nous effectuons dans le cadre de cette expérience consiste à demander à un utilisateur qui a un minimum de connaissances en termes de biotopes de bactéries, d'énoncer des règles qui lui semblent évidentes que nous traduisons en

langage Ruta. Pour imposer le moins possible de contraintes, nous avons donné la liberté à l'utilisateur d'utiliser n'importe quelles formes de mots (expressions exactes, lemmes, étiquettes morpho-syntaxiques).

Nous vérifions par la suite l'existence de ces règles ou de versions modifiées de ces règles dans l'ensemble des règles inférées par l'algorithme $\text{WHISK}_{\text{MotLemPos}}$.

Résultats

En consultant de manière non approfondie quelques textes du corpus BB BioNLP-ST 2013, l'utilisateur énonce des règles dont quelques unes sont traduites par :

```
1-Token{FEATURE("lemma", "tsetse")} Token{FEATURE("lemma", "fly")
  ->MARKONCE(Habitat,1,2)};
2-Token{FEATURE("lemma", "insect")->MARKONCE(Habitat)};
3-Token{FEATURE("lemma", "intestine")->MARKONCE(Habitat)};
4-Token{FEATURE("lemma", "animal")->MARKONCE(Habitat)};
5-Token{FEATURE("lemma", "human")->MARKONCE(Habitat)};
```

Ces règles ont du sens car quand on parle de mouches, d'insectes, d'animaux, d'intestins ou d'humains dans des textes qui parlent de biotopes de bactéries, c'est généralement pour faire référence à des hotes potentiels ou des lieux de prolifération de bactéries. Nous remarquons également que les règles que les utilisateurs ont tendance à écrire sont généralement des règles génériques car ils ont tendance à penser au cas général et non aux cas particuliers qui sont plus rares dans des textes de domaine.

La figure 8.17 montre l'ensemble de règles produites par $\text{WHISK}_{\text{MotLemPos}}$ en apprenant sur l'ensemble du corpus.

Concernant la première règle de l'utilisateur, nous remarquons qu'il existe une règle plus spécifique que cette dernière dans cet ensemble et qui est la suivante :

```
Token{REGEXP("bacteria")} # Token{FEATURE("lemma", "tsetse")
->MARKONCE(Habitat, 3, 4)} Token{FEATURE("lemma", "fly")};
```

En effet, la règle de l'utilisateur n'a pas été gardée telle quelle car elle fait des erreurs. Elle permet, par exemple, d'annoter *tsetse fly* dans le terme *blood-sucking tsetse fly* qui figure dans le texte présenté dans la figure 8.18 comme exemple positf alors que c'est le terme en entier qui doit être annoté comme exemple positif. D'où l'inférence d'une règle plus spécifique. D'autres règles, en revanche, ont été inférées pour couvrir d'autres exemples relatifs aux *tsetse flies* auxquels il n'est pas intuitif de penser comme *tsetse host cell*, *tsetse midgut*, *tsetse midgut and haemolymph*, *tsetse immune system*, *gut of the tsetse fly* ou encore les termes *tsetse* et *fly* utilisés parfois seuls.

La deuxième règle de l'utilisateur est générique et fait des erreurs comme par exemple la couverture du terme *insects* dans *dead insects* ou le terme *insect* dans *haemolymph of the insect*. Ces termes complexes doivent, en effet, être annotés en entier comme des exemples positifs. La règle de l'utilisateur a donc été remplacée dans l'ensemble des règles apprises par $\text{WHISK}_{\text{MotLemPos}}$ par les règles suivantes :

```
//******(1)******//
Token{REGEXP("bacteria")} # Token{FEATURE("lemma", "tsetse")->MARKONCE(Habitat, 3, 4)}
    Token{FEATURE("lemma", "fly")}; //tsetse fly; tsetse flies
Token{FEATURE("lemma", "tsetse")->MARKONCE(Habitat, 1, 3)} Token Token{FEATURE("lemma", "cell")};
//tsetse host cell
Token{FEATURE("lemma", "tsetse")->MARKONCE(Habitat)} Token{FEATURE("lemma", "host")}; //tsetse
Token{REGEXP("by")} # Token{FEATURE("lemma","tsetse")->MARKONCE(Habitat)}; //tsetse
Token{FEATURE("lemma", "tsetse")->MARKONCE(Habitat, 1, 4)} Token Token Token{FEATURE("lemma", "haemolymph")}
//tsetse midgut and haemolymph
Token{FEATURE("lemma", "tsetse")->MARKONCE(Habitat, 1, 2)} Token{FEATURE("lemma", "midgut")}; //tsetse midgu
Token{FEATURE("lemma", "tsetse")->MARKONCE(Habitat)} Token{FEATURE("lemma", "midgut")}; //tsetse
Token{FEATURE("lemma","micro-organism")} # Token{FEATURE("lemma","tsetse")->MARKONCE(Habitat)}; //tsetse
Token{FEATURE("lemma", "tsetse")->MARKONCE(Habitat, 1, 3)} Token{FEATURE("lemma", "immune")} Token;
//tsetse immune system
Token{FEATURE("lemma","tsetse")->MARKONCE(Habitat)}#Token{FEATURE("lemma","(")} Token{FEATURE("lemma", "6")}
//tsetse
Token{FEATURE("lemma", "gut")->MARKONCE(Habitat, 1, 5)} Token Token Token{FEATURE("lemma", "tsetse")} Token;
//gut of the tsetse fly
Token{FEATURE("lemma", "glossinidius")} # Token{FEATURE("lemma", "tsetse")->MARKONCE(Habitat)};//tsetse
Token{FEATURE("lemma","fly")->MARKONCE(Habitat)} Token{REGEXP("transmits")}; //fly; flies
Token{FEATURE("lemma", "fly")->MARKONCE(Habitat)} Token{REGEXP("live")}; //fly; flies
//******(2)******//
Token{REGEXP("dead")->MARKONCE(Habitat, 1, 5)} Token{FEATURE("lemma", "insect")} Token Token Token;
//dead insects or even animals
Token{REGEXP("insect")->MARKONCE(Habitat)}; //insect; insects
Token{REGEXP("dead")->MARKONCE(Habitat, 1, 2)} Token; // dead insects; dead larvae
Token{FEATURE("lemma", "insect")->MARKONCE(Habitat)} # Token{FEATURE("lemma", "human")}; //insect; insects
Token{FEATURE("lemma","insect")->MARKONCE(Habitat)} # Token{REGEXP("In")}; //insect; insects
Token{FEATURE("lemma", "haemolymph")->MARKONCE(Habitat, 1, 4)} Token Token{FEATURE("lemma", "the")} Token;
//haemolymph of the insect
//******(3)******//
Token{FEATURE("lemma", "intestine")->MARKONCE(Habitat)} # Token{FEATURE("lemma", "many")};
//intestine; intestines
Token{FEATURE("lemma", "small")->MARKONCE(Habitat, 1, 2)} Token{FEATURE("lemma", "intestine")};
//small intestine; small intestines
Token{FEATURE("lemma", "animal")->MARKONCE(Habitat, 1, 2)} Token{REGEXP("intestines")}; // animal intestines
Token{FEATURE("postag", "NN")->MARKONCE(Habitat, 1, 4)} Token Token Token{REGEXP("feces")};
//animal intestines and feces; kangaroo and bird feces
//******(4)******//
Token{FEATURE("lemma", "animal")->MARKONCE(Habitat)} # Token{FEATURE("lemma", "death")}; //animal; animals
Token{REGEXP("bacteria")} # Token{FEATURE("lemma", "animal")->MARKONCE(Habitat)}; //animal; animals
Token{FEATURE("postag","DT")} # Token{REGEXP("animal")->MARKONCE(Habitat)}; //animal; animals
Token{FEATURE("lemma", ",")} # Token{FEATURE("lemma", "animal")->MARKONCE(Habitat)} #
    Token{FEATURE("lemma", "find")}; //animal; animals
Token{->MARKONCE(Habitat, 1, 4)} Token{REGEXP("tract")} Token Token{FEATURE("lemma", "animal")};
// respiratory tract of animals
Token{FEATURE("lemma","animal")->MARKONCE(Habitat)} # Token{REGEXP("diseases")}; //animal; animals
Token{FEATURE("lemma", "for")} # Token{FEATURE("lemma", "animal")->MARKONCE(Habitat)} #
    Token{FEATURE("lemma", "strain")}; //animal; animals
Token{FEATURE("lemma", "plant")} # Token{FEATURE("lemma", "animal")->MARKONCE(Habitat)}; //animal; animals
Token{FEATURE("lemma", "animal")->MARKONCE(Habitat, 1, 2)} Token{REGEXP("intestines")}; //animal intestines
Token{FEATURE("lemma", "into")} # Token{FEATURE("lemma", "animal")->MARKONCE(Habitat)}; //animal; animals
Token{FEATURE("lemma", "intestinal")->MARKONCE(Habitat, 1, 5)} Token{FEATURE("lemma", "tract")} # Token
    Token{REGEXP("animals")}; //intestinal tract of animals; intestinal tract of humans and animals
Token{REGEXP("tract")} # Token{FEATURE("lemma","animal")->MARKONCE(Habitat)};  //animal; animals
Token{FEATURE("lemma",",")} # Token{FEATURE("lemma","animal")->MARKONCE(Habitat)} #
    Token{FEATURE("lemma","have")} # Token{FEATURE("lemma","in")}; //animal; animals
//******(5)******//
Token{FEATURE("lemma", "human")->MARKONCE(Habitat)}; //human; humans
Token{REGEXP("human")->MARKONCE(Habitat, 1, 2)} Token{FEATURE("lemma", "body")};//human body; human bodies
Token{FEATURE("lemma", "human")->MARKONCE(Habitat, 1, 3)} Token{FEATURE("lemma", "urogenital")} Token;
//human urogenital tract
Token{REGEXP("human")->MARKONCE(Habitat, 1, 2)} Token{FEATURE("lemma", "cell")}; //human cell; human cells
Token{FEATURE("lemma", "human")->MARKONCE(Habitat, 1, 3)} Token{FEATURE("lemma", "immune")} Token;
//human immune system
Token{FEATURE("lemma", "interior")->MARKONCE(Habitat, 1, 4)} Token Token{FEATURE("lemma", "human")} Token;
//interior of human cells
```

FIGURE 8.17 – Règles produites par $\text{WHISK}_{\text{MotLemPos}}$ en apprenant sur l'ensemble du corpus.

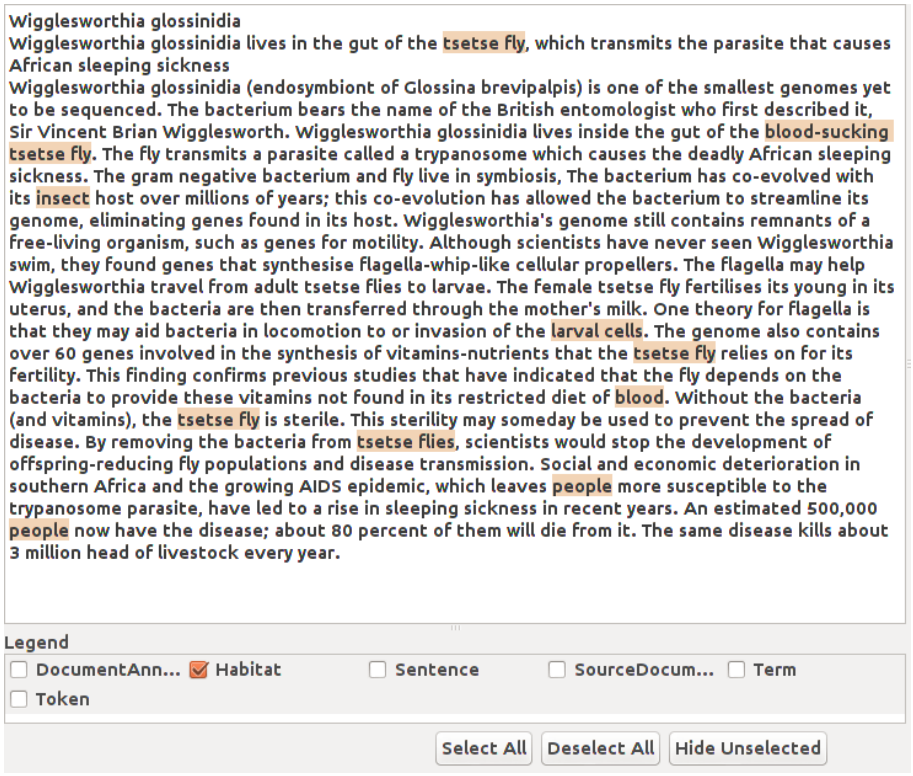


Figure 8.18 – Texte 4.

```
1-Token{REGEXP("insect")->MARKONCE(Habitat)};
2-Token{FEATURE("lemma","insect")->MARKONCE(Habitat)} #
  Token{FEATURE("lemma","human")};
3-Token{FEATURE("lemma","insect")->MARKONCE(Habitat)} #
  Token{REGEXP("In")};
```

La première règle permet de couvrir tous les mots *insect* au singulier car il n'y a pas d'erreurs sur ces mots. Les deuxième et troisième règles sont des versions plus spécifiques de la règle de l'utilisateur qui ne couvrent pas les erreurs faites par cette dernière. D'autres règles qui couvrent des termes complexes relatifs aux insectes ont été apprises à part.

La troisième règle de l'utilisateur a également remplacée par une règle plus spécifique qui fait moins d'erreurs et qui est la suivante :

```
Token{FEATURE("lemma","intestine")->MARKONCE(Habitat)} #
Token{FEATURE("lemma","many")};
```

Pour couvrir des termes plus complexes comme *animal intestines* ou *small intestines* qui font partie des exemples que l'utilisateur cherche à couvrir, d'autres règles ont été inférées.

```
1-Token{FEATURE("lemma","animal")->MARKONCE(Habitat)} #
  Token{FEATURE("lemma","death")};
2-Token{REGEXP("bacteria")} # Token{FEATURE("lemma","animal")
  ->MARKONCE(Habitat)};
```

```
3-Token{FEATURE("postag","DT")} # Token{REGEXP("animal")
  ->MARKONCE(Habitat)};
4-Token{FEATURE("lemma", ",")} # Token{FEATURE("lemma", "animal")
  ->MARKONCE(Habitat)} # Token{FEATURE("lemma", "find")};
5-Token{FEATURE("lemma","animal")->MARKONCE(Habitat)} #
  Token{REGEXP("diseases")};
6-Token{FEATURE("lemma", "for")} # Token{FEATURE("lemma", "animal")
  ->MARKONCE(Habitat)} # Token{FEATURE("lemma", "strain")};
7-Token{FEATURE("lemma","plant")} # Token{FEATURE("lemma","animal")
  ->MARKONCE(Habitat)};
8-Token{FEATURE("lemma","into")} # Token{FEATURE("lemma","animal")
  ->MARKONCE(Habitat)};
9-Token{REGEXP("tract")} # Token{FEATURE("lemma","animal")
  ->MARKONCE(Habitat)};
10-Token{FEATURE("lemma",",")} # Token{FEATURE("lemma","animal")
   ->MARKONCE(Habitat)} # Token{FEATURE("lemma","have")} #
   Token{FEATURE("lemma","in")};
```

Le terme *animal* est un terme très fréquent dans le corpus. Il est aussi bien utilisé dans des termes simples (animal, animals) que dans des termes complexes (animal intestines, respiratory tract of animals, intestinal tract of animals, etc.). Dans un corpus beaucoup plus volumineux, la règle de l'utilisateur aurait été gardée comme telle car le nombre d'erreurs qu'elle ferait serait négligeable devant le nombre d'exemples positifs qu'elle couvre. Seulement, dans notre corpus cette règles fait beaucoup d'erreurs. Le nombre élevé des versions spécifiques de cette règle apprises par $\text{WHISK}_{\text{MotLemPos}}$ s'explique par les différentes manières d'emploi du terme et l'absence d'un contexte commun à ces emplois.

Nous remarquons que la cinquième règle de l'utilisateur a été gardée comme telle. En effet, malgré les erreurs que cette règle fait, c'est négligeable devant les exemples positifs qu'elle couvre (3 négatifs pour 95 positifs) sachant que l'algorithme $\text{WHISK}_{\text{MotLemPos}}$ tel que nous l'avons paramétré autorise 1 négatif pour 20 positifs.

Pour conclure, nous remarquons qu'aucune des règles de l'utilisateur n'a été ignorée. Elles ont été soit gardées comme telles soient améliorées en en créant des versions plus spécifiques et plus précises.

Nous pouvons constater, d'après les deux expériences réalisées, que la stabilité des règles est intrinsèquement assurée par l'algorithme WHISK que ce soit dans le cas où l'utilisateur ajoute de nouveaux exemples ou dans le cas où il introduit des règles. Nous pensons que cette stabilité est due aux facteurs suivants :

— L'algorithme $\text{WHISK}_{\text{MotLemPos}}$ favorise les règles qui couvrent le plus d'exemples positifs. Dans le cadre de cette politique, pour chaque nouvel exemple introduit, l'algorithme essaie de le couvrir en favorisant les règles qui couvrent également d'autres exemples. Par conséquent, si cet exemple ressemble à des exemples déjà couverts, il risque d'apprendre la même règle qui couvrait ces derniers ou une version un peu plus spécifique mais pas complètement différente. Si, en revanche, le nouvel exemple ne ressemble à aucun autre exemple couvert, l'algorithme apprend une nouvelle règle qui couvre uniquement cet exemple.

— De la même manière que l'algorithme WHISK, l'utilisateur, par son esprit synthétique, essaie d'introduire des règles qui couvrent beaucoup d'exemples positifs et non un seul exemple particulier. C'est la raison pour laquelle ses règles s'expriment d'une manière très proche pour ne pas dire similaire à celle des règles apprises.

Conclusion

Nous avons présenté dans ce chapitre une étude expérimentale (sur les corpus BioNLP-ST 2013 et SyntSem) qui a pour but d'évaluer les différentes propositions détaillées dans les deux chapitres précédents. Une étude de différentes combinaisons d'informations linguistiques utilisées dans l'expression des règles inférées par nos différentes versions étendues de WHISK_{R} ont permis d'évaluer l'impact de ces informations sur les performances des règles et sur leur nombre. En effet, l'utilisation de l'expression exacte des mots dans les règles ($\text{WHISK}_{\text{Mot}}$) a permis d'obtenir les meilleures valeurs de précision alors que l'utilisation à la fois des lemmes et des étiquettes morpho-syntaxiques ($\text{WHISK}_{\text{LemPos}}$) a permis d'obtenir les meilleures valeurs de rappel ainsi que le nombre le plus réduit de règles. L'utilisation de la stratégie d'apprentissage sur un corpus réduit a permis non seulement la non discrimination des exemples positifs non encore annotés par l'utilisateur à une itération donnée mais aussi un gain considérable dans le temps d'apprentissage. Le module d'apprentissage actif proposé (IAL4Sets) a également permis d'améliorer significativement les performances du module d'apprentissage de base de l'algorithme WHISK grâce à l'introduction de la notion de similarité distributionnelle qui permet de proposer à l'utilisateur des exemples sémantiquement proches des exemples positifs déjà couverts. Même si nous n'avons pas fait de propositions pour assurer la stabilité de l'ensemble des règles, nous avons montré expérimentalement dans ce chapitre que ni l'ajout de nouveaux exemples ni l'écriture de nouvelles règles ne cause des modifications majeures sur les règles. Ceci est essentiellement dû à la stabilité intrinsèque de l'algorithme WHISK.

Chapitre 9

Conclusion et Perspectives

9.1 Rappel des objectifs

L'Extraction d'Information (EI) est une technologie émergente du Traitement Automatique du Langage (TAL) qui consiste à traiter du texte non structuré écrit en langage naturel afin d'en extraire des informations et des faits spécifiques et de les stocker dans des *templates* ou des bases de données pour des traitements ultérieurs. Les techniques d'EI peuvent être divisées en techniques à base de règles et techniques à base d'apprentissage statistique. D'un côté, les techniques à base d'apprentissage statistique sont actuellement plus utilisées dans le monde académique pour leur robustesse face au bruit dans les données non structurées. D'un autre côté, les techniques à base de règles dominent le monde industriel car les règles sont plus faciles à manipuler et à interpréter par un être humain et par conséquent plus faciles à investiguer pour corriger d'éventuelles erreurs.

De nos jours, nous observons également l'apparition de plus en plus de systèmes d'EI interactifs. Cela répond à un double objectif. Le premier consiste à réduire le coût de mise en place de systèmes d'EI quand on ne dispose pas d'une quantité suffisante de données d'apprentissage annotées ou de l'expertise nécessaire en ingénierie de connaissances. Le deuxième consiste à permettre à l'utilisateur d'investiguer les erreurs faites par le système d'EI et de les corriger.

Notre travail s'inscrit dans le cadre de l'extraction d'information interactive à base de règles. Notre objectif consistait à produire une approche générique itérative interactive d'EI à base de règles pour permettre à l'utilisateur de travailler à la fois sur les règles d'EI et les exemples d'apprentissage tout en minimisant au maximum l'effort humain requis. Étant donné que l'utilisateur est amené à écrire des règles et lire et modifier des règles inférées, le langage de règles utilisé par l'utilisateur et le module d'apprentissage de règles doit être le même. Nous avons donc décidé d'utiliser le langage Ruta et d'étendre la version de l'algorithme WHISK basée sur Ruta implémentée dans le système TextRuler (WHISK_R) pour l'adapter à nos besoins.

9.2 Bilan

Nous avons commencé, dans ce travail, par déterminer les spécifications et les enjeux de notre approche. Pour pouvoir écrire ou modifier des règles, l'utilisateur a besoin de comprendre ces règles qui doivent être facilement interprétables. Les règles doivent également avoir des expressions assez génériques pour ne pas faire de surapprentissage, pour avoir des expressions moins complexes et pour être moins nombreuses. L'utilisateur a également besoin de suivre l'évolution des règles qu'il écrit ou modifie au fil des itérations, d'où la nécessité d'assurer une certaine stabilité de l'ensemble des règles pour garantir l'absence de modifications majeures des règles d'une itération à une autre. Pour pouvoir interagir avec le module d'apprentissage de règles, l'utilisateur ne doit pas attendre longtemps la réponse de ce dernier. Autrement dit, l'apprentissage de règles doit se faire d'une manière assez rapide.

Pour répondre aux spécifications et aux besoins de l'approche proposée, nous proposons :

- **Une chaîne d'annotation linguistique et un langage de règles expressif** – Le choix des annotations avec lesquelles annoter le texte de départ ainsi que le langage dans lequel écrire et inférer les règles joue un rôle très important dans la compréhensibilité et la généricité des règles.
- **Une stratégie d'apprentissage sur un corpus réduit** – Cette stratégie cherche à éviter la discrimination des exemples positifs non encore annotés par l'utilisateur à une itération donnée du processus d'apprentissage interactif. Elle permet également de réduire le temps d'apprentissage.
- **Un concordancier** – Cet outil permet à l'utilisateur d'écrire des règles prospectives, une règle prospective étant une règle qui a uniquement pour but de chercher des exemples dans le texte en s'appuyant sur des mots clés dont dispose l'utilisateur. Un concordancier est d'une grande utilité dans une approche interactive dans la mesure où il permet à l'utilisateur de réduire son espace de recherche d'exemples à l'ensemble des exemples couverts par les règles prospectives qu'il écrit.
- **Un module d'apprentissage actif (IAL4Sets)** – Ce module étend le module d'apprentisage actif de l'algorithme WHISK en lui ajoutant des notions de distance ou de similarité distributionnelle qui permet de proposer à l'utilisateur des exemples sémantiquement proches des exemples positifs déjà couverts.

Le système IRIES dans lequel nous avons mis en place les différents modules proposés étend l'interface de visualisation du système TextRuler et l'algorithme WHISK adapté au langage Ruta (WHISK_{R}) implémenté dans TextRuler.

Les modules composant le système IRIES ont été évalués sur deux corpus : le corpus de BioNLP-ST 2013 et le corpus SyntSem. Une étude de différentes combinaisons d'informations linguistiques utilisées dans l'expression des règles inférées par nos différentes versions étendues de WHISK_{R} a permis d'évaluer l'impact de ces informations sur les performances des règles (précision, rappel et F-mesure) ainsi que sur la taille de l'ensemble des règles. Alors que l'utilisation de l'expression exacte des mots ($\text{WHISK}_{\text{Mot}}$) a permis d'obtenir les meilleures valeurs de précision, la combinaison des lemmes avec les étiquettes morpho-syntaxiques ($\text{WHISK}_{\text{LemPos}}$)

a permis d'obtenir les meilleures valeurs de rappel ainsi que le nombre le plus réduit de règles inférées. La stratégie d'apprentissage sur un corpus réduit a permis un gain considérable dans le temps d'apprentissage sans dégrader les performances. En apprenant sur environ un quart des phrases du corpus BioNLP-ST 2013, nous avons pu réduire le temps d'apprentissage d'environ quatre cinquièmes sans dégrader les performances. Enfin, le module d'apprentissage actif proposé (IAL4Sets) a permis d'améliorer significativement les performances de l'apprentissage actif de base de l'algorithme WHISK (Baseline) grâce à la notion de similarité distributionnelle que nous avons introduite et qui permet de proposer à l'utilisateur des exemples sémantiquement proches des exemples positifs déjà couverts. L'algorithme WHISK étant intrinsèquement stable, nous n'avons pas proposé dans ce travail un module qui assure la stabilité des règles. Cependant, nous avons mené deux expérimentations qui ont conclu que l'ajout d'exemples d'apprentissage ou de règles par l'utilisateur ne modifie pas complètement l'ensemble total des règles. Les modifications apportées, si elles ont lieu, se résument à des généralisations ou spécifications de certaines règles ou à l'ajout de nouvelles règles, ce qui n'affecte pas la stabilité de l'ensemble total des règles.

9.3 Perspectives

Une étude théorique sur la stabilité de l'ensemble de règles produit par des algorithmes d'apprentissage de règles d'EI doit être menée de manière rigoureuse pour juger de la légitimité de cette question de stabilité sachant qu'elle n'a jamais été évoquée jusque-là dans des travaux d'apprentissage de règles. Si cette étude révèle l'existence d'algorithmes d'apprentissage de règles non intrinsèquement stables, des pistes de modules assurant la stabilité des règles peuvent être trouvées du côté de l'apprentissage incrémental. En effet, un algorithme d'apprentissage incrémental, comme l'a défini Polikar et al. (2001), doit répondre aux critères suivants :

- il doit être capable d'apprendre des connaissances supplémentaires à partir de nouvelles données ;
- il ne doit pas nécessiter l'accès aux données d'origine utilisées pour apprendre le classifieur actuel ;
- il doit préserver les connaissances déjà acquises ;
- il doit être capable d'apprendre de nouvelles classes susceptibles d'être introduites avec de nouvelles données.

Ces critères semblent s'accorder avec ce que nous attendons de notre apprentissage de règles. Cependant, l'utilisation de l'apprentissage incrémental ne doit pas être systématique. En effet, l'utilisateur doit avoir le choix entre appliquer l'apprentissage incrémental ou ne pas l'appliquer selon qu'il est satisfait de l'ensemble des règles d'extraction d'information courant ou non.

- Si l'ensemble des règles le satisfait, il choisit d'appliquer l'apprentissage incrémental car il ne veut pas que l'algorithme d'apprentissage réapprenne à chaque itération des nouvelles règles à partir de zéro mais plutôt qu'il garde les anciennes règles et leur ajoute de nouvelles connaissances à partir des nouveaux exemples étiquetés.

— Si l'ensemble des règles ne le satisfait pas, il ne choisit pas d'appliquer l'apprentissage incrémental car les règles inférées de zéro peuvent avoir des expressions moins complexes.

Nous nous sommes focalisée dans ce travail sur la mise en place des piliers d'une approche d'apprentissage interactif de règles d'EI générique. Nous avons pris le soin de les évaluer un par un séparément mais l'évaluation de l'approche globale reste à faire. De nouvelles métriques à l'instar de ENUA (Expected Number of User Actions) (Kristjansson et al., 2004) doivent, en effet, être définies pour mesurer le taux de l'intervention humaine ou le taux d'effort humain économisé.

Notre approche peut être appliquée dans le cadre de l'annotation sémantique et plus spécifiquement l'annotation sémantique au regard d'ontologies où il est question, par exemple, d'étendre les ontologies par une base de règles d'extraction d'information (Ma et al., 2009). Cette base de règles peut être constituée en s'appuyant sur notre méthode interactive, les règles de la base étant associées aux éléments de l'ontologie (aussi bien les concepts que les relations).

En effet, l'annotation sémantique au regard d'ontologies repose généralement sur des ontologies enrichies de connaissances linguistiques et lexicales qui permettent de mettre en correspondance des éléments de l'ontologie avec des fragments de texte. L'enrichissement de l'ontologie par ce type de connaissances pose problème car la taille de l'ontologie peut augmenter fortement, ce qui rend son utilisation plus difficile.

Pour résoudre ce problème, Ma et al. (2009) ont proposé de construire une extension qui soit indépendante de l'ontologie et qui permet de faire le lien entre l'ontologie et le texte à annoter au regard de cette ontologie. Cette extension est, en effet, une base de règles d'EI dont les règles peuvent être construites en utilisant notre approche interactive.

La figure 9.1 illustre la méthodologie de constitution de l'extension :

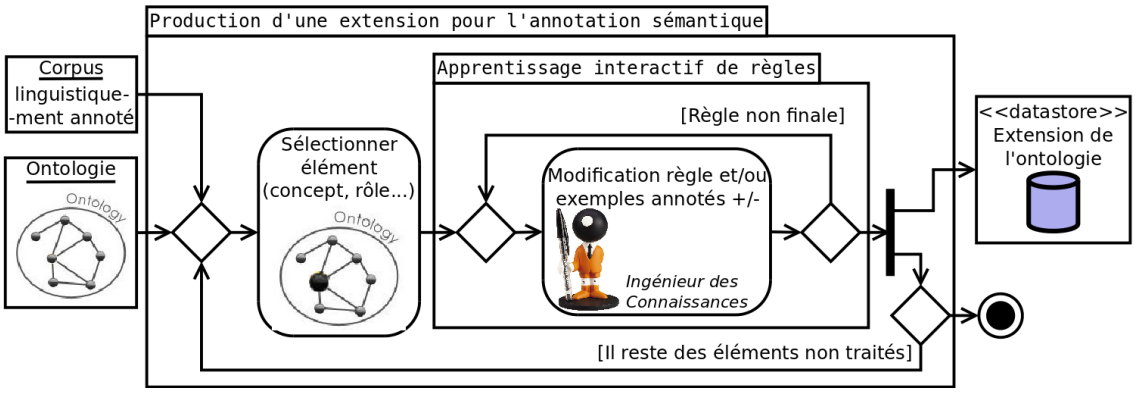


FIGURE 9.1 – Méthodologie de production d'une extension d'une ontologie

1. Les différents éléments de l'ontologie (concepts, rôles, etc.) sont traités successivement de manière à venir alimenter l'extension de l'ontologie de manière incrémentale (paquetage *Production d'une extension pour l'annotation sémantique* sur la figure 9.1).

2. Chaque ensemble de règles d'annotation d'un élément de l'ontologie est construit de manière itérative et interactive, jusqu'à aboutir à un ensemble couvrant de manière satisfaisante les portions de texte qui doivent être mises en correspondance avec l'élément de l'ontologie traité (paquetage *Apprentissage interactif de règles* sur la figure 9.1). Un utilisateur intervient à chaque itération en travaillant de manière duale sur les règles d'annotation (intension) ainsi que sur des exemples d'annotation (extension).

La généricité de notre approche fait qu'elle peut annoter n'importe quelle cible (un concept, une entité nommée, une relation, etc.). Seulement, dans une ontologie les différents éléments sont reliés entre eux. Cette hiérarchie doit donc être exploitée au mieux afin d'établir un certain ordre et une certaine structure entre les différentes règles d'extraction d'information. Certaines questions doivent par conséquent être traitées :

— quelle relation doit exister entre une règle qui annote un concept et une autre qui annote son fils ?
— quelle relation doit exister entre une règle qui annote un concept et une autre qui annote une instance de ce concept ?
— quelle relation doit exister entre une règle qui annote une relation entre deux concepts et les règles qui annotent les concepts impliqués ?
— etc.

Notre approche peut donc être évaluée dans sa globalité dans le cadre de l'annotation sémantique au regard d'ontologies.

Avoir dans un même ensemble de règles des règles qui dénotent des concepts différents soulève une autre problématique qui est le conflit qui peut survenir entre ces règles. En effet, on parle de conflit quand deux segments de texte qui se chevauchent sont couverts par deux règles différentes. Plusieurs stratégies de résolution de ce genre de conflits existent dans la littérature (voir section 2.6.2). Nous préconisons la considération des règles comme une collection non ordonnée où les règles s'appliquent indépendamment les unes des autres car cette méthode d'organisation offre à l'utilisateur plus de flexibilité dans la définition des règles sans souci des éventuels recouvrements avec les règles existantes. Dans ce cas de figure, deux solutions peuvent être adoptées pour résoudre les conflits entre les règles :

— favoriser la règle qui couvre le segment de texte le plus long (c'est le cas dans GATE (Cunningham, 2002)) ;
— fusionner les segments de texte qui se chevauchent si leurs règles correspondantes paratagent la même action ou utiliser des politiques plus sophistiquées comme celle définie dans (Reiss et al., 2008) sinon.

Dans un système interactif, nous pouvons tout à fait développer un module qui identifie les règles conflictuelles et les présente à l'utilisateur laissant à ce dernier le choix de l'action à mener pour résoudre le conflit. En effet, l'utilisateur peut avoir recours à différentes politiques de résolution de conflits selon la nature du conflit rencontré à chaque étape.

Bibliographie

Aitken, S. (2002). Learning Information Extraction Rules : An Inductive Logic Programming approach. In *Proceedings of ECAI 2002 (European Conference on Artificial Intelligence)* (pp. 355–359).

Akbik, A., Konomi, O., & Melnikov, M. (2013). Propminer : A Workflow for Interactive Information Extraction and Exploration using Dependency Trees. In *ACL (Conference System Demonstrations)* (pp. 157–162).

Appelt, D. E. (1999). Introduction to Information Extraction. *AI Commun.*, *12*, 161–172.

Aramaki, Eiji and Imai, Takeshi and Miyo, Kengo and Ohe, Kazuhiko. (2006). Automatic deidentification by using sentence features and label consistency. In *i2b2 Workshop on Challenges in Natural Language Processing for Clinical Data* (pp. 10–11).

Atzmueller, M., Kluegl, P., & Puppe, F. (2008). Rule-Based Information Extraction for Structured Data Acquisition using TextMarker. In *LWA-2008 (Special Track on Knowledge Discovery and Machine Learning)* (pp. 1–7).

Audibert, L. (2003). Etude des critères de désambiguïsation sémantique automatique : résultats sur les cooccurrences. In *Actes de taln'2013* (pp. 35–44).

Bank, M., & Schierle, M. (2012). A Survey of Text Mining Architectures and the UIMA Standard. In *LREC* (pp. 3479–3486).

Banko, M., Cafarella, M. J., Soderland, S., Broadhead, M., & Etzioni, O. (2007). Open Information Extraction from the Web. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence* (pp. 2670–2676).

Banko, M., & Etzioni, O. (2008). The Tradeoffs Between Open and Traditional Relation Extraction. In *ACL-HLT* (pp. 28–36).

Bannour, S., Audibert, L., & Nazarenko, A. (2011). Mesures de similarité distributionnelle entre termes. In *22es journées francophones d'ingénierie des connaissances (IC2011)* (pp. 523–538).

Bannour, S., Audibert, L., & Soldano, H. (2013). Ontology-based semantic annotation : an automatic hybrid rule-based method. In *Proceedings of BioNLP Shared Task 2013 Workshop.*

Basili, R., Pazienza, M. T., & Vindigni, M. (2000). Corpus-driven learning of Event Recognition Rules. In *Proceedings ECAI Workshop on Machine Learning for Information Extraction.*

Baum, L. E., & Petrie, T. (1966). Statistical inference for probabilistic functions of finite state Markov chains. *Annals of Mathematical Statistics*, *37*, 1554–1563.

Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The Semantic Web. *Scientific American*, *284*, 34–43.

Bikel, D. M., Schwartz, R., & Weischedel, R. M. (1999). An algorithm that learns what&lsquo ;s in a name. *Mach. Learn.*, *34* (1-3), 211–231.

Bilotti, M. W., Katz, B., & Lin, J. (2004). What works better for question answering : Stemming or morphological query expansion. In *Proceedings of the information retrieval for question answering (ir4qa) workshop at sigir* (pp. 1–3).

Bontcheva, K., Tablan, V., Maynard, D., & Cunningham, H. (2004). Evolving GATE to Meet New Challenges in Language Engineering. *Nat. Lang. Eng.*, *10*, 349–373.

Bossy, R., Golik, W., Ratkovic, Z., Bessières, P., & Nédellec, C. (2013). BioNLP Shared Task 2013 - An overview of the Bacteria Biotope Task. In *Proceedings of BioNLP Shared Task 2013 Workshop.*

Brauer, F., Rieger, R., Mocan, A., & Barczynski, W. M. (2011). Enabling Information Extraction by Inference of Regular Expressions from Sample Entities. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management* (pp. 1285–1294).

Brin, S. (1999). Extracting Patterns and Relations from the World Wide Web. In *Selected papers from the International Workshop on The World Wide Web and Databases* (pp. 172–183).

Brunet, E. (s. d.). Le lemme comme on lâaime..

Buitelaar, P., Cimiano, P., Haase, P., & Sintek, M. (2009). Towards Linguistically Grounded Ontologies. In *Proceedings of the 6th European Semantic Web Conference on The Semantic Web : Research and Applications* (pp. 111–125).

Califf, M. E., & Mooney, R. J. (1998). Relational Learning of Pattern-Match Rules for Information Extraction. In *Working Notes of AAAI Spring Symposium on Applying Machine Learning to Discourse Processing* (pp. 6–11).

Califf, M. E., & Mooney, R. J. (2003). Bottom-up relational learning of pattern matching rules for information extraction. *J. Mach. Learn. Res.*, *4*, 177–210.

Cardie, C., & Pierce, D. (1998). *Proposal for an Interactive Environment for Information Extraction* (Rapport technique).

Caruana, R., Hodor, P., & Rosenberg, J. (2000). High precision information extraction. In *KDD-2000 Workshop on Text Mining.*

Chai, J. Y., Biermann, A. W., & Guinn, C. I. (1999). Two dimensional generalization in information extraction. In *Proceedings of the sixteenth national conference on Artificial intelligence and the eleventh Innovative applications of artificial intelligence conference* (pp. 431–438).

Chieu, H. L. (2002). Named entity recognition : a maximum entropy approach using global information. In *Proceedings of COLING02* (pp. 190–196).

Chieu, H. L., & Ng, H. T. (2002). Named entity recognition : A maximum entropy approach using global information. In *Coling.*

Chieu, H. L., & Ng, H. T. (2003). Named Entity Recognition with a Maximum Entropy Approach. In *Proceedings of CoNLL-2003* (pp. 160–163).

Chiticariu, L., Krishnamurthy, R., Li, Y., Raghavan, S., Reiss, F. R., & Vaithyanathan, S. (2010). SystemT : An Algebraic Approach to Declarative Information Extraction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics* (pp. 128–137).

Chiticariu, L., Li, Y., Raghavan, S., & Reiss, F. (2010). Enterprise information extraction : recent developments and open challenges. In *Proceedings of the ACM SIGMOD International Conference on Management of Data* (pp. 1257–1258).

Chiticariu, L., Li, Y., & Reiss, F. R. (2013). Rule-Based Information Extraction is Dead ! Long Live Rule-Based Information Extraction Systems ! In *EMNLP* (pp. 827–832).

Choueka, Y., & Lusignan, S. (1985). Disambiguation by short contexts. *Computers and the Humanities*, *19*(3), 147–157.

Ciravegna, F. (2001). (LP)2, an Adaptive Algorithm for Information Extraction from Web-related Texts. In *Proceedings of the IJCAI-2001 Workshop on Adaptive Text Extraction and Mining.*

Ciravegna, F. (2003). *(LP)2 : Rule Induction for Information Extraction Using Linguistic Constraints* (Rapport technique).

Ciravegna, F., Dingli, A., Wilks, Y., & Petrelli, D. (2002). Amilcare : adaptive information extraction for document annotation. In *SIGIR* (pp. 367–368).

Cleveland, W. S. (1979). Robust Locally Weighted Regression and Smoothing Scatterplots. *Journal of the American Statistical Association*, *74*, 829–836.

Cohn, D., Atlas, L., & Ladner, R. (1994). Improving Generalization with Active Learning. *Mach. Learn.*, *15*, 201–221.

Collier, N., Nobata, C., & Tsujii, J. I. (2000). Extracting the names of genes and gene products with a hidden Markov model. In *Proceedings of the 18th conference on computational linguistics - volume 1* (pp. 201–207).

Culotta, A., Kristjansson, T. T., McCallum, A., & Viola, P. A. (2006). Corrective feedback and persistent learning for information extraction. *Artif. Intell.*, *170*, 1101–1122.

Culotta, A., & McCallum, A. (2004). Confidence Estimation for Information Extraction. In *Proceedings of HLT-NAACL 2004 : Short Papers* (pp. 109–112).

Cunningham, H. (2002). GATE, a General Architecture for Text Engineering. *Computers and the Humanities*, *36*, 223–254.

Cunningham, H. (2005). Information Extraction, Automatic. *Encyclopedia of Language and Linguistics, 2nd Edition.*

Cunningham, H., Maynard, D., & Tablan, V. (2000). *JAPE : a Java Annotation Patterns Engine (Second Edition)* (Rapport technique). Sheffield, Department of Computer Science.

Dagan, I., & Engelson, S. P. (1995). Committee-Based Sampling For Training Probabilistic Classifiers. In *Proceedings of the Twelfth International Conference on Machine Learning* (pp. 150–157).

DeJong, G. (1982). An Overview of the FRUMP System. In *Strategies for Natural Language Processing* (pp. 149–176).

Ding, L., Finin, T., Joshi, A., Pan, R., Cost, R. S., Peng, Y., ... Sachs, J. (2004). Swoogle : A Search and Metadata Engine for the Semantic Web. In *Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management* (pp. 652–659).

Doddington, G. R., Mitchell, A., Przybocki, M. A., Ramshaw, L. A., Strassel, S., & Weischedel, R. M. (2004). The Automatic Content Extraction (ACE) Program

- Tasks, Data, and Evaluation. In *LREC*.

Du, J., Zhang, Z., Yan, J., Cui, Y., & Chen, Z. (2010). Using Search Session Context for Named Entity Recognition in Query. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 765–766).

Eckstein, B., Kluegl, P., & Puppe, F. (2011). Towards Learning Error-Driven Transformations for Information Extraction. In *Proceedings of the LWA 2011 - Learning, Knowledge, Adaptation.*

Eichler, K., Hemsen, H., Löckelt, M., Neumann, G., & Reithinger, N. (2008). Interactive Dynamic Information Extraction. In *KI* (pp. 54–61).

Ekbal, A., & Bandyopadhyay, S. (2008). Improving the Performance of a NER System by Post-processing and Voting. In *Proceedings of the 2008 Joint IAPR International Workshop on Structural, Syntactic, and Statistical Pattern Recognition* (pp. 831–841).

Embley, D. W. (2004). Toward Semantic Understanding : An Approach Based on Information Extraction Ontologies. In *Proceedings of the 15th Australasian Database Conference* (pp. 3–12).

Es-salihe, M., & Bond, S. (2006). *Étude des frameworks UIMA, GATE et OpenNLP.* Consulté sur `http://www.crim.ca/fr/r-d/technologie_internet/documents/Etude-UIMA-GATE-OpenNLP.pdf`

Etzioni, O., Banko, M., Soderland, S., & Weld, D. S. (2008). Open Information Extraction from the Web. *Commun. ACM*, *51*, 68–74.

Etzioni, O., Fader, A., Christensen, J., Soderland, S., & Mausam, M. (2011). Open Information Extraction : The Second Generation. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence* (pp. 3–10).

Fader, A., Soderland, S., & Etzioni, O. (2011). Identifying Relations for Open Information Extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing* (pp. 1535–1545).

Fagin, R., Kimelfeld, B., Reiss, F., & Vansummeren, S. (2013). Spanners : A Formal Framework for Information Extraction. In *Proceedings of the 32Nd Symposium on Principles of Database Systems* (pp. 37–48).

Fellbaum, Christiane. (1998). *WordNet An Electronic Lexical Database.*

Ferrucci, D., & Lally, A. (2004). UIMA : an architectural approach to unstructured information processing in the corporate research environment. *Nat. Lang. Eng.*, *10*, 327–348.

Finn, A., & Kushmerick, N. (2003). Active learning selection strategies for information extraction. In *Proceedings of the Workshop on Adaptive Text Extraction and Mining. held in conjunction with ECML.*

Finn, A., & Kushmerick, N. (2004). Multi-level Boundary Classification for Information Extraction. In *ECML'04* (pp. 111–122).

Freitag, D., & Kushmerick, N. (2000). Boosted Wrapper Induction. In *AAAI/IAAI* (pp. 577–583).

Freitag, D., & McCallum, A. (s. d.).
In *AAAI/IAAI* (pp. 584–589).

Freitag, D., & Mccallum, A. K. (1999). Information Extraction with HMMs and

Shrinkage. In *Proceedings of the AAAI-99 Workshop on Machine Learning for Information Extraction* (pp. 31–36).

Fürnkranz, J. (1999). Separate-and-Conquer Rule Learning. *Artif. Intell. Rev.*, *13*, 3–54.

Gaizauskas, R., Demetriou, G., & Humphreys, K. (2000). Term recognition and classification in biological science journal articles. In *Proc. of the computional terminology for medical and biological applications workshop of the 2 nd international conference on nlp* (pp. 37–44).

Gardner, J. J., & Xiong, L. (2008). HIDE : An Integrated System for Health Information DE-identification. In *Proceedings of the Twenty-First IEEE International Symposium on Computer-Based Medical Systems* (pp. 254–259).

Golik, W., Bossy, R., Ratkovic, Z., & Claire, N. (2013). Improving Term Extraction with Linguistic Analysis in the Biomedical Domain. *Proceedings of the 14th International Conference on Intelligent Text Processing and Computational Linguistics (CICLing'13), Special Issue of the journal Research in Computing Science*, 24–30.

Grishman, R. (1996). *The TIPSTER Text Phase II Architecture Design, Version 2.2* (Rapport technique). New York University.

Grishman, R. (2012). Information Extraction :Capabilities and Challenges.

Gruber, T. R. (1993). A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition*, *5*, 199–220.

Hamon, T., & Aubin, S. (2006). Improving Term Extraction with Terminological Resources. In *FinTAL '06* (pp. 380–387).

Heer, J., Card, S. K., & Landay, J. A. (2005). Prefuse : A Toolkit for Interactive Information Visualization. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 421–430).

Heller, B. W., Veltink, P. H., Rijkhoff, N. J., Rutten, W. L., & Andrews, B. J. (1993). Reconstructing muscle activation during normal walking : a comparison of symbolic and connectionist machine learning techniques. *Biological Cybernetics*, *69*, 327–335.

Hobbs, J. R., Bear, J., Israel, D., & Tyson, M. (1993). FASTUS : A finite-state processor for information extraction from real-world text. In (pp. 1172–1178).

Hobbs, J. R., & Riloff, E. (2010). Information Extraction. In *Handbook of Natural Language Processing, Second Edition.*

Hsu, C.-N., & Dung, M.-T. (1998). Generating finite-state transducers for semi-structured data extraction from the Web. *Inf. Syst.*, *23*, 521–538.

Huysmans, J., Baesens, B., & Vanthienen, J. (2006). *Using Rule Extraction to Improve the Comprehensibility of Predictive Models* (Rapport technique). KU Leuven.

Isozaki, H., & Kazawa, H. (2002). Efficient Support Vector Classifiers for Named Entity Recognition. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING'02)* (pp. 390–396).

Jain, A., Ipeirotis, P., & Gravano, L. (2009). Building Query Optimizers for Information Extraction : The SQoUT Project. *SIGMOD Rec.*, *37*, 28–34.

Jayram, T. S., Krishnamurthy, R., Raghavan, S., Vaithyanathan, S., & Zhu, H. (2006). Avatar Information Extraction System. *IEEE Data Eng. Bull.*, *29*,

40–48.

Jiang, J. (2012). Information Extraction from Text. In *Mining Text Data* (pp. 11–41).

Jones, M. P., & Martin, J. H. (1997). Contextual spelling correction using latent semantic analysis. In *Proceedings of the fifth conference on applied natural language processing* (pp. 166–173).

Jones, R., Ghani, R., Mitchell, T., & Riloff, E. (2003). Active Learning for Information Extraction with Multiple View Feature Sets. In *Proceedings of the ECML-2004 workshop on Adaptive Text Extraction and Mining (ATEM-2003).*

Kambhatla, N. (2004). Combining Lexical, Syntactic, and Semantic Features with Maximum Entropy Models for Extracting Relations. In *Proceedings of the ACL 2004 on Interactive Poster and Demonstration Sessions.*

Kazama, J., Makino, T., Ohta, Y., & Tsujii, J. (2002). Tuning support vector machines for biomedical named entity recognition. In *Proceedings of the acl-02 workshop on natural language processing in the biomedical domain - volume 3* (pp. 1–8).

Kim, J.-T., & Moldovan, D. I. (1995). Acquisition of Linguistic Patterns for Knowledge-Based Information Extraction. *IEEE Trans. on Knowl. and Data Eng.*, *7*, 713–724.

Kluegl, P., Atzmueller, M., Hermann, T., & Puppe, F. (2009). A Framework for Semi-Automatic Development of Rule-based Information Extraction Applications. In *Proc. LWA 2009 (KDML - Special Track on Knowledge Discovery and Machine Learning)* (pp. 56–59).

Kluegl, P., Atzmueller, M., & Puppe, F. (2009). TextMarker : A Tool for Rule-Based Information Extraction . In *Proceedings of the Biennial GSCL Conference 2009, 2nd UIMA@GSCL Workshop* (pp. 233–240).

Krishnan, V., & Manning, C. D. (2006). An Effective Two-Stage Model for Exploiting Non-Local Dependencies in Named Entity Recognition. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics* (pp. 1121–1128).

Kristjansson, T., Culotta, A., Viola, P., & McCallum, A. (2004). Interactive Information Extraction with Constrained Conditional Random Fields. In *Proceedings of the 19th National Conference on Artifical Intelligence* (pp. 412–418).

Kushmerick, N., Weld, D., & Doorenbos, B. (1997). Wrapper induction for information extraction. In *Proc. Int. Joint Conf. Artificial Intelligence.*

Lafferty, J. D., McCallum, A., & Pereira, F. C. N. (2001). Conditional Random Fields : Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of the Eighteenth International Conference on Machine Learning* (pp. 282–289).

Leek, T. (1997). *Information Extraction Using Hidden Markov Models* (Mémoire de Master non publié). UC San Diego.

Lehnert, W. G., Cardie, C., Fisher, D., Riloff, E., & Williams, R. (1991). University of Massachusetts : description of the CIRCUS system as used for MUC-3. In *MUC* (pp. 223–233).

Lemaire, B. (2008). Limites de la lemmatisation pour l'extraction de significations.

In *9e Journées internationales d'Analyse Statistique des Données Textuelles* (pp. 725–732).

Lewis, D. D., & Catlett, J. (1994). Heterogeneous uncertainty sampling for supervised learning. In *Proceedings of ICML-94, 11th International Conference on Machine Learning* (pp. 148–156).

Li, Y., & Bontcheva, K. (2007). Hierarchical, Perceptron-like Learning for Ontology-based Information Extraction. In *Proceedings of the 16th International Conference on World Wide Web* (pp. 777–786).

Li, Y., Chiticariu, L., Yang, H., Reiss, F. R., & Carreno-fuentes, A. (2012). WizIE : A Best Practices Guided Development Environment for Information Extraction. In *Proceedings of the ACL 2012 System Demonstrations* (pp. 109–114).

Li, Y., Chu, V., Blohm, S., Zhu, H., & Ho, H. (2011). Facilitating Pattern Discovery for Relation Extraction with Semantic-signature-based Clustering. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management* (pp. 1415–1424).

Li, Y., Reiss, F., & Chiticariu, L. (2011). SystemT : A Declarative Information Extraction System. In *ACL (System Demonstrations)* (pp. 109–114).

Liu, H., Christiansen, T., Baumgartner, W. A., & Verspoor, K. (2012). BioLemmatizer : a lemmatization tool for morphological processing of biomedical text. *Journal of biomedical semantics*, *3*.

Ma, Y., Audibert, L., & Nazarenko, A. (2009). Ontologies étendues pour l'annotation sémantique. In *20es Journées Francophones d'Ingénierie des Connaissances* (pp. 205–216).

Maedche, A., Neumann, G., & Staab, S. (2003). Intelligent Exploration of the Web. In (pp. 345–359).

Makhoul, J., Kubala, F., Schwartz, R., & Weischedel, R. (1999). Performance Measures For Information Extraction. In *Proceedings of DARPA Broadcast News Workshop* (pp. 249–252).

Maynard, D., Tablan, V., Ursu, C., Cunningham, H., & Wilks, Y. (2001). Named Entity Recognition from Diverse Text Types. In *Proceedings of the Recent Advances in Natural Language Processing 2001 Conference* (pp. 257–274).

McCallum, A., Freitag, D., & Pereira, F. C. N. (2000). Maximum Entropy Markov Models for Information Extraction and Segmentation. In *Proceedings of the Seventeenth International Conference on Machine Learning* (pp. 591–598).

McCallum, A., & Li, W. (2003). Early Results for Named Entity Recognition with Conditional Random Fields, Feature Induction and Web-enhanced Lexicons. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003* (pp. 188–191).

McDowell, L. (2006). Ontology-driven information extraction with ontosyphon. In *International Semantic Web Conference.*

Miller, G. A. (1995). WordNet : a lexical database for English. *Commun. ACM*, *38*, 39–41.

Muslea, I., Minton, S., & Knoblock, C. A. (2000). Selective Sampling With Redundant Views. In *AAAI/IAAI* (pp. 621–626).

Muslea, I., Minton, S., & Knoblock, C. A. (2002). Active + Semi-supervised Learning = Robust Multi-View Learning. In *Proceedings of the Nineteenth International*

Conference on Machine Learning (pp. 435–442).

Nédellec, C., & Nazarenko, A. (2005). Ontology and Information Extraction : a necessary symbiosis. In *Ontology Design and Population* (pp. 155–170).

Nédellec, C., Nazarenko, A., & Bossy, R. (2009). Information Extraction. In *Handbook on Ontologies* (pp. 663–685).

Nigel, C. N., Collier, N., & ichi Tsujii, J. (1999). Automatic term identification and classification in biology texts. In *Proc. of the 5th nlprs* (pp. 369–374).

Olsson, F. (2009). *A literature survey of active machine learning in the context of natural language processing* (Rapport technique).

Papazian, F., Bossy, R., & Nédellec, C. (2012). AlvisAE : A Collaborative Web Text Annotation Editor for Knowledge Acquisition. In *Proceedings of the Sixth Linguistic Annotation Workshop* (pp. 149–152).

Peng, F., Feng, F., & McCallum, A. (2004). Chinese Segmentation and New Word Detection Using Conditional Random Fields. In *Proceedings of the 20th International Conference on Computational Linguistics.*

Petasis, G., Karkaletsis, V., Paliouras, G., Androutsopoulos, I., & Spyropoulos, C. D. (2002). Ellogon : A New Text Engineering Platform. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC 2002)* (pp. 72–78).

Piskorski, J., & Yangarber, R. (2013). Information Extraction : Past, Present and Future. In *Multi-source, Multilingual Information Extraction and Summarization* (pp. 23–49).

Polikar, R., Udpa, L., Udpa, S., Member, S., Member, S., & Honavar, V. (2001). Learn++ : An Incremental Learning Algorithm for Supervised Neural Networks. *IEEE Transactions on System, Man and Cybernetics (C), Special Issue on Knowledge Management*, *31*, 497–508.

Probst, K., & Ghani, R. (2007). Towards 'Interactive' Active Learning in Multi-view Feature Sets for Information Extraction. In *Proceedings of the 18th European conference on Machine Learning* (pp. 683–690).

Rabiner, L. (1989). A tutorial on hidden Markov models and selected applications inspeech recognition. , *77*, 257–286.

Ray, S., & Craven, M. (2001). Representing Sentence Structure in Hidden Markov Models for Information Extraction. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence* (pp. 1273–1279).

Reeve, L., & Han, H. (2005). Survey of semantic annotation platforms. In *Proceedings of the 2005 ACM symposium on Applied computing* (pp. 1634–1638).

Reiss, F., Raghavan, S., Krishnamurthy, R., Zhu, H., & Vaithyanathan, S. (2008). An Algebraic Approach to Rule-Based Information Extraction. In *Proceedings of the 2008 IEEE 24th International Conference on Data Engineering* (pp. 933–942).

Reymond, D. (2001). Dictionnaires distributionnels et étiquetage lexical de corpus. In *Actes des 3e Rencontres des Étudiants Chercheurs en Informatique pour le Traitement Automatique des Langues* (pp. 473–482).

Riloff, E. (1993). Automatically constructing a dictionary for information extraction tasks. In *Proceedings of the eleventh national conference on Artificial intelligence* (pp. 811–816).

Riloff, E. (1996). Automatically generating extraction patterns from untagged text. In *Proceedings of the thirteenth national conference on Artificial intelligence* (pp. 1044–1049).

Sager, N., Friedman, C., & Lyman, M. S. (1987). *Medical Language Processing : Computer Management of Narrative Data*.

Sarawagi, S. (2008). Information Extraction. *Found. Trends databases*, *1*, 261–377.

Sarma, A. D., Jain, A., & Srivastava, D. (2010). I4E : interactive investigation of iterative information extraction. In *SIGMOD Conference* (pp. 795–806).

Schäfer, U. (2006). Middleware for Creating and Combining Multi-dimensional NLP Markup. In *Proceedings of the 5th Workshop on NLP and XML : Multi-Dimensional Markup in Natural Language Processing* (pp. 81–84).

Schmid, H. (1994). Probabilistic Part-of-Speech Tagging Using Decision Trees. In *International Conference on New Methods in Language Processing* (pp. 44–49).

Settles, B. (2004). Biomedical Named Entity Recognition Using Conditional Random Fields and Rich Feature Sets. In *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and Its Applications* (pp. 104–107).

Settles, B. (2009). *Active Learning Literature Survey* (Computer Sciences Technical Report). University of Wisconsin–Madison.

Seymore, K., Mccallum, A., & Rosenfeld, R. (1999). Learning Hidden Markov Model Structure for Information Extraction. In *AAAI 99 Workshop on Machine Learning for Information Extraction* (pp. 37–42).

Shen, W., Doan, A., Naughton, J. F., & Ramakrishnan, R. (2007). Declarative Information Extraction Using Datalog with Embedded Extraction Predicates. In *Proceedings of the 33rd International Conference on Very Large Data Bases* (pp. 1033–1044).

Siefkes, C., & Siniakov, P. (2005). An Overview and Classification of Adaptive Approaches to Information Extraction. , *3730*, 172–212.

Skounakis, M., Craven, M., & Ray, S. (2003). Hierarchical Hidden Markov Models for Information Extraction. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence* (pp. 427–433).

Smith, L., Rindflesch, T., & Wilbur, W. J. (2004). MedPost : a part-of-speech tagger for bioMedical text. *Bioinformatics (Oxford, England)*, *20*, 2320–2321.

Soderland, S., Cardie, C., & Mooney, R. (1999). Learning Information Extraction Rules for Semi-structured and Free Text. In *Machine Learning* (pp. 233–272).

Soderland, S., Fisher, D., Aseltine, J., & Lehnert, W. (1995). CRYSTAL Inducing a Conceptual Dictionary. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence* (pp. 1314–1319).

Soderland, S. G. (1997). *Learning Text Analysis Rules for Domain-specific Natural Language Processing* (Thèse de doctorat non publiée).

Sönströd, C., Johansson, U., & König, R. (2007). Towards a Unified View on Concept Description. In *DMIN* (pp. 59–65).

Spasic, I., Sarafraz, F., Keane, J. A., & Nenadic, G. (2010). Medication information extraction with linguistic pattern matching and semantic rules. *JAMIA*, *17*, 532–535.

Studer, R., Benjamins, V. R., & Fensel, D. (1998). Knowledge Engineering : Principles and Methods. *Data Knowl. Eng.*, *25*, 161–197.

Sun, A., Naing, M.-M., Lim, E.-P., & Lam, W. (2003). Using support vector machines for terrorism information extraction. In *ISI'03 : Proceedings of the 1st NSF/NIJ conference on Intelligence and security informatics* (pp. 1–12).

Takeuchi, K., & Collier, N. (2002). Use of support vector machines in extended named entity recognition. In *Proceedings of the 6th conference on natural language learning - volume 20* (pp. 1–7).

Thompson, C., Califf, M., & Mooney, R. (1999). Active Learning for Natural Language Parsing and Information Extraction. In *Proceedings of the International Conference on Machine Learning* (pp. 406–414).

Turmo, J., Ageno, A., & Català, N. (2006). Adaptive information extraction. *ACM Comput. Surv.*, *38*.

Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory*.

Viterbi, A. J. (1967). Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm. *IEEE Transactions on Information Theory*, *IT-13*, 260–269.

Wang, D. Z., Michelakis, E., Franklin, M. J., Garofalakis, M. N., & Hellerstein, J. M. (2010). Probabilistic declarative information extraction. In *ICDE* (pp. 173–176).

Wellner, B., Huyck, M., Mardis, S., Aberdeen, J., Morgan, A., Peshkin, L., ... Hirschman, L. (2007). Rapidly retargetable approaches to de-identification in medical records. *Journal of the American Medical Informatics Association : JAMIA*, *14*, 564–573.

Wimalasuriya, D. C., & Dou, D. (2010). Ontology-based information extraction : An introduction and a survey of current approaches. *J. Inf. Sci.*, *36*, 306–323.

Wisniewski, G., & Gallinari, P. (2007). Relaxation Labeling for Selecting and Exploiting Efficiently Non-local Dependencies in Sequence Labeling. In *PKDD* (Vol. 4702, pp. 312–323).

Wu, F., Hoffmann, R., & Weld, D. S. (2008). Information Extraction from Wikipedia : Moving Down the Long Tail. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 731–739).

Wu, T., & Pottenger, W. M. (2005). A Semi-Supervised Active Learning Algorithm for Information Extraction from Textual Data. *JASIST*, *56*, 258–271.

Xu, J., & Croft, W. B. (1998). Corpus-based stemming using cooccurrence of word variants. *ACM Trans. Inf. Syst.*, *16*(1), 61–81.

Yangarber, R. (2001). *Scenario Customization for Information Extraction* (Thèse de doctorat non publiée). New York University.

Yangarber, R. (2003). Counter-training in discovery of semantic patterns. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics* (pp. 343–350).

Yangarber, R., Grishman, R., Tapanainen, P., & Huttunen, S. (2000). Automatic acquisition of domain knowledge for Information Extraction. In *Proceedings of the 18th conference on Computational linguistics* (pp. 940–946).

Yarowsky, D. (1993). One sense per collocation. In *Proceedings of the workshop on*

human language technology (pp. 266–271).

Yates, A., Banko, M., Broadhead, M., Cafarella, M. J., Etzioni, O., & Soderland, S. (2007). TextRunner : Open Information Extraction on the Web. In *Proceedings of Human Language Technologies : The Annual Conference of the North American Chapter of the Association for Computational Linguistics : Demonstrations* (pp. 25–26).

Yildiz, B., & Miksch, S. (2007). ontoX - A Method for Ontology-Driven Information Extraction. In *ICCSA (3)* (pp. 660–673).

Zaragoza, H., & Gallinari, P. (1998). Coupled Hierarchical IR and Stochastic Models for Surface Information Extraction. In *BCS-IRSG Annual Colloquium on IR Research.* BCS.

Zhou, G., & Su, J. (2002). Named entity recognition using an hmm-based chunk tagger. In *Proceedings of the 40th annual meeting on association for computational linguistics* (pp. 473–480).

Zhou, Z.-H., & Jiang, Y. (2003). Medical diagnosis with C4.5 rule preceded by artificial neural network ensemble. *IEEE Transactions on Information Technology in Biomedicine*, *7*, 37–42.