Apprentissage non-supervisé de la morphologie des langues à l'aide de modèles bayésiens non-paramétriques

Kevin Löser

université PARIS-SACLAY

UNIVERSITÉ PARIS SUD
Comprendre le monde, construire l'avenir

NNT : 2019SACLS203

Thèse de doctorat

Apprentissage non-supervisé de la morphologie des langues à l'aide de modèles bayésiens non-paramétriques

Thèse de doctorat de l'Université Paris-Saclay
préparée à l'Université Paris-Sud

Ecole doctorale n°580 Sciences et technologies de l'information et de la communication (STIC)
Spécialité de doctorat : Informatique

Thèse présentée et soutenue à Orsay, le 9 juillet 2019, par

M. KEVIN LÖSER

Composition du Jury :

M. PIERRE ZWEIGENBAUM Directeur de Recherche, CNRS (LIMSI)	Président
M. BENOÎT CRABBÉ Professeur, Université Paris Diderot (UFRL)	Rapporteur
M. CHRISTOPHE CERISARA Chargé de Recherche, Université Henri-Poincaré (LORIA)	Rapporteur
M. ALEXANDRE ALLAUZEN Professeur, Université Paris-Sud (LIMSI)	Directeur de thèse

Remerciements

Je souhaite remercier en premier lieu mon directeur de thèse, M. Alexandre Allauzen, Professeur à l'université Paris Sud, pour sa disponibilité, pour le temps conséquent qu'il m'a accordé et pour tous les échanges que nous avons pu avoir au cours de ces cinq années.

Un grand merci aux rapporteurs de cette thèse, M. Christophe Cerisara, Chargé de Recherche au Laboratoire lorrain de recherche en informatique et ses applications, et M. Benoît Crabbé, Maître de Conférences à l'université Paris Diderot pour l'intérêt qu'ils ont porté à mon travail.

J'associe à ces remerciements M. Pierre Zweigenbaum, Directeur de Recherche à l'université Paris Sud pour avoir accepté d'examiner mon travail.

Je remercie également M. François Yvon, Directeur du Laboratoire d'informatique pour la mécanique et les sciences de l'ingénieur, Mme Anne Vilnat, Directrice adjointe de ce même laboratoire, et M. Jean-Luc Gauvain, Directeur du groupe Traitement du Langage Parlé pour m'avoir accueilli au sein de leur équipe.

Table des matières

Introduction

0.1 Les morphèmes en linguistique

En linguistique, les *morphèmes* désignent des unités linguistiques à une échelle intermédiaire entre le mot et le phonème, caractérisées par le fait d'être les unités significatives minimales (c'est-à-dire les plus petites unités porteuses de sens). Par exemple, le mot *parcourir* est composé de trois morphèmes : le préfixe *par-*, la racine *-cour-* et la désinence *-ir* (marque de l'infinitif). Chacune de ces unités est :

— *significative* dans le sens où la supprimer ou la remplacer par une autre unité (par exemple supprimer ou remplacer *par-* pour former par exemple : *courir*, *discourir*, *accourir*) entraîne une modification du sens du mot.

— *minimale* dans le sens où chacune de ces unités ne peut pas à son tour être elle-même décomposée en sous-unités significatives.

Cette définition n'est pas sans poser de nombreuses difficultés. Par exemple, considérons les décompositions en morphèmes de différentes formes du verbe *aller* : *v-ais*, *v-as*, *all-ons*, *ir-ons*. Dans chacune de ces formes, on trouve une racine (*v-*, *all-*, *ir-*) porteuse du sens de l'action d'aller ainsi qu'un suffixe *-ais*, *-as*, *-ons* qui modifie le sens en désignant différentes personnes. La question qui se pose est : faut-il considérer *v-*, *all-*, *ir-* comme trois unités distinctes, ou comme trois réalisations différentes d'une même unité abstraite ? La deuxième option est suggérée par le fait que les formes considérées sont toutes des conjugaisons d'un même verbe, que pour la grande majorité des verbes les différentes formes conjuguées possèdent un morphème "racine" en commun, et donc qu'il serait plus élégant d'expliquer les différentes formes du verbe irrégulier *aller* comme composées elles aussi d'une racine abstraite commune, dont la réalisation concrète est modifiée par les autres morphèmes présents. Un autre argument en faveur de cette explication est que *v-*, *all-*, *ir-* ont un sens identique (celui de l'action d'aller). Cette constatation a entraîné les linguistes à établir une distinction entre *morphèmes* (unités abstraites) et *morphes* (les différentes réalisations concrètes d'une unité abstraite). Ainsi dans notre exemple, *v-*, *all-*, *ir-* sont des *morphes* distincts qui sont tous les trois des réalisations concrètes d'un même *morphème* abstrait. On dit alors que *v-*, *all-*, *ir-* sont des *allomorphes*.

D'autres langues présentent des difficultés supplémentaires. Par exemple, en arabe, *'aktubu* (1ps, présent), *taktubu* (2ps, présent), *yaktubu* (3ps, présent),

katabtu (1ps, passé), *katabta* (2ps, passé), *kataba* (3ps, passé) [1] sont toutes des formes conjuguées du verbe *kataba* (lire). Ces différentes formes peuvent être expliquées de la manière suivante : entre les trois consonnes *k-t-b* porteuses du sens de "lire" (et ayant donc le rôle d'une racine) sont intercalés différents "schèmes" modifiant le sens en désignant une personne : *'a_ _u_u* pour désigner la 1ère personne du singulier au présent, *_a_a_tu* pour désigner la 1ère personne du singulier au passé, *ta_ _u_u* pour désigner la 2ème personne du singulier au présent, *_a_a_ta* pour désigner la 2ème personne du singulier au passé, et ainsi de suite... Ces mêmes schèmes vocaliques sont utilisés pour conjuguer de nombreux verbes en arabe : ainsi le verbe *darasa* (étudier) dont la racine est donnée par les trois consonnes *d-r-s* aura comme formes conjuguées : *'adrusu* (1ps, présent), *tadrusu* (2ps, présent), *yadrusu* (3ps, présent), *darastu* (1ps, passé), *darasta* (2ps, passé), *darasa* (3ps, passé). Ainsi les morphèmes en présence sont d'une part les racines à trois consonnes (*k-t-b*, *d-r-s*, ...), d'autre part les schèmes s'intercalant entre les consonnes (*'a_ _u_u*, *_a_a_tu*, ...). Nous constatons donc que dans la langue arabe (et c'est une caractéristique de la plupart des langues sémitiques comme aussi l'hébreu ou l'araméen), lorsque nous parlons de morphèmes comme d'"unités minimales", "unité" n'est pas à comprendre au sens de "chaîne contigue de phonèmes" : ici il s'agit de chaînes discontinues susceptibles d'accueillir d'autres chaînes discontinues par intercalation.

0.2 Le rôle des morphèmes en traitement des langues

En traitement automatique des langues (TAL), on adopte un point de vue moins rigoureux qu'en linguistique et en général on ignore toutes les difficultés inhérentes à la définition des morphèmes, les questions d'allomorphie et les distinctions entre *morphe* et *morphème*, puisqu'on se préoccupe moins de bien fonder des notions sur le plan théorique, que de produire des modèles implémentables et efficaces. Toutefois, pour des raisons différentes qu'en linguistique, les morphèmes jouent également un rôle important.

0.2.1 L'apprentissage statistique

L'approche moderne en TAL est l'*apprentissage statistique* qui consiste à développer des systèmes de TAL non pas en codant "à la main" des règles linguistiques (les *systèmes à base de règles* étant une approche qui fut autrefois plus répandue), mais en inférant ces règles à l'aide de statistiques obtenues sur un corpus (éventuellement annoté par des linguistes). Par exemple, si on voulait concevoir un système de traduction automatique :

— Avec l'approche *à base de règles*, on écrirait sous forme informatique un dictionnaire bilingue associant à chaque mot de la langue source ses tra-

1. 1ps : première personne du singulier, 2ps : deuxième personne du singulier, 3ps : troisième personne du singulier.

ductions possibles, et on coderait aussi (à l'aide de linguistes) des règles de réordonnancement pour que les phrases traduites soient conformes à l'ordre des mots dans la langue cible.

— Avec l'approche par *apprentissage statistique*, on fournirait à une machine un corpus bilingue, c'est-à-dire un grand nombre d'exemples de phrases dans la langue source associées à leur traduction (réalisée par des humains) dans la langue cible. Puis un algorithme calculerait des statistiques sur ce corpus, par exemple :

  — pour tout couple de mots (w_s, w_c) où w_s est un mot de la langue source et w_c est un mot de la langue cible, le nombre de fois où w_c apparaît dans une phrase traduite du corpus lorsque w_s apparaît dans la phrase correspondante en langue source. Ceci constituerait le *modèle de traduction* et aiderait à la traduction de mots individuels.

  — pour tout couple de mots (w_1, w_2) de la langue cible, le nombre de fois que w_2 apparaît à la suite de w_1 dans la langue cible. Ceci constituerait le *modèle de langue* et aiderait à agencer les mots traduits de sorte à avoir un ordre conforme à celui de la langue cible.

  Une fois ces statistiques calculées, étant donnée une phrase source non observée dans le corpus, on pourrait produire une phrase en langue cible, à l'aide d'un algorithme qui rechercherait une phrase (en langue cible) effectuant un bon compromis entre les indications données par le modèle de traduction et par le modèle de langue.

Le terme d'*apprentissage* est du au fait que dans cette dernière approche, la machine semble "découvrir" par elle-même des règles de traduction et généraliser ces règles à des exemples de phrases non rencontrées dans le corpus d'apprentissage.

Selon cette approche, la qualité d'un système sera d'autant plus grande que le corpus d'apprentissage qui lui aura été fourni sera de taille suffisamment importante pour épuiser les possibiltés combinatoires de la langue à traiter. Par exemple, un corpus trop petit couvrira une partie insuffisante du lexique de la langue : que faire alors lorsqu'il sera demandé au système de traduire un mot qu'il n'a pas rencontré dans le corpus [2] ? Ce *problème de parcimonie des données* est lié à la structure même des langues où la majorité des évènements possibles sont des évènements rares (par exemple, la plupart des mots qu'on rencontrera dans un dictionnaire seront des mots rares), et est d'autant plus saillant que pour de nombreuses langues, les données informatiques sont en quantité trop limitée pour obtenir des systèmes de qualité. Ce problème est lié au problème du *sur-apprentissage* (*overfitting*), à savoir que pour des évènements rares, un système statistique aura tendance à accorder trop de crédit aux statistiques du corpus relatives à cet évènement rare. Par exemple, si dans un corpus le mot rare *chryséléphantines* n'apparait que dans une phrase ("*Ces anciennes chryséléphantines grecques ont connu un nouvel attrait pour les collectionneurs*"), un système se fiant aux statistiques de succession de mots dans

2. de tels mots sont appelés OOV : *out-of-vocabulary words*

le corpus pourra croire à tort que le mot *chryséléphantines* est nécessairement suivi du mot *grecques* en français, et pourra par exemple mal traduire en français la phrase "*In the 20th century, a few European chryselephantine sculptures of Greek inspiration were produced*".

0.2.2 Réduire la parcimonie des données à l'aide de la morphologie

Les problèmes de parcimonie des données et de sur-apprentissage apparaissent lorsque la diversité du lexique est trop grande. Une solution est alors de travailler à une échelle plus élémentaire que le mot, d'où l'utilité de la morphologie en traitement des langues. C'est particulièrement le cas dans les langues qualifiées de *morphologiquement riches* (comme le turc, le finnois ou le hongrois), où les mots sont formés en combinant un grand nombre de morphèmes qui déterminent chacun un aspect du sens. Dans certaines langues, cette combinatoire est telle qu'un seul mot peut exprimer un sens qui ne pourrait être traduit que par une phrase entière dans d'autres langues (en inuit, le seul mot "*angyaghillangyugtug*" signifie "*il veut acheter un grand bateau*".[3]). En effet, lorsque la combinatoire morphologique est riche, les formes de mots prolifèrent et il devient d'autant plus difficile pour un corpus d'offrir une couverture suffisante du lexique possible. Décomposer un mot en ses morphèmes constituants permet alors de "factoriser" le lexique et de travailler à une échelle présentant un moins grand nombre d'évènements rares. Par exemple, considérons en français les neuf verbes *compliquer*, *comparer*, *impliquer*, *parer*, *répliquer*, *réparer*, *porter*, *comporter*, *importer*, ainsi que leurs formes conjuguées au présent de l'indicatif. Prendre en compte l'ensemble de ces formes en travaillant à l'échelle du mot nécessiterait $9 \times (5+1) = 54$ mots (car 9 verbes, et pour chaque verbe l'infinitif ainsi que 5 personnes puisque la 1ère et 3ème personne du singulier sont les mêmes). En revanche, en travaillant sur le lexique de morphèmes contenant les préfixes *com-, im-, ré-*, les racines *-pliqu-, -par-, -port-* et les suffixes *-er,-e,-es,-ons,-ez,-ent*, le lexique nécessaire est réduit à $3 + 3 + 6 = 12$ morphèmes. Cette factorisation se base sur les aspects *concaténatifs* de la morphologie du français : c'est-à-dire que les morphèmes considérés sont des chaînes de caractères (ou de phonèmes) contiguës, et les formes de mots capturées par ce lexique de morphèmes s'obtiennent par concaténation de ces morphèmes.

De même, dans l'exemple de la langue arabe évoqué plus haut, distinguer les racines consonantiques des schèmes venant s'y intercaler permettrait de factoriser le lexique des verbes. Il s'agirait pour cela de traiter des phénomènes de *morphologie non-concaténative* : en effet, il ne s'agit plus d'une simple concaténation de morphèmes se manifestant par des chaînes de caractères (ou de phonèmes) contiguës. La morphologie non-concaténative pose des problèmes supplémentaires en TAL, notamment sur le plan algorithmique, et dans cette thèse nous n'aborderons que la morphologie concaténative.

3. *angya = bateau, ghilla = grand, ng = acquérir, yug = volonté, tug =* 3è personne du singulier

Enfin évoquons deux problèmes de TAL apparentés à l'apprentissage de morphologie :

— La *tokénisation* (segmentation en mots) : il s'agit d'une tâche consistant à segmenter un flux continu de phonèmes (ou de caractères) en mots. Cette tâche est utile notamment dans le traitement des langues non écrites pour lesquelles les seules données sont des enregistrements sur lesquels ne figurent pas de coupures explicites entre les mots. Un premier traitement des données préalable à tout traitement ultérieur de telles langues nécessite de les segmenter en mots. De même qu'en apprentissage de morphologie, il s'agit de découvrir dans des chaînes de caractères/phonèmes des unités minimales permettant d'expliquer un grand nombre de formes possibles. La différence principale avec la morphologie est une différence d'échelle : alors qu'en morphologie, on cherchera à segmenter des mots en phonèmes, en tokénisation on cherchera à segmenter des phrases en mots.

— La *compression de texte* : puisque la connaissance de la morphologie d'une langue permet de se ramener à un lexique moins divers, elle permet aussi d'encoder du texte à moindre coût.

0.3 But

Dans cette thèse, nous aborderons le problème de l'apprentissage de la morphologie dans un cadre *non-supervisé*, c'est-à-dire à partir de texte brut dénué de toute annotation par des linguistes, et à l'aide de modèles n'ayant recours à aucune connaissance experte. Notre but est d'apporter les contributions suivantes :

— Présenter un panorama des méthodes existantes de traitement de la morphologie, ainsi que quelques points de repère pour s'orienter dans la diversité de ces méthodes.

— Décrire une application des *méthodes bayésiennes non-paramétriques* au traitement de la morphologie. Ces méthodes permettent de modéliser des données en recherchant un compromis entre deux objectifs opposés : la fidélité du modèle aux données, et la simplicité du modèle. Nous présenterons ces méthodes d'une part sous l'aspect formel de modèles probabilistes en utilisant une formalisation mathématique, d'autre part sous leurs aspects algorithmiques en abordant les difficultés liées à une implémentation computationnellement efficace de celles-ci.

— Évaluer la pertinence des segmentations morphologiques obtenues par ces méthodes, à l'aune de segmentations de référence produites par des linguistes.

Le contenu des chapitres est le suivant :

— **Chapitre 1 :** Nous présentons un aperçu général de la littérature concernant le traitement de la morphologie, en présentant de manière plus approfondie deux méthodes ayant connu un succès notoire : *Linguistica* et

Morfessor. Enfin nous proposons une grille de classification des différentes approches selon certains axes pouvant servir de points de repère dans la littérature.

— **Chapitre 2 :** Nous introduisons les aspects mathématiques des processus de Dirichlet et de Pitman-Yor, et présentons comme application de la théorie un premier algorithme de segmentation morphologique se basant sur une explication des formes de mots comme échantillons d'un modèle unigramme de morphèmes.

— **Chapitre 3 :** Nous introduisons un formalisme mathématique afin de définir les processus de Pitman-Yor hiérarchiques qui permettent de modéliser de manière plus fine des dépendances entre morphèmes. Comme application, nous présentons et évaluons deux algorithmes de segmentation : l'un se basant sur un modèle n-gramme de morphèmes, l'autre sur un modèle semi-Markov caché regroupant les morphèmes en classes morphologiques et induisant une représentation de la morphologie de la langue comme un automate à états finis.

— **Chapitre 4 :** Nous traitons la question du choix des hyperparamètres dans un processus de Pitman-Yor hiérarchique et décrivons une stratégie d'échantillonnage des hyperparamètres selon un *a posteriori* bayésien. Les aspects algorithmiques du calcul de cet *a posteriori* sont traités. Enfin nous évaluons l'influence du rééchantillonnage des hyperparamètres sur la performance de nos méthodes.

Chapitre 1

L'apprentissage non-supervisé de morphologie

1.1 Approches par *letter successor varieties*

1.1.1 Un critère distributionnel de segmentation de chaînes de phonèmes

Le linguiste Zellig Harris publia un des articles initiateurs de la recherche en segmentation morphologique non-supervisée [30]. Le but de cet article n'était pas de décrire un algorithme de segmentation pouvant être utilisé pour le traitement informatique des langues, mais de fournir aux linguistes un critère permettant de délimiter des unités telles que des morphèmes ou des mots dans un flux continu de phonèmes. De plus, Harris cherchait un critère purement distributionnel, c'est-à-dire un critère uniquement basé sur les statistiques d'apparition des phonèmes dans les discours des locuteurs, et ne nécessitant pas de prendre en compte l'aspect sémantique, c'est-à-dire les modifications de sens que certaines chaînes de phonèmes imposaient au discours.

L'idée de Harris découle de l'observation suivante : il est possible de segmenter une chaîne de phonèmes en cherchant les endroits où cette chaîne s'articule, c'est-à-dire en cherchant les segments de cette chaîne qui auraient pu être remplacés par une grande variété d'autres segments tout en conservant une phrase bien formée. Par exemple, considérons la phrase (non segmentée) `he'squicker`. Considérons d'une part les segments qui auraient pu être substitués au segment `-icker`, par exemple :

— `-ick` pour former la phrase : *he's quick*

— `-aint` pour former la phrase : *he's quaint*

— `-alified` pour former la phrase : *he's qualified*

— `-iteacharacter` pour former la phrase : *he's quite a character*

A présent regardons ce qu'il en est du segment `-quick-` :

— `-fast-` pour former la phrase : *he's faster*

— `-slow-` pour former la phrase : *he's slower*

— `-bigg-` pour former la phrase : *he's bigger*

— `-small-` pour former la phrase : *he's smaller*

— et ainsi de suite... toutes les racines d'adjectifs admettant une forme comparative finissant en `-er` peuvent être utilisées

— sans compter d'autres exemples comme `-clev-` pour former la phrase : *he's clever*

Ainsi, le segment `-quick-` possède une bien plus grande variété de segments qui lui sont substituables que le segment `-icker`. Selon le critère de Harris, on en déduit que le segment `-quick-` est un bien meilleur candidat pour être un segment de la phrase que le segment `-icker`.

Dans l'article de Harris, le cadre est le suivant : on cherche à segmenter une certaine chaîne de phonèmes (*"utterance"*) U en se reférant à un corpus C de chaînes de phonèmes "identiques à U". La première procédure proposée pour mettre en oeuvre ce critère est la suivante : si $U = u_1 u_2 \dots u_l$ (où u_i sont des phonèmes), pour tout préfixe $u_1^k = u_1 \dots u_{k-1}$ ($k \leq l$) de U, on calcule la *variété de successeurs* de u_1^k, notée $S(u_1^k)$, de la manière suivante :

— Initialement, $S(u_1^k)$ est un ensemble vide.

— A chaque fois que u_1^k apparaît comme préfixe d'une chaîne de phonèmes $c \in C$, on ajoute à $S(u_1^k)$ le phonème de c qui succède immédiatement à u_1^k.

Par exemple, si on souhaite segmenter U : `he'squicker` et que le corpus de chaînes de phonèmes à disposition contient : `he'squick, he'squaint, he'squalified, he'squiteacharacter, he'sfaster, he'sslower, he'sbigger, he'ssmaller, he'sclever`, alors par exemple :

— La variété de successeurs S(`he'squ`) contiendra : `i,a`.

— La variété de successeurs S(`he's`) contiendra : `q,f,s,b,c`.

Harris observe alors que lorsqu'on suit l'évolution de la taille de $S(u_1^k)$ le long de la chaîne (si tant est que le corpus de référence est suffisamment grand), cette taille aura tendance à décroître, puis à effectuer un pic à l'endroit d'une articulation, puis décroître à nouveau, et ainsi de suite. Afin de segmenter la chaîne, Harris propose d'introduire une coupure à chaque position k où $|S(u_1^k)|$ présente un pic (sans pour autant définir rigoureusement ce qu'il considère comme un pic).

1.1.2 Raffinements

Harris propose ensuite plusieurs raffinements de cette méthode :

— **Variétés de prédécesseurs :** D'abord, il constate un problème de *parcimonie* du corpus lorsque la chaîne à segmenter est trop longue : pour des k élevés, même si la position k est une coupure linguistiquement pertinente, il se peut qu'il y ait trop peu de chaînes dans le corpus de référence qui commencent par u_1^k pour fournir une bonne estimation de $|S(u_1^k)|$. Une solution proposée est de prendre aussi en compte les *variétés de prédecesseurs*, c'est-à-dire pour tout suffixe $u_j^l = u_j \dots u_l$ de U, compter le nombre de phonèmes différents qui précèdent u_j^l et introduire une coupure là où ce nombre présente un pic.

— **Variétés d'insertions :** Pour chaque position k de U, recenser les chaînes du corpus qui commencent par u_1^k et terminent pas u_k^l. Pour chaque chaîne de la forme : $c = u_1 \dots u_{k-1} \underbrace{c_{k+1} \dots c_{k+i}}_{\text{insertion}} u_k \dots u_l$, ajouter $c_{k+1} \dots c_{k+i}$ à un ensemble $I(k)$ (*variété d'insertions*). Ensuite compter le nombre de phonèmes différents apparaissant en première position d'une chaîne de $I(k)$, et le nombre de phonèmes différents apparaissant en dernière position d'une chaîne de I, et introduire une coupure là où la somme de ces deux nombres présente un pic. Par exemple, si on souhaite segmenter `thisisnew`, et que l'ensemble des chaînes de référence est `theirisisnew, thosepeoplesaidthattheirisisnew, thisirisisnew, thiscarisnew, thishouseisnew, thiscityisnew` alors :
  — La variété $I(2)$ des insertions entre `th-` et `-isisnew` est : `-eir-` (*the iris is new*) et `-osepeoplesaidthattheir-` (*those people said that the iris is new*). Le phonèmes apparaissant au début d'une chaîne de $I(2)$ sont : `e,o` et les phonèmes apparaissant à la fin d'une chaîne de $I(2)$ sont : `r`. Le nombre à considérer est donc : $|\{\texttt{e}, \texttt{o}\}| + |\{\texttt{r}\}| = 3$.
  — La variété $I(4)$ des insertions entre `this-` et `-isnew` est : `-iris-, -car-, -house-, -city-` et le nombre à considérer est donc : $|\{\texttt{i}, \texttt{c}, \texttt{h}\}| + |\{\texttt{s}, \texttt{r}, \texttt{e}, \texttt{y}\}| = 7$.
  — Le critère favorise donc la segmentation `this-isnew` par rapport à la segmentation `th-isisnew`.

1.1.3 Implémentation

L'article [27], présente une implémentation sur machine de ces critères, et les évalue en calculant la précision et le rappel des segmentations produites sur des données de test. De nombreuses variantes sont comparées, notamment :

— Plutôt que de chercher les pics de $|S(u_1^k)|$, chercher les pics d'entropie de la distribution $\mathbb{P}(c|u_1^k) = \dfrac{f(u_1^k c)}{f(u_1^k)}$, où $f(u_1^k)$ désigne la fréquence d'apparition de u_1^k en préfixe dans le corpus. L'*entropie du successeur* de u_1^k vaut alors $-\sum_{u \in \text{phonèmes}} \mathbb{P}(c|u_1^k) \log \mathbb{P}(c|u_1^k)$ et quantifie le degré d'incertitude quant au phonème qui succèderait à u_1^k.

— De même pour l'entropie du prédecesseur.

— Introduire une coupure à la position k soit lorsque $|S(u_1^k)|$ dépasse un certain seuil fixé à l'avance, soit lorsque k est un maximum local (c'est-à-dire : $|S(u_1^k)| > |S(u_1^{k-1})|$ et $|S(u_1^k)| > |S(u_1^{k+1})|$). De même considérant la taille de la variété des prédecesseurs, ou l'entropie du successeur/prédecesseur.

Les conclusions tirées des évaluations de l'algorithme sont les suivantes : les meilleurs résultats sont obtenus lorsqu'à la fois les variétés de successeurs et de prédecesseurs sont prises en comptes, et lorsqu'une coupure est introduite dès lors que : soit la variété de successurs excède un seuil important et la variété de prédecesseurs un seuil plus modéré, soit l'inverse. Utiliser un critère de maximum local pour placer une coupure donne en général de moins bons résultats. Enfin les performances sont similaires selon qu'on se réfère à la taille de la variété des successeurs/prédecesseurs ou à l'entropie.

1.1.4 Recherche de paradigmes

On peut évoquer l'article de Nikolai Andreev [32], contemporain des travaux de Harris, et reprenant des idées similaires. Cet article est présenté et commenté en français dans [9]. Le cadre est le même que l'article de Harris (à ceci près que des chaînes de caractères sont considérées plutôt que des chaînes de phonèmes), et brièvement, l'idée de l'algorithme est la suivante.

La première étape est de repérer un premier affixe (préfixe ou suffixe), appelé *bootstrap affix* car il servira à initier le processus de segmentation. Pour cela, on calcule un modèle unigramme de caractères $\mathbb{P}(c)$ en comptant les fréquences d'apparition des caractères dans le corpus. De plus, pour un certain nombre de positions $k \in \{1,2,3\} \cup \{-1,-2,-3\}$, on calcule la distribution $\mathbb{P}(c|k)$ de caractères apparaissant à la position k dans les chaînes de caractères du corpus. Ici, les positions négatives $k = -1, -2, -3$ désignent les positions en partant de la fin de la chaîne. Ainsi, si le corpus est contient des chaînes $w^1, \ldots, w^n$, et que chaque chaîne s'écrit $w^i = w_1^i w_2^i \ldots w_{l_i}^i$ avec : $(w_k^i)_{1 \leq k \leq l_i}$ les caractères de la chaîne et l_i la longueur de la chaîne, on définit :

$$\mathbb{P}(c) = \frac{\sum_{i=1}^{n} \sum_{k=1}^{l_i} \mathbf{1}_{w_k^i = c}}{\sum_{i=1}^{n} l_i} \quad \text{et} \quad \mathbb{P}(c|k) \begin{array}{l} = \dfrac{\sum_{i=1}^{n} \mathbf{1}_{w_k^i = c}}{n} \text{ si } k \in \{1,2,3\} \\ = \dfrac{\sum_{i=1}^{n} \mathbf{1}_{w_{l_i - k}^i = c}}{n} \text{ si } k \in \{-1,-2,-3\} \end{array}$$

Ensuite, un ensemble d'*informants* est calculé : Andreev définit un informant comme un caractère dont la probabilité d'apparition est fortement corrélée à une certaine position dans les chaînes. Les informants sont calculés en cherchant, pour tous les $k = \pm 1, 2, 3$, les caractères dont la probabilité $\mathbb{P}(c|k)$ dépasse de moitié la probabilité $\mathbb{P}(c)$. Ensuite, on essaye d'étendre les informants en affixes (préfixe ou suffixe) à l'aide d'un critère similaire à celui employé par Harris : étant donné un informant dans une chaîne du corpus, on cherche à l'étendre

à droite ou à gauche, en lui ajoutant les caractères situés à sa droite (ou à sa gauche). Plus précisément :

1. Chaque informant est considéré comme un "germe" d'affixe.
2. Si a est un germe d'affixe apparaissant entre les positions j et k dans une chaîne $w_1 \dots w_l$, alors le caractère situé immédiatement à gauche de a est w_{j-1} et le caractère situé immédiatement à droite est w_k. On compte alors la fréquence P_{gauche} d'apparition de w_{j-1} juste à gauche de a dans le corpus, et la fréquence P_{droite} d'apparition de w_k juste à droite de a dans le corpus.
3. Si P_{droite} excède un certain seuil, on étend le germe d'affixe vers la droite. De même, si P_{gauche} excède un certain seuil, on étend le germe d'affixe vers la gauche. Si les deux probabilités sont en-dessous du seuil, le germe d'affixe est rejeté. Sinon, on essaye de continuer l'extension du germe en répetant cette étape.
4. On choisit comme *bootstrap affix* le premier germe d'affixe qu'on a réussi à prolonger soit vers la gauche jusqu'au début d'une chaîne (auquel cas c'est un préfixe), soit vers la droite jusqu'à la fin d'une chaîne (auquel cas c'est un suffixe).

Une fois un *bootstrap affix* obtenu, Andreev décrit une procédure itérative pour élargir l'ensemble des morphèmes obtenus. Pour ne pas alourdir notre exposition, nous ne décrirons pas cette procédure en détail car un grand nombre d'heuristiques relativement arbitraires sont employées, mais l'idée générale est la suivante :

1. Considérer l'ensemble des chaînes dans lesquelles apparaît le *bootstrap affix*. En ôtant cet affixe à chacune des chaînes considérées, on obtient un ensemble de *racines candidates*.
2. Pour chaque racine candidate trouvée, regarder l'ensemble des mots du corpus où cette racine apparaît, et calculer dans cette ensemble (par une procédure similaire au calcul du *bootstrap affix*) un nouvel ensemble d'*affixes candidats*.
3. Pour chaque affixe parmi les nouveaux candidats, calculer de nouveau l'ensemble de racines dans lesquelles cet affixe apparaît. Dans ce nouvel ensemble de racines, ne garder que celles qui apparaissent aussi avec le *bootstrap affix* dans une des chaînes du corpus, et les ajouter aux racines candidates.
4. Répéter ces étapes jusqu'à épuisement des racines et affixes candidats.

Après terminaison de la procédure, on obtient donc un ensemble de racines et un ensemble d'affixes susceptibles d'êtres concaténés à ces racines, ce qui correspond à la notion linguistique de *paradigme*. On peut dès lors ignorer dans le corpus toutes les chaînes rentrant dans ce paradigme, et chercher un nouveau paradigme parmi les chaînes restantes.

Le résultat final de l'algorithme d'Andreev est donc une représentation de la morphologie dont la nature est différente de celle de l'algorithme de Harris. Alors

que dans l'algorithme de Harris, le résultat est une segmentation des chaînes du corpus, dans l'algorithme d'Andreev, le résultat est plus riche dans le sens où les chaînes du corpus sont regroupées en paradigmes, où chaque paradigme consiste en un certain ensemble de racines pouvant recevoir un certain ensemble d'affixes. En revanche, dans l'algorithme d'Andreev, une chaîne ne peut être coupée qu'en deux segments (la racine et l'affixe), tandis que l'algorithme de Harris autorise un nombre potentiellement illimité de segments.

1.2 Approches par longueur de description minimale

Les méthodes que nous avons présentées plus haut s'appuient sur des critères de coocurrence entre caractères/phonèmes voisins pour placer des coupures dans les chaînes du corpus à segmenter. Bien que conceptuellement simples, elles nécessitent d'avoir recours à un grand nombre d'heuristiques et de seuils arbitrairement fixés. Dans cette section, nous décrivons des approches présentant une assise plus rigoureuse, car s'appuyant sur un principe général d'inférence statistique appelé le *principe de longueur de description minimale* (MDL). Dans ces approches, le point de vue adopté sur la morphologie a un caractère plus global : l'accent est mis sur l'inférence d'un lexique de segments et d'un modèle explicatif de la formation des chaînes du corpus, plutôt que sur des conditions locales déterminant le placement des coupures.

1.2.1 Longueur de description d'un jeu de données

1.2.1.1 Complexité de Kolmogorov

Dans l'article [38], le mathématicien Andreï Kolmogorov propose de définir quantitativement la complexité d'un objet combinatoire $\mathbf{x}$ (par exemple : une chaîne de caractères, un graphe...) comme la longueur minimale d'un programme capable de produire $\mathbf{x}$ (et écrit dans un langage de programmation universel, par exemple dans un langage de symboles pouvant être lu par une machine de Turing universelle). Cette *complexité de Kolmogorov* correspond bien à l'idée intuitive qu'un objet sera d'autant plus complexe qu'il sera long à décrire.

Par exemple, considérons les deux chaînes de bits suivantes :

$$\mathbf{x} = 0001000100010001000100010001000100010001000100010001$$
$$\mathbf{y} = 0100110100101010000010111001101011100000101011100$$

La chaîne $\mathbf{x}$ présente une bien moindre complexité que la chaîne $\mathbf{y}$ qui semble avoir été générée aléatoirement. Par exemple, il sera beaucoup plus facile pour un humain de retenir la première que la seconde. Et en effet, en termes de complexité de Kolmogorov, la première chaîne peut être produite par le programme suivant :

```
for i = 1...48 do
    if i mod 4 = 0 then
        print 1
    else
        print 0
    end if
end for
```

alors que pour la deuxième chaîne, il est difficile d'imaginer un programme autre que le programme trivial suivant :

print 0100110100101010000010111001101011100001010111 00

Notons qu'il y a une ambigüité dans la définition de la complexité de Kolmogorov, puisque la longueur d'un programme générant un objet dépend du langage dans lequel est écrit ce programme. Il s'avère que d'après un *théorème d'invariance* prouvé par Kolmogorov [38], pour des objets suffisamment complexes, les complexités c_A et c_B (relativement à deux langages "universels" différents A et B) sont équivalentes à une constante près, et donc qu'à une relation de proportionnalité près, les complexités de Kolmogorov peuvent être définies indépendamment d'un langage particulier.

Cette notion de complexité présente un intêret surtout théorique, car deux obstacles la rendent inapplicable en pratique :

— Son incalculabilité : calculer la complexité de Kolmogorov d'un objet $\mathbf{x}$ de "taille" finie (par exemple une chaîne de bits de longueur finie) nécessiterait de rechercher, parmi l'ensemble des programmes générant $\mathbf{x}$ et de longueur inférieure au programme trivial "print $\mathbf{x}$", celui qui est de longueur minimale. Or une telle recherche prendrait un temps exponentiel, donc la complexité de Kolmogorov est donc *pratiquement* incalculable. Mais de plus, dès que cet ensemble de programmes contiendrait un programme contenant une boucle infinie, on se heurterait au *problème de l'arrêt*, du au fait qu'il est impossible de déterminer de manière générale en un temps fini si un programme se termine ou non. On a donc en plus une incalculabilité *théorique*.

— La dépendance au langage utilisé : le théorème d'invariance de Kolmogorov est un théorème asymptotique, mais n'assure pas que pour des objets de petite complexité, il n'y ait pas de grandes différences dans les longueurs des programmes générant $\mathbf{x}$ selon le langage utilisé.

1.2.1.2 Le principe de longueur de description minimale

L'intêret de cette notion dans le cadre de l'apprentissage non-supervisé est le suivant : la description la plus courte possible d'un jeu de données sera celle qui exploitera le plus possible les régularités de celui-ci et s'approchera le mieux de la "loi cachée" derrière ce jeu de données. Ce qui revient à dire qu'apprendre de manière non-supervisée un jeu de données équivaut à chercher sa compression optimale.

Toutefois, pour rendre ce principe applicable, on a vu qu'il est nécessaire de renoncer à utiliser un langage de programmation universel pour représenter les données, et de se restreindre à des classes de modèles moins expressives. L'article de Rissanen [50] propose un cadre pour des implémentations pratiques du principe MDL. Son idée est de séparer la longueur de description des données en longueur de description d'un modèle, et longueur de description des données à l'aide de ce modèle :

— On souhaite décrire un jeu de données $\mathcal{X} = (\mathbf{x_1}, \ldots, \mathbf{x_n})$.

— On dispose pour cela d'une famille de modèles $\mathcal{M}$. Chaque modèle $M \in \mathcal{M}$ peut être décrit par un code de longueur $L(M)$.

— Etant donné un modèle $M \in \mathcal{M}$, chaque $\mathbf{x_i} \in \mathcal{X}$ peut être décrit à l'aide de ce modèle par un code de longueur $L(\mathbf{x_i}|M)$

— Le modèle préconisé par le principe MDL sera donc celui qui minimisera la longueur totale des codes nécessaires pour décrire le modèle et les données à l'aide du modèle :

$$M_{\text{optimal}} = \underset{M \in \mathcal{M}}{\operatorname{argmin}} \Big(\underbrace{L(M)}_{\text{(1) : description du modèle}} + \underbrace{\sum_{\mathbf{x_i} \in \mathcal{X}} L(\mathbf{x_i}|M)}_{\text{(2) : description des données à l'aide du modèle}} \Big)$$

Dans la terminologie de l'apprentissage statistique classique, le terme (2) joue le rôle de la *fonction d'erreur*, et le terme (1) joue le rôle du *terme de régularisation*. Si (1) était minimisé isolément, on obtiendrait un modèle trivial inadapté aux données, et si (2) était minimisé isolément, les données seraient surapprises et le modèle serait incapable de généralisation. Chercher à minimiser la somme de ces deux termes revient donc à trouver un compromis entre simplicité du modèle et adaptation aux données.

Nous présentons deux méthodes d'apprentissage non-supervisé de la morphologie utilisant le principe MDL, et qui ont toutes les deux obtenu de très bons résultats. Contrairement aux méthodes présentées à la section 1.1, nous verrons que ces méthodes ont en commun de se présenter sous deux aspects indépendants : d'une part, la spécification d'une classe de modèles, d'autre part, un ensemble d'heuristiques d'initialisation et de recherche afin de trouver un modèle optimal.

1.2.2 *Linguistica*

1.2.2.1 Modèles de racines, suffixes et signatures et longueurs de description associées

Le système *Linguistica* de John Goldsmith [19] est un système d'apprentissage non-supervisé de morphologie (permettant un degré partiel de supervision). Le but de ce système est de segmenter un corpus de mots de manière à ce que chaque mot soit coupé en une racine et un suffixe, et de manière à ce que les

T (racines)	F (suffixes)	Σ (signatures)
cat dog hat John jump laugh sav the walk	NULL ed ing s e es	$\sigma_1 = \begin{Bmatrix} \texttt{ptr(cat)} \\ \texttt{ptr(dog)} \\ \texttt{ptr(hat)} \end{Bmatrix} \begin{Bmatrix} \texttt{ptr(NULL)} \\ \texttt{ptr(s)} \end{Bmatrix}$ $\sigma_2 = \{\texttt{ptr(sav)}\} \begin{Bmatrix} \texttt{ptr(e)} \\ \texttt{ptr(es)} \\ \texttt{ptr(ing)} \end{Bmatrix}$ $\sigma_3 = \begin{Bmatrix} \texttt{ptr(jump)} \\ \texttt{ptr(laugh)} \\ \texttt{ptr(walk)} \end{Bmatrix} \begin{Bmatrix} \texttt{ptr(NULL)} \\ \texttt{ptr(ed)} \\ \texttt{ptr(ing)} \\ \texttt{ptr(s)} \end{Bmatrix}$ $\sigma_4 = \begin{Bmatrix} \texttt{ptr(John)} \\ \texttt{ptr(the)} \end{Bmatrix} \{\texttt{ptr(NULL)}\}$

Paradigmes représentés
cat, cats
dog, dogs
hat, hats
save, saves, saving
jump, jumped, jumping, jumps
laugh, laughed, laughing, laughs
walk, walked, walking, walks
John
the

FIGURE 1.1 – Un exemple de modèle (T, F, Σ) et le lexique qu'il est capable de couvrir

ensembles de mots partageant une même racine puissent être regroupés en paradigmes. Dans ce système, un modèle est spécifié par la donnée de trois ensembles (T, F, Σ) :

— T est un ensemble de *racines*.

— F est un ensemble de *suffixes*.

— Σ est un ensemble de *signatures*. Une signature σ est la donnée d'un ensemble $\mathbf{t} \subset T$ de racines et d'un ensemble $\mathbf{f} \subset F$ de suffixes telles que les racines de $\mathbf{t}$ et les suffixes de $\mathbf{f}$ puissent être concaténés pour former des mots.

— Ceci avec la contrainte qu'une racine ne peut appartenir qu'à une seule signature.

La figure 1.1 représente un exemple d'un tel modèle. Pour se placer dans le cadre MDL, il convient aussi de définir la manière dont un modèle sera décrit afin de calculer sa longueur de description. La description suivante est adoptée :

— T est représenté par une liste de chaînes de caractères. Chaque chaîne de caractères t de longueur $|t|$ (sur un alphabet à 26 lettres) nécessite $|t| \log 26$

bits pour être représentée. De plus, il faut stocker la longueur de la liste, ce qui nécessite $\lambda(|T|)$ bits (où $\lambda(k)$ est une fonction représentant le nombre de bits requis pour coder l'entier k). La longueur de description de T est donc : $L(T) = \lambda(|T|) + \sum_{t \in T} |t| \log 26$.

— De même pour S : $L(S) = \lambda(|S|) + \sum_{s \in S} |s| \log 26$.

— Une signature est représentée par deux listes : une liste de pointeurs vers des racines, et une liste de pointeurs vers des suffixes. En considérant que les racines sont issues d'une distribution de probabilité $freq(t)$ donnée par leurs fréquences dans le corpus, on peut donc coder un pointeur vers t avec $-\log freq(t)$ bits (en utilisant par exemple un code de Huffman). Il en est de même pour les suffixes. Ainsi, la longueur de description d'une signature $\sigma = (\mathbf{t}, \mathbf{f})$ vaut :

$$\begin{aligned} L(\sigma) = & \lambda(|\mathbf{t}|) && \text{(codage de la taille de } \mathbf{t}) \\ & + \lambda(|\mathbf{f}|) && \text{(codage de la taille de } \mathbf{f}) \\ & + \sum_{t \in \mathbf{t}} -\log freq(t) && \text{(codage des pointeurs vers les } t \in \mathbf{t}) \\ & + \sum_{f \in \mathbf{f}} -\log freq(f) && \text{(codage des pointeurs vers les } f \in \mathbf{f}) \end{aligned}$$

et la longueur de description de l'ensemble Σ des signatures vaut donc :

$$\begin{aligned} L(\Sigma) = & \lambda(|\Sigma|) && \text{(codage de la taille de} \Sigma) \\ & + \sum_{\sigma \in \Sigma} L(\sigma) && \text{(codage de chaque signature)} \end{aligned}$$

— La longueur de description totale d'un modèle $M = (T, F, \Sigma)$ est donc : $L(M) = L(T) + L(F) + L(\Sigma)$.

La dernière étape nécessaire pour se placer dans le cadre MDL est de définir les longueurs de description des données à l'aide d'un modèle. Les données sont une liste de mots $\mathbf{w} = (w_1, \ldots, w_{|W|})$. Pour décrire un mot w de racine t et de suffixe f à l'aide d'un modèle $M = (T, F, \Sigma)$, il faut :

— Spécifier la signature $\sigma = (\mathbf{t}, \mathbf{f})$ auquel le mot appartient : ceci peut se coder sur $-\log freq(\sigma)$ bits.

— Spécifier, parmi les racines de $\mathbf{t}$, laquelle est la racine de w. Ceci peut se coder sur $-\log freq(t|\sigma)$.

— Spécifier, parmi les suffixes de $\mathbf{f}$, lequel est le suffixe de w. Ceci peut se coder sur $-\log freq(f|\sigma)$.

— La longueur de description $L(w|M)$ du mot $w = t + f$ à l'aide du modèle M est donc :

$$L(w|M) = \underbrace{-\log freq(\sigma)}_{\text{pointeur vers la signature}} \quad \underbrace{-\log freq(t|\sigma)}_{\text{pointeur vers la racine}} \quad \underbrace{-\log freq(f|\sigma)}_{\text{pointeur vers le suffixe}}$$

La longueur de description de l'ensemble du corpus à l'aide d'un modèle M est donc :

$$L(\mathbf{w}|M) = \lambda(|\mathbf{w}|) + \sum_{w \in \mathbf{w}} L(w|M)$$

Ce qui nous donne enfin la longueur de description totale des données, qui est la fonction qu'on cherchera à minimiser au cours de l'apprentissage :

$$M_{\text{optimal}} = \underset{M}{\text{argmin}}\ L(M) + L(\mathbf{w}|M)$$

Signalons que pour ne pas alourdir notre présentation, nous avons omis le fait que Linguistica autorise également une structure récursive dans les signatures, c'est-à-dire que l'ensemble des racines (resp. des suffixes) d'une signature peut comprendre à la fois :

— des pointeurs vers des racines (resp. suffixes) "simples".

— des pointeurs vers *d'autres signatures* pour représenter des racines (resp. suffixes) "complexes", et permettre des analyses du type : $\underbrace{\underbrace{\underbrace{\texttt{sav}}_{t'}\ \underbrace{\texttt{ing}}_{f'}}_{t=\sigma'}\ \underbrace{\texttt{s}}_{f}}_{\sigma}$

1.2.2.2 Recherche d'un modèle optimal

Nous avons décrit la structure des modèles, et défini leur longueur de description. A présent, il s'agit de trouver un modèle de longueur de description minimale. Pour cela, *Linguistica* utilise un algorithme de recherche locale qui se déroule de la manière suivante :

1. Initialiser un modèle M_0 à l'aide d'heuristiques.
2. Etant donné un modèle M_k à la k-ème itération de la recherche, appliquer différentes heuristiques à M_k de manière à obtenir plusieurs "modifications" de ce modèle. Pour chacune de ces modifications, sélectionner celle de longueur de description minimale : M_{k+1}. Si $L(M_{k+1}) < L(M_k)$, réitérer la recherche sur M_{k+1}. Sinon, cela signifie qu'on a atteint en M_k un minimum local pour la longueur de description, et la recherche est terminée.

Heuristiques d'initialisation Une heuristique d'initialisation proposée utilise l'algorithme EM :

1. Au début, chaque mot est segmenté aléatoirement en une racine et un suffixe (par exemple en le coupant en un point choisi uniformément sur la longueur du mot).

2. Ensuite, pour chaque mot w, tous les points de coupure possibles $i \in \{1, \ldots, |w|\}$ sont évalués avec une probabilité définie par :

$$\mathbb{P}(w = w_0^i + w_i^l) = \frac{1}{Z} \exp\Big(- \big(i \log freq(\text{racine} = w_0^i) + (l-i) \log freq(\text{suffixe} = w_i^l)\big)\Big)$$

où :

— l est la longueur du mot

— Z est une constante de normalisation choisie de sorte à bien obtenir : $\sum_{i=1}^{l} \mathbb{P}(w = w_0^i + w_i^l) = 1$

— $freq(\text{racine} = w_0^i)$ est la fréquence d'apparition de w_0^i comme racine parmi les segmentations courantes. De même pour $freq(\text{suffixe} = w_i^l)$.

3. Ensuite, les points de coupure de chaque mot sont rééchantillonnés selon les distributions $\mathbb{P}(w = w_0^i + w_i^l)$, puis on retourne à l'étape 2.

4. ...et ainsi de suite jusqu'à convergence du processus.

On obtient de la sorte un ensemble initial de racines (T_0) et de suffixes (F_0). Reste à initialiser l'ensemble des signatures Σ_0 pour achever l'initialisation du modèle. Cet ensemble est initialisé en calculant tous les couples maximaux formés d'un ensemble de racines ($\mathbf{t}$) et d'un ensemble de suffixes ($\mathbf{f}$) tels que l'ensemble des chaînes obtenues en concaténant une racine $t \in \mathbf{t}$ et un suffixe $f \in \mathbf{f}$ soient présentes dans le corpus. Chaque couple maximal $(\mathbf{t}, \mathbf{f})$ de la sorte constitue une signature.

Les ensembles T_0, F_0, Σ_0 ainsi obtenus forment le modèle initial, qu'on cherche par la suite à raffiner par un algorithme de recherche locale.

Heuristiques de recherche locale L'optimisation du modèle s'effectue en partant du modèle initial, en lui appliquant différentes modifications, en retenant la modification de longueur de description minimale, puis en réitérant l'optimisation sur la modification retenue, jusqu'à convergence vers un minimum local de la longueur de description minimale. Brièvement, les modifications envisagées à chaque étape sur un modèle sont :

— Pour chaque suffixe, essayer toutes les coupures en deux possibles de ce suffixe. (Par exemple, essayer de remplacer le suffixe "simple" `ings` (en anglais) par un suffixe "complexe" représenté par une signature $\sigma' = \underbrace{\{\texttt{ing}\}}_{\mathbf{t}'} \underbrace{\{\texttt{s}\}}_{\mathbf{f}'}$).

— Si tous les suffixes d'une signature commencent par le même caractère,

essayer de “déplacer” ce caractère vers les stems. Par exemple :

$$\begin{Bmatrix}\texttt{oscilla}\\ \texttt{vibra}\\ \texttt{sta}\\ \texttt{crea}\end{Bmatrix}\begin{Bmatrix}\texttt{te}\\ \texttt{tes}\\ \texttt{ting}\\ \texttt{tion}\end{Bmatrix} \xrightarrow[\text{modification}]{} \begin{Bmatrix}\texttt{oscillat}\\ \texttt{vibrat}\\ \texttt{stat}\\ \texttt{creat}\end{Bmatrix}\begin{Bmatrix}\texttt{e}\\ \texttt{es}\\ \texttt{ing}\\ \texttt{ion}\end{Bmatrix}$$

— Lorsqu'une signature contient un faible nombre de racines ou un faible nombre de suffixes, essayer de la supprimer et considérer tous les mots qu'elle engendre comme des racines munies du suffixe `NULL`. Par exemple :

$$\begin{Bmatrix}\texttt{lo}\\ \texttt{bo}\end{Bmatrix}\begin{Bmatrix}\texttt{ok}\\ \texttt{ot}\end{Bmatrix} \xrightarrow[\text{modification}]{} \begin{Bmatrix}\texttt{look}\\ \texttt{loot}\\ \texttt{book}\\ \texttt{boot}\\ \dots\end{Bmatrix}\{\texttt{NULL}\}$$

1.2.3 *Morfessor*

Nous décrivons à présent *Morfessor*, une famille de méthodes de segmentation morphologique non-supervisée déclinées en plusieurs variantes dans [12], [13], [14], [15], et qui est à présent utilisée dans de nombreux systèmes de traitement des langues. *Linguistica* décrivait la morphologie d'une langue en termes de signatures représentant des paradigmes, en imposant des analyses où chaque mot est découpé en une racine et un suffixe. *Morfessor* adopte une description différente : un mot est considéré comme une séquence de morphèmes (éventuellement affectés à des classes), et il n'y a pas de limite au nombre de morphèmes possibles constituant un mot. Cette méthode est donc particulièrement adaptée aux langues *agglutinantes*, où de nombreux mots sont constitués de longues séquences de morphèmes dérivationnels, et aux langues permettant des *mots composés* où plusieurs racines peuvent intervenir dans le même mot.

1.2.3.1 Version *Baseline* (2002)

Modèle La version initiale de *Morfessor* [12] se situe dans le cadre MDL, et de même que *Linguistica*, consiste à minimiser la somme des longueurs de description d'un modèle et des données à partir de celui-ci. Le modèle est un dictionnaire de morphèmes, et les données sont décrites à l'aide de pointeurs vers les entrées de ce dictionnaire. L'objectif de minimisation de la longueur de description favorise le choix de segments qui, étant fréquemment réutilisés, permettent une économie de description. Plus précisément :

— Le modèle M est un dictionnaire de morphèmes $(w_1, \dots, w_m)$. Chaque morphème est encodé comme une chaîne de caractères suivie d'un caractère séparateur. Chaque caractère nécessite $\log 27$ bits pour être encodé (26 lettres de l'alphabet + 1 caractère séparateurs). Donc en tout, la longueur de description de M vaut : $(\log 27) \cdot \sum_i 1 + |w_i|$ bits.

— Les données sont une liste de mots. Chaque mot est décrit à l'aide d'une suite de pointeurs vers des entrées du dictionnaire de morphèmes. Par exemple, `morphologiquement` pourra être décrit comme : `ptr(morpho) ptr(logi) ptr(que) ptr(ment)`. Si $freq(\mu)$ est la fréquence d'apparition d'un morphème μ parmi l'ensemble des segmentations, un pointeur `ptr(`μ`)` pourra être codé sur $-\log freq(\mu)$ bits. Ainsi la description des données à l'aide d'un modèle nécessite $\sum_\mu -\log freq(\mu)$, en sommant sur l'ensemble des occurrences de morphèmes dans les données segmentées.

— La longueur de description totale à minimiser est donc :

$$L = (\log 27) \cdot \left(\sum_i 1 + |w_i| \right) + \sum_\mu -\log freq(\mu)$$

Recherche La recherche d'un minimum de la longueur de description s'effectue en re-segmentant chaque mot tour à tour jusqu'à convergence. Pour segmenter un mot w :

1. On calcule la longueur de description L_0 dans le cas où le mot n'est pas segmenté. On calcule aussi, pour $i = 1, \ldots, |w|-1$, la longueur de description L_i dans le cas où le mot est coupé à la position i. Parmi toutes ces options, on choisit celle qui donne la longueur de description minimale. (NB : si une des options nécessite de segmenter un morphème qui n'est pas encore dans le dictionnaire, il faut donc tenir compte de l'ajout de ce morphème dans le dictionnaire en calculant la longueur de description).
2. Si l'option retenue est de ne pas segmenter le mot, on passe au mot suivant. Sinon, si on a coupé le mot à une position i on réapplique la même procédure sur chacun des deux segments obtenus (c'est-à-dire w_0^i et $w_i^{|w|}$).

1.2.3.2 *Categories-ML* (2004)

La deuxième version de *Morfessor* [13] sort du cadre MDL, et consiste en un objectif de maximisation de vraisemblance d'un modèle probabiliste génératif. Un désavantage majeur du critère de maximum de vraisemblance est que celui-ci mène à des modèles trop fidèles aux données pour pouvoir se généraliser à des cas nouveaux : ici cet écueil est évité en recourant à un ensemble d'heuristiques. Par ailleurs, les morphèmes sont regroupés en catégories, et des règles de succession entre ces catégories sont prises en compte (c'est-à-dire une *morphotactique* qui est essentiellement une *syntaxe* à l'échelle du mot).

Modèle Le modèle sous-jacent est un modèle probabiliste *génératif*, c'est-à-dire un modèle expliquant, pour chaque mot des données, sa formation comme résultat d'un scénario faisant intervenir des évènements aléatoires. Il s'agit ici d'un *modèle de Markov caché*, c'est-à-dire que la formation d'un mot est expliquée comme :

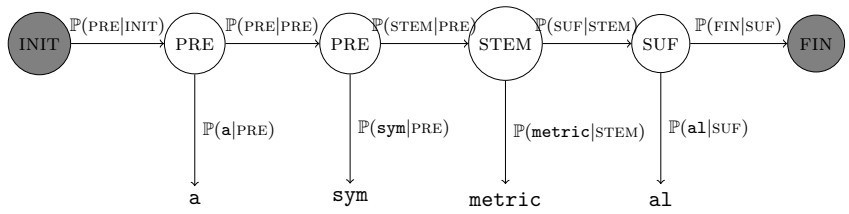


FIGURE 1.2 – Exemple de scénario génératif pour le modèle HMM de *Categories-ML*. Chaque flèche représente une étape générative du scénario, et la probabilité totale de ce scénario est le produit des probabilités indiquées sur chaque flèche.

1. La génération d'une séquence aléatoire de catégories $C_0, C_1, \ldots, C_{k+1}$, où C_0 est une catégorie "initiale", C_{k+1} est une catégorie "terminale" signalant la fin de la séquence, et C_i $(i = 1 \ldots k)$ une catégorie morphologique parmi : PRE (préfixe), SUF (suffixe), STEM (racine), NOISE ("bruit"). La séquence est une chaîne de Markov, c'est-à-dire que la probabilité d'une catégorie C_i en i-ème position est entièrement déterminée par la catégorie précédente C_{i-1} selon une loi conditionnelle $\mathbb{P}(C_i|C_{i-1})$.
2. Pour chaque catégorie C_i $(i = 1 \ldots k)$ de la séquence, la génération d'un morphème aléatoire μ_i selon une loi conditionnée sur C_i : $\mathbb{P}(\mu_i|C_i)$.
3. Enfin le mot obtenu est la concaténation des morphèmes générés : $w = \mu_1 \ldots \mu_k$.

De plus, les probabilités de transitions entre catégories sont contraintes de sorte à rester conformes à l'expression régulière suivante : $(\text{PRE}^* \text{ STEM SUF}^*)^+$. Ainsi par exemple, un préfixe ne pourra pas être immédiatement suivi par un suffixe, ce qui se traduit par la contrainte $\mathbb{P}(\text{SUF}|\text{PRE}) = 0$. Les probabilités de transition qui ne contredisent pas cette expression régulière sont considérées comme des paramètres du modèle, et sont évaluées durant l'inférence.

Inférence Les étapes de l'inférence sont :

1. Initialiser une segmentation du corpus à l'aide de la version *Baseline*.
2. Assigner leurs catégories aux morphèmes de ces segmentations initiales à l'aide d'heuristiques : essentiellement, des scores $p(\text{PRE}|\mu)$ (resp. $p(\text{STEM}|\mu)$, $p(\text{SUF}|\mu)$) agrégeant différentes heuristiques quantifient à quel point un morphème μ "ressemble à un préfixe" (resp. à une racine, à un suffixe). Par exemple, une heuristique utilisée pour déterminer la "ressemblance à un préfixe" est de regarder l'entropie des successeurs de μ sur le corpus : si en moyenne, à chaque fois que μ apparaît, le caractère le suivant immédiatement est difficile à prédire, on assignera un score élevé à $p(\text{PRE}|\mu)$. Autre exemple : le score de "ressemble à une racine" sera d'autant plus

élevé que μ est long. Enfin, on définit $p(\text{NOISE}|\mu) = (1 - p(\text{PRE}|\mu))(1 - p(\text{STEM}|\mu))(1 - p(\text{SUF}|\mu))$.

3. La donnée d'un ensemble initial de segmentations (et d'affectations de catégories) permet de calculer un modèle initial, c'est-à-dire, les probabilités de transition entre catégories $\mathbb{P}(C'|C)$ et d'émission de morphèmes $\mathbb{P}(\mu|C)$.
4. Ce modèle initial est ensuite optimisé avec l'algorithme EM (en gardant les points de coupure fixés). Autrement dit, on alterne entre :
   — Rééchantillonner les affectations de catégories à l'aide des distributions $\mathbb{P}(C'|C)$ et $\mathbb{P}(\mu|C)$.
   — Recalculer les distributions $\mathbb{P}(C'|C)$ et $\mathbb{P}(\mu|C)$ à l'aide des nouvelles catégories échantillonnées.
   — ...et ainsi de suite jusqu'à convergence.
5. Puis une batterie d'heuristiques est appliquée pour raffiner les segmentations et les catégories. Elles consistent essentiellement à :
   — Trouver les morphèmes μ qui peuvent s'écrire comme concaténation de deux morphèmes préexistants μ_1 et μ_2, et les scinder en ces deux morphèmes (si cela ne viole pas les contraintes morphotactiques ou n'entraîne pas l'apparition de morphèmes NOISE).
   — Essayer de supprimer les morphèmes NOISE en les fusionnant avec leurs voisins.
6. A nouveau optimiser les paramètres $\mathbb{P}(C'|C)$ et $\mathbb{P}(\mu|C)$ avec EM (en gardant les points de coupure fixés). Puis rééchantillonner toutes les segmentations et catégories (algorithme de Viterbi) puis revenir à l'étape 4.

1.2.3.3 *Categories-MAP* (2005)

La troisième version [14] reprend la version *Categories-ML* en ajoutant un *a priori* sur les modèles :

— Dans *Categories-ML*, le but est de trouver un modèle de vraisemblance maximale, c'est-à-dire de résoudre :

$$M^* = \underset{M \in \text{modèles}}{\text{argmax}} \ \mathbb{P}(\text{données}|M)$$

en contraignant les modèles considérés par certaines heuristiques.

— Dans *Categories-MAP*, on considère d'une part la même probabilité $\mathbb{P}(\text{données}|M)$ que ci-dessus, mais de plus un *a priori* probabiliste $\mathbb{P}(M)$ remplace les contraintes heuristiques sur les modèles. L'objectif est donc de maximiser la distribution jointe sur les données *et* les modèles ainsi définie :

$$M^* = \underset{M \in \text{modèles}}{\text{argmax}} \ \mathbb{P}(\text{données}, M) = \underset{M \in \text{modèles}}{\text{argmax}} \ \mathbb{P}(\text{données}|M)\mathbb{P}(M)$$

L'a priori est défini comme :

$$\mathbb{P}(M) \propto \prod_{\mu \in \text{types de morphèmes}} \mathbb{P}(\text{meaning}(\mu)) \cdot \mathbb{P}(\text{form}(\mu))$$

Ce que les auteurs appellent $\mathbb{P}(\text{meaning}(\mu))$ est un score heuristique basé sur des statistiques d'utilisation de μ comme sa fréquence, sa longueur, ou la perplexité quand au morphème qui le suit ou qui le précède. Le facteur $\mathbb{P}(\text{form}(\mu))$ est calculé en considérant deux cas : soit μ est représenté comme concaténation de deux autres morphèmes présents dans le lexique, soit simplement comme une chaîne de caractères. Cet *a priori* introduit donc une dimension hiérarchique dans le lexique de morphèmes inféré, et dans les analyses morphologiques qui en résultent.

— **Cas 1 (représentation hiérarchique de μ) :** Si on choisit de représenter μ comme concaténation de deux autres types de morphèmes μ_1 et μ_2 existant dans le lexique, et de catégories respectives C_1 et C_2 alors

$$\mathbb{P}(\text{form}(\mu)) = \mathbb{P}(\text{cas 1})\mathbb{P}(C_1|\text{cas 1})\mathbb{P}(\mu_1|C_1)\mathbb{P}(C_2|C_1)\mathbb{P}(\mu_2|C_2)$$

  — $\mathbb{P}(\text{cas 1})$ est la probabilité empirique qu'un morphème soit représenté hérarchiquement, autrement la proportion de morphèmes dans le lexique représentés comme concaténation de deux autres morphèmes.

  — $\mathbb{P}(C_1|\text{cas 1})$ est la proportion, parmi les morphèmes représentés hiérarchiquement comme $\mu = \mu_1\mu_2$, de ceux pour lesquels μ_1 a reçu la catégorie C_1.

— **Cas 2 (cas de base) :** μ est considéré simplement comme une chaîne de caractères $c_1 \dots c_n$ sans sous-structure, et

$$\mathbb{P}(\text{form}(\mu)) = (1 - \mathbb{P}(\text{cas 1})) \prod_{i=1}^{n} \mathbb{P}(c_i)$$

où le facteur $\prod_{i=1}^{n} \mathbb{P}(c_i)$ joue le rôle d'un modèle unigramme de caractères.

L'ensemble des facteurs $\mathbb{P}(\text{form}(\mu))$ de l'a priori forment un équivalent probabiliste de la longueur de description du modèle. Par exemple, la quantité $-\log \mathbb{P}(\text{form}(\mu))$ n'est rien d'autre que la longueur de description de μ

— à l'aide de pointeurs vers des catégories C_1, C_2 et morphèmes μ_1, μ_2 dans le premier cas.

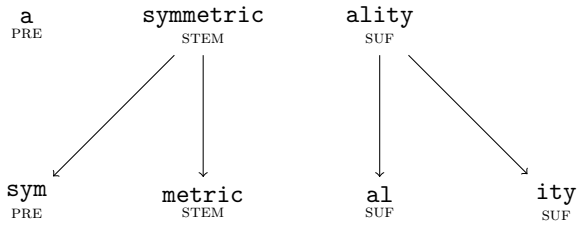
— en le codant comme une chaîne de caractères dans le second cas.

De par la manière dont l'a priori est spécifié, il se peut donc par exemple que le mot `asymmetricality` soit analysé en :

$$\underset{\text{PRE}}{\texttt{a}} \cdot \underset{\text{STEM}}{\texttt{symmetric}} \cdot \underset{\text{SUF}}{\texttt{ality}}$$

mais que par ailleurs le morphème `symmetric` tombe sous les **cas 1** et que dans le lexique de morphèmes inféré, les morphèmes $\underset{\text{PRE}}{\texttt{sym}}$ et $\underset{\text{STEM}}{\texttt{metric}}$ soient aussi

présents (et de même pour al SUF et ity SUF), auquel cas on pourrait compléter l'analyse par la représentation hiérarchique suivante :



1.2.3.4 Versions ultérieures

Plusieurs versions ultérieures de *Morfessor* ont été proposées, que nous ne décrirons pas toutes ici.

Par exemple, *Morfessor FlatCat* (2014) [25] est une version utilisant un *modèle de Markov caché* similaire à *Categories-MAP*, mais où la possibilité de représenter hiérarchiquement le lexique de morphèmes à été supprimée, dans le but de faciliter l'apprentissage semi-supervisé. Selon les auteurs, l'absence de hiérarchie facilite l'apprentissage semi-supervisé car étant donné un corpus annoté en segmentations et catégories de morphèmes, il est facile de trouver les paramètres de vraisemblance maximale : ces paramètres sont les probabilités de transition entre catégories $\mathbb{P}(C_i|C_{i-1})$ et d'émission de morphèmes $\mathbb{P}(\mu|C)$, et il suffit pour maximiser leur vraisemblance de compter des fréquences d'évènements sur le corpus. En revanche, il n'y a pas de moyen aussi évident de choisir quand représenter ou non un morphème de manière hiérarchique (**cas 1** versus **cas 2** ci-dessus) de sorte à maximiser la vraisemblance du modèle, ce qui rend l'aspect hiérarchique difficile à implémenter en supervision.

Allomorfessor (2008) [37] est une extension de *Morfessor* ajoutant une modélisation des phénomènes d'*allomorphie*, c'est-à-dire du fait qu'un même morphème "abstrait" puisse se réaliser sous différentes formes concrètes selon le contexte phonologique ou morphologique où il apparaît : par exemple, en allemand `Wald` (singulier) → `Wälder` (pluriel) ou `Maus` (singulier) → `Mäuser` (pluriel). Dans ces exemples, les racines `Wald` et `Maus` se manifestent sous des formes différentes selon qu'elles sont modifiées ou non par le suffixe "pluriel" `-er`. *Allomorfessor* prend en compte ces phénomènes en autorisant dans son modèle génératif des *mutations* comme le fait de remplacer une lettre par une autre, ou de supprimer une lettre.

1.3 Méthodes bayésiennes non-paramétriques

La troisième famille de méthodes que nous allons évoquer, et dans laquelle s'inscrivent les travaux de cette thèse, sont les méthodes bayésiennes non-paramétriques (BNP). Elles s'appuient toutes sur un modèle probabiliste génératif des données, paramétré par un certain vecteur (potentiellement infini) de paramètres. De

plus, les paramètres sont eux-mêmes soumis à un *a priori* probabiliste. Le modèle génératif prend donc la forme d'une distribution jointe sur les observations générées *et* sur les paramètres :

$$\mathbb{P}(\mathbf{x}, \theta) = \underbrace{\mathbb{P}(\mathbf{x}|\theta)}_{\text{modèle génératif des données}} \cdot \underbrace{\mathbb{P}(\theta)}_{\textit{a priori} \text{ sur les paramètres}}$$

$\mathbf{x}$: observation $\quad \theta$: vecteur de paramètres

On peut alors considérer la distribution *a posteriori* $\mathbb{P}(\theta|\mathcal{X})$ sur les paramètres, qui étant donné un jeu de données $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, détermine la probabilité que ce soit le jeu de paramètres θ qui ait engendré ce jeu de données. Cette distribution peut s'exprimer par la règle de Bayes :

$$\mathbb{P}(\theta|\mathcal{X}) = \frac{\mathbb{P}(\mathcal{X}|\theta) \cdot \mathbb{P}(\theta)}{\int_\theta \mathbb{P}(\mathcal{X}|\theta) \cdot \mathbb{P}(\theta)} = \frac{\prod_{i=1}^n \mathbb{P}(\mathbf{x}_i|\theta) \cdot \mathbb{P}(\theta)}{\int_\theta \prod_{i=1}^n \mathbb{P}(\mathbf{x}_i|\theta) \cdot \mathbb{P}(\theta)}$$

Le cadre est donc similaire à l'*estimation maximum a posteriori*, qui est par exemple le cadre employé dans *Morfessor Categories-MAP* (section 1.2.3.3). Toutefois, à la différence de celle-ci, on ne cherchera pas à trouver un jeu de paramètres θ maximisant la probabilité *a posteriori*. Le modèle prescrit par l'inférence bayésienne sera celui obtenu en effectuant une moyenne de tous les modèles possibles $\{\mathbb{P}(\mathbf{x}|\theta)\}_\theta$ en les pondérant par leur probabilité *a posteriori*. Autrement dit, le modèle qu'on obtiendra ainsi sera celui qui assigne à une observation $\mathbf{x}$ la probabilité :

$$\mathbb{P}(\mathbf{x}|\mathcal{X}) = \int_\theta \mathbb{P}(\mathbf{x}|\theta) \cdot \mathbb{P}(\theta|\mathcal{X})$$

L'enjeu est donc de pouvoir *marginaliser* le vecteur de paramètres θ. Dans le cas général, cela nécessiterait de recourir à des méthodes d'échantillonnage approximatif, ce qui serait particulièrement difficile lorsque la dimension du vecteur de paramètres à échantillonner est grande, voire infinie. Or, pour certaines familles particulières d'*a priori* (appelés *processus de Dirichlet* ainsi que leur généralisation, les *processus de Pitman-Yor*), il est possible de marginaliser θ sans avoir à calculer l'intégrale même de manière approchée. De plus, il se trouve que ces *a priori* contraignent les vecteurs de paramètres de sorte que les distributions de probabilités qui en découlent suivent une *loi de Zipf*, ce qui les rend particulièrement adéquates pour modéliser des phénomènes linguistiques.

Les méthodes BNP se rapprochent des approches par longueur de description minimale (MDL) en ce sens que toutes deux tendront à un compromis entre la simplicité du modèle et sa fidélité aux données. Pour les méthodes BNP, ces deux exigences ne sont pas quantifiées en termes de longueur de description mais de probabilités (ce qui, quitte à considérer l'entropie des distributions en jeu, revient au même) : l'*a priori* $\mathbb{P}(\theta)$ favorisera des modèles simples, et la probablité conditionnelle $\mathbb{P}(\mathcal{X}|\theta)$ favorisera des modèles fidèles aux données. Toutefois, à la différence des méthodes MDL, le but ici n'est pas de trouver un modèle

optimisant un critère quantitatif, mais de moyenner un ensemble de modèles selon une distribution *a posteriori*. De plus, les méthodes BNP introduisent une exigence supplémentaire qui intéresse particulièrement le traitement des langues, à savoir le fait que les distributions intervenant dans le modèle génératif devront suivre une loi de Zipf.

1.3.1 Modèles n-grammes

Dans sa thèse [24], Sharon Goldwater cherche à rendre compte de l'acquisition du vocabulaire dans l'enfance à partir d'un flux de phonèmes en termes d'apprentissage bayésien, et se propose d'aborder le problème de la *tokénisation* (découpage de mots dans des chaînes de phonèmes) à l'aide de méthodes bayésiennes non-paramétriques.

1.3.1.1 Modèle unigramme

Sa première méthode s'appuie sur un modèle *unigramme* de mots, c'est-à-dire un modèle probabiliste génératif expliquant la formation d'une chaîne de phonèmes s comme la concaténation d'unités $s = w_1 w_2 \dots w_n$, où chaque w_i est un échantillon d'une distribution catégorique $\mathbb{P}(w|\theta)$. Le vecteur de paramètres θ est un vecteur de dimension infinie attribuant une probabilité à chaque chaîne de phonèmes possible, et est soumis à un *a priori* appelé *Processus de Dirichlet* et noté $DP(\alpha, G_0)$, où :

— α est un nombre réel strictement positif.

— G_0 est une distribution sur les chaînes de phonèmes appelée *distribution de base*. Cette distribution est "agnostique" : il n'est pas nécessaire qu'elle soit adaptée d'une quelconque manière au problème ou à la langue à traiter. Par exemple, la distribution de base choisie dans [24] est un modèle unigramme de phonèmes, la probabilité pour chaque phonème étant donnée par sa fréquence d'occurence dans le corpus.

— Ces deux données constituent les *hyperparamètres* du processus.

Les bases théoriques de ces objets seront présentées en détail dans la suite de cette thèse, et nous verrons notamment qu'à aucun moment il n'est nécessaire d'avoir une représentation explicite du vecteur de paramètres infini θ car il n'interviendra qu'en tant que variable marginalisée. La théorie montre que, sachant qu'un ensemble de mots $\mathcal{X} = \{w_1, w_2, \dots, w_n\}$ a été engendré par le passé, la probabilité que le scénario génératif considéré engendre w vaut :

$$\mathbb{P}(w|\mathcal{X}) = \int_\theta \mathbb{P}(w|\theta) \cdot \mathbb{P}(\theta|\mathcal{X})$$
$$= \frac{\overbrace{\left(\sum_{i=1}^{n} \mathbf{1}(w_i = w)\right)}^{\text{nombre de } w_i \text{ égaux à } w} + \alpha G_0(w)}{n + \alpha}$$

Pour segmenter un ensemble de chaînes de phonèmes $\mathbf{s} = \{s_1, \ldots, s_m\}$, la stratégie adoptée par Sharon Goldwater est le *Gibbs sampling*, c'est-à-dire :

1. Partir d'un ensemble aléatoire de segmentations.
2. Pour chaque $i = 1 \ldots m$, re-segmenter s_i à l'aide de toutes les autres segmentations (de $(s_j)_{j \neq i}$), c'est-à-dire en utilisant la distribution $\mathbb{P}(w|(s_j)_{j\neq i})$ (telle que donnée dans l'expression **??** ci-dessus) comme probabilité d'émission d'un mot.
3. et répéter ceci jusqu'à convergence.

Les segmentations ainsi obtenues représentent, pour chaque chaîne de phonèmes s, la suite de mots $s = w_1 w_2 \ldots w_n$ expliquant sa formation. Chacun de ces mots à été engendré par une distribution $\mathbb{P}(w|\text{données}) = \int_\theta \mathbb{P}(w, \theta|\text{données}) d\theta$ où θ a été marginalisé. Le lexique issu de ces segmentations suit une loi de Zipf : cela garantit que sa distribution ressemblera aux distributions typiquement rencontrées dans les langues naturelles.

1.3.1.2 Modèles d'ordre supérieur

Sharon Goldwater [24] propose par la suite d'étendre ces idées à un modèle bi-gramme. Elle procède par un système de "repli" (*back-off*), qui consiste à remplacer la distribution de base G_0 d'un processus de Dirichlet par une distribution aléatoire qui est elle-même engendrée par un processus de Dirichlet.

Plus précisément, il s'agit de considérer :

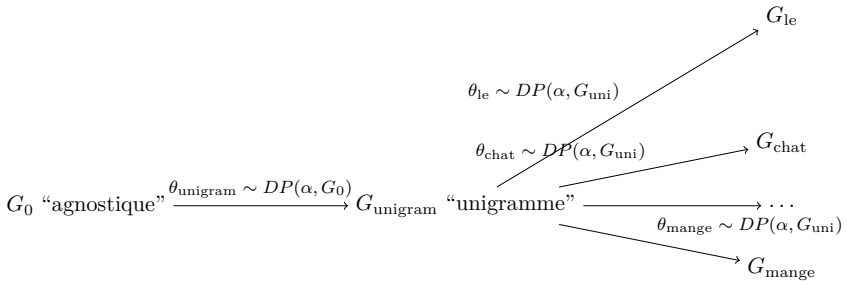
— Une distribution de base G_0 telle qu'à la section précédente.

— Une distribution G_{unigram} paramétrée par un vecteur de paramètres $\theta_{\text{unigram}} \sim DP(\alpha, G_0)$.

— Pour chaque chaîne de phonèmes w, une distribution G_w paramétrée par un vecteur de paramètres $\theta_w \sim DP(\alpha, G_{\text{unigram}})$. Elle représentera la distribution de mots succédant au mot w.

L'ensemble des paramètres est donc donné par un vecteur $\theta = \big(\theta_{\text{unigram}}, (\theta_w)_w\big)$, et étant donné un tel vecteur, les probabilités $\mathbb{P}(w_i|w_{i-1};\theta)$ du modèle bigramme (probabilité d'engendrer le mot w_i à la suite du mot w_{i-1}) sont alors données par :

$$\mathbb{P}(w_i|w_{i-1};\theta) = G_{w_{i-1}}(w_i)$$



$$\mathbb{P}(\text{chat}|\text{le}, \mathcal{S}) = \hat{G}_{\text{le}}(\text{chat}) = \frac{\#\{(\text{le}, \text{chat}) \in \mathcal{S}\} + \alpha \hat{G}_{\text{unigram}}(\text{chat})}{\#\{(\text{le}) \in \mathcal{S}\} + \alpha}$$

$$\hat{G}_{\text{unigram}}(\text{chat}) = \frac{\#\{(\text{chat}) \in \mathcal{S}\} + \alpha G_0(\text{chat})}{\#\{\text{nombre de mots découpés dans } \mathcal{S}\} + \alpha}$$

FIGURE 1.3 – Système de *back-off* pour un modèle bi-gramme de mots dont les distributions sont issues d'un processus de Dirichlet, et probabilités associées conditionnellement à un ensemble $\mathcal{S}$ de segmentations

Toutefois, de même que pour le modèle unigramme, on ne considérera jamais explicitement le vecteur de paramètres θ, celui-ci sera toujours marginalisé, conditionnellement à un ensemble de segmentations $\mathcal{S}$. Ainsi plutôt que de considérer les distributions G_{unigram} et G_w (pour chaque chaîne de phonèmes w) qui dépendent de θ, on considérera les moyennes $\hat{G}_{\text{unigram}}$ et $\hat{G}_w$ de ces distributions lorsque θ sera marginalisé selon sa distribution *a posteriori* :

$$\hat{G}_{\text{unigram}} = \int_\theta G_{\text{unigram}} \cdot \mathbb{P}(\theta|\mathcal{S}) d\theta$$

$$\forall \text{ chaîne de phonèmes } w, \hat{G}_w = \int_\theta G_{\text{w}} \cdot \mathbb{P}(\theta|\mathcal{S}) d\theta$$

La théorie des processus de Dirichlet que nous présenterons au cours de cette thèse nous permet d'avoir une expression explicite de $\hat{G}_{\text{unigram}}$ et de $\hat{G}_w$, donnée dans la figure 1.3.

L'article [42] étend l'idée d'un système de *back-off* en généralisant la méthode précédente dans deux directions :

— L'ordre n du modèle n-gramme est quelconque. Ainsi, si $1 \leq k \leq n$, pour tout $(k-1)$-uplet de mots $(w_{-1}, \ldots, w_{-k+1})$, une distribution $G_{w_{-k+1},\ldots,w_{-1}}$ représentant la distribution de mots succédant à ce $(k-1)$-uplet est considérée, et ses paramètres sont engendrés par un processus de Dirichlet de

$$G_{w_{-n+1}\dots w_{-1}} \longleftarrow G_{w_{-n+2}\dots w_{-1}} \longleftarrow \cdots \longleftarrow G_{\emptyset} \longleftarrow G_{\text{char}}$$

$$G_{\text{char}} == \prod_{i=1}^{l} H_{c_1\dots c_{i-1}}(c_i)$$

$$\cdots \leftarrow H_{c_{-m}\dots c_{-1}} \longleftarrow H_{c_{-m+1}\dots c_{-1}} \longleftarrow \cdots \longleftarrow H_{c_{-1}} \longleftarrow H_{\emptyset} \longleftarrow H_{\text{base}}$$

FIGURE 1.4 – Système de *back-off* pour un modèle n-gramme de mots utilisant comme distribution de base un modèle ∞-gramme de caractères tel que présenté dans l'article [42]

distribution de base $G_{w_{-k+2},\dots,w_{-1}}$. Autrement, le *back-off* s'effectue en tronquant le mot le plus ancien du contexte. En bas de la chaîne de *back-off*, la distribution unigramme se replie sur une distribution de base G_{char} qui est un modèle de caractères.

— La distribution de base G_{char} est remplacée par un "modèle ∞-gramme de caractères" (il pourrait aussi s'agir de phonèmes, mais l'article [42] de Mochihashi traite de la langue écrite). Un tel modèle prend en compte une distribution $H_{c_{-m}\dots c_{-1}}$ pour chaque chaîne de caractères possible $c_{-m}\dots c_{-1}$ de longueur m, et représentant les caractères successeurs de cette chaîne. Les paramètres de cette distribution sont engendrés par un $DP(\alpha, H_{c_{-m+1}\dots c_{-1}})$ c'est-à-dire que de même que pour les distributions de mots, le *back-off* s'effectue en tronquant le caractère le plus ancien du contexte. En bas de la chaîne, la distribution unigramme de caractères $H_{\emptyset}$ se replie sur une distribution uniforme H_{base} sur les caractères de la langue. La probabilité attribuée à une chaîne de caractères $c_1c_2\dots c_l$ par ce modèle ∞-gramme vaut :

$$\begin{aligned} G_{\text{char}}(c_1\dots c_l) &= H_{\emptyset}(c_1)\cdot H_{c_1}(c_2)\cdot H_{c_1c_2}(c_3)\cdot\ldots\cdot H_{c_1\dots c_{l-1}}(c_l) \\ &= \prod_{i=1}^{l} H_{c_1\dots c_{i-1}}(c_i) \end{aligned}$$

1.3.2 *Adaptor grammars*

Les modèles présentés dans la section 1.3.1 consituent une version bayésienne non-paramétrique des modèles n-grammes classiques, en ce sens qu'ils ajoutent à ceux-ci un système de "mise en cache" d'échantillons engendrés dans le passé, encourageant la réutilisation d'éléments et la convergence vers une loi de Zipf. Les *adaptor grammars* ([35], [33], [34]) consistent à appliquer cette même idée aux grammaires hors-contexte probabilistes (PCFG). Dans une PCFG, à chaque symbole non-terminal est associé une distribution sur les sous-arbres qu'il est

susceptible d'engendrer. Dans une *adaptor grammar*, on associe de plus à chaque symbole non-terminal un *cache* contenant tous les sous-arbres qu'il a engendré auparavant. Un symbole non-terminal réutilisera un sous-arbre avec une probabilité proportionnelle à son nombre d'utilisations dans le passé, mais une masse de probabilité résiduelle constante sera allouée à la création de nouveaux sous-arbres, selon une *distribution de base* donnée par une PCFG classique.

1.3.2.1 Grammaires hors-contexte probabilistes

Une PCFG est définie par :

— Un ensemble $\mathcal{N}$ de *symboles non-terminaux*.

— Un ensemble $\mathcal{T}$ de *symboles terminaux*.

— La spécification d'un *symbole de départ* $S \in \mathcal{N}$ parmi les symboles non-terminaux.

— Un ensemble $\mathcal{R}$ de *règles de réécriture* de la forme $\left[A \xrightarrow{\theta_{A \to B_1 \dots B_r}} B_1 \dots B_r\right]$ où $A \in \mathcal{N}$ et $B_1, \dots, B_r \in \mathcal{N} \cup \mathcal{T}$. $\theta_{A \to B_1 \dots B_r}$ est la *probabilité de déclenchement* d'une règle. Par économie de notation, on désignera par $\beta = B_1 \dots B_r$ la suite de symboles à droite d'une règle. Pour chaque symbole non-terminal $A \in \mathcal{N}$, on souhaite avoir une distribution de probabilité sur les règles de réécriture associées à ce symbole, c'est-à-dire que :

$$\sum_{[A \to \beta] \in \mathcal{R}_A} \theta_{A \to \beta} = 1$$

où $\mathcal{R}_A \subset \mathcal{R}$ désigne le sous-ensemble des règles ayant A pour terme de gauche.

Une PCFG engendre alors une chaîne aléatoire de symboles terminaux, selon le processus génératif suivant :

1. Initialement, on part d'une chaîne formée d'un unique symbole de départ S.
2. La chaîne de symboles courante est réécrite en appliquant la procédure suivante :
   (a) Chaque symbole terminal de la chaîne est laissé tel quel.
   (b) Pour chaque occurence de symbole non terminal A présent dans la chaîne, on choisit (aléatoirement) une règle de réécriture $[A \to \beta] \in \mathcal{R}_A$, selon les probabilités $\theta_{A \to \beta}$. Dans la chaîne, on remplace alors l'occurence de A en question par la chaîne de symboles β.
3. L'étape 2 est réitérée jusqu'à ce que la chaîne ne contienne plus que des symboles terminaux.

Ce processus engendre donc également un *arbre* dont les feuilles sont les symboles de la chaîne engendrée, dont les noeuds internes sont les symboles non-terminaux apparus au cours du processus, et où une arête relie deux noeuds dès

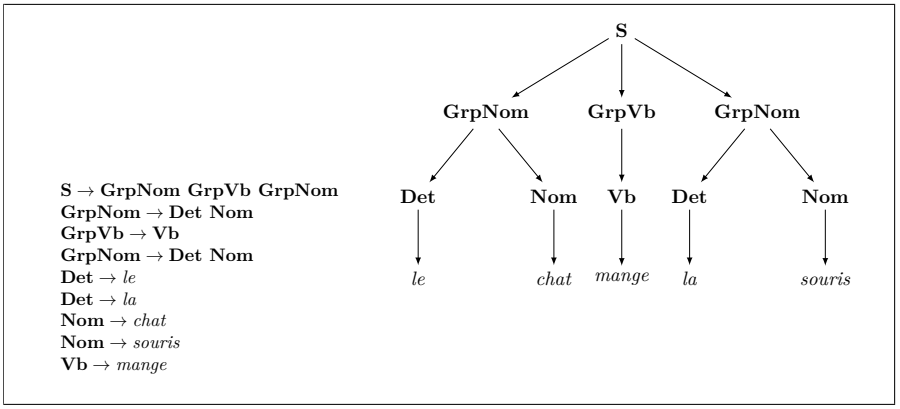


FIGURE 1.5 – Un exemple de grammaire hors contexte et d'arbre engendré par cette grammaire. Les symboles non-terminaux sont en gras et les terminaux en italique.

lors que le noeud fils est apparu suite au déclenchement d'une règle de réécriture appliquée au noeud père.

Selon ce processus génératif, la probabilité d'un arbre est le produit des probabilités des règles de réécriture qui ont été déclenchées pour obtenir cet arbre. La probabilité d'une chaîne de symboles est la somme des probabilités des arbres possédant cette chaîne de symboles comme "feuillage".

1.3.2.2 Adapteurs

Le système de "cache" introduit dans les *adaptor grammars* consiste à modifier le processus de réécriture des symboles non-terminaux dans les PCFG. Dans une PCFG, pour réécrire un symbole non-terminal A, on choisit aléatoirement une règle de la forme $[A \to \beta]$ à lui appliquer, selon les probabilités $\theta_{A\to\beta}$. Ainsi la génération d'un arbre sera indépendante des arbres engendrés par le passé, et ne dépendra que de la spécification de la grammaire. Dans une *adaptor grammar*, on dispose d'un ensemble de règles et de leurs probabilités (identique à celui d'une PCFG), mais une dépendance est introduite par rapport aux arbres qui ont été engendrés par le passé, en procédant de la manière suivante :

— Parmi tous les arbres engendrés par le passé, on recense tous les sous-arbres de racine A (notons $\mathcal{X}(A)$ cet ensemble). Pour tout sous-arbre T de racine A, notons $n(T)$ le nombre de fois qu'il apparaît dans les arbres passés. On choisit alors de générer un sous-arbre T comme expansion de A avec une probabilité proportionnelle à $n(T)$.

— Avec une probabilité résiduelle proportionnelle à α, on étend A comme pour une PCFG, en choisissant aléatoirement une règle $[A \to \beta]$ selon les probabilités $\theta_{A\to\beta}$.

$\mathbf{S} \rightarrow \mathbf{Mots}$
$\mathbf{Mots} \xrightarrow{p_1} \underline{\mathbf{Mot}}$
$\mathbf{Mots} \xrightarrow{1-p_1} \mathbf{Mots}\ \underline{\mathbf{Mot}}$
$\underline{\mathbf{Mot}} \xrightarrow{1} \mathbf{Chars}$
$\mathbf{Chars} \xrightarrow{p_2} \mathbf{Char}$
$\mathbf{Chars} \xrightarrow{1-p_2} \mathbf{Chars}\ \mathbf{Char}$
$\mathbf{Char} \xrightarrow{freq(a)} a$
$\mathbf{Char} \xrightarrow{freq(b)} b$
$\mathbf{Char} \xrightarrow{freq(c)} c$
...

FIGURE 1.6 – Exemple d'*adaptor grammar* équivalente au modèle unigramme de segmentation en mots de la section 1.3.1.1. Les non-terminaux sont en gras, les terminaux en italique, et les adapteurs sont soulignés. p_1 est la probabilité de cesser la génération d'une suite de mots pour former une phrase. p_2 est la probabilité de cesser la génération d'une suite de caractères pour former un mot. Le modèle de séquence de caractères servant de distribution de base au modèle de mots est un modèle unigramme dont les probabilités sont données par les fréquences $freq(x)$ d'apparition du caractère x dans les données d'apprentissage.

— Cette opération est répétée sur chaque symbole non-terminal, jusqu'à ce qu'il ne reste plus que des symboles terminaux.

Les probabilités de réécriture de A sont donc les suivantes :

$$\forall T \in \mathcal{X}(A), \mathbb{P}\left(\text{remplacer } A \text{ par } T\right) = \frac{n(T)}{\alpha + \sum_{T' \in \mathcal{X}(A)} n(T')}$$

$$\forall [A \rightarrow B_1 \dots B_r] \in \mathcal{R}, \mathbb{P}\left(\text{ajouter } B_1, \dots, B_r \text{ comme fils de } A\right) = \frac{\alpha \cdot \theta_{A \rightarrow B_1 \dots B_r}}{\alpha + \sum_{T' \in \mathcal{X}(A)} n(T')}$$

1.3.2.3 Remarques

— Il n'est pas nécessaire que les règles sus-citées prenant en compte le cache soient appliquées à *tous* les non-terminaux. Dans les articles, [35], [33], [34], ces règles ne sont appliquées qu'à un sous-ensemble de non-terminaux appelé *non-terminaux adapteurs*. Les autres non-terminaux sont développés en appliquant les règles d'une PCFG classique.

— Nous avons décrit ci-dessus le processus génératif. L'*inférence* (c'est-à-dire l'échantillonnage d'un arbre étant donnée la chaîne de symboles devant constituer son feuillage) utilise un algorithme de programmation dynamique appelé *inside-outside algorithm*.

— Dans [35], les *adaptor grammars* sont présentées comme un cadre général pour définir des modèles bayésiens non-paramétriques de séquences de

symboles. En effet, ce cadre est général dans le sens où les modèles markoviens présentés plus haut (section 1.3.1) peuvent être spécifiés comme un cas particulier d'*adaptor grammar*. Par exemple, l'*adaptor grammar* décrite dans la figure **??** est équivalente au modèle unigramme de Sharon Goldwater décrit dans la section 1.3.1.1

1.4 Autres approches

Plutôt que d'épuiser l'ensemble de la littérature concernant la segmentation morphologique, nous avons décidé de présenter plus haut quelques familles d'approches qui nous semblaient les plus adéquates pour introduire les méthodes que nous aborderons dans la suite de cette thèse. Dans cette section, nous évoquons brièvement quelques approches que nous avons laissées de côté.

— L'article [48] décrit un modèle log-linéaire de segmentation non-supervisée (autorisant un degré de supervision). Ce modèle définit une distribution jointe sur un corpus C ainsi que ses segmentations S, prenant la forme : $\mathbb{P}(C,S) = \frac{1}{Z}\exp\left(\sum_\mu \lambda_\mu n_\mu(S) + \sum_c \lambda_c n_c(S) + L(S)\right)$ où :

  — Pour chaque morphème candidat μ, λ_μ est un score quantifiant à quel point μ est un bon candidat, et $n_\mu(S)$ est le nombre de fois que μ apparaît comme morphème dans les segmentations.

  — Pour chaque *contexte* de morphème c (un contexte étant défini comme la donnée des trigrammes de caractères entourant un morphème), λ_c est un score quantifiant à quel point il est plausible que c soit un contexte de morphème, et $n_c(S)$ est le nombre de fois que c apparaît comme contexte de morphème parmi les segmentations.

  — $L(S)$ est un terme de régularisation (assimilable à une longueur de description) incitant le modèle à utiliser un lexique de morphèmes suffisamment restreint, tout en pénalisant un excès de segmentation (pour éviter par exemple de converger vers un modèle dégénéré ou les morphèmes sont les caractères).

  — Les paramètres du modèle sont donc les familles de poids λ_μ et λ_c.

  Selon les auteurs, l'avantage de ce modèle est que contrairement à des modèles s'appuyant par exemple sur un scénario génératif markovien, il ne fait pas de forte hypothèses d'indépendance : par exemple, l'utilisation de *features* de contexte rend la génération d'un morphème à la fois dépendante de ce qui le précède et de ce qui le suit, contrairement aux modèles markoviens où la génération d'un morphème ne dépend que des morphèmes générés avant.

— L'article [52] présente un modèle supervisé de segmentation à l'aide de *champs aléatoires conditionnels* (*conditional random fields* ou CRFs). Le problème de segmentation morphologique est présenté comme un problème d'étiquetage de caractères dans une séquence, les étiquettes à attribuer étant : S (morphème à caractère unique), B (début d'un morphème à

plusieurs caractères), M (intérieur d'un morphème à plusieurs caractères), E (fin d'un morphème à plusieurs caractères).

— L'article [53] exploite des représentations vectorielles de mots (obtenues par *analyse sémantique latente* : LSA) pour définir une mesure de similarité sémantique entre mots afin de guider l'apprentissage de paradigmes morphologiques.

— A partir de beaucoup de texte non annoté, et d'une liste de *racines* candidates (faible degré de supervision), l'article [61] décrit comme classifier les mots d'une langue en paires (*racine*, *inflexion*) en agrégeant différentes mesures de similarité.

— ParaMor [**?**] est un algorithme permettant (dans un esprit similaire à *Linguistica*) de regrouper les mots d'un lexique en paradigmes de type $\{$*ensemble de racines*$\} - \{$*ensemble de suffixes*$\}$.

— L'article [16] montre comment des paradigmes linguistiques peuvent se modéliser élégamment et de manière probabiliste et générative par un modèle graphique non orienté dont :

  — Les noeuds sont des variables aléatoires représentant les différentes formes d'un paradigme. Par exemple, en utilisant 7 noeuds pour modéliser la conjugaison d'un verbe : 6 noeuds ($X_{1\text{sg}}$, $X_{2\text{sg}}$, $X_{3\text{sg}}$, $X_{1\text{pl}}$, $X_{2\text{pl}}$, $X_{3\text{pl}}$) pour les différentes personnes et un noeud (X_{lem}) pour la racine du verbe.

  — Les arêtes (représentant les dépendances entre les différentes formes) sont des distribution jointes sur les valeurs que peuvent prendre les variables aléatoires à leurs extrémités. En tant que distribution jointes sur des paires de chaînes de caractères, elles peuvent être représentées par un transducteur. Par exemple, pour modéliser la conjugaison d'un verbe, on pourra employer 6 transducteurs, un transducteur pour relier chaque personne à la racine.

  La théorie des modèles graphiques probabilistes fournit alors de nombreux algorithmes pour l'inférence des paramètres, et pour l'échantillonnage d'un tel modèle, en verrouillant éventuellement certaines variables (afin par exemple, étant données les formes singulières d'un verbe, d'en déduire sa racine et ses formes plurielles). Un intêret majeur d'une telle approche est qu'elle se dispense entièrement de l'aspect concaténatif de la morphologie : à aucune étape il n'est requis d'introduire des points de coupure dans un mot pour découper des morphèmes. Ainsi cette approche pourrait être adéquate pour capturer des phénomènes de morphologie non-concaténative (comme l'alternance vocalique, ou les systèmes de racines à trois consonnes entre lesquelles des voyelles sont intercalées, fréquents dans les langues sémitiques).

1.5 Critères de classification

Nous concluons notre revue des travaux sur l'apprentissage de la morphologie en classifiant les approches que nous avons évoquées selon différents critères. Les critères qui nous ont semblé utiles afin de s'orienter dans la diversité des approches sont :

— **La représentation de la morphologie :** les approches diffèrent de par la manière dont la morphologie est représentée, et on peut les classifier selon la richesse de la représentation de la morphologie sur laquelle elles s'appuient. Parmi les approches présentées ici, il est possible de distinguer les représentations suivantes :

  — *Points de coupure* : la représentation la plus simple, consistant simplement à placer des points de coupure dans des séquences de caractères/phonèmes afin d'en délimiter les unités (mots ou morphèmes). Ex : méthode de Harris (section 1.1.1).

  — *Lexique de morphèmes* : une représentation plus riche que la précédente, où en plus de points de coupure, un lexique d'unités est inféré. Ex : modèle unigramme de Goldwater (section 1.3.1.1).

  — *Morphotactique* : une représentation plus riche que les deux précédentes. En plus d'un lexique de morphèmes, on cherche à inférer une grammaire décrivant la manière dont les morphèmes se succèdent. Cette grammaire peut prendre la forme de modèles markoviens (ex : modèles bi-gramme de Goldwater et n-gramme de Mochihashi, section 1.3.1.2), de modèles de Markov cachés où les morphèmes sont regroupés en classes (ex : Morfessor Categories-ML, section 1.2.3.2), ou de grammaires hors contextes (ex : adaptor grammars, section 1.3.2). Ces représentations abordent la morphologie sous l'angle de séquences de morphèmes, et sont donc de nature à capturer les aspects concaténatifs de la morphologie.

  — *Paradigmes* : une représentation en termes d'ensembles de racines susceptibles de recevoir certains ensembles d'affixes. Ex : Linguistica (section 1.2.2).

— **Recours à un modèle génératif :** Certaines approches s'appuient sur un modèle génératif, expliquant les données comme étant issues d'un certain scénario (par exemple un scénario de concaténation de morphèmes selon certaines règles probabilistes, ex : les modèles bayésiens non-paramétriques que nous avons présenté). De tels modèles sont donc susceptibles de générer des formes de mots. D'autres approches ne font pas appel à un tel scénario, et consistent simplement à fournir des critères pour déterminer si un segment donné doit être retenu ou non comme une unité viable (ex : la méthode de Harris, le modèle log-linéaire de l'article [48]).

— **Possibilité de supervision ou d'injection de connaissances expertes :** Selon les méthodes, plus ou moins de marge est laissée à l'introduction d'*a priori* pour guider le modèle. Ces *a priori* peuvent prendre la

forme d'une semi-supervision (par exemple, en utilisant un petit corpus annoté à côté d'un gros corpus non annoté pour guider l'apprentissage), ou de connaissances expertes qui peuvent être exprimées par des *a priori* probabilistes ou par des paramètres fixés à la main.

— **Utilisation d'heuristiques :** Ce critère consiste à classifier les méthodes selon leur recours à des heuristiques et à des paramètres arbitraires dont la validité est testée empiriquement.

Méthode	**Représentation de la morphologie**	**Modèle génératif**	**Possibilité de supervision**	**Heuristiques**
Harris [29]	Points de coupure	Non	Non	Nombreux seuils arbitraires
Andreev [32]	Points de coupure, paradigmes *racine-affixe*	Non	Non	Nombreux seuils arbitraires
Linguistica [19]	Paradigmes *racine-suffixe* avec représentation hiérarchique possible	Oui	Segmentations préalables	Pour la recherche locale
Morfessor Baseline [12]	Lexique de morphèmes	Non	Segmentations préalables	Non
Categories-ML [13]	Morphotactique : séquences de préfixes/suffixes/racines	Oui	Segmentations préalable	Pour affecter des morphèmes à des classes
Categories-MAP [14]	Morphotactique : séquences de préfixes/suffixes/racines avec représentation hiérarchique possible	Oui	Non	Présentes sous forme d'*a priori* probabiliste
FlatCat [25]	Morphotactique : séquences de préfixes/suffixes/racines	Oui	Segmentations préalables	Présentes sous forme d'*a priori* probabiliste
Goldwater-unigramme [24]	Lexique de morphèmes	Oui	Segmentations préalables	Non
Mochihashi [42]	Lexique de morphèmes et modèle n-gramme de succession	Oui	Segmentations préalables	Non
Adaptor grammars [35]	Morphotactique : PCFG	Oui	Segmentations préalables, spécification de la PCFG	Non
Poon [48]	Points de coupure	Non	Segmentations préalables	Non
Ruokolainen [52]	Points de coupure	Non	Segmentations préalables	Non
Schone, Jurafsky [53]	Paradigmes : paires *racine, inflexion*	Non	Racines candidates	Analyse sémantique latente, agrégation de diverses mesures de similarité
Dreyer, Eisner [16]	Paradigmes : distribution jointe sur les différentes formes	Oui	Très flexible	Non

FIGURE 1.7 – Classification de différentes méthodes de traitement de la morphologie

Chapitre 2

Processus de Pitman-Yor

Dans ce chapitre, nous présentons l'approche bayésienne en statistiques. L'approche bayésienne consiste à ne pas envisager le choix d'un modèle comme la recherche d'un jeu de paramètres optimal, mais plutôt à obtenir à partir des données une distribution de probabilités sur l'espace des paramètres, puis à moyenner les différents modèles possibles selon cette distribution. Nous présenterons ensuite les aspects théoriques des processus de Dirichlet et de Pitman-Yor sur lesquels se fonderont les modèles présentés dans les chapitres suivants. En plus de suivre un formalisme bayésien, ces processus aléatoires donnent lieu à des modèles non-paramétriques, c'est-à-dire dont l'espace des paramètres est de dimension infinie. Ce caractère non-paramétrique les rend adaptés à la modélisation des langues, où les mots (ou autres évènements linguistiques) peuvent se rencontrer dans une variété potentiellement illimitée. En dernière partie du chapitre, nous présentons une première application de ces processus à la segmentation morphologique.

2.1 Approche bayésienne et distribution de Dirichlet

2.1.1 Critère de vraisemblance maximale

Considérons le problème suivant :

— On dispose d'un ensemble d'observations $\mathbf{e} = \{e_1, e_2, \ldots, e_r\}$ appartenant à un certain ensemble $\mathcal{E}$.

— On dispose par ailleurs d'une famille de distributions de probabilités $P(.|\theta)$ paramétrisée par un paramètre $\theta \in \Theta \subseteq \mathbb{R}^d$.

On cherche à modéliser les données $\mathbf{e}$ à l'aide de la famille $P(.|\theta)$, de manière à obtenir un "bon" modèle génératif de ces données, c'est-à-dire un modèle capable de rendre compte de celles-ci tout en conservant la capacité de généraliser à de nouvelles données. Il reste donc à préciser le critère qu'il convient d'adopter pour aboutir à un tel modèle.

Une critère classique consiste à rechercher le paramètre de *vraisemblance maximale*, c'est à dire le paramètre θ^* pour lequel la probabilité de générer le jeu d'observations $\mathbf{e}$ est maximal :

$$\theta^* = \text{argmax}_\theta \prod_{e_i \in \mathbf{e}} P(e_i|\theta^*)$$

Un problème lié à ce critère est celui du *sur-apprentissage* : lorsque la famille de distributions $P(.|\theta)$ est assez riche (sa richesse pouvant notamment se mesurer à la quantité de paramètres), cette famille autorisera des choix de paramètres modélisant trop fidèlement les données pour permettre au modèle appris de généraliser. Autrement dit, le désavantage du critère de vraisemblance maximale est que celui-ci ne guide le choix des paramètres qu'à l'aune de la fidélité du modèle $P(.|\theta)$ aux données, sans prendre en compte le fait que les données (surtout si celles-ci sont en faible quantité) puissent ne pas représenter de manière suffisamment exhaustive le phénomène à modéliser.

Exemple 1. (Lancer de dés) Supposons qu'on dispose d'une suite de quelques lancers de dés $\mathbf{e} = \{2, 5, 6, 4, 1, 2, 1, 3, 4, 5, 3\}$. On cherche à modéliser le comportement du dé par une distribution discrète, paramétrée par un vecteur $\theta \in \left\{0 \leq \theta_1, \ldots, \theta_6 \leq 1 | \sum_{k=1}^6 \theta_k = 1\right\}$ tel que θ_k soit la probabilité d'apparition du chiffre k.

La *vraisemblance* d'un θ (c'est-à-dire la probabilité attribuée aux données par le choix de ce paramètre vaut) :

$$L(\theta) = \prod_{e_i \in \mathbf{e}} P(e_i|\theta)$$

et donc, en notant $n_\mathbf{e}(k)$ le nombre de fois que le chiffre k apparaît dans les données :

$$L(\theta) = \prod_{k=1}^{6} P(k|\theta)^{n_\mathbf{e}(k)} = \prod_{k=1}^{6} \theta_k^{n_\mathbf{e}(k)}$$

$$\log L(\theta) = \sum_{k=1}^{6} n_\mathbf{e}(k) \cdot \log \theta_k$$

Le paramètre maximisant cette valeur peut se trouver en cherchant lorsque son gradient est normal à l'hyperplan d'équation $\sum_{k=1}^6 \theta_k = 1$ dans lequel se situe la variété des paramètres, c'est-à-dire lorsque toutes les coordonnées de son gradient sont égales.

$$d(\log L(\theta)) = \sum_{k=1}^{6} \frac{n_\mathbf{e}(k)}{\theta_k} d\theta_k$$

Il faut donc que $\frac{n_\mathbf{e}(1)}{\theta_1} = \ldots = \frac{n_\mathbf{e}(6)}{\theta_6} = \alpha$. Donc $\forall k, \theta_k = \frac{n_\mathbf{e}(k)}{\alpha}$ et pour que les θ_k somment à un, il faut que $\alpha = N_\mathbf{e}$ (le nombre total d'observations) et donc

$\forall k, \theta_k = \frac{n_{\mathbf{e}}(k)}{N_{\mathbf{e}}}$, c'est-à-dire que les θ_k sont simplement la fréquence d'apparition du chiffre k dans les données.

Dans le cas des données que nous avons données en exemple, tous les chiffres apparaissent deux fois, sauf le chiffre 6 n'apparaissant qu'une seule fois. Ainsi, en se fiant au critère de vraisemblance maximale, nous obtiendrions un modèle attribuant au chiffre 6 une probabilité d'apparition deux fois plus faible qu'aux autres chiffres. Ce qui va à l'encontre du bon sens, qui dirait plutôt que les données sont en quantité trop réduite pour pouvoir en déduire que non seulement le dé est biaisé, mais présente de plus un biais aussi important.

2.1.2 Approche bayésienne

Une approche permettant de remédier au comportement indésirable du critère de vraisemblance maximale est l'*approche bayésienne*. Cette approche consiste à prendre en compte un *degré de croyance* dans les modèles exprimés par le choix du paramètre θ. Cette prise en compte s'effectue de la manière suivante :

— Un *a priori* sur le choix de θ est défini sous la forme d'une distribution $P(\theta)$ attribuant des poids plus élevés aux choix de paramètre les plus "pertinents". La question de cette "pertinence" dépend du phénomène à modéliser, et constitue par conséquent une forme de *connaissance experte*. Par exemple, en ce qui concerne le lancer de dés, on peut partir de l'a priori qu'un dé est en général bien équilibré et choisir une distribution $P(\theta)$ possèdant un maximum en $\theta = (\frac{1}{6}, \ldots, \frac{1}{6})$ tout en conservant de la masse de probabilité pour des valeurs non-équilibrées de θ afin de permettre l'éventualité d'un dé déséquilibré. Si aucune connaissance de ce type n'est disponible sur le phénomène à modéliser, on peut toujours utiliser un a priori *agnostique* en choisissant par exemple une distribution $P(\theta)$ uniforme.

— Ensuite, après avoir obtenu r observations, le degré de croyance est mis à jour en calculant l'*a posteriori* $P(\theta|\mathbf{e}) = \dfrac{P(\theta, \mathbf{e})}{P(\mathbf{e})} = \dfrac{P(\mathbf{e}|\theta)P(\theta)}{\int_{\theta' \in \Theta} P(\mathbf{e}|\theta')P(\theta')}$.

Remarquons que cette approche consiste a remplacer le choix d'un paramètre "optimal" par le calcul d'une distribution a posteriori sur l'ensemble des paramètres (c'est-à-dire des modèles) possibles. Ainsi, après observation des données, le modèle probabiliste obtenu ne sera pas un unique modèle pris dans la famille $P(.|\theta)$, mais une moyenne pondérée de tous ces modèles, le poids d'un modèle reflétant le degré de croyance a posteriori en celui-ci. En termes mathématiques, ceci signifie que le modèle prédictif inféré sera obtenu en *marginalisant* θ selon la distribution a posteriori. Ainsi la probabilité d'un évènement $x \in \mathcal{E}$ vaudra :

$$P(x|\mathbf{e}) = \int_{\theta} P(x|\theta)P(\theta|\mathbf{e})$$

Exemple 2. (Lancer de dés, suite) Choisissons un a priori uniforme sur le

paramètre θ :

$$P(\theta) = \frac{1}{\int_{\theta_k \geq 0, \sum_{k=1}^{6} \theta_k = 1} d\theta_1 \dots d\theta_6} = \frac{1}{\int_{(\theta_1,\dots,\theta_5)\in\Delta_5} d\theta_1 \dots d\theta_5}$$

où Δ_n désigne le ensemble $\{\theta \in \mathbb{R}^n | \forall k, \theta_k \geq 0 \text{ et } \sum_{k=1}^{n} \theta_k < 1\}$. Etant données les observations $\mathbf{e}$, d'après la règle de Bayes la distribution a posteriori sur θ s'exprime comme :

$$P(\theta|\mathbf{e}) = \frac{P(\mathbf{e}|\theta)P(\theta)}{\int_\theta P(\mathbf{e}|\theta)P(\theta)} = \frac{\theta_1^{n_\mathbf{e}(1)} \dots \theta_6^{n_\mathbf{e}(6)}}{\int_\theta \theta_1^{n_\mathbf{e}(1)} \dots \theta_6^{n_\mathbf{e}(6)} d\theta_1 \dots d\theta_6}$$

Posons $B(\alpha_1, \dots, \alpha_n) = \int_{\left(\theta_k \geq 0, \sum_{k=1}^{n} \theta_k = 1\right)} \theta_1^{\alpha_1 - 1} \dots \theta_n^{\alpha_n - 1} d\theta_1 \dots d\theta_n$ (pour des $\alpha_k \geq 1$).

Le calcul de cette intégrale multiple (l'intégrale *bêta multivariée*) nous servira pour ce calcul et aussi pour la suite.

En posant $\theta_n = 1 - \theta_1 - \dots - \theta_{n-1}$, elle peut se réexprimer comme :

$$\begin{aligned} B(\alpha_1, \dots, \alpha_n) &= \int_{\Delta_{n-1}} \theta_1^{\alpha_1 - 1} \dots \theta_{n-1}^{\alpha_{n-1} - 1} \left(1 - \sum_{k=1}^{n-1} \theta_k\right)^{\alpha_n - 1} d\theta_{1\dots(n-1)} \\ &= \int_{\Delta_{n-2}} \theta_1^{\alpha_1 - 1} \dots \theta_{n-2}^{\alpha_{n-2} - 1} \left(\int_{\theta_{n-1}=0}^{1-\sum_{k=1}^{n-2}\theta_k} \theta_{n-1}^{\alpha_{n-1}-1} \left(1 - \sum_{k=1}^{n-2} \theta_k\right)^{\alpha_n - 1} d\theta_{n-1} \right) d\theta_{1\dots(n-2)} \end{aligned}$$

En intégrant par parties l'intégrale à l'intérieur des parenthèses, on obtient :

$$\int_{\theta_{n-1}=0}^{1-\sum_{k=1}^{n-2}\theta_k} \theta_{n-1}^{\alpha_{n-1}-1} \left(1 - \sum_{k=1}^{n-1} \theta_k\right)^{\alpha_n - 1} d\theta_{n-1} = \frac{\alpha_n - 1}{\alpha_{n-1}} \int_{\theta_{n-1}=0}^{1-\sum_{k=1}^{n-2}\theta_k} \theta_{n-1}^{\alpha_{n-1}} \left(1 - \sum_{k=1}^{n-1} \theta_k\right)^{\alpha_n - 2} d\theta_{n-1}$$

ce qui implique que :

$$\begin{aligned} B(\alpha_1, \dots, \alpha_n) &= \frac{\alpha_n - 1}{\alpha_{n-1}} B(\alpha_1, \dots, \alpha_{n-2}, \alpha_{n-1} + 1, \alpha_n - 1) \\ &= \frac{(\alpha_n - 1)(\alpha_n - 2)\dots 1}{\alpha_{n-1}(\alpha_{n-1}+1)\dots(\alpha_{n-1}+\alpha_n - 2)} B(\alpha_1, \dots, \alpha_2, \alpha_{n-1} + \alpha_n - 1, 1) \quad \text{(par récurrence)} \\ &= \frac{(\alpha_n - 1)!(\alpha_{n-1} - 1)!}{(\alpha_{n-1} + \alpha_n - 2)!} B(\alpha_1, \dots, \alpha_2, \alpha_{n-1} + \alpha_n - 1, 1) \end{aligned}$$

Ensuite, comme B est une fonction symétrique (indépendante de l'ordre de ses arguments), il est facile de répéter la procédure et de montrer par récurrence que :

$$B(\alpha_1, \dots, \alpha_n) = \frac{(\alpha_1 - 1)! \dots (\alpha_n - 1)!}{(\alpha_1 + \dots + \alpha_n - n)!} \underbrace{B(\alpha_1 + \dots + \alpha_n - n + 1, 1, \dots, 1)}_{n \text{ arguments}}$$

Calculons $B(\alpha_1 + \ldots + \alpha_n - n + 1, 1, \ldots, 1)$:

$$B\left(\sum_i \alpha_i - n + 1, 1, \ldots, 1\right) = \int_{\theta_1=0}^{1} \theta_1^{\sum_i \alpha_i - n} \underbrace{\left(\int_{(\theta_i \geq 0, \theta_2 + \ldots + \theta_{n-1} < 1-\theta_1)} d\theta_2 \ldots d\theta_{n-1}\right)}_{= \frac{(1-\theta_1)^{n-2}}{(n-2)!}} d\theta_1$$

$$= \frac{1}{(n-2)!} \int_{\theta_1=0}^{1} \theta_1^{\sum_i \alpha_i - n} (1-\theta_1)^{n-2} d\theta_1$$

Rappelons que la valeur des intégrales du type $\int_0^1 t^p (1-t)^q dt = \frac{p! \cdot q!}{(p+q+1)!}$ se calcule facilement par une suite d'intégrations par parties. On obtient :

$$B\left(\sum_i \alpha_i - n + 1, 1, \ldots, 1\right) = \frac{1}{(n-2)!} \cdot \frac{(\sum_i \alpha_i - n)!(n-2)!}{(\sum_i \alpha_i - 1)!}$$
$$= \frac{(\sum_i \alpha_i - n)!}{(\sum_i \alpha_i - 1)!}$$

Puis, en réinjectant ce résultat dans l'expression de $B(\alpha_1, \ldots, \alpha_n)$ obtenue plus haut, on obtient :

$$B(\alpha_1, \ldots, \alpha_n) = \frac{(\alpha_1 - 1)! \ldots (\alpha_n - 1)!}{(\alpha_1 + \ldots + \alpha_n - n)!} \cdot \frac{(\sum_i \alpha_i - n)!}{(\sum_i \alpha_i - 1)!}$$
$$= \frac{(\alpha_1 - 1)! \ldots (\alpha_n - 1)!}{(\alpha_1 + \ldots + \alpha_n - 1)!}$$

qu'on peut réexprimer en introduisant la fonction Gamma $\Gamma(n) = (n-1)!$ connue pour être un prolongement de la factorielle à l'ensemble des réels :

$$B(\alpha_1, \ldots, \alpha_n) = \frac{\Gamma(\alpha_1) \ldots \Gamma(\alpha_n)}{\Gamma(\alpha_1 + \ldots + \alpha_n)}$$

Ainsi, pour en revenir à l'exemple du lancer de dés, nous avons vu que :

$$P(\theta|\mathbf{e}) = \frac{\theta_1^{n_\mathbf{e}(1)} \ldots \theta_6^{n_\mathbf{e}(6)}}{\int_\theta \theta_1^{n_\mathbf{e}(1)} \ldots \theta_6^{n_\mathbf{e}(6)} d\theta_1 \ldots d\theta_6}$$
$$= \frac{\theta_1^{n_\mathbf{e}(1)} \ldots \theta_6^{n_\mathbf{e}(6)}}{B(1 + n_\mathbf{e}(1), \ldots, 1 + n_\mathbf{e}(6))}$$

En suivant l'approche bayésienne, un modèle prédictif s'obtient à partir de cette distribution a posteriori en moyennant tous les modèles (i.e : en margina-

lisant θ) selon cette distribution :

$$\begin{aligned} P(k|\mathbf{e}) &= \int_\theta \underbrace{P(k|\theta)}_{=\theta_k} \cdot P(\theta|\mathbf{e}) d\theta \\ &= \frac{1}{B\left(1+n_\mathbf{e}(1), \ldots, 1+n_\mathbf{e}(6)\right)} \int_\theta \theta_1^{n_\mathbf{e}(1)} \ldots \theta_k^{1+n_\mathbf{e}(k)} \ldots \theta_6^{n_\mathbf{e}(6)} \\ &= \frac{B\left(1+n_\mathbf{e}(1), \ldots, 2+n_\mathbf{e}(k), \ldots, 1+n_\mathbf{e}(6)\right)}{B\left(1+n_\mathbf{e}(1), \ldots, 1+n_\mathbf{e}(6)\right)} \\ &= \frac{1+n_\mathbf{e}(k)}{6+N_\mathbf{e}} \end{aligned}$$

Ainsi dans notre exemple où $\mathbf{e} = \{2, 5, 6, 4, 1, 2, 1, 3, 4, 5, 3\}$ (tous les nombres apparaissent deux fois sauf 6 n'apparaissant qu'une seule fois), $P(6|\mathbf{e})$ vaudra $\frac{2}{17}$. Nous voyons donc que le fait de moyenner sur tous les modèles entraîne un biais moindre vers l'hypothèse d'un dé deséquilibré que le critère de vraisemblance maximale. Remarquons également que dans l'expression de $P(k|\mathbf{e})$, la taille $N_\mathbf{e}$ du jeu d'observations intervient de telle sorte que lorsque $N_\mathbf{e} \to \infty$, $P(k|\mathbf{e})$ convergera vers les fréquences d'apparition de chaque chiffre dans les données (et donc se rapprochera du modèle donné par le critère de vraisemblance maximale). Au contraire, plus les observations sont en quantité réduite, plus $P(k|\mathbf{e})$ se rapprochera d'une distribution uniforme, conformément à notre a priori agnostique. Une interprétation de ces remarques est la suivante : "Lorsque j'ai un gros jeu de données, celui-ci est plutôt exhaustif et reflète le comportement général du phénomène à modéliser, donc je peux me fier aux fréquences d'apparition des évènements dans mes données. En revanche, lorsque les données sont en nombre faible, je préfère me replier sur l'a priori qui m'a été fourni".

2.1.3 Distribution de Dirichlet

Nous avons vu ci-dessus que l'a posteriori sur les paramètres d'une distribution discrète soumis à un a priori uniforme prend une forme particulière. Nous généralisons ce résultat dans cette section.

Definition 3. (Distribution de Dirichlet) Soient $\alpha_1, \ldots, \alpha_n > 0$. La *distribution de Dirichlet* de paramètres $\alpha_1, \ldots, \alpha_n$ notée $Dir(\alpha_1, \ldots, \alpha_n)$ est définie sur le simplexe $\{\theta \in \mathbb{R}_+^n | \theta_1 + \ldots + \theta_n = 1\}$ par la densité de probabilité :

$$p(\theta) d\theta_{1\ldots n} = \frac{\theta_1^{\alpha_1 - 1} \ldots \theta_n^{\alpha_n - 1}}{B(\alpha_1, \ldots, \alpha_n)} d\theta_{1\ldots n}$$

Remarque. Lorsque $\alpha_1 = \ldots = \alpha_n = 1$, $Dir(\alpha_1, \ldots, \alpha_n)$ est une distribution uniforme sur le simplexe.

Propriété 4. *L'espérance d'une distribution de Dirichlet vaut*

$$\mathbb{E}[Dir(\alpha_1, \ldots, \alpha_n)] = \left(\frac{\alpha_1}{\sum_i \alpha_i}, \ldots, \frac{\alpha_n}{\sum_i \alpha_i} \right)$$

De plus, lorsque les $\alpha_i > 1$, une telle distribution possède un unique mode en

$$\theta^* = \left(\frac{\alpha_1 - 1}{\sum_i \alpha_i - n}, \ldots, \frac{\alpha_n - 1}{\sum_i \alpha_i - n} \right)$$

On peut notamment voir les vecteurs engendrés par une distribution de Dirichlet comme des *vecteurs de probabilités*, puisque leurs coordonnées sont positives et somment à un.

L'intêret de cette famille de distributions pour notre propos est qu'elle est *conjuguée* à la famille des lois discrètes : c'est-à-dire que si les paramètres d'une loi discrète sont soumis à un a priori de Dirichlet, alors après observation d'échantillons du phénomène à modéliser, l'a posteriori reste une distribution de Dirichlet, et ses nouveaux paramètres peuvent se calculer facilement à partir des données. L'avantage de cette propriété de conjugaison est donc que le calcul des a posteriori est immédiat et ne nécessite pas d'intégrations, ce qui serait en général le cas. Ceci est résumé par la propriété suivante :

Propriété 5. ***(Conjugaison)*** *Si $P(x|\theta)$ une loi discrète à n issues possibles $x_1, \ldots, x_n$ et de paramètre $\theta \in \mathbb{R}^n$, si ce dernier est soumis à un a priori $\theta \sim Dir(\alpha_1, \ldots, \alpha_n)$, et si $\boldsymbol{e}$ est un ensemble d'observations alors l'a posteriori $P(\theta|\boldsymbol{e})$ est une distribution de Dirichlet $Dir\,(\alpha_1 + n_{\boldsymbol{e}}(x_1), \ldots, \alpha_n + n_{\boldsymbol{e}}(x_n))$.*

De même, étant donné un ensemble d'observations, le modèle prédictif correspondant à la moyenne de tous les modèles selon la distribution a posteriori $P(\theta|\mathbf{e})$ peut s'obtenir en marginalisant θ sans avoir à explicitement calculer cette moyenne :

Propriété 6. ***(Modèle a posteriori)*** *Si les paramètres d'une loi discrète $(P(x_k) = \theta_k)_{1 \leq k \leq n}$ sont soumis à un a priori de Dirichlet $\theta \sim Dir(\alpha_1, \ldots, \alpha_n)$ alors étant donné un ensemble d'observations $\mathbf{e}$, la probabilité a posteriori d'un évènement x_k vaut :*

$$P(x_k|\boldsymbol{e}) = \frac{\alpha_k + n_{\boldsymbol{e}}(x_k)}{\sum_i \alpha_i + N_{\boldsymbol{e}}}$$

Preuve.

$$\begin{aligned}
P(x_k|\mathbf{e}) &= \int_\theta P(x_k|\theta) P(\theta|\mathbf{e}) d\theta = \int_\theta \theta_k \frac{\theta_1^{\alpha_1 + n_{\mathbf{e}}(1) - 1} \ldots \theta_n^{\alpha_n + n_{\mathbf{e}}(n) - 1}}{B(\alpha_1 + n_{\mathbf{e}}(1), \ldots \alpha_n + n_{\mathbf{e}}(n))} d\theta \\
&= \frac{\int_\theta \theta_1^{\alpha_1 + n_{\mathbf{e}}(1) - 1} \ldots \theta_k^{\alpha_k + n_{\mathbf{e}}(k)} \ldots \theta_n^{\alpha_n + n_{\mathbf{e}}(n) - 1} d\theta}{B(\alpha_1 + n_{\mathbf{e}}(1), \ldots, \alpha_n + n_{\mathbf{e}}(n))} \\
&= \frac{B(\alpha_1 + n_{\mathbf{e}}(1), \ldots, \alpha_k + n_{\mathbf{e}}(k) + 1, \ldots, \alpha_n + n_{\mathbf{e}}(n))}{B(\alpha_1 + n_{\mathbf{e}}(1), \ldots, \alpha_n + n_{\mathbf{e}}(n))} \\
&= \frac{\Gamma(\alpha_1 + \ldots + \alpha_n + N_{\mathbf{e}})}{\Gamma(\alpha_1 + \ldots + \alpha_n + N_{\mathbf{e}} + 1)} \frac{\Gamma(\alpha_k + n_{\mathbf{e}}(k) + 1)}{\Gamma(\alpha_k + n_{\mathbf{e}}(k))} \\
&= \frac{\alpha_k + n_{\mathbf{e}}(k)}{\alpha_1 + \ldots + \alpha_n + N_{\mathbf{e}}}
\end{aligned}$$

□

Enfin, remarquons que nous pouvons adopter un autre point de vue sur la définition de la distribution de Dirichlet : étant donné qu'elle génère les paramètres d'une loi discrète (qui sont en même nombre que les valeurs que cette loi peut prendre), nous pouvons voir une distribution de Dirichlet comme un processus générant des distributions discrètes, c'est-à-dire une *distribution sur des distributions*.

Posons $\alpha = \sum_i \alpha_i$ et $G_0(x_k) = \dfrac{\alpha_k}{\sum_i \alpha_i}$ pour tout x_k dans le ensemble $\mathcal{D} = \{x_1, \ldots, x_n\}$ de la loi discrète à modéliser. Notons $DP(\alpha, G_0)$ la loi de probabilité sur l'ensemble $Prob(\mathcal{D})$ des distributions de probabilités sur $\mathcal{D}$ telle que $DP(\alpha, G_0)$ affecte à une distribution G la même valeur que $Dir(\alpha_1, \ldots, \alpha_n)$ affecte au vecteur de paramètres de G. Cette nouvelle définition n'est qu'un changement de point de vue (qui va nous servir par la suite), où plutôt que de considérer une loi de Dirichlet comme un processus générant les paramètres d'une distribution discrète, on la considère comme un processus générant cette distribution elle-même.

Conditionnellement à un ensemble $\mathbf{e}$ d'observations, la formule du modèle a posteriori se réécrit alors :

$$P(x|\mathbf{e}) = \frac{n_{\mathbf{e}}(x) + \alpha G_0(x)}{N_{\mathbf{e}} + \alpha}$$

Lorsque $\mathbf{e} = \emptyset$, ce modèle est la distribution G_0. La distribution G_0 correspond donc à l'a priori. De plus, lorsque $N_{\mathbf{e}} \to \infty$, le modèle a posteriori va converger vers les fréquences d'apparition des évènements dans les observations, mais d'autant plus lentement que α est élevé. α peut donc s'interpréter comme la *force* de l'a priori : plus α est élevé, plus l'a priori possèdera de l'inertie et sera peu enclin à être remis en question par les observations.

2.2 Modèles non-paramétriques

Modéliser des données à l'aide de modèles paramétriques nécessite de fixer d'avance un nombre de paramètres et donc d'imposer un degré de complexité au modèle qui sera inféré. Or dans les problèmes de traitement des langues, la complexité requise pour obtenir un bon modèle peut varier d'une langue à l'autre, et il n'est pas évident de déduire des données à modéliser un nombre de paramètre adéquat pour les représenter sans pour autant les sur-apprendre. Par exemple, lorsqu'on essayera d'apprendre la morphologie d'une langue à partir de données non annotées, fixer un nombre de paramètres reviendra à fixer la taille du lexique de morphèmes à inférer, alors que la taille de ce lexique nous est a priori inconnue.

Les modèles dits *non-paramétriques* remédient à ce problème en ne limitant pas le nombre de paramètres à inférer, mais en permettant de l'adapter à la complexité des données. Cette adaptation nécessite de trouver un compromis entre deux exigences opposées : autoriser d'une part le nombre de paramètres à grandir afin de modéliser les données avec fidélité, mais d'autre part suffisamment le contraindre afin de ne pas aboutir à un modèle dont la haute complexité

ne servirait qu'apprendre les données par cœur qui se révelerait inefficace sur des exemples hors des données d'apprentissage.

2.2.1 Processus de Dirichlet

Le *processus de Dirichlet* est une généralisation non-paramétrique de la distribution de Dirichlet présentée à la section précédente. Nous avons vu qu'une distribution de Dirichlet $Dir(\alpha_1, \ldots, \alpha_n)$ servant d'a priori pour les paramètres de multinomiales sur un ensemble fini $\mathcal{D} = \{x_1, \ldots, x_n\}$ pouvait prendre la forme d'une distribution sur des distributions $DP(\alpha, G_0)$ où G_0 est une *distribution de base* sur $\mathcal{D}$ telle que $\alpha G_0(x_i) = \alpha_i$. Les vecteurs de paramètres engendrés par $Dir(\alpha_1, \ldots, \alpha_n)$ correspondent aux distributions sur $\mathcal{D}$ engendrées par $DP(\alpha, G_0)$.

L'avantage du second point de vue adopté est qu'il peut facilement se généraliser au cas d'un ensemble infini. Cette généralisation est le *processus de Dirichlet*.

Definition 7. (Processus de Dirichet) Soit $\mathcal{D}$ un ensemble mesurable de cardinal quelconque et $Prob(\mathcal{D})$ l'ensemble des distributions de probabilités sur $\mathcal{D}$. Alors le *processus de Dirichlet* $DP(\alpha, G_0)$ où $\alpha > 0$ et $G_0 \in Prob(\mathcal{D})$ est l'unique distribution sur $Prob(\mathcal{D})$ telle que :
si $G \sim DP(\alpha, G_0)$ alors pour toute partition finie $\{A_1, \ldots, A_n\}$ de $\mathcal{D}$,

$$(G(A_1), \ldots, G(A_n)) \sim Dir\left(\alpha G_0(A_1), \ldots, \alpha G_0(A_n)\right)$$

— α est le *paramètre de force* du processus de Dirichlet

— G_0 est sa *distribution de base*

Pour comprendre en quoi le processus de Dirichlet est une généralisation infinie de la distribution de Dirichlet, on peut considérer le fait suivant. Toute application $\pi : \mathcal{D} \to X$ de $\mathcal{D}$ vers un ensemble fini $X = \{x_i\}_{i=1}^n$ permet de ramener des distributions sur $\mathcal{D}$ à des distributions sur X : si G est une distribution sur $\mathcal{D}$, alors $\pi G(x) = G(\pi^{-1}(x))$ est une distribution sur X attribuant à chaque $x \in X$ la masse de son image réciproque. Dire que $G \sim DP(\alpha, G_0)$ revient alors à dire que pour toute application de $\mathcal{D}$ vers un ensemble fini, les paramètres de la distribution (aléatoire) πG suivront une loi de Dirichlet de paramètres $(\alpha \cdot \pi G_0(x_1), \ldots, \alpha \cdot \pi G_0(x_n))$. En résumé : "un processus de Dirichlet engendre des distributions qui, vues à travers n'importe quelle application vers un ensemble fini, se réduisent à une distribution dont les paramètres suivent une loi de Dirichlet".

2.2.2 Le processus de Dirichlet comme a priori bayésien

De même que la distribution de Dirichlet, le processus de Dirichlet peut servir d'a priori sur des distributions, mais à support infini.

Supposons qu'on dispose d'un jeu d'observations $\mathbf{e}$ dans un ensemble infini $\mathcal{D}$. On souhaite modéliser ces observations par une distribution $G \in Prob(\mathcal{D})$

en plaçant un processus de Dirichlet comme a priori sur cette distribution : $G \sim DP(\alpha, G_0)$.

Première étape : Traitons le cas simple où $\mathbf{e} = \emptyset$. En absence d'observations préalables, quel degré de croyance peut on attribuer au fait qu'un certain $x \in \mathcal{D}$ soit observé ?

En considérant la partition de $\mathcal{D}$ formée du singleton $A_1 = \{x\}$ et de son complémentaire $A_2 = \mathcal{D} \setminus \{x\}$, on peut se ramener au cas où G est une multinomiale sur un ensemble à deux éléments $X = \{1, 2\}$ dont les paramètres (θ_1, θ_2) sont engendrés par une distribution de Dirichlet $Dir\left(\alpha G_0(A_1), \alpha G_0(A_2)\right)$.

La probabilité d'apparition de 1 vaut :

$$P(1) = \frac{B\left(1 + \alpha G_0(A_1), \alpha G_0(A_2)\right)}{B\left(\alpha G_0(A_1), \alpha G_0(A_2)\right)} = \frac{\alpha G_0(A_1)}{\alpha G_0(A_1) + \alpha G_0(A_2)}$$

(d'après les propriétés de la fonction B).

Or comme A_1 et A_2 sont complémentaires, $G_0(A_1) + G_0(A_2) = 1$ et donc $P(1) = G_0(A_1)$.

Conclusion : lorsque $\mathbf{e} = \emptyset$, $P(x|\mathbf{e}) = G_0(x)$ ce qui nous confirme que la distribution de base G_0 joue le rôle d'un a priori.

Deuxième étape : Traitons maintenant le cas général où le support de $\mathbf{e}$ est un ensemble $\{x_1, \ldots, x_r\}$ tel que l'observation x_i apparait $n_{\mathbf{e}}(x_i)$ fois dans $\mathbf{e}$. Pour se ramener à une distribution de Dirichlet, considérons la partition finie de $\mathcal{D}$ où $A_i = \{x_i\}$ pour $i \geq 1$, et $A_0 = \mathcal{D} \setminus \mathbf{e}$. On est alors ramené au cas d'une multinomiale sur un ensemble fini $X = \{0, \ldots, r\}$ dont les paramètres sont engendrés par $Dir(\alpha G_0(A_0), \ldots, \alpha G_0(A_r))$, où tous les éléments de $\mathcal{D}$ qui n'ont pas encore été observés sont représentés par l'indice 0, et où les observations ont pour support $\bar{\mathbf{e}} = \{1, \ldots, r\}$, l'observation i apparaissant $n_{\bar{\mathbf{e}}}(i)$ fois.

Soit $x \in \mathcal{D}$, calculons la probabilité a posteriori $P(x|\mathbf{e})$.

— **Premier cas :** Si $x = x_i \in \mathbf{e}$. Alors cette probabilité correspond à la probabilité a posteriori que l'indice i apparaisse. D'après les propriétés de la distribution de Dirichlet, cette probabilité vaut $\dfrac{n_{\bar{\mathbf{e}}}(i) + \alpha G_0(A_i)}{\sum_{j=1}^{r} n_{\bar{\mathbf{e}}}(j) + \alpha \sum_{j=0}^{r} G_0(A_j)}$ et comme $\{A_j\}_{j=0}^{r}$ forme une partition, on a donc :

$$P(x|\mathbf{e}) = \frac{n_{\bar{\mathbf{e}}}(i) + \alpha G_0(A_i)}{\alpha + \sum_{j=1}^{r} n_{\bar{\mathbf{e}}}(j)} = \frac{n_{\mathbf{e}}(x_i) + \alpha G_0(A_i)}{\alpha + \sum_{j=1}^{r} n_{\mathbf{e}}(x_j)}$$

— **Deuxième cas :** Si $x \notin \mathbf{e}$ alors $P(x|\mathbf{e})$ correspond à la probabilité a posteriori que l'indice 0 apparaisse, et vaut donc :

$$P(x|\mathbf{e}) = \frac{\alpha G_0(A_0)}{\alpha + \sum_{j=1}^{r} n_{\bar{\mathbf{e}}}(j)} = \frac{\alpha G_0(A_0)}{\alpha + \sum_{j=1}^{r} n_{\mathbf{e}}(x_j)}$$

De ces remarques, on peut déduire la procédure suivante pour échantillonner la distribution a posteriori $P(x|\mathbf{e})$:

— $\forall 1 \leq i \leq r$, retourner x_i avec une probabilité proportionnelle à $n_{\mathbf{e}}(x_i)$.
— sinon, retourner un échantillon de G_0 avec une probabilité proportionnelle à α.

On retrouve un comportement similaire aux distributions de Dirichlet, à savoir que : lorsque les observations sont en petit nombre ($\sum_i n_{\mathbf{e}}(x_i) \ll \alpha$), la deuxième option va l'emporter et on va avoir recours à l'a priori pour modéliser $P(x|\mathbf{e})$. En revanche, lorsque les observations sont en grand nombre ($\sum_i n_{\mathbf{e}}(x_i) \gg \alpha$), c'est aux fréquences des observations qu'on va avoir recours.

2.2.3 Le processus de restaurant chinois

2.2.3.1 Définition

Le processus de restaurant est un processus probabiliste modélisant les observations qui précèdent.

Definition 8. (Processus de restaurant chinois)

— Soit un restaurant où K tables sont ouvertes. Il n'y a pas de limite au nombre de tables que le restaurant peut accueillir. A chaque table $k \in \{1, \ldots, K\}$, $n_k \geq 1$ clients sont assis, soit un nombre total de clients $N = \sum_{k=1}^{K} n_k$. De plus, chaque table k possède une certaine étiquette $l_k \in \mathcal{D}$, où $\mathcal{D}$ est un certain ensemble éventuellement infini d'étiquettes. Au fur et à mesure que des clients entrent dans le restaurant, la configuration du restaurant change, donc nous utiliserons une variable de temps $t \in \mathbb{N}$ qui désignera l'état du restaurant après l'entrée du t-ème client. A l'état initial, le restaurant est vide : $K^{t=0} = 0$ et $N^{t=0} = 0$. Puis des clients entrent au fur et à mesure et s'assoient à une table selon deux alternatives :
— *Première alternative* : $\forall 1 \leq k \leq K$, le $(t+1)$-ème client est assis à la k-ème table avec une probabilité proportionnelle à n_k^t. L'état du restaurant au temps $t+1$ correspond donc à :

$$\begin{cases} K^{t+1} = K^t \\ N^{t+1} = N^t + 1 \\ n_k^{t+1} = n_k^t + 1 \quad & \forall j \neq k, n_j^{t+1} = n_j^t \\ \forall j, l_j^{t+1} = l_j^t \end{cases}$$

— *Seconde alternative* : Avec une probabilité résiduelle proportionnelle à $\alpha > 0$, une nouvelle table est ouverte ($K_{t+1} = K^t + 1$). Cett nouvelle table est étiquetée par un échantillon d'une distribution $G_0 \in Prob(\mathcal{D})$. Le $(t+1)$-ème client s'assoit à cette nouvelle table. L'état $t+1$ correspond donc à :

$$\begin{cases} K^{t+1} = K^t + 1 \\ N^{t+1} = N^t + 1 \\ n_{K^{t+1}}^{t+1} = 1 \quad & \forall 1 \leq j \leq K^t, n_j^{t+1} = n_j^t \\ l_{K^{t+1}} \sim G_0 \quad & \forall 1 \leq j \leq K^t, l_j^{t+1} = l_j^t \end{cases}$$

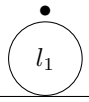
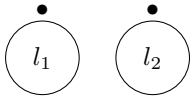
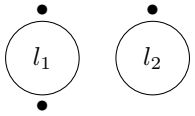
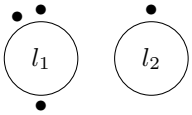
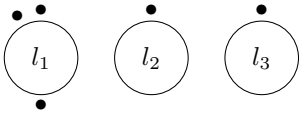
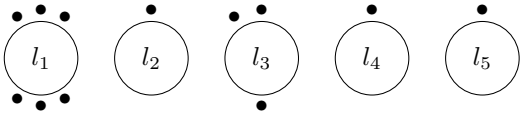
$t = 1$		Un premier client entre. Une table est ouverte et étiquetée par un échantillon $l_1 \sim G_0$.
$t = 2$		Un deuxième entre. Deux choix : l'asseoir à la première table avec une probabilité $\frac{1}{1+\alpha}$ ou ouvrir une nouvelle table avec une probabilité $\frac{\alpha}{1+\alpha}$. Le deuxième choix est retenu et la nouvelle table est étiquetée avec un échantillon $l_2 \sim G_0$.
$t = 3$		Un troisième entre. Probabilités de $\frac{1}{2+\alpha}$ pour chaque table, et $\frac{\alpha}{2+\alpha}$ pour l'ouverture d'une nouvelle table. La table 1 est choisie.
$t = 4$		Un quatrième entre. Probabilités : $\frac{2}{3+\alpha}$ pour la table 1, $\frac{1}{3+\alpha}$ pour la table 2, $\frac{\alpha}{3+\alpha}$ pour une nouvelle table. Le premier choix est retenu, en effet la table 1 commence à acquérir de plus en plus de poids.
$t = 5$		Néanmoins à tout moment, bien qu'avec une probabilité rapidement décroissante, l'ouverture de nouvelles tables reste possible et il n'y a pas de limite au nombre potentiel de tables.
$t = 12$		Au fur et à mesure que le processus se déroule, un petit nombre de tables concentre un grand nombre de clients, et un grand nombre de tables concentre un petit nombre de clients.

FIGURE 2.1 – Exemple de déroulement d'un $CRP(\alpha, G_0)$

Nous désignerons par $CRP(\alpha, G_0)$ le processus aléatoire décrit ci-dessus. Un échantillon de $CRP(\alpha, G_0)$ est donc une suite de quadruplets $\left(K^t, N^t, (n_k^t)_{1\leq k\leq K^t}, (l_k^t)_{1\leq k\leq K^t}\right)_{t\in\mathbb{N}}$. Notons que évidemment, $N^t = t$.

2.2.3.2 Comportement asymptotique

Nous présentons d'abord deux propriétés du comportement asymptotique d'un processus de restaurant chinois.

Propriété 9. ***(Croissance du nombre de tables)*** *Le nombre de tables dans un $CRP(\alpha, G_0)$ évolue logarithmiquement au cours du temps :*

$$\mathbb{E}[K^t] \underset{t\to\infty}{\sim} \alpha \log t$$

Preuve. Pour tout t, $K^t - K^{t-1}$ vaut 1 lorsqu'une nouvelle table a été crée au temps t (donc avec probabilité $\frac{\alpha}{t-1+\alpha}$), et vaut 0 sinon.
donc, $\mathbb{E}[K^t - K^{t-1}] = \frac{\alpha}{t-1+\alpha}$.

Par linéarité de l'espérance,
$\mathbb{E}[K^t] = \underbrace{\mathbb{E}[K^1]}_{=1} + \sum_{s=2}^{t} \mathbb{E}[K^s - K^{s-1}]$
$\mathbb{E}[K^t] = 1 + \alpha \sum_{s=2}^{t} \frac{1}{s-1+\alpha}$
or $\frac{1}{s-1+\alpha} \underset{s\to\infty}{\sim} \frac{1}{s}$
donc $\mathbb{E}[K^t] \underset{t\to\infty}{\sim} \alpha \sum_{s=1}^{t} \frac{1}{s}$
et il est connu que la sommes partielles de la série harmonique sont équivalentes à log.
donc, $\mathbb{E}[K^t] \underset{t\to\infty}{\sim} \alpha \log t$ □

Propriété 10. ***(Croissance du nombre de clients à une table)*** *La population des tables évolue en moyenne linéairement en fonction du temps. Si à un temps t_0, les nombres de clients à chaque table, c'est-à-dire les variables $\left(n_k^{t_0}\right)_{1\leq k\leq K^{t_0}}$, sont connus, alors :*

$$\forall t > t_0, \forall k_0 \in \{1, \dots, K^{t_0}\}, \mathbb{E}\left[n_{k_0}^t | (n_k^{t_0})_{k\in\{1,\dots,K^{t_0}\}}\right] = \frac{t+\alpha}{t_0+\alpha} n_{k_0}^{t_0}$$

Preuve. Considérons une table k_0. Soit $q^t = \left(n_{k_0}^t, \alpha + \sum_{j\neq k_0} n_j^t\right)$.
Avec une probabilité $\frac{n_{k_0}^t}{t+\alpha}$, au temps $t+1$, le client entrant s'assoiera à la table k_0, auquel cas $q^{t+1} = q^t + (1,0)$. Sinon, avec une probabilité $\frac{\alpha+\sum_{j\neq k_0} n_j^t}{t+\alpha}$, il s'assoiera à une autre table, auquel cas $q^{t+1} = q^t + (0,1)$.
Donc, $\mathbb{E}[q^{t+1}|q^t] = \frac{n_{k_0}^t}{t+\alpha}\left((1,0) + q^t\right) + \frac{\alpha + \sum_{j\neq k_0} n_j^t}{t+\alpha}\left((0,1) + q^t\right)$
$\mathbb{E}[q^{t+1}|q^t] = q^t + \frac{1}{t+\alpha}\left(n_{k_0}^t, \alpha + \sum_{j\neq k_0} n_j^t\right)$
$\mathbb{E}[q^{t+1}|q^t] = \frac{t+1+\alpha}{t+\alpha} q^t$
Par ailleurs,

$$\begin{aligned}
\mathbb{E}[q^{t+s}|q^t] &= \sum_{q^{t+s-1}} \mathbb{E}[q^{t+s}|q^{t+s-1}] \cdot P(q^{t+s-1}|q^t) \\
&= \sum_{q^{t+s-1}} \underbrace{\frac{t+s+\alpha}{t+s-1+\alpha} q^{t+s-1}}_{\text{d'après ce qui précède}} \cdot P(q^{t+s-1}|q^t) \\
&= \frac{t+s+\alpha}{t+s-1+\alpha} \sum_{q^{t+s-1}} \cdot P(q^{t+s-1}|q^t) \\
&= \frac{t+s+\alpha}{t+s-1+\alpha} \mathbb{E}[q^{t+s-1}|q^t]
\end{aligned}$$

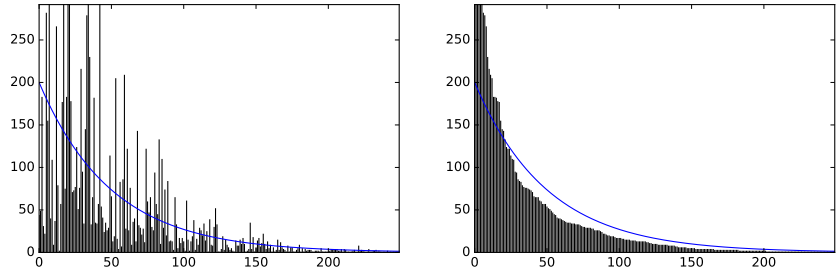


FIGURE 2.2 – Représentation d'un échantillon de $CRP(\alpha = 50)$ au bout de 10000 itérations. En haut, les tables sont disposées le long de l'axe des abscisses selon leur ordre d'apparition, et la hauteur de chaque table correspond à son nombre de clients. En bas, les tables ont été ordonnées en nombre décroissant de clients. La courbe bleue est la fonction $e^{-k/\alpha}$ renormalisée avec une constante appropriée.

donc, par récurrence :

$$\begin{aligned}\mathbb{E}[q^{t+s}|q^t] &= \left(\prod_{r=0}^{s-1} \frac{t+r+1+\alpha}{t+r+\alpha}\right) q^t \\ &= \frac{t+s+\alpha}{t+\alpha} q^t\end{aligned}$$

puisque le premier membre de q^t vaut $n^t_{k_0}$, le résultat voulu en découle. □

Ces deux résultats permettent de se faire une idée de l'allure de la répartition asymptotique des clients. Puisque le nombre de tables évolue logarithmiquement en fonction du temps (propriété 9), si nous notons T_k le temps de création de la k-ème table, on a : $K^{T_k} \underset{k\to\infty}{\sim} \alpha \log T_k$ et donc, $T_k = \Theta\left(e^{\frac{1}{\alpha}K^{T_k}}\right)$. Comme $K^{T_k} = k$, $T_k = \Theta\left(e^{k/\alpha}\right)$, c'est-à-dire que les temps d'apparition des tables suivent une progression d'allure exponentielle.

De plus, au moment T_k de la création de la k-ème table, on a $n_k^{T_k} = 1$. D'après la propriété 10, on sait que l'espérance du nombre de clients à la k-ème table à un temps ultérieur $t > T_k$ (conditionnellement à l'état du restaurant au temps T_k) vaudra donc $\frac{t+\alpha}{T_k+\alpha}$. Autrement, dit, asymptotiquement, la part des clients qui seront alloués à la table k vaudra en moyenne $\frac{1}{t}\frac{t+\alpha}{T_k+\alpha} \underset{t\to\infty}{\sim} \frac{1}{T_k+\alpha} = \Theta_{k\to\infty}\left(e^{-k/\alpha}\right)$.

Les clients auront donc tendance à s'agglutiner autour d'un nombre restreint de tables qui en général ont été créées au début du processus. Puisque la proportion de clients alloués à la k-ème table suit une loi exponentielle de paramètre $\frac{1}{\alpha}$, et puisque la médiane d'une telle loi vaut $\alpha \log 2$, on peut en déduire l'interprétation heuristique suivante du paramètre de force α : *asymptotiquement,*

la moitié des clients sera concentrée sur environ $\alpha \log 2$ *tables, et l'autre moitié sur environ* $\alpha(\log t - \log 2)$ *tables.*

2.2.3.3 Lien avec les processus de Dirichlet

Le processus de restaurant chinois est une métaphore utile pour implémenter un processus de Dirichlet. Supposons que nous disposons d'un jeu de données $\mathbf{e} \subset \{x_1, \ldots, x_r\}$ sur un ensemble $\mathcal{D}$ où l'observation x_i apparaît $n_{\mathbf{e}}(x_i)$ fois et que nous souhaitions modéliser ce jeu de données par une distribution G soumise à un a priori de Dirichlet $DP(\alpha, G_0)$. Rappelons que le cadre bayésien consiste à adopter le point de vue suivant : l'observation de $\mathbf{e}$ modifie le degré de croyance qu'on peut attribuer aux différentes options possibles pour G, ce qui se traduit par une distribution *a posteriori* $P(G|\mathbf{e})$. On peut alors "apprendre" du jeu de données $\mathbf{e}$ un modèle prédictif en faisant une moyenne pondérée de toutes les valeurs possibles de G selon les poids donnés par la distribution *a posteriori* $P(G|\mathbf{e})$. Autrement dit, le modèle prédictif prescrit par l'approche bayésienne s'écrirait :

$$G_{\text{Bayes}} = \int_{G \in Prob(\mathcal{D})} G \cdot P(G|\mathbf{e})$$

Calculer cette moyenne pose deux problèmes computationnels lorsque le ensemble $\mathcal{D}$ est infini (ce qui sera le cas en traitement de la langue, où $\mathcal{D}$ représentera par exemple l'ensemble des chaînes de caractères sur un alphabet).

— Cela nécessiterait d'une part de pouvoir représenter une distribution sur un ensemble infini. Sur un ensemble fini, il suffirait d'avoir pour chaque évènement un paramètre correspondant à la probabilité de cet évènement pour représenter de manière univoque une distribution. Mais cette représentation ne peut pas se généraliser à un ensemble infini car elle nécessiterait de stocker un nombre infini de paramètres.

— Il faudrait d'autre part calculer une intégrale sur $Prob(\mathcal{D})$ qui est un espace de dimension infinie, ce qui nécessiterait des méthodes approchées (de Monte-Carlo par exemple).

Le CRP est défini de manière à ce que pour une configuration donnée du restaurant à un temps t : $(K^t, N^t, (n_k^t)_{1 \leq k \leq K^t}, (l_k^t)_{1 \leq k \leq K^t})$, la probabilité d'assigner une étiquette l au prochain client [1] vaut : $\frac{\sum_{k|l_k=l} n_k^t + \alpha G_0(l)}{\alpha + N^t} = \frac{n_l^t + \alpha G_0(l)}{\alpha + N^t}$ (où n_l^t désigne le nombre de clients portant une étiquette l au temps t). Or, nous avons vu à la section 2.2.2 que conditionnellement à un ensemble d'observations $\mathbf{e}$, la probabilité a posteriori d'un évènement x pour une distribution soumise à un a priori de Dirichlet vaut $P(x|\mathbf{e}) = \frac{n_{\mathbf{e}}(x) + \alpha G_0(x)}{\alpha + \sum_{y \in \mathbf{e}} n_{\mathbf{e}}(y)}$. Il en découle la relation suivante entre CRP et processus de Dirichlet :

1. Par économie de langage, nous dirons qu'une étiquette l est assignée à un client pour dire qu'un client s'asseoit à une table d'étiquette l. Nous dirons qu'un client porte une étiquette l pour dire qu'il est assis à une table d'étiquette l.

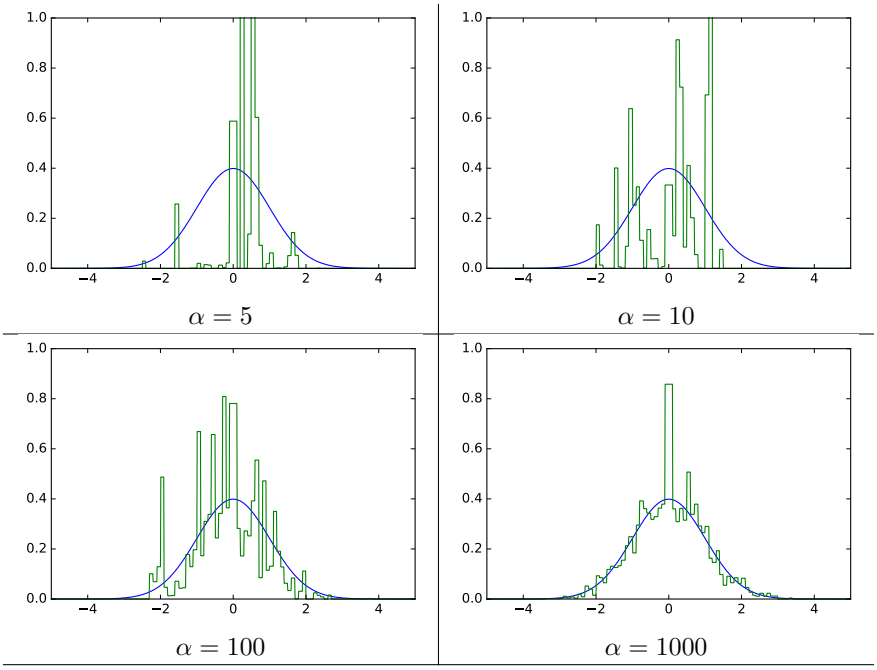


FIGURE 2.3 – Représentation des densités d'échantillons de processus de Dirichlet pour différentes valeurs de α et pour une distribution de base $\mathcal{N}(0, 1)$. La densité de la gaussienne normale est représentée en bleu.

Propriété 11. *Soit G_0 une distribution de base sur un ensemble $\mathcal{D}$. Soit $G \sim DP(\alpha, G_0)$ une distribution aléatoire et $X_1, X_2, \ldots, X_N, X$ des variables aléatoires indépendantes distribuées selon G. Soit $\boldsymbol{e} = (x_1, \ldots, x_N)$ un N-uplet d'éléments de $\mathcal{D}$ où $x \in \mathcal{D}$ apparaît $n_{\boldsymbol{e}}(x)$ fois. Alors la probabilité $P(X = x | X_1 = x_1 \ldots X_N = x_N)$ correspond à la probabilité que le $(N+1)$-ème client reçoive l'étiquette x dans un $CRP(\alpha, G_0)$ où au temps N, N clients sont assis et où $\forall l \in \mathcal{D}$, $n_{\boldsymbol{e}}(l)$ clients portent l'étiquette l.*

Par ailleurs, le CRP offre aussi un moyen de produire des échantillons approximatifs de $DP(\alpha, G_0)$: la propriété suivante exprime le fait que les proportions de clients par étiquette convergent lorsque $t \to \infty$ vers un échantillon de $DP(\alpha, G_0)$.

Propriété 12. *Soit un processus de restaurant chinois $CRP(\alpha, G_0)$. Soit $G^t(l)$ la distribution (aléatoire) des clients par étiquette au temps t : $G^t(l) = n_l^t/t$. Alors pour tout l, lorsque $t \to \infty$, $G^t(l)$ converge presque sûrement vers une limite $G^\infty(l)$. G^∞ est une distribution aléatoire dont la loi est donnée par $DP(\alpha, G_0)$.*

Ainsi, les distributions générées par un DP ressemblent aux distributions asymptotiques des étiquettes dans un CRP. Nous avons vu auparavant que

lorsque $t \to \infty$ dans un CRP, il y aura d'une part un petit nombre de tables accueillant chacune un grand nombre de clients, d'autre part, un grand nombre de tables accueillant chacune un petit nombre de clients. Par conséquent, un petit nombre d'étiquettes (celles ayant été assignées aux tables peuplées) auront été assignées à un grand nombre de clients, et du reste, comme G_0 a été échantillonné un grand nombre de fois pour étiqueter le grand nombre de tables peu peuplées, la distribution des étiquettes sur les tables peu peuplées sera similaire à G_0. Cette description grossière de l'allure des distributions d'étiquettes lorsque $t \to \infty$ dans un CRP (c'est-à-dire des distributions engendrées par un DP) nous suggère que les distributions engendrées par un DP ressembleront à des perturbations *parcimonieuses*[2] de G_0.

Cette allure de perturbations parcimonieuses des échantillons de $DP(\alpha, G_0)$ peut être observée sur la figure 2.3, où nous voyons que α contrôle le degré de parcimonie des échantillons : plus α est grand, plus les échantillons ressembleront à la distribution de base. Plus α est petit, plus les échantillons ressembleront à un *Dirac*[3] dont le support est distribué selon G_0.

2.2.4 Processus de Pitman-Yor

Nous introduisons ces processus bayésiens non-paramétriques afin de les utiliser par la suite pour modéliser des distributions apparaissant dans des tâches de traitement des langues, comme les probabilités d'apparition d'un mot ou d'un morphème dans un contexte donné.

Or une célèbre loi empirique sur l'usage d'une mots d'une langue est la *loi de Zipf*, qui affirme que la fréquence d'apparition f_w d'un mot dans un texte est environ inversement proportionnelle à son rang r_w dans la liste des mots classés par fréquence décroissante : $f_w \propto r_w^{-s}$ avec $s \approx 1$. Nous souhaiterions par conséquent que les *a priori* définis par les processus bayésiens non-paramétriques tendent à favoriser des distributions de cette forme.

Qu'en est-il pour les processus de Dirichlet ? Nous avons montré à la section 2.2.3.2 qu'en moyenne, la part des clients alloués à la k-ème table d'un CRP décroît de manière exponentielle en fonction de k. Les distributions engendrées par un processus de Dirichlet ne suivront donc pas la loi de Zipf. Par conséquent, nous allons considérer une modification des processus de Dirichlet appelée *processus de Pitman-Yor* qui remédie à ce problème.

2.2.4.1 Definition

Definition 13. (Processus de Pitman-Yor) Considérons un processus de restaurant chinois modifié que nous désignerons par $CRP(\alpha, d, G_0)$, où la politique d'assignation de tables aux clients est modifiée de manière à dépendre d'un paramètre supplémentaire $d \in [0, 1[$ appelé *paramètre de décompte*. Dans

2. Une distribution est qualifiée de *parcimonieuse* lorsqu'une grande masse de probabilité sera assignée à un petit nombre d'évènements et une petite masse de probabilité à un grand nombre d'évènements.

3. Une distribution pour laquelle toute la masse est allouée à un singleton

ce processus modifié, le t-ème client s'asseoit à la k-ème table avec une probabilité proportionnelle à $n_k^{t-1} - d$, ou s'asseoit à une table nouvellement ouverte (et étiquetée avec un échantillon de G_0) avec une probabilité proportionnelle à $\alpha + K^{t-1}d$.

Soit $G^t(l) = n_l^t/t$. Il s'agit d'une distribution aléatoire sur les étiquettes, qui converge lorsque $t \to \infty$ vers une distribution aléatoire $G^\infty(l)$. Le *processus de Pitman-Yor* (de paramètre de force α, de paramètre de décompte d et de distribution de base G_0) noté $PYP(\alpha, d, G_0)$ est défini comme la loi de G^∞.

Ainsi, pour chaque table k, une masse de probabilité constante égale à d est retirée à la probabilité de peupler k. De plus, la masse de probabilité totale retirée aux tables (valant $K^t d$ au temps t) est ajoutée à la probabilité d'ouvrir une nouvelle table. Les processus de Pitman-Yor favorisent donc l'ouverture de nouvelles tables par rapport aux processus de Dirichlet, et ce d'autant plus que d est proche de 1. En particulier, dans la construction même du $PYP(\alpha, d, G_0)$, on peut remarquer les deux cas limites suivants :

— Lorsque $d = 0$, aucune masse de probabilité n'est retirée aux tables, et donc on obtient exactement le même comportement que le CRP utilisé pour représenter les processus de Dirichlet.

— Lorsque $d \approx 1$, aucune masse de probabilité ne sera allouée aux tables, et le processus ne consistera qu'en la création de nouvelles tables, c'est-à-dire en des échantillonages successifs de G_0. La distribution asymptotique obtenue sera donc G_0.

2.2.4.2 Loi de Zipf

De même que pour les CRP associés aux processus de Dirichlet, nous pouvons calculer un équivalent asymptotique du nombre de tables lorsque $t \to \infty$, ce qui nous permettra d'en déduire la vitesse de décroissance des distributions engendrées par un PYP.

Propriété 14. ***(Croissance du nombre de tables)*** *Le nombre de tables dans un $CRP(\alpha, d, G_0)$ est asymptotiquement équivalent à une puissance de d :*

$$\mathbb{E}[K^t] \underset{t\to\infty}{\sim} \left(1 + \frac{\alpha}{d}\right) t^d$$

Démonstration. $\mathbb{E}[K^t - K^{t-1}]$ est égal à la probabilité qu'une nouvelle table soit ouverte au temps t, c'est-à-dire à : $\dfrac{\alpha + \mathbb{E}[K^{t-1}]d}{\alpha + t - 1}$. donc

$$\mathbb{E}[K^t] - \mathbb{E}[K^{t-1}] = \frac{\left(\mathbb{E}[K^{t-1}] + \frac{\alpha}{d}\right) d}{\alpha + t - 1}$$

en posant $u_t = \mathbb{E}[K^t] + \frac{\alpha}{d}$, on a alors :

$$u_t - u_{t-1} = \frac{u_{t-1} d}{\alpha + t - 1}$$
$$\frac{u_t}{u_{t-1}} = 1 + \frac{d}{\alpha + t - 1}$$

et donc puisque $u_1 = \mathbb{E}[K^1] + \frac{\alpha}{d} = 1 + \frac{\alpha}{d}$,

$$\begin{aligned} u_t &= \left(1 + \frac{\alpha}{d}\right) \prod_{s=2}^{t} \frac{u_s}{u_{s-1}} \\ &= \left(1 + \frac{\alpha}{d}\right) \prod_{s=2}^{t} \left(1 + \frac{d}{\alpha + t - 1}\right) \\ &= \left(1 + \frac{\alpha}{d}\right) \prod_{s=1}^{t-1} \frac{\alpha + t + d}{\alpha + t} \\ &= \left(1 + \frac{\alpha}{d}\right) \frac{\Gamma(\alpha + t + d)}{\Gamma(\alpha + t)} \end{aligned}$$

On peut obtenir un équivalent de cette quantité en utilisant la formule de Stirling :

$$\begin{aligned} u_t &\underset{t\to\infty}{\sim} \left(1 + \frac{\alpha}{d}\right) \frac{(\alpha + t + d)^{\alpha + t + d - \frac{1}{2}} e^{-(\alpha + t + d)}}{(\alpha + t)^{\alpha + t - \frac{1}{2}} e^{-(\alpha + t)}} \\ &\underset{t\to\infty}{\sim} \left(1 + \frac{\alpha}{d}\right) \left(1 + \frac{d}{\alpha + t}\right)^{\alpha + t - \frac{1}{2}} (\alpha + t + d)^d e^{-d} \\ &\underset{t\to\infty}{\sim} \left(1 + \frac{\alpha}{d}\right) t^d e^{-d} \exp\left(\left(\alpha + t - \frac{1}{2}\right) \log\left(1 + \frac{d}{\alpha + t}\right)\right) \\ &\underset{t\to\infty}{\sim} \left(1 + \frac{\alpha}{d}\right) t^d e^{-d} \exp\left(\left(\alpha + t - \frac{1}{2}\right) \left(\frac{d}{\alpha + t} + o\left(\frac{1}{t}\right)\right)\right) \\ &\underset{t\to\infty}{\sim} \left(1 + \frac{\alpha}{d}\right) t^d e^{-d} e^{d + o(1)} \\ &\underset{t\to\infty}{\sim} \left(1 + \frac{\alpha}{d}\right) t^d \end{aligned}$$

$\mathbb{E}[K^t] = u_t - \frac{\alpha}{d}$ et $u_t \underset{t\to\infty}{\longrightarrow} \infty$ donc $\mathbb{E}[K^t] \underset{t\to\infty}{\sim} \left(1 + \frac{\alpha}{d}\right) t^d$.

□

On peut en déduire la forme des échantillons de $PYP(\alpha, d, G_0)$ par un raisonnement analogue à celui de la section 2.2.3.2. L'équivalent précédent permet

d'affirmer que pour un t élevé, il y aura environ $\left(1+\frac{\alpha}{d}\right)t^d$ tables dans le restaurant. En notant T_k le temps d'apparition de la k-ème table, on aura alors

$$T_k \underset{k\to\infty}{\sim} \left(1+\frac{\alpha}{d}\right)^{-1/d} k^{1/d}$$

.

De plus, de même que pour les processus de Dirichlet, on peut montrer que conditionnellement à l'état du restaurant donné à un temps t_1, l'espérance du nombre de clients à la table k à un temps ultérieur $t_2 > t_1$ vérifie la condition suivante :

$$\mathbb{E}\left[\left.\frac{n_k^{t_2}-d}{t_2+\alpha}\right| n_k^{t_1}\right] = \frac{n_k^{t_1}-d}{t_1+\alpha}$$

En remplaçant t_1 par T_k et t_2 par un temps $t \gg T_k$, et sachant que $n_k^{T_k} = 1$, on obtient :

$$\mathbb{E}\left[\left.\frac{n_k^{t}-d}{t+\alpha}\right| n_k^{T_k}\right] = \frac{1-d}{T_k+\alpha}$$

ce qui permet de montrer que, pour $k \to \infty$ et $t \gg T_k$:

$$\frac{n_k^t}{t} \simeq k^{-1/d}$$

2.3 Un algorithme de segmentation

Comme illustration des processus de Pitman-Yor en traitement des langues, nous présentons un premier algorithme de segmentation morphologique.

Quelques définitions et notations préliminaires :

- Un *mot* est une chaîne de caractères $w = c_1 \dots c_s$ où les c_i sont des caractères d'un alphabet donné. Nous notons $|w| = s$ la longueur du mot w.
- Si $w = c_1 \dots c_s$ est un mot, nous notons w_i^j la sous-chaîne de caractères $c_{i+1}c_{i+2}\dots c_j$. On convient que si $i = j$, w_i^j est une chaîne de caractères vide.
- Si w est un mot, nous appelons *segmentation* de w une liste d'entiers $\mathbf{s} = (s_0, s_1, \dots s_r)$ telle que $0 = s_0 < s_1 < s_2 < \dots < s_r = |w|$. Nous notons $\mathrm{Seg}(w)$ l'ensemble des segmentations de w.

Un algorithme de segmentation morphologique est spécifié de la manière suivante :

- En entrée, nous disposons d'une liste de mots (un *lexique*) $L = (w_1, \dots w_{|L|})$.
- En sortie, nous souhaite obtenir :
  - Une ensemble de *morphèmes* $M = \{\mu_1, \dots, \mu_{|M|}\}$, un morphème étant une chaîne de caractères.
  - Pour chaque mot w du lexique, une segmentation $\mathbf{s} = (s_0, \dots, s_r)$ tel que chaque segment $w_{s_i}^{s_{i+1}}$ soit un morphème de M.

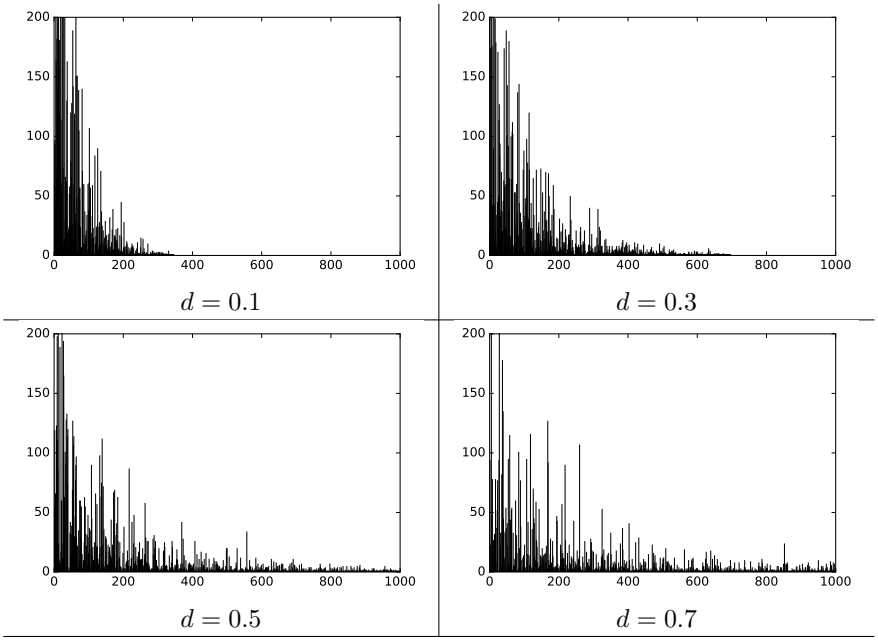


FIGURE 2.4 – Représentation des distributions de clients par table générées par un processus de Pitman-Yor pour $\alpha = 50$ et différentes valeurs de d.

2.3.1 Modèle génératif

2.3.1.1 Scénario

L'algorithme s'appuie sur un modèle probabiliste génératif, c'est-à-dire : une histoire expliquant comment le lexique observé est venu à apparaître, en termes d'un scénario où un certain nombre de distributions de probabilités ont été échantillonnées selon une certaine séquence.

Le modèle génératif que nous proposons a la structure suivante :

1. En première étape, une distribution sur des chaînes de caractères G est échantillonnée à partir d'un processus de Pitman-Yor $PYP(\alpha, d, G_0)$ où G_0 est une certaine distribution de base sur des chaînes de caractères, reflétant les séquences de caractères typiques apparaissant dans la langue traitée. Dans un premier temps, nous choisirons comme distribution de base G_0 un modèle trigramme de caractères appris sur le lexique. Dans le scénario proposé, le rôle de G sera de représenter une distribution sur les morphèmes de la langue. Initialement, G sera une distribution relativement agnostique favorisant des chaînes de caractères typiques, mais le but est qu'au cours de l'inférence, la masse de G vienne à se concentrer sur un ensemble restreint de chaînes de caractères qui correspondront aux morphèmes inférés par le modèle.

2. Ensuite, un lexique et des segmentations sont engendrés en échantillonnant tour à tour des mots et leur segmentation. Chaque mot et sa segmentation sont échantillonnés de la manière suivante :
   (a) A l'étape $k = 1$, un premier morphème est engendré en échantillonnant G : $\mu_1 \sim G$, puis on passe à l'étape 2.
   (b) A l'étape $k > 1$, avec une probabilité $p_{\text{arrêt}}$, la génération du mot se termine. Sinon, avec une probabilité $1 - p_{\text{arrêt}}$ un morphème suivant est échantillonné : $\mu_k \sim G$, puis on passe à l'étape $k + 1$.
   (c) La génération s'arrête presque sûrement après un nombre fini r d'étapes. Le mot obtenu est la concaténation des morphèmes qui ont été engendrés : $w = \mu_1 \dots \mu_r$. On obtient aussi une segmentation $\mathbf{s} = (s_0, \dots, s_r)$ de ce mot donnée par : $w_{s_{k-1}}^{s_k} = \mu_k$.

$G \sim PYP(\alpha, d, G_0)$ $\forall w \in L, w = \mu_1 \dots \mu_r$ avec $\mu_k \sim G$ et $r \sim \text{Geom}(1 - p_{\text{arrêt}})$
$\mathbb{P}(L, S, G) = \mathbb{P}(L, S\|G) \underbrace{\mathbb{P}(G)}_{\text{donnée par le PYP}}$ $\mathbb{P}(L, S\|G) = \prod_{i=1}^{\|L\|} \mathbb{P}(w_i, \mathbf{s}_i\|G)$ Pour tout mot $w \in L$ et pour toute segmentation $\mathbf{s} = (s_0, \dots, s_r)$ de ce mot : $\mathbb{P}(w, \mathbf{s}\|G) = G(w_{s_0}^{s_1})(1 - p_{\text{arrêt}}) \cdot G(w_{s_1}^{s_2})(1 - p_{\text{arrêt}}) \cdot \dots \cdot G(w_{s_{r-1}}^{s_r}) p_{\text{arrêt}}$

FIGURE 2.5 – Un modèle génératif du lexique et de ses segmentations

Ce scénario génératif définit une distribution jointe $\mathbb{P}(L, S, G)$ sur toutes les variables du problème, donnée dans la figure 2.5.

2.3.1.2 Justification

On pourrait douter de l'adéquation de ce scénario pour modéliser la morphologie d'un lexique : comment se fait-il qu'en n'imposant aucune autre restriction sur la distribution G que le fait d'être engendré par un PYP, on puisse attendre que sa masse se concentre sur des unités linguistiquement pertinentes ?

Une idée de justification est la suivante. Comme on l'a vu plus haut, les PYP engendrent des distributions où une grande masse de probabilité est concentrée sur un petit nombre d'éléments. Ainsi, imposer que G soit engendrée par un PYP entraîne que lors de l'inférence, le modèle aura tendance à converger vers des G qui permettent d'expliquer un grand nombre de formes de mots avec un petit nombre de chaînes de caractères, c'est-à-dire que G capturera des unités qui sont réutilisées dans beaucoup de mots, ce qui est le cas par exemple des morphèmes qui apparaissent comme préfixes ou comme suffixes. Parallèlement, les PYP autorisent des distributions à "queue lourde", ce qui permettra aussi à G de prendre en compte des classes de morphèmes qui présentent une plus grande variété, comme par exemple les racines.

2.3.2 Inférence

2.3.2.1 Déroulement

Etant donné un lexique, pour produire des segmentations et un lexique de morphème, on souhaite échantillonner la distribution $\mathbb{P}(S|L) = \mathbb{P}(\mathbf{s}_1, \ldots, \mathbf{s}_{|L|}|w_1, \ldots, w_{|L|})$ où :

— Les $w_1, \ldots, w_{|L|}$ sont les mots du lexique à segmenter.

— Pour tout $1 \leq i \leq |L|$, $\mathbf{s}_i$ est la segmentation du mot w_i.

Obtenir directement un échantillon de $\mathbb{P}(S|L)$ est intraitable : il s'agit d'une distribution jointe sur un grand nombre de variables, avec des corrélations complexes. Une stratégie qu'on peut adopter est la méthode d'*échantillonnage de Gibbs*. C'est une méthode approchée d'échantillonnage qui consiste à obtenir un échantillon d'une distribution jointe en échantillonnant alternativement chaque variable conditionnellement aux autres. Dans notre cas, un échantillonnage de Gibbs se déroulerait par exemple comme suit :

1. Initialiser les segmentations $\mathbf{s}_1, \ldots, \mathbf{s}_{|L|}$ de manière aléatoire.
2. Pour i de 1 à $|L|$, échantillonner $\mathbf{s}_i$ selon la distribution $\mathbb{P}(\mathbf{s}_i|\mathbf{s}_{-i}, L)$ (**Notation :** $\mathbf{s}_{-i}$ désigne l'ensemble des segmentations auquel on a ôté la i-ème segmentation, c'est-à-dire : $\mathbf{s}_1, \ldots, \mathbf{s}_{i-1}, \mathbf{s}_{i+1}, \ldots, \mathbf{s}_{|L|}$).
3. Répéter l'étape précédente jusqu'à convergence du modèle.

Il s'agit donc à présent de détailler comment on peut échantillonner une segmentation du mot w_i étant donné les mots w_{-i} et des segmentations de ces mots.

2.3.2.2 Probabilité d'une segmentation conditionnellement aux précédentes

Dans notre modèle génératif, la probabilité jointe d'un mot w et de sa segmentation $\mathbf{s}$ conditionnellement à G vaut :

$$\begin{aligned}\mathbb{P}(w, \mathbf{s}|G) &= G(w_{s_0}^{s_1})(1 - p_{\text{arrêt}}) \cdot G(w_{s_1}^{s_2})(1 - p_{\text{arrêt}}) \cdot \ldots \cdot G(w_{s_{r-1}}^{s_r})p_{\text{arrêt}} \\ &= (1 - p_{\text{arrêt}})^{r-1} p_{\text{arrêt}} \prod_{k=1}^{r} G(w_{s_{k-1}}^{s_k})\end{aligned}$$

Donc étant donné un ensemble de segmentations antérieures $\mathbf{s}_{-i}$ de mots w_{-i}, la probabilité jointe de w et de sa segmentation $\mathbf{s}$ peut s'obtenir en marginalisant G conditionnellement à $\mathbf{s}_{-i}$ et w_{-i} :

$$\begin{aligned}\mathbb{P}(w, \mathbf{s}|\mathbf{s}_{-i}, w_{-i}) &= \int_G \mathbb{P}(w, \mathbf{s}|G)\mathbb{P}(G|\mathbf{s}_{-i}, w_{-i}) \\ &= (1 - p_{\text{arrêt}})^{r-1} p_{\text{arrêt}} \int_G \prod_{k=1}^{r} G(w_{s_{k-1}}^{s_k})\mathbb{P}(G|\mathbf{s}_{-i}, w_{-i})\end{aligned}$$

Bien que les termes $G(w_{s_{k-1}}^{s_k})$ ne soient pas des variables aléatoires indépendantes (conditionnellement à $\mathbf{s}_{-i}, w_{-i}$), on peut se placer dans une approximation où ils le sont, quitte à appliquer une correction de Metropolis-Hastings par la suite afin d'échantillonner la vraie distribution. Cette approximation permet d'échanger produit et intégrale dans l'expression ci-dessus, et on obtient :

$$\mathbb{P}(w, \mathbf{s}|\mathbf{s}_{-i}, w_{-i}) \simeq (1 - p_{\text{arrêt}})^{r-1} p_{\text{arrêt}} \prod_{k=1}^{r} \int_G G(w_{s_{k-1}}^{s_k}) \mathbb{P}(G|\mathbf{s}_{-i}, w_{-i})$$

$\mathbb{P}(G|\mathbf{s}_{-i}, w_{-i})$ est entièrement déterminé par l'ensemble des segments qui ont été produits au cours des segmentations antérieures. Autrement dit, en notant $\mu_1, \ldots, \mu_t$ l'ensemble des morphèmes qui ont été découpés parmi les segmentations $\mathbf{s}_{-i}$ (avec répétition si le morphème a été découpé plusieurs fois), on a : $\mathbb{P}(G|\mathbf{s}_{-i}, w_{-i}) = \mathbb{P}(G|\mu_1, \ldots, \mu_t)$.

On a vu à la section 2.2.4 que la probabilité marginale $\int_G G(\mu)\mathbb{P}(G|\mu_1, \ldots, \mu_t)$ correspondant à la probabilité pour une distribution G engendrée par un $PYP(\alpha, d, G_0)$ d'émettre un prochain élément μ (sachant que les éléments précédemment émis par G sont $\mu_1, \ldots, \mu_t$) peut être calculée en maintenant un $CRP(\alpha, d, G_0)$

— contenant p clients : un client pour chaque μ_i, affecté à une table d'étiquette μ_i

— et en utilisant la formule suivante (en reprenant les notations de la section 2.2.3) :

$$\mathbb{P}(\mu|\mu_1, \ldots, \mu_t) = \int_G G(\mu)\mathbb{P}(G|\mu_1, \ldots, \mu_t) = \frac{(\alpha + K^t d)G_0(\mu) + \sum_{k=1}^{K^t} \mathbf{1}_{l_k=\mu}(n_k^t - d)}{\alpha + N^t}$$

Ainsi, étant donné cette formule et notre approximation où nous considérons indépendantes les probabilités d'émission des morphèmes de la segmentation qu'on souhaite échantillonner, on obtient :

$$\mathbb{P}(w, \mathbf{s}|\mathbf{s}_{-i}, w_{-i}) = (1-p_{\text{arrêt}})^{r-1} p_{\text{arrêt}} \prod_{j=1}^{r} \underbrace{\mathbb{P}(w_{s_{j-1}}^{s_j}|\mu_1, \ldots, \mu_t)}$$
$$= \frac{(\alpha + K^t d) \cdot G_0(w_{s_{j-1}}^{s_j}) + \sum_{k=1}^{K^t} \mathbf{1}_{l_k = w_{s_{j-1}}^{s_j}} (n_k^t - d)}{\alpha + N^t}$$

A présent, détaillons les aspects algorithmiques de l'inférence, à savoir :

— Etant donnée la distribution conditionnelle $\mathbb{P}(\mu|\mu_1, \ldots, \mu_t)$ sur les morphèmes, comment échantillonner une nouvelle segmentation.

— Comment maintenir le $CRP(\alpha, d, G_0)$ qui nous permet de calculer cette distribution conditionnelle.

2.3.2.3 Echantillonnage d'une segmentation

Une approche naïve pour échantillonner une segmentation serait d'énumérer toutes les segmentations possibles, de calculer chacune leur probabilité puis de normaliser la distribution sur les segmentations obtenues. Pour une segmentation d'un mot w, chaque interstice entre deux caractères peut être ou non un point de coupure, et il y a donc $2^{|w|-1}$ segmentations possibles : on voit donc que cette approche est computationnellement intraitable.

Une approche moins naïve est de traiter le problème par programmation dynamique.

Quelques notations :

— On note $\text{Seg}_{u,v}(w) \subset \text{Seg}(w)$ (pour $0 \leq u < v \leq |w|$) l'ensemble des segmentations du mot w où (u, v) apparaît comme segment, c'est-à-dire les segmentations de la forme $(\underbrace{0, *, \ldots, *, u}_{\in \text{Seg}(w_0^u)}, \underbrace{v, *, \ldots, *, |w|}_{\in \text{Seg}(w_v^{|w|})})$.

— Les segmentations $\mathbf{s} \in \text{Seg}_{u,v}(w)$ s'obtiennent toutes de manière unique comme une concaténation $\mathbf{s} = \boldsymbol{\sigma} \odot \boldsymbol{\sigma'}$ d'une segmentation $\boldsymbol{\sigma} \in \text{Seg}(w_0^u)$ et d'une segmentation $\boldsymbol{\sigma'} \in \text{Seg}(w_v^{|w|})$. On notera $\odot$ l'opérateur de concaténation, et on a alors : $\text{Seg}_{u,v}(w) = \text{Seg}(w_0^u) \odot \text{Seg}(w_v^{|w|})$.

— Si $\mathbf{s}$ est une segmentation d'un mot w, on notera ses points de coupure $s_0, \ldots, s_{|\mathbf{s}|}$, de sorte que : $0 = s_0 < s_1 < \ldots < s_{|\mathbf{s}|-1} < s_{|\mathbf{s}|} = |w|$.

— Pour ne pas alourdir les notations, dans cette section toutes les distributions sont implicitement conditionnées à l'ensemble des morphèmes $\mu_1, \ldots, \mu_t$ qui ont été obtenus dans l'ensemble des segmentations antérieures.

Probabilité de présence d'un segment Calculons la probabilité jointe de générer un mot w et sa segmentation $\mathbf{s}$ tel que (u, v) apparaisse comme segment dans $\mathbf{s}$:

$$\begin{aligned}
\sum_{\mathbf{s} \in \text{Seg}_{u,v}(w)} \mathbb{P}(\mathbf{s}, w) &= \sum_{\mathbf{s} \in \text{Seg}_{u,v}(w)} (1 - p_{\text{arrêt}})^{|\mathbf{s}|-1} p_{\text{arrêt}} \cdot \mathbb{P}(w_{s_0}^{s_1}) \ldots \mathbb{P}(w_{s_{|\mathbf{s}|-1}}^{s_{|\mathbf{s}|}}) \\
&= \sum_{\boldsymbol{\sigma} \in \text{Seg}(w_0^u)} \sum_{\boldsymbol{\sigma'} \in \text{Seg}(w_v^{|w|})} \underbrace{(1 - p_{\text{arrêt}})^{|\boldsymbol{\sigma}|} \cdot \mathbb{P}(w_{\sigma_0}^{\sigma_1}) \ldots \mathbb{P}(w_{\sigma_{|\boldsymbol{\sigma}|-1}}^{\sigma_{|\boldsymbol{\sigma}|}})}_{\text{proba d'émission des morphèmes de } \boldsymbol{\sigma}} \\
&\qquad \underbrace{\cdot (1 - p_{\text{arrêt}}) \mathbb{P}(w_u^v)}_{\text{proba d'émission du morphème au segment } (u,v)} \\
&\qquad \underbrace{\cdot (1 - p_{\text{arrêt}})^{|\boldsymbol{\sigma'}|-1} p_{\text{arrêt}} \cdot \mathbb{P}(w_{\sigma'_0}^{\sigma'^1}) \ldots \mathbb{P}(w_{\sigma'_{|\boldsymbol{\sigma'}|-1}}^{\sigma'_{|\boldsymbol{\sigma'}|}})}_{\text{proba d'émission des morphèmes de } \boldsymbol{\sigma'}}
\end{aligned}$$

$$\sum_{\mathbf{s}\in \mathrm{Seg}_{u,v}(w)} \mathbb{P}(\mathbf{s},w) = \left(\sum_{\boldsymbol{\sigma}\in \mathrm{Seg}(w_0^u)} \underbrace{(1-p_{\text{arrêt}})^{|\boldsymbol{\sigma}|-1} p_{\text{arrêt}} \cdot \mathbb{P}(w_{\sigma_0}^{\sigma_1}) \dots \mathbb{P}(w_{\sigma_{|\boldsymbol{\sigma}|-1}}^{\sigma_{|\boldsymbol{\sigma}|}})}_{=\mathbb{P}(\boldsymbol{\sigma},w_0^u)} \right) \frac{1-p_{\text{arrêt}}}{p_{\text{arrêt}}} \cdot (1-p_{\text{arrêt}}) \mathbb{P}(w_u^v) \cdot \left(\sum_{\boldsymbol{\sigma'}\in \mathrm{Seg}(w_v^{|w|})} \underbrace{(1-p_{\text{arrêt}})^{|\boldsymbol{\sigma'}|-1} p_{\text{arrêt}} \cdot \mathbb{P}(w_{\sigma'_0}^{\sigma'_1}) \dots \mathbb{P}(w_{\sigma'_{|\boldsymbol{\sigma'}|-1}}^{\sigma'_{|\boldsymbol{\sigma'}|}})}_{=\mathbb{P}(\boldsymbol{\sigma'},w_v^{|w|})} \right)$$

En notant :

$$\alpha(u) = \sum_{\boldsymbol{\sigma}\in \mathrm{Seg}(w_0^u)} \mathbb{P}(\boldsymbol{\sigma}, w_0^u) = \mathbb{P}(w_0^u)$$
$$\beta(v) = \sum_{\boldsymbol{\sigma'}\in \mathrm{Seg}(w_v^{|w|})} \mathbb{P}(\boldsymbol{\sigma'}, w_v^{|w|}) = \mathbb{P}(w_v^{|w|})$$

on a alors :

$$\mathbb{P}(\mathbf{s}\in \mathrm{Seg}_{u,v}(w), w) = \alpha(u) \cdot \frac{(1-p_{\text{arrêt}})^2}{p_{\text{arrêt}}} \cdot \beta(v)$$

Donc le mot w étant fixé, la probabilité conditionnelle que (u,v) apparaisse comme segment vaut :

$$\mathbb{P}\big(\mathbf{s}\in \mathrm{Seg}_{u,v}(w) | w\big) = \frac{\sum_{\mathbf{s}\in \mathrm{Seg}_{u,v}(w)} \mathbb{P}(\mathbf{s},w)}{\sum_{\mathbf{s}\in \mathrm{Seg}(w)} \mathbb{P}(\mathbf{s},w)} = \frac{\alpha(u) \cdot \frac{(1-p_{\text{arrêt}})^2}{p_{\text{arrêt}}} \cdot \beta(v)}{\alpha(|w|)}$$

Calcul des $\alpha(u)$ et $\beta(v)$ Pouvoir calculer efficacement $\alpha(u)$ et $\beta(v)$ nous permettra d'obtenir un algorithme efficace pour échantillonner une segmentation. Pour cela, on peut observer que les α et β vérifient une relation de récurrence.

Tout d'abord, remarquons qu'on peut partitionner les segmentations $\mathrm{Seg}(w_0^u)$ en les distinguant selon leur avant-dernier point de coupure, ce que l'on peut exprimer par la relation suivante :

$$\mathrm{Seg}(w_0^u) = \coprod_{k=0}^{u-1} \underbrace{\mathrm{Seg}(w_0^k) \odot (u)}_{\text{segmentations de la forme } (0,\dots,k,u)}$$

Ainsi la sommation sur $\boldsymbol{\sigma} \in \mathrm{Seg}(w_0^u)$ peut se réexprimer comme une double somme :

$$\alpha(u) = \sum_{\boldsymbol{\sigma}\in\mathrm{Seg}(w_0^u)} \mathbb{P}(\boldsymbol{\sigma}, w_0^u) = \sum_{k=0}^{u-1} \sum_{\boldsymbol{\sigma}\in\mathrm{Seg}(w_0^k)} \mathbb{P}(\boldsymbol{\sigma} \odot (u), w_0^u) \quad = \sum_{k=0}^{u-1} \sum_{\boldsymbol{\sigma}\in\mathrm{Seg}(w_0^k)} \mathbb{P}(\boldsymbol{\sigma}, w_0^k) \frac{1-p_{\text{arrêt}}}{p_{\text{arrêt}}} \mathbb{P}(w_k^u) p_{\text{arrêt}}$$

Le terme $\frac{1-p_{\text{arrêt}}}{p_{\text{arrêt}}}$ est une terme de "correction" pour remplacer le facteur $p_{\text{arrêt}}$ apparaissant dans $\mathbb{P}(\boldsymbol{\sigma}, w_0^k)$ par $(1 - p_{\text{arrêt}})$

$$\begin{aligned}\alpha(u) &= \sum_{k=0}^{u-1} \underbrace{\left(\sum_{\boldsymbol{\sigma}\in\mathrm{Seg}(w_0^k)} \mathbb{P}(\boldsymbol{\sigma}, w_0^k) \right)}_{=\alpha(k)} \cdot (1 - p_{\text{arrêt}}) \mathbb{P}(w_k^u) \\ &= (1 - p_{\text{arrêt}}) \sum_{k=0}^{u-1} \alpha(k) \cdot \mathbb{P}(w_k^u)\end{aligned}$$

en prenant comme condition de bord $\alpha(0) = \dfrac{p_{\text{arrêt}}}{1 - p_{\text{arrêt}}}$.

Un raisonnement similaire s'applique aux $\beta(v)$: on peut partitionner les segmentations $\mathrm{Seg}(w_v^{|w|})$ en les distinguant selon leur deuxième point de coupure, ce qui donne la relation ensembliste suivante :

$$\mathrm{Seg}(w_v^{|w|}) = \coprod_{k=v+1}^{|w|} (v) \odot \mathrm{Seg}(w_k^{|w|})$$

ce qui permet de réexprimer $\beta(v)$ en faisant intervenir une double somme :

$$\beta(v) = \sum_{\boldsymbol{\sigma'}\in\mathrm{Seg}(w_v^{|w|})} \mathbb{P}(\boldsymbol{\sigma'}, w_v^{|w|}) = \sum_{k=v+1}^{|w|} \sum_{\boldsymbol{\sigma'}\in\mathrm{Seg}(w_k^{|w|})} \mathbb{P}((v) \odot \boldsymbol{\sigma'}, w_v^{|w|})$$

$$= \sum_{k=v+1}^{|w|} \sum_{\boldsymbol{\sigma'}\in\mathrm{Seg}(w_k^{|w|})} \mathbb{P}(w_v^k)(1 - p_{\text{arrêt}}) \mathbb{P}(\boldsymbol{\sigma'}, w_k^{|w|}) = (1 - p_{\text{arrêt}}) \sum_{k=v+1}^{|w|} \mathbb{P}(w_v^k) \cdot \beta(k)$$

en prenant comme condition de bord $\beta(|w|) = \dfrac{p_{\text{arrêt}}}{1 - p_{\text{arrêt}}}$.

Ces deux relations de récurrence permettent de calculer toutes les valeurs de $\alpha(u), \beta(v)$, $0 \leq u, v \leq |w|$ par programmation dynamique (figure 2.6), en remplissant un tableau unidimensionnel. Calculer $\alpha(u)$ nécessite de calculer une somme des $\alpha(k)_{0\leq k<u}$ (pondérées par les probabilités $\mathbb{P}(w_k^u)$), c'est-à-dire u additions. Donc remplir le tableau $\alpha(u)$ pour tous les $u \in \{0, \ldots, |w|\}$ nécessite : $1 + 2 + \ldots + |w|$ additions et la complexité de ce calcul est donc quadratique. Il en est de même pour β.

$\alpha(0), \beta(|w|) \leftarrow \frac{p_{\text{arrêt}}}{1-p_{\text{arrêt}}}$
for $u = 1, \ldots, |w|$ **do**
  $\alpha(u) \leftarrow 0$
  for $0 \leq k \leq (u-1)$ **do**
    $\alpha(u) \leftarrow \alpha(u) + \alpha(k) \cdot \mathbb{P}(w_k^u)$
  end for
end for
for $v = (|w|-1), \ldots, 0$ **do**
  $\beta(v) \leftarrow 0$
  for $(v+1) \leq k \leq |w|$ **do**
    $\beta(v) \leftarrow \beta(v) + \beta(k) \cdot \mathbb{P}(w_v^k)$
  end for
end for
return α, β

FIGURE 2.6 – Algorithme de programmation dynamique pour calculer les α et β

"Forward filtering - backward sampling" On a vu plus haut qu'étant donné un mot w, la probabilité que le segment (u, v) apparaisse dans une segmentation de w vaut :

$$\mathbb{P}(\mathbf{s} \in \text{Seg}_{u,v}(w)|w) = \frac{\alpha(u) \cdot \frac{(1-p_{\text{arrêt}})^2}{p_{\text{arrêt}}} \cdot \beta(v)}{\alpha(|w|)}$$

Nous allons à présent voir comment échantillonner les points de coupure pour obtenir une segmentation, en commençant par la fin du mot.

Les segmentations de w peuvent se partitionner en distinguant les segmentations selon leur avant-dernier point de coupure :

$$\text{Seg}(w) = \coprod_{k=0}^{|w|-1} \underbrace{\left(\text{Seg}(w_0^k) \odot (|w|) = \text{Seg}_{k,|w|}(w)\right)}_{\text{segmentations dont l'avant-dernier point de coupure est } k}$$

Le dernier point de coupure de toute segmentation de w est fixé et correspond à la fin du mot. Ensuite, échantillonner l'avant-dernier point de coupure revient à déterminer dans lequel des ensembles de cette partition notre segmentation va tomber.

Ainsi, on placera l'avant-dernier point de coupure à une position $k \in \{0, \ldots, |w|-1\}$ avec une probabilité donnée par la formule suivante (à noter que $k = 0$ signifie

que le mot ne sera pas coupé) :

$$\begin{aligned}\mathbb{P}(\text{avant-dernier point de coupure à } k|w) &= \mathbb{P}\left(\mathbf{s} \in \text{Seg}_{k,|w|}(w)|w\right) \\ &= \frac{\alpha(k)\frac{(1-p_{\text{arrêt}})^2}{p_{\text{arrêt}}}\beta(|w|)}{\alpha(|w|)} \\ &= \frac{\alpha(k)(1-p_{\text{arrêt}})}{\alpha(|w|)}\end{aligned}$$

Puis la procédure peut se répéter, en remplaçant le mot à segmenter par w_0^k, sachant que les valeurs de $\alpha(u)$ resteront les mêmes pour ce mot, à ceci près que seuls les $\alpha(u)$ pour $u \in 0, \ldots, k$ seront considérés. Cette procédure, résumée dans la figure 2.7, est un type d'algorithme appelé *forward filtering - backward sampling* : les points de coupure sont échantillonnés de la droite vers la gauche, mais leurs probabilités sont pondérées par les $\alpha(k)$ qui sont été calculés par récurrence de gauche à droite.

procedure SEGMENTER(w)
  if $|w| > 0$ **then**
    for $0 \leq k \leq (|w| - 1)$ **do**
      $P(k) \leftarrow \frac{\alpha(k)(1-p_{\text{arrêt}})}{\alpha(|w|)}$
    end for
    $k_0 \leftarrow$ Echantillonner(P)
    $\boldsymbol{\sigma} \leftarrow$SEGMENTER($w_0^{k_0}$)
    return $\boldsymbol{\sigma} \odot (|w|)$
  else
    return (0)
  end if
end procedure

FIGURE 2.7 – Algorithme de *forward filtering - backward sampling* pour échantillonner une segmentation

2.3.3 Expériences

2.3.3.1 Données et évaluation

Les expériences avec le modèle que nous avons présenté ont été réalisées sur le jeu de données anglais du *Morpho Challenge 2005*[4]. Le *Morpho Challenge 2005* fut une compétition organisée par les concepteurs de *Morfessor* pour comparer différents algorithmes de segmentation morphologique non supervisée. Les

4. Les données et leur description sont disponibles à l'adresse : `http://morpho.aalto.fi/events/morphochallenge2005/datasets.shtml`

données d'entraînement que nous avons utilisées sont une liste de mots (ainsi que leur fréquence d'apparition) extraite d'une variété de publications et de romans du *Gutenberg project*, un échantillon du corpus *Gigaword* anglais, ainsi que le corpus *Brown* entier. Elles contiennent 167377 *types* de mots et 24447034 *tokens*. Les données de test sont une liste de 532 mots anglais ainsi que leur segmentation, réalisée par des linguistes. Le modèle est évaluée en calculant la précision, le rappel et la F-mesure des segmentation générées par rapport à celles des données test. Ces trois quantités sont calculées sur les points de coupure des segmentations, c'est-à-dire que :

— La précision est, parmi les points de coupure générés par le modèle, la proportion des points de coupure qui correspondent à des points de coupure donnés dans les références.

— Le rappel est, parmi les points de coupure donnés dans les références, la proportion de points de coupure qui correspondent à des points de coupure qui ont été trouvés par le modèle.

— La F-mesure est la moyenne harmonique de la précision et du rappel : $\text{F-mesure} = 2 \cdot \frac{\text{Précision} \cdot \text{Rappel}}{\text{Précision} + \text{Rappel}}$

2.3.3.2 Expériences et résultats

Dans les expériences réalisées, nous avons cherché à determiner l'influence de deux facteurs sur la qualité des segmentations :

— Le *paramètre de force* α du PYP déterminant la distribution sur les morphèmes : nous avons varié la valeur de se paramètre ($\alpha = 1, 10, 100, 500, 1000$) tout en gardant un *paramètre de décompte* fixé à $d = 0.1$.

— La manière dont les données étaient présentées à l'algorithme. En effet, celles-ci pouvaient être présentées par *types* (en présentant chaque forme de mot avec la même fréquence lors de l'apprentissage, c'est-à-dire selon une distribution uniforme) ou par *occurences* (en présentant chaque forme de mot avec une fréquence égale à sa fréquence d'usage dans le texte). De plus, nous avons essayé plusieurs compromis entre ces deux alternatives, de la manière suivante. Pour une forme w, notons $n(w)$ le nombre de fois qu'elle apparaît dans les données d'apprentissage, et définissons $P_\gamma(w) = \frac{n(w)^\gamma}{\sum_{w'} n(w')^\gamma}$ une distribution de probabilité (paramétrée par $\gamma \in [0, 1]$ que nous appellerons *paramètre de lissage*) sur les formes de mots présentes dans les données d'apprentissage, correspondant à la probabilité (à chaque itération) de présenter w à l'algorithme. Alors choisir $\gamma = 0$ correspond à présenter les exemples selon une distribution uniforme (donc par *types*), et choisir $\gamma = 1$ correspond à présenter les exemples par *occurences*. Lorsque $0 < \gamma < 1$, on réalise une interpolation entre ces deux options.

Les expériences révèlent que le paramètre de force exerce une influence moindre sur la qualité des segmentations que le paramètre de lissage. En particulier, les meilleures performances sont obtenues lorsque $\gamma = \frac{1}{2}$, c'est-à-dire

γ : paramètre de *lissage* du corpus α : paramètre de force du PYP	$\gamma = 0$	$\gamma = 0.25$	$\gamma = 0.5$	$\gamma = 0.75$	$\gamma = 1$
$\alpha = 1$	P 62.8 R 52.7 F 57.2 S 13.7	P 66.5 R 51.6 F 58.1 S 12.7	P 70.1 R 53.6 F **60.7** S 12.5	P 67.1 R 52.7 F **59.0** S 12.9	P 64.2 R 50.7 F 56.7 S 12.9
$\alpha = 10$	P 63.6 R 53.8 F 58.2 S 13.9	P 65.6 R 52.1 F 58.1 S 13.0	P 69.9 R 52.8 F **60.1** S 12.4	P 67.3 R 52.1 F 58.8 S 12.7	P 60.7 R 47.9 F 53.5 S 12.9
$\alpha = 100$	P 61.3 R 51.6 F 56.0 S 13.8	P 65.4 R 52.1 F 58.0 S 13.1	P 69.3 R 52.6 F **59.8** S 12.4	P 66.9 R 52.1 F 58.6 S 12.8	P 62.5 R 48.0 F 54.3 S 12.6
$\alpha = 500$	P 62.0 R 51.7 F 56.4 S 13.7	P 64.8 R 51.5 F 57.4 S 13.0	P 69.0 R 51.2 F 58.8 S 12.1	P 67.0 R 50.3 F 57.4 S 12.3	P 59.1 R 46.7 F 52.2 S 13.0
$\alpha = 1000$	P 59.5 R 50.7 F 54.8 S 13.9	P 66.1 R 51.7 F 58.0 S 12.8	P 69.0 R 51.4 F **58.9** S 12.2	P 66.4 R 51.1 F 57.8 S 12.6	P 59.0 R 46.5 F 52.0 S 12.9

FIGURE 2.8 – Evaluation des segmentations produites par un modèle unigramme de morphèmes non-paramétrique, au bout de 1673770 itérations (dix *epochs*). P = Précision, R = Rappel, F = F-mesure, S = Taux de segmentation. Ces quatres valeurs sont exprimées en pourcentages. Les cinq meilleures F-mesures sont en gras, les cinq meilleures précisions en rouge, les cinq meilleurs rappels en bleu, et les cinq taux de segmentation les plus élevés en vert.

lorsqu'on réalise un compromis à mi-chemin entre une présentation par types et une présentation par occurences. Ceci peut s'interpréter en considérant le fait que présenter les exemples exclusivement par types ou exclusivement par occurences induit entraîne les deux cas des désavantages :

— Lorsque les exemples sont présentés par types, des mots d'usage fréquent (comme les pronoms, les prépositions, et plus généralement les mots à fonction grammaticale : *mots-outils*) sont présentés aussi fréquemment que des mots rares (comme des mots composés, des mots appartenant à un domaine spécialisé, des emprunts à d'autres langues, des noms propres...). Par conséquent, les exemples auxquels l'algorithme est exposé sont très hétérogènes, ce qui rend difficile la détection de régularités (segments d'usage fréquent) parmi ceux-ci. Le fait qu'on observe un taux de segmentation plus élevé lorsque $\gamma = 0$ laisse supposer que l'algorithme a tendance à excessivement segmenter : en effet, l'hétérogénéité des exemples fait que les segments d'usage fréquent susceptibles d'être détectés seront assez courts, parfois réduits à des caractères isolés.

— Lorsque les exemples seront présentés par occurences, une grande partie des exemples auxquels l'algorithme sera exposé seront des mots-outils sans

structure morphologique, et donc qui auront tendance à être expliqués par l'algorithme comme un morphème unique. L'algorithme apprendra alors "par cœur" ces mots-outils, et une trop faible partie de son temps d'apprentissage sera dédiée à décomposer des mots plus complexes en morphèmes, ce qui entraînera une qualité moindre de ses analyses morphologiques.

Conclusion

La première famille de distributions que nous avons présentée dans ce chapitre sont les distributions de Dirichlet, qui sont utilisées dans un cadre bayésien afin d'engendrer des vecteurs de paramètres pour une distribution de probabilités. L'approche bayésienne requiert de calculer, suite à l'observation de données, des distributions *a posteriori* sur l'espace des paramètres (en appliquant la formule de Bayes). En général, cela nécessiterait de calculer une intégrale sur l'espace des paramètres (ce qui, pour une distribution quelconque, ne pourrait que se faire de manière approchée et avec éventuellement un certain coût de calcul). Ce qui rend les distributions de Dirichlet particulièrement adaptées au cadre bayésien, c'est que les distributions *a posteriori* restent des distributions de Dirichlet, et qu'elles sont obtenues simplement par un changement de paramètres de la distribution *a priori*, les nouveaux paramètres étant facilement calculables à partir des données. Les processus de Dirichlet sont une généralisation en dimension infinie des distributions de Dirichlet, et les processus de Pitman-Yor sont une généralisation des processus de Dirichlet qui introduisent un nouveau *paramètre de décompte* et produisent des lois de probabilités suivant une loi de Zipf. Les processus de restaurant chinois permettent d'implémenter les processus de Dirichlet en donnant un moyen algorithmique d'échantillonner les distributions *a posteriori* après observation de données. Enfin, en utilisant un processus de Pitman-Yor pour engendrer des "dictionnaires probabilistes de morphèmes", nous obtenons un modèle génératif des mots d'une langue comme suites finies de morphèmes. L'inférence de ce modèle se fait en alternant échantillonnage des segmentations conditionnellement au modèle, et échantillonnage du modèle conditionnellement aux segmentations. L'échantillonnage des segmentations est rendu efficace par une méthode de programmation dynamique. L'échantillonnage du modèle se fait en maintenant un processus de restaurant chinois, où les tables correspondent aux types de morphèmes, et le nombre de clients à une table donnée au nombre d'occurrences du type correspondant. Nous avons observé expérimentalement que le paramètre de force du processus de Pitman-Yor influençait moins les résultats que le paramètre du lissage du corpus (ce dernier gouvernant à quel point les mots à segmenter sont échantillonés plutôt selon leur fréquence dans le corpus, ou de manière équiprobable pour tous les types).

Chapitre 3

Processus hiérarchiques

Le modèle unigramme de morphèmes que nous avons présenté au chapitre précédent suppose que les morphèmes sont générés de manière indépendante. Pour modéliser la morphologie, on souhaiterait permettre des dépendances entre morphèmes : par exemple, on souhaiterait modéliser que certains morphèmes n'apparaissent qu'en début ou en fin de mot, ou alors que certains morphèmes ne peuvent être suivis que par un nombre réduit d'autres morphèmes. Dans ce chapitre, nous introduisons le formalisme des *processus de Pitman-Yor hiérarchiques*, ceux-ci permettant de générer des familles de distributions conditionnelles et donc des dépendances entre variables. Les *CRP hiérarchiques* permettent alors de calculer les distributions *a posteriori*. Nous appliquons ce formalisme à deux modèles de morphologie. Le premier est une version bayésienne non-paramétrique d'un modèle *n-grammes* de morphèmes. Son caractère bayésien non-paramétrique permet de prendre en compte une variété potentiellement infinie de morphèmes. Le second est une version bayésienne non-paramétrique d'un modèle semi-Markov caché, la variable cachée étant les classes auxquels les morphèmes sont affectés. En plus de prendre en compte une variété potentiellement infinie de morphèmes, il autorise aussi une variété potentiellement infinie de classes.

3.1 Hiérarchies d'adapteurs

3.1.1 Exemple introductif : modèle n-gramme de morphèmes

Au chapitre précédent, nous avons décrit un modèle unigramme de morphèmes où la distribution sur les morphèmes était soumise à un *a priori* de Pitman Yor. Pour résumer :

— Chaque morphème était généré indépendamment des précédents selon une distribution G, elle-même engendrée par un $PYP(\alpha, d, G_0)$ où G_0 était une certaine distribution de base "agnostique" sur l'ensemble des mots

pouvant être formés à partir d'un alphabet.

— La probabilité de générer un morphème μ étant donné un ensemble S de morphèmes précédemment engendrés, c'est-à-dire la probabilité $\mathbb{P}(\mu|S) = \int_G \mathbb{P}(\mu, G|S)dG$ obtenue en marginalisant G était donnée par :

$$\frac{n_\mu - k_\mu d + (\alpha + Kd)G_0(\mu)}{N + \alpha}$$

où, étant donné un CRP représentant l'ensemble des morphèmes précédemment engendrés (avec un client par occurrence de morphème, assis à une table étiquetée avec le morphème correspondant) :

— n_μ est le nombre de clients dont la table est étiquetée par μ

— k_μ est le nombre de tables étiquetées par μ

— N est le nombre total de clients (c'est-à-dire le nombre total d'occurrences de morphèmes)

— K est le nombre total de tables

La question que nous abordons à présent est : comment généraliser ceci à un modèle n-gramme ?

Il sera pour cela nécessaire d'introduire des distributions $G_{\mu_{-n+1}\dots\mu_{-1}}$ afin de modéliser, pour chaque contexte $\mu_{-n+1}\dots\mu_{-1}$ de $(n-1)$ morphèmes, les morphèmes qui lui succèderont. Plutôt qu'une unique distribution G engendrée par un PYP (ou un processus de Dirichlet), on considèrera alors une *famille* de distributions $\left(G_{\mu_{-n+1}\dots\mu_{-1}}\right)_{\mu_{-n+1}\dots\mu_{-1}}$.

De plus, on souhaiterait prendre en compte les dépendances entre ces distributions, notamment le fait que les contextes similaires présenteront des distributions similaires quant aux morphèmes qui leur succèderont. Ceci en considérant que deux contextes sont d'autant plus "similaires" qu'ils ont un long suffixe commun. Par exemple, les morphèmes succédant aux contextes `re-plie-` et à `dé-plie-` se ressembleront davantage que pour les contextes `re-plie-` et `re-voit-`.

Pour prendre en compte ces dépendances, la solution que nous adopterons est d'utiliser une distribution de base commune pour des contextes partageant les mêmes $(n-2)$ morphèmes en suffixe, c'est-à-dire que : étant donné des morphèmes $\mu_{-n+2}\dots\mu_{-1}$, alors pour tout morphème μ_{-n+1}, la distribution $G_{\mu_{-n+1}\dots\mu_{-1}}$ sera engendrée par un processus de Pitman-Yor (ou de Dirichlet) de distribution de base $G_{\mu_{-n+2}\dots\mu_{-1}}$.

De même, toutes les distributions $G_{\mu_{-n+2}\dots\mu_{-1}}$ seront engendrées par une distribution de base $G_{\mu_{-n+3}\dots\mu_{-1}}$, et ainsi de suite (on dira qu'elles se *replieront* sur cette distribution de base). Au bout de cet *arbre de repli*, on aura les distributions $G_{\mu_{-1}}$ qui se replieront sur une distribution $G_\emptyset$ ($\emptyset$ représentant un contexte vide). $G_\emptyset$ se repliera elle-même sur un modèle de chaîne de caractères $G_{\mathbf{base}}$ analogue à la distribution que nous avons noté G_0 au chapitre précédent.

En termes plus formels, la situation que nous considérons peut se décrire de la manière suivante :

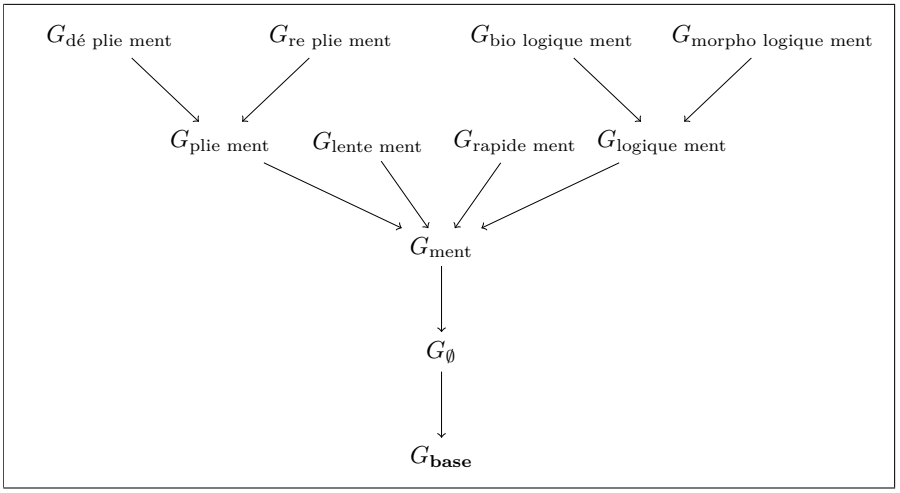


FIGURE 3.1 – Portion d'un arbre de repli. Chaque flèche $G_{c_1} \leftarrow G_{c_2}$ signifie que G_{c_2} est engendrée par un processus de Pitman-Yor (ou de Dirichlet) de distribution de base G_{c_1}. La seule distribution sans parent est $G_{\mathbf{base}}$: elle est fixée et fait partie des hyperparamètres du processus.

— La donnée d'un ensemble $\mathcal{C}$ de *contextes* : l'ensemble des suites d'au plus $(n-1)$ morphèmes.

— La donnée d'un *schéma de repli* : une application $\beta : \mathcal{C} \to \mathcal{C} \cup \{\mathbf{base}\}$ qui :

  — A toute séquence de morphèmes de longueur ≥ 1 associe la même séquence où le morphème le plus ancien a été tronqué : $\beta(\mu_{-n+i} \dots \mu_{-1}) = \mu_{-n+i+1} \dots \mu_{-1}$

  — A la séquence vide ($\emptyset$) associe **base**

  dans l'exemple de la figure 3.1, le schéma de repli est représenté par les flèches.

— La donnée, pour tout contexte $c \in \mathcal{C}$, d'un *adapteur*[1] noté η_c, c'est-à-dire d'une application qui à toute distribution associe une distribution sur des distributions. Ici on pourra par exemple utiliser pour tout contexte c l'adapteur $\eta_c : G \mapsto DP(\alpha, G)$ (pour définir une hiérarchie de processus de Dirichlet) ou l'adapteur $\eta_c : G \mapsto PYP(\alpha, d, G)$ (pour définir une hiérarchie de processus de Pitman-Yor).[2]

1. Pour rester fidèle à la terminologie des *adaptor grammars*

2. Auquel cas α et d seront des hyperparamètres fixés et communs à tous les G_c. Notons que puisque l'adapteur η_c dépend de c, ce formalisme permet aussi d'utiliser des hyperparamètres différents pour les G_c, ou encore de faire en sorte que certains G_c (mais pas tous) partagent leurs hyperparamètres.

Etant donnés $\mathcal{C}$, $\beta, G_{\mathbf{base}}$ et les $(\eta_c)_{c\in\mathcal{C}}$, on peut alors considérer la famille de distributions aléatoires $(G_c)_{c\in\mathcal{C}}$ définie par :

$$\forall c \in \mathcal{C},\ G_c \sim \eta_c(G_{\beta(c)})$$

ce qui, dans notre exemple, donne bien le schéma génératif souhaité :

$$G_{\mu_{-n+1}\dots\mu_{-1}} \sim PYP(\alpha, d, G_{\mu_{-n+2}\dots\mu_{-1}})$$

$$G_{\mu_{-n+2}\dots\mu_{-1}} \sim PYP(\alpha, d, G_{\mu_{-n+3}\dots\mu_{-1}})$$

$$\dots$$

$$G_{\mu_{-2}\mu_{-1}} \sim PYP(\alpha, d, G_{\mu_{-1}})$$

$$G_{\mu_{-1}} \sim PYP(\alpha, d, G_{\emptyset})$$

$$G_{\emptyset} \sim PYP(\alpha, d, G_{\mathbf{base}})$$

3.1.2 Définition formelle

Afin de généraliser l'exemple précédent, nous définissons une *hiérarchie d'adapteurs* comme la donnée de :

- Un ensemble de *contextes* $\mathcal{C}$
- Un *arbre de repli* : une application $\beta : \mathcal{C} \to \mathcal{C} \cup \{\mathbf{base}\}$ telle que pour tout contexte $c \in \mathcal{C}$, il existe un rang k tel que $\underbrace{\beta \circ \dots \circ \beta}_{k \text{ itérations}}(c) = \mathbf{base}$ [3]
- Pour chaque $c \in \mathcal{C}$, un *adapteur* η_c, c'est-à-dire une application $Prob(Prob(X)) \to Prob(X)$ qui à toute distribution de probabilité sur un ensemble X associe une distribution sur les distributions sur X.

Une hiérarchie d'adapteurs définit alors une famille de variables aléatoires $(G_c)_{c\in\mathcal{C}}$ vérifiant la relation suivante :

$$G_c \sim \eta_c(G_{\beta(c)})$$

Remarque. Les applications de la forme $\eta : G \mapsto DP(\alpha, G)$ sont des exemples d'adapteurs. Lorsque des adapteurs de cette forme seront utilisés, on parlera de *processus de Dirichlet hiérarchique*. De même, lorsque des adapteurs de la forme $\eta : G \mapsto PYP(\alpha, d, G)$ seront utilisés, on parlera de *processus de Pitman-Yor hiérarchique*.

3. Cette condition traduit la structure d'*arbre*, nécessaire pour éviter des cycles de repli infinis.

3.1.3 Restaurant chinois hiérarchique

Nous avons vu présenté au chapitre précédent le *processus de restaurant chinois* qui permet, étant donné une distribution G engendrée par un processus de Dirichlet (ou de Pitman-Yor), et un ensemble d'observations $\mathbf{e} = \{x_1, \ldots, x_r\}$ issues de G ($x_i \sim G$), de calculer la distribution a posteriori sur la prochaine observation en marginalisant G.

Nous montrons à présent comment cette construction peut se généraliser au cas de processus de Dirichlet (ou de Pitman-Yor) hiérarchiques. Dans ce qui suit, nous considérons le cas d'un processus de Pitman-Yor hiérarchique, car celui-ci englobe le cas d'un processus de Dirichlet hiérarchique (en effet, un processus de Dirichlet est un processus de Pitman-Yor dont le paramètre de décompte vaut 0). Autrement dit, on considère le cas d'une hiérarchie d'adapteurs où tous les adapteurs sont de la forme $G \mapsto PYP(\alpha, d, G)$.

A chaque distribution aléatoire G_c de la hiérarchie est donc associé un restaurant chinois noté $CRP(c)$ (d'hyperparamètres α_c, d_c), et nous noterons :

— $K(c)$ le nombre de tables qu'il contient.

— $N(c)$ le nombre de clients qu'il contient.

— $n_k(c)$ le nombre de clients assis à la k-ème table et $l_k(c)$ son étiquette.

— $n_l(c)$ le nombre de clients assis à une table d'étiquette l, autrement dit : $n_l(c) = \sum_{k \in \text{tables}} \mathbf{1}_{l_k(c)=l} \cdot n_k(c)$

Ici, les observations et les distributions les modélisant sont associées à un contexte : par conséquent, les observations prennent la forme d'éléments $c \to x$ avec $c \in \mathcal{C}$ et $x \in X$. Les étiquettes des tables sont prises dans l'ensemble X.

3.1.3.1 Distribution de probabilité associée à un CRP

Chaque $CRP(c)$ induit une distribution de probabilité $\mathbb{P}(x|CRP(c))$ qui représente une probabilité d'observer x dans le contexte c. La distribution de probabilité induite par $CRP(c)$ est définie par :

$$\mathbb{P}(x|CRP(c)) = \frac{n_x(c) - d_c + (\alpha_c + K(c)d_c)\overbrace{\mathbb{P}\left(x|CRP(\beta(c))\right)}^{\text{distribution de base}}}{N(c) + \alpha_c}$$

3.1.3.2 Assignation d'une table à un nouveau client

Un nouveau client s'asseoit dans le restaurant $CRP(c)$ selon la politique suivante :

— Pour tout indice k correspondant à une table déjà existante dans $CRP(c)$, on l'asseoit avec une probabilité proportionnelle à $n_k(c) - d_c$ à cette table.

— Avec une masse de probabilité résiduelle proportionnelle à $\alpha_c + K(c)d_c$, on crée une nouvelle table pour l'asseoir, et ici intervient l'aspect hiérarchique : si $\beta(c) \neq \textbf{base}$, on fait entrer un nouveau client dans $CRP(\beta(c))$ (la politique que nous décrivons est donc récursivement appliquée à $CRP(\beta(c))$,

ce client reçoit une étiquette l, et la nouvelle table de $CRP(c)$ est donc étiquetée avec l.

Ainsi, chaque création de table dans $CRP(c)$ correspond à l'entrée d'un nouveau client dans $CRP(\beta(c))$. Il y a donc une correspondance entre les tables de $CRP(c)$ et les clients de $CRP(\beta(c))$.

procedure $CRP(c)$.AJOUT()
  for $k = 1 \dots K(c)$ **do**
    $P[k] \leftarrow \frac{n_k(c) - d_c}{N(c) + \alpha_c}$ ▷ probabilité d'assignement à la k-ème table
  end for
  $P[K(c)+1] \leftarrow \frac{(\alpha_c + K(c)d_c)\mathbb{P}(x|CRP(\beta(c)))}{N(c) + \alpha_c}$ ▷ probabilité d'ajout de nouvelle table
  $k \leftarrow$ ECHANTILLONNER(P)
  if $1 \leq k \leq K(c)$ **then** ▷ ajout à une table existante
    $n_k(c) \leftarrow n_k(c) + 1$
  else if $k = K(c) + 1$ **then** ▷ création d'une nouvelle table
    $K(c) \leftarrow K(c) + 1$
    $n_k(c) \leftarrow 1$
    if $\beta(c) =$ **base then**
      $l_k(c) \leftarrow$ECHANTILLONNER($G_{\mathbf{base}}$)
    else
      $l_k(c) \leftarrow CRP(\beta(c))$.AJOUT() ▷ repli
    end if
  end if
  $N(c) \leftarrow N(c) + 1$
  return $l_k(c)$
end procedure

FIGURE 3.2 – Ajout d'un nouveau client à $CRP(c)$

3.1.3.3 Conditionnement à un ensemble d'observations

Etant donné un ensemble d'observations $\mathbf{e} = \{c_1 \to x_1, \dots, c_r \to x_r\}$, décrivons à présent comment *mettre en cache les observations*, c'est-à-dire comment mettre à jour les $CRP(c)$ afin de représenter les distributions conditionnelles $\mathbb{P}(x|c, \mathbf{e})$.

Essentiellement, il s'agit, pour chaque observation $c_i \to x_i$, d'asseoir un client à $CRP(c_i)$ selon la politique du restaurant, mais en imposant qu'il reçoive l'étiquette x_i (quitte à créer une nouvelle table).

Par conséquent, pour chaque observation $c_i \to x_i$, on assoiera un client à $CRP(c_i)$ selon la politique suivante :

— Pour toute table d'indice k et étiquetée par x_i, le client s'assoiera à la table k avec une probabilité proportionnelle à $n_k(c_i) - d_{c_i}$.

— Avec une probabilité résiduelle proportionnelle à $(\alpha_{c_i}+K(c_i)d_{c_i})\mathbb{P}(x_i|CRP(\beta(c_i))$, une nouvelle table sera créée pour le client, et recevra l'étiquette x_i. La création de cette nouvelle table a une répercussion dans la hiérarchie : on lui fait correspondre l'entrée d'un nouveau client dans $CRP(\beta(c_i))$ en imposant (application récursive de la politique) qu'il reçoive une étiquette x_i (autrement dit, on met en cache l'observation $\beta(c_i) \to x_i$ *sauf si* $\beta(c_i) =$ **base**).

procedure CACHE($c \to x$)
  for $k = 1 \dots K(c)$ **do**
    $Q[k] \leftarrow n_k(c) - d_c$ ▷ probabilité d'assignement à la k-ème table
  end for
  $Q[K(c)+1] \leftarrow (\alpha_c + K(c)d_c)\mathbb{P}(x|CRP(\beta(c)))$ ▷ probabilité d'ajout de nouvelle table
  $P \leftarrow$NORMALISER(Q) ▷ pour obtenir une distribution de probabilité
  $k \leftarrow$ ECHANTILLONNER(P)
  if $1 \leq k \leq K(c)$ **then** ▷ ajout à une table existante
    $n_k(c) \leftarrow n_k(c) + 1$
  else if $k = K(c)+1$ **then** ▷ création d'une nouvelle table
    $K(c) \leftarrow K(c) + 1$
    $n_k(c) \leftarrow 1$
    if $\beta(c) \neq$ **base then** CACHE($\beta(c) \to x$) ▷ repli
    end if
  end if
  $N(c) \leftarrow N(c) + 1$
end procedure

FIGURE 3.3 – Mise en cache d'une observation

3.1.3.4 Remarque

La procédure de mise en cache décrite ci-dessus n'est pas déterministe : pour un même ensemble d'observations **e**, elle peut donner lieu à plusieurs configurations différentes de CRP, et les distributions induites $\mathbb{P}(x|CRP(c))$ varieront selon la configuration adoptée. La probabilité a posteriori $\mathbb{P}(x|\mathbf{e})$ s'obtient en effectuant une moyenne sur toutes les configurations possibles, pondérée par la probabilité (sachant **e**) de chacune de ces configurations. Autrement dit :

$$\mathbb{P}(x|\mathbf{e}) = \mathbb{E}\Big[\mathbb{P}(x|CRP(c)) \cdot \mathbb{P}(CRP(c)|\mathbf{e})\Big]$$

en pratique, lors de l'inférence, on approximera cette probabilité par $\mathbb{P}(x|CRP(c))$ tout en rééchantillonnant régulièrement la configuration du restaurant. La confi-

guration peut par exemple se rééchantillonner en vidant le restaurant puis en mettant à nouveau en cache les observations.

3.1.3.5 Exemple

Reprenons l'exemple d'un modèle bigramme de morphèmes tel que présenté à la section 3.1.1. Supposons que l'on dispose comme observations de deux mots segmentés en morphèmes : `logique ment` et `morpho logique ment`, et qu'on souhait mettre à jour les CRP de la hiérarchie de sorte à tenir compte de ces segmentations.

Les observations à mettre en cache sont donc :

$$\mathbf{e} = \Big\{ \begin{array}{l} \texttt{I} \rightarrow \texttt{logique}, \\ \texttt{logique} \rightarrow \texttt{ment}, \\ \texttt{ment} \rightarrow \texttt{F}, \\ \texttt{I} \rightarrow \texttt{morpho}, \\ \texttt{morpho} \rightarrow \texttt{logique}, \\ \texttt{logique} \rightarrow \texttt{ment}, \\ \texttt{ment} \rightarrow \texttt{F} \end{array} \Big\}$$

où `I` (respectivement `F`) est un symbole spécial de début de mot (respectivement de fin de mot).

Ici, l'ensemble des contextes observés est : $\mathcal{C} = \Big\{ \texttt{I}, \texttt{logique}, \texttt{ment}, \texttt{morpho}, \emptyset \Big\}$ (où $\emptyset$ est le contexte vide) et l'arbre de repli est donné dans la figure 3.1.3.5

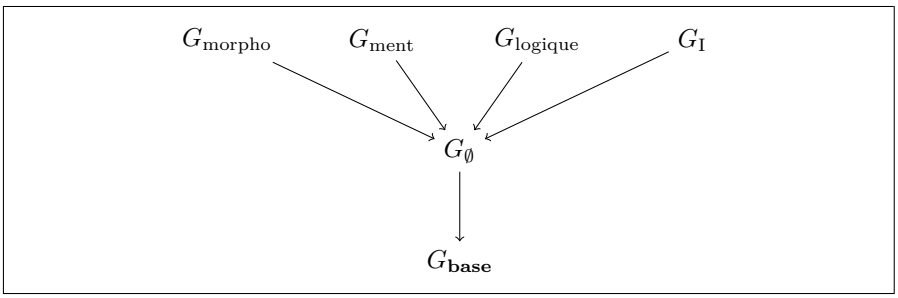


FIGURE 3.4 – Exemple d'arbre de repli

Pour chaque contexte observé c, on devra donc maintenir un $CRP(c)$: la figure 3.1.3.5 est un exemple d'état que peut prendre la hiérarchie de CRP, une fois que toutes les observations de $\mathbf{e}$ ont été mises en cache.

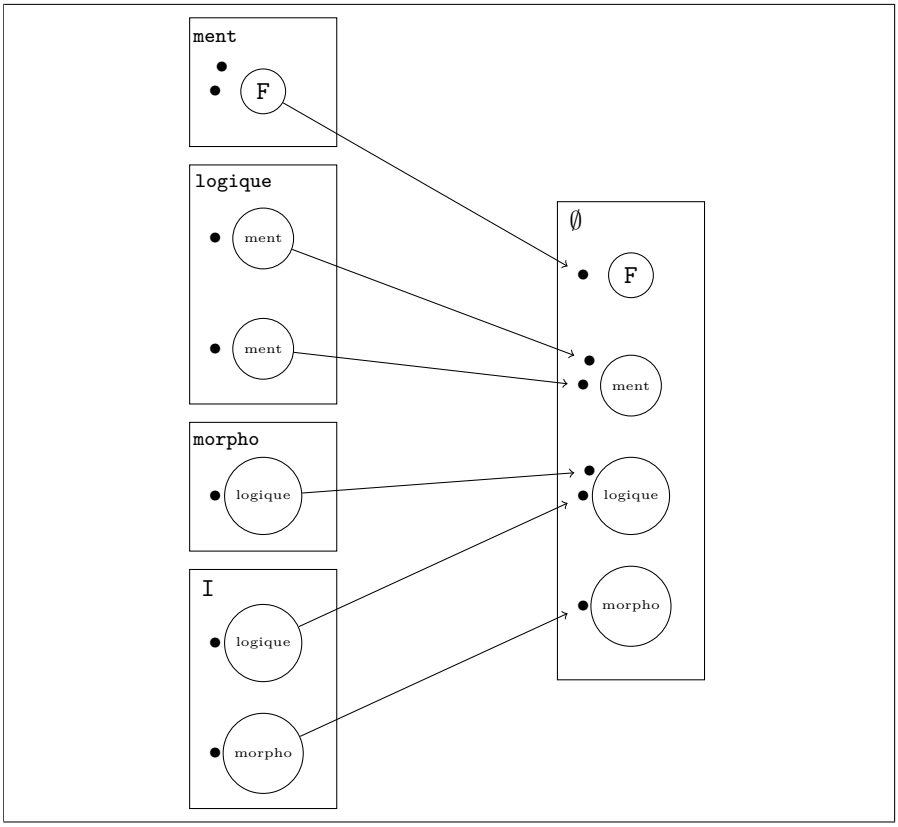


FIGURE 3.5 – Exemple d'état pour un CRP hiérarchique

3.2 Modèle n-gramme de séquences de morphèmes

Dans cette section, nous considérons le modèle n-gramme pris en exemple introductif dans la section 3.1.1 et décrivons son utilisation afin de produire des segmentations de mots en morphèmes.

3.2.1 Description

L'ensemble des *contextes* $\mathcal{C}$ est l'ensemble des suites finies de la forme $\mu_1 \dots \mu_r$ de longueur $r = 0 \dots n$, telles que chaque μ_i est une chaîne de caractères (un morphème potentiel) en autorisant éventuellement μ_1 à être un symbole spécial

(I) de "début de mot". On le munit d'un *arbre de repli* défini comme :

$$\begin{aligned}\beta : \mathcal{C} &\to \mathcal{C} \cup \{\mathbf{base}\} \\ \mu_1 \dots \mu_r \text{ avec } 1 \leq r \leq k &\mapsto \mu_2 \dots \mu_r \\ \emptyset &\mapsto \mathbf{base}\end{aligned}$$

A chaque $c \in \mathcal{C}$ est associé un adapteur de la forme $G \mapsto PYP(\alpha_c, d_c, G)$. Le tout forme un PYP hiérarchique donnant lieu à une famille de distributions aléatoires $(G_c)_{c\in\mathcal{C}}$. La distribution située au plus bas de la hiérarchie est une distribution $G_{\mathbf{base}}$. $G_{\mathbf{base}}$ et les G_c sont des distributions sur l'ensemble des chaînes de caractères auquel on a ajouté `F` (symbole spécial de fin de mot).

La famille $(G_c)_{c\in\mathcal{C}}$ définit un modèle n-gramme (aléatoire) de morphèmes susceptible de générer des séquences de morphèmes selon la distribution jointe :

$$\mathbb{P}\left(\mu_1 \dots \mu_r | (G_c)_{c\in\mathcal{C}}\right) = \prod_{i=0}^{r+1} G_{\mu_{i-n+1}^{i}}(\mu_i)$$

où :

— $\mu_{l+1} = \mathtt{F}$ par convention.

— la notation μ_i^j désigne :

$$\begin{aligned}\mu_i^j &= \emptyset \text{ (morphème vide)} && \text{si } i = j \\ &= \mathtt{I}\mu_1\mu_2 \dots \mu_j && \text{si } i < 0 \leq j \leq r \text{ (padding en début de séquence)} \\ &= \mu_{i+1}\mu_{i+2} \dots \mu_j && \text{si } 0 \leq i < j \leq r \\ &= \mu_{i+1} \dots \mu_l\mathtt{F} && \text{si } 0 \leq i \leq l < j \text{ (padding en fin de séquence)}\end{aligned}$$

Une séquence de morphèmes $\mu_1 \dots \mu_l$ peut être assimilée à un couple $(w, \mathbf{s})$ où :

— w est un mot (assimilé à la concaténation des μ_i).

— $\mathbf{s}$ est une *segmentation* de w, c'est-à-dire une liste d'entiers $\mathbf{s} = (s_0, \dots, s_r)$ telle que $0 = s_0 < s_1 < \dots < s_r = |w|$. μ_i est alors assimilé à $w_{s_{i-1}}^{s_i}$.

Le tout définit donc une distribution jointe :

$$\mathbb{P}\Big((w, \mathbf{s}) = \mu_1 \dots \mu_r, (G_c)_{c\in\mathcal{C}}\Big) = \mathbb{P}\left(\mu_1 \dots \mu_r | (G_c)_{c\in\mathcal{C}}\right) \cdot \mathbb{P}\left((G_c)_{c\in\mathcal{C}}\right)$$

— Le terme $\mathbb{P}\left(\mu_1 \dots \mu_r | (G_c)_{c\in\mathcal{C}}\right)$ est la probabilité, pour un modèle n-gramme donné (c'est-à-dire en fixant les distributions $(G_c)_{c\in\mathcal{C}}$), d'engendrer la séquence de morphèmes $\mu_1 \dots \mu_r$.

— Le terme $\mathbb{P}\left((G_c)_{c\in\mathcal{C}}\right)$ est l'a priori (probabiliste) sur les modèles n-grammes défini par la hiérarchie de PYP.

3.2.2 Inférence

On dispose d'un corpus $L = (w_1, \ldots, w_{|L|})$ de mots non segmentés, et on souhaite produire des segmentations de ces mots, c'est-à-dire échantillonner la distribution jointe $\mathbb{P}(\mathbf{s}_1, \ldots, \mathbf{s}_{|L|}|L)$

On effectue pour cela un *échantillonnage de Gibbs*, pour chaque $i = 1 \ldots |L|$, on échantillonne $\mathbf{s}_i$ conditionnellement à l'ensemble de toutes les autres segmentations (qu'on notera $\mathbf{s}_{-i}$), et on répète cette boucle jusqu'à convergence.

Le conditionnement à $\mathbf{s}_{-i}$ se fait en mettant en cache dans la hiérarchie de CRP toutes les transitions du type $\underset{\text{(contexte)}}{\mu_{i-n+1}^{i}} \rightarrow \underset{\text{(successeur)}}{\mu_i}$. Une fois ces observations mises en cache, la hiérarchie de CRP représente un modèle n-gramme de morphèmes dont la probabilité est donnée par :

$$\mathbb{P}\left(\mu_1 \ldots \mu_r | \left(CRP(c)\right)_{c \in \mathcal{C}}\right) = \prod_{i=0}^{r+1} \mathbb{P}\left(\mu_i | CRP(\mu_{i-n+1}^{i})\right)$$

où $\mathbb{P}\left(\mu_i | CRP(\mu_{i-n+1}^{i})\right)$ est la distribution (sur des morphèmes potentiels) induite par le CRP associé au contexte μ_{i-n+1}^{i}.

Echantillonner $\mathbf{s}_i$ revient dès lors à échantillonner une séquence de morphèmes $\mu_1 \ldots \mu_r$ dont la concaténation donne w_i.

Il nous reste donc à décrire comment échantillonner une telle segmentation avec un modèle n-gramme fixé, que nous noterons :

$$P_{\text{ngram}}(\mu_i | \mu_{i-n+1}^{i}) = \mathbb{P}\left(\mu_i | CRP(\mu_{i-n+1}^{i})\right)$$

3.2.3 Echantillonnage d'une segmentation

Nous généralisons l'algorithme *forward-backward* présenté au chapitre précédent à un modèle n-gramme. Il s'agit encore de calculer par programmation dynamique des tableaux α et β qui pourront par la suite être utilisés pour échantillonner des points de coupure, à ceci près que le fait que $n \geq 1$ entraîne que les tableaux seront multidimensionnels.

Les tableaux α et β sont indexés par des suites de points de coupure de la forme $(i_{-n+k}, \ldots, i_0)$, de longueur au plus n, et pris dans l'ensemble :

$$I = \overbrace{\left\{(i_{-n+1}, \ldots, i_0) \mid 0 < i_{-n+1} < \ldots < i_0 \leq |w|\right\}}^{=\mathring{I} \text{ (“intérieur de “}I\text{)}} \cup \underbrace{\left\{(i_{-n+k}, \ldots, i_0) \mid 1 \leq k \leq n \text{ et } 0 = i_{-n+k} < \ldots < i_0 \leq |w|\right\}}_{=\partial I \text{ (“bord” de } I\text{)}}$$

Pour toute suite $(i_{-n+k}, \ldots, i_0)$, α est défini par récurrence par les égalités suivantes :

— si $(i_{-n+1},\dots,i_0) \in \mathring{I}$ (cas récurrent) :

$$\alpha(i_{-n+1},\dots,i_0) = \sum_{0\le i_n<i_{-n+1}} \alpha(i_{-n},\dots,i_{-1})\cdot P_{\text{ngram}}\left(w_{i_{-1}}^{i_0} \mid w_{i_{-n}}^{i_{-n+1}},\dots,w_{i_{-2}}^{i_{-1}}\right)$$

— si $(i_{-n+k},\dots,i_0) \in \partial I$ (condition de bord) :

$$\begin{aligned}\alpha(i_{-n+k},\dots,i_0) =& P_{\text{ngram}}\left(w_{i_{-n+k}}^{i_{-n+k+1}} \mid \mathtt{I}\right)\\ &\times P_{\text{ngram}}\left(w_{i_{-n+k+1}}^{i_{-n+k+2}} \mid \mathtt{I}, w_{i_{-n+k}}^{i_{-n+k+1}}\right)\\ &\times P_{\text{ngram}}\left(w_{i_{-n+k+2}}^{i_{-n+k+3}} \mid \mathtt{I}, w_{i_{-n+k}}^{i_{-n+k+1}}, w_{i_{-n+k+1}}^{i_{-n+k+2}}\right)\\ &\times \dots \times P_{\text{ngram}}\left(w_{i_{-1}}^{i_0} \mid \mathtt{I}, w_{i_{-n+k}}^{i_{-n+k+1}},\dots,w_{i_{-2}}^{i_{-1}}\right)\end{aligned}$$

de même, pour β :

— si $(i_{-n+1},\dots,i_0) \in \mathring{I}$: (cas où le "bord gauche" n'est pas atteint)

  — si $i_0 < |w|$: (cas "intérieur")

$$\beta(i_{-n+1},\dots,i_0) = \sum_{i_0<i_1\le|w|} P_{\text{ngram}}\left(w_{i_0}^{i_1} \mid w_{i_{-n+1}}^{i_{-n+2}},\dots,w_{i_{-1}}^{i_0}\right)\cdot\beta(i_{-n+2},\dots,i_1)$$

  — si $i_0 = |w|$: (cas où seul le "bord droit" est atteint)

$$\beta(i_{-n+1},\dots,i_0) = P_{\text{ngram}}\left(\mathtt{F} \mid w_{i_{-n+1}}^{i_{-n+2}},\dots,w_{i_{-1}}^{i_0}\right)$$

— si $(i_{-n+k},\dots,i_0) \in \partial I$: (cas où le "bord gauche" est atteint)

  — si $i_0 < |w|$: (cas où seul le 'bord gauche" est atteint)

$$\beta(i_{-n+k},\dots,i_0) = \sum_{i_0<i_1\le|w|} P_{\text{ngram}}\left(w_{i_0}^{i_1} \mid w_{i_{-n+k}}^{i_{-n+k+1}},\dots,w_{i_{-1}}^{i_0}\right)\cdot\beta(i_{-n+k},\dots,i_1)$$

  — si $i_0 = |w|$: (cas où le "bord gauche" *et* le "bord droit" sont atteints)

$$\beta(i_{-n+k},\dots,i_0) = P_{\text{ngram}}\left(\mathtt{F} \mid w_{i_{-n+k}}^{i_{-n+k+1}},\dots,w_{i_{-1}}^{i_0}\right)$$

Remarque. Le calcul des α, β nécessite, pour chaque case, un calcul en $\mathcal{O}(|w|)$. Le nombre de cases à remplir correspond au cardinal de I, qui vaut $\sum_{k=1}^{n}\binom{|w|}{k}$ et est d'ordre $\mathcal{O}\left(|w|^n\right)$. Donc la complexité du calcul de α et β est en $\mathcal{O}\left(|w|^{n+1}\right)$.

Définis ainsi, pour tout $(i_{-n+k},\dots,i_0) \in I$, α et β permettent de calculer la probabilité que w soit généré et que lors de sa génération, $w_{i_{-n+k}}^{i_{-n+k+1}},\dots,w_{i_{-1}}^{i_0}$ apparaissent comme segments consécutifs : cette probabilité est donnée par le produit $\alpha(i_{-n+k},\dots,i_0)\cdot\beta(i_{-n+k},\dots,i_0)$.

Afin d'échantillonner une segmentation de w, on procède de la manière suivante :

1. Calculer les probabilités $\mathbb{P}(k) = \alpha(0,k) \cdot \beta(0,k)$ pour $k = 1 \dots |w|$. Puis normaliser ces probabilités (afin d'obtenir les probabilités que w_0^k apparaisse comme segment) et échantillonner un premier point de coupure k_1 selon celles-ci.
2. De manière générale, en supposant que des points de coupure $0 = k_0 < k_1 < \dots k_r < |w|$ ont déjà été placés, pour échantillonner le prochain point de coupure :
   — si $r < n-1$: utiliser les probabilités $\mathbb{P}(k) = \alpha(k_0, \dots, k_r, k) \cdot \beta(k_0, \dots, k_r, k)$ pour échantillonner le prochain point de coupure.
   — si $r \geq n-1$: utiliser les probabilités $\mathbb{P}(k) = \alpha(k_{r-n+2}, \dots, k_r, k) \cdot \beta(k_{r-n+2}, \dots, k_r, k)$ pour échantillonner le prochain point de coupure.

   ...et ce jusqu'à placer le dernier point de coupure à la fin du mot (en position $|w|$), étape à laquelle se termine l'échantillonnage de la segmentation.

3.2.4 Inférence du modèle orthographique

Nous appelerons *modèle orthographique* la distribution de probabilité (à support infini sur l'ensemble des chaînes de caractères) servant de distribution de base pour la génération de morphèmes potentiels, et reflétant les séquences de caractères typiques apparaissant dans la langue traitée. Dans le chapitre précédent, le modèle orthographique était la distribution de base (que nous avons noté G_0) du PYP engendrant la distribution unigramme. Ici, le modèle orthographique est la distribution située à la base de la hiérarchie, que nous avons noté $G_{\mathbf{base}}$.

Au chapitre précédent, nous avons utilisé comme modèle orthographique un simple modèle trigramme de caractères appris sur les données préalablement à l'inférence des segmentations. Il est également possible d'inférer le modèle orthographique en même temps que les segmentations, en le considérant comme un paramètre supplémentaire du modèle, engendré par un processus de Pitman-Yor hiérarchique.

Soit $p \in \mathbb{N} \cup \{\infty\}$. Considérons le processus de Pitman-Yor hiérarchique défini par :

— L'ensemble des *contextes* $\mathcal{C}$ est l'ensemble des chaînes de caractères de longueur $0 \leq l \leq p-1$.

— L'*arbre de repli* est l'application

$$\begin{aligned} \beta : \mathcal{C} &\longrightarrow \mathcal{C} \cup \{\mathbf{base}\} \\ c_{-l} \dots c_{-1} &\mapsto c_{-l+1} \dots c_{-1} \\ \emptyset \text{ (contexte vide)} &\mapsto \mathbf{base} \end{aligned}$$

— A chaque contexte c est attaché un adapteur $\eta_c : G \mapsto PYP(\alpha, d, G)$ (on utilisera les mêmes hyperparamètres α, d pour tous les adapteurs).

— La distribution de base $H_{\mathbf{base}}$ est une distribution uniforme sur l'ensemble des caractères de la langue traitée.

Une famille de distribution conditionnelles $(H_c)_{c\in\mathcal{C}}$ engendrée par ce processus hiérarchique définit un modèle p-gramme de caractères que nous noterons G_{ortho} et défini par :

$$\begin{aligned} G_{\text{ortho}}\big(\underbrace{c_1c_2\dots c_l\mathtt{f}}_{\text{chaîne de caractères}}\big) =& H_{\tau_p(\emptyset)}(c_1)\times H_{\tau_p(c_1)}(c_2)\times H_{\tau_p(c_1c_2)}(c_3)\\ &\times\dots\times H_{\tau_p(c_1\dots c_{l-1})}(c_l)\times H_{\tau_p(c_1\dots c_l)}(\mathtt{f})\\ &=\prod_{i=1}^{l+1} H_{\tau_p(c_1\dots c_{i-1})}(c_i) \end{aligned}$$

en notant :

— $\mathtt{f}$ un symbole spécial de fin de chaîne ($c_{l+1}=\mathtt{f}$).

— τ_p un opérateur de troncature qui ne conserve que les $p-1$ derniers caractères d'une chaîne (de sorte à obtenir un contexte dans $\mathcal{C}$) : $\tau_p(c_1\dots c_m)=c_{m-p+2}\dots c_m$.

Mettre en cache une chaîne de caractère $c_1\dots c_l$ revient alors à mettre en cache dans le PYP hiérarchique associé à G_{ortho} l'ensemble des transitions $\tau_p(c_1\dots c_{i-1})\to c_i$ pour $i=1\dots(l+1)$.

3.2.5 Expériences

Nous consignons ici les résultats d'expériences réalisées sur les mêmes données et selon le même protocole qu'avec le modèle unigramme dans le chapitre précédent, en utilisant cette fois un modèle bi-gramme de morphèmes. Les hyperparamètres α_c, d_c des différents PYP de la hiérarchie sont tous fixés à une même valeur. Les d_c sont fixés à 0.1, et les α_c sont fixés à une valeur commune α que nous avons fait varier selon les expériences. Comme modèle orthographique, nous avons utilisé d'une part un simple modèle trigramme appris sur les données avant l'inférence des segmentations (figure 3.6), d'autre part un modèle trigramme non-paramétrique inféré simultanément aux segmentations (figure 3.7).

— Le modèle bigramme obtient de meilleures performances que le modèle unigramme. De plus, le fait d'inférer un modèle orthographique non-paramétrique permet un gain de performances supplémentaire.

— Les meilleurs rappels (ainsi que les taux de segmentation les plus élevés) sont obtenus pour $\gamma\leq 0.5$ et les meilleures précisions pour $\gamma\geq 0.5$. Ceci appuie notre interprétation des expériences présentées au chapitre précédent, selon laquelle une présentation par types aura tendance à entraîner une sur-segmentation de par l'hétérogénéité des données auxquelles sera

γ : paramètre de *lissage* du corpus α : paramètre de force du PYP	$\gamma = 0$	$\gamma = 0.25$	$\gamma = 0.5$	$\gamma = 0.75$	$\gamma = 1$
$\alpha = 1$	P 42.0 R 71.0 F 52.8 S 27.6	P 47.3 R 71.1 F 56.8 S 24.6	P 45.5 R 65.2 F 53.6 S 23.5	P 47.0 R 63.9 F 54.2 S 22.2	P 51.3 R 62.7 F 56.4 S 20.0
$\alpha = 10$	P 42.9 R 71.7 F 53.7 S 27.4	P 45.2 R 70.5 F 55.0 S 25.5	P 45.7 R 67.1 F 54.3 S 24.1	P 47.0 R 61.8 F 53.4 S 21.5	P 50.7 R 61.9 F 55.7 S 20.0
$\alpha = 100$	P 45.8 R 71.5 F 55.8 S 25.6	P 45.3 R 68.9 F 54.6 S 24.9	P 47.7 R 65.0 F 55.0 S 22.3	P 56.5 R 61.9 F **59.1** S 17.9	P 51.1 R 53.8 F 52.4 S 17.2
$\alpha = 500$	P 48.2 R 69.2 F 56.8 S 23.5	P 52.5 R 67.1 F 58.9 S 20.9	P 57.5 R 73.2 F **64.4** S 20.8	P 54.7 R 58.8 F 56.7 S 17.6	P 58.0 R 57.4 F 57.7 S 16.2
$\alpha = 1000$	P 53.8 R 66.5 F **59.4** S 20.2	P 54.3 R 64.0 F 58.8 S 19.3	P 57.3 R 71.0 F **63.4** S 20.3	P 56.2 R 63.1 F **59.5** S 18.4	P 58.2 R 58.8 F 58.5 S 16.6

FIGURE 3.6 – Evaluation des segmentations produites par un modèle bigramme de morphèmes, au bout de 1673770 itérations (dix *epochs*). Le modèle orthographique est un modèle trigramme calculé sur les données avant l'inférence des segmentations. P = Précision, R = Rappel, F = F-mesure, S = Taux de segmentation. Ces quatres valeurs sont exprimées en pourcentages. Les cinq meilleures F-mesures sont en gras, les cinq meilleures précisions en rouge, les cinq meilleurs rappels en bleu, et les cinq taux de segmentation les plus élevés en vert.

exposé l'algorithme. Une conséquence de la sur-segmentation est l'augmentation du rappel et la diminution de la précision.

3.3 Modèle semi-markovien de classes morphologiques

Nous considérons à présent un modèle semi-markovien caché (SHMM : *semi hidden Markov model*). Un tel modèle modélise une forme de mot comme résultant de la génération (de manière markovienne) d'une suite aléatoire d'identifiants de classes morphologiques, chaque identifiant générant à son tour un morphème. L'inférence d'un tel modèle à partir de données non segmentées permet ainsi d'obtenir :

— des segmentations en morphèmes des mots présents dans les données.

— un regroupement des morphèmes obtenus en différentes classes morphologiques.

γ : paramètre de *lissage* du corpus α : paramètre de force du PYP	$\gamma = 0$	$\gamma = 0.25$	$\gamma = 0.5$	$\gamma = 0.75$	$\gamma = 1$
$\alpha = 1$	P 48.0 R 71.0 F 57.3 S 24.2	P 51.5 R 73.8 F 60.7 S 23.5	P 42.0 R 71.1 F 52.8 S 27.7	P 62.0 R 65.9 F 63.9 S 17.4	P 69.7 R 60.4 F 64.7 S 14.2
$\alpha = 10$	P 50.7 R 85.2 F 63.6 S 27.5	P 61.0 R 83.6 F **70.5** S 22.4	P 57.1 R 79.7 F **66.5** S 22.9	P 63.5 R 67.0 F 65.2 S 17.3	P 59.6 R 63.2 F 61.4 S 17.4
$\alpha = 100$	P 47.3 R 72.1 F 57.2 S 25.0	P 49.1 R 72.8 F 58.7 S 24.3	P 55.0 R 79.4 F 65.0 S 23.6	P 68.4 R 62.8 F 65.5 S 15.0	P 77.1 R 56.4 F 65.1 S 12.0
$\alpha = 500$	P 55.9 R 68.6 F 61.6 S 20.1	P 48.6 R 79.1 F 60.2 S 26.6	P 56.9 R 68.0 F 62.0 S 19.6	P 70.1 R 62.3 F **66.0** S 14.6	P 84.7 R 61.5 F **71.3** S 11.9
$\alpha = 1000$	P 53.7 R 76.1 F 63.0 S 23.2	P 64.2 R 67.1 F 65.6 S 17.1	P 57.8 R 75.0 F 65.3 S 21.2	P 75.8 R 62.2 F **68.3** S 13.4	P 35.2 R 65.2 F 45.7 S 30.3

FIGURE 3.7 – Evaluation des segmentations produites par un modèle bigramme de morphèmes, au bout de 1673770 itérations (dix *epochs*). Le modèle orthographique est un modèle trigramme de caractères engendré par un PYP hiérarchique, tel que décrit à la section 3.2.4.

— une chaîne de Markov modélisant la manière dont ces classes se succèdent, constituant une *morphotactique* (c'est-à-dire une grammaire à l'échelle des mots régissant les successions de morphèmes).

Notre but dans cette section ne se limite pas à obtenir des bonnes segmentations : nous souhaiterions déterminer si, de manière non-supervisée, l'algorithme est capable d'inférer une morphotactique linguistiquement pertinente.

3.3.1 Génération de formes de mots

Notons n l'ordre markovien du modèle, et K un ensemble (éventuellement infini) d'*identifiants de classes*, contenant deux symboles spéciaux : `I` ("classe initiale") et `F` ("classe finale"). Les paramètres du modèle sont deux familles de distributions :

— Des distributions $\left(G^{\kappa}_{\kappa_1 \ldots \kappa_{m-1}}(\kappa_m)\right)_{\kappa_1,\ldots,\kappa_{m-1} \in K}$ où $1 \leq m \leq n$. Ces distributions déterminent un *modèle de séquence de classes* qui est une chaîne de Markov d'ordre n sur K. Etant données ces distributions, la probabilité

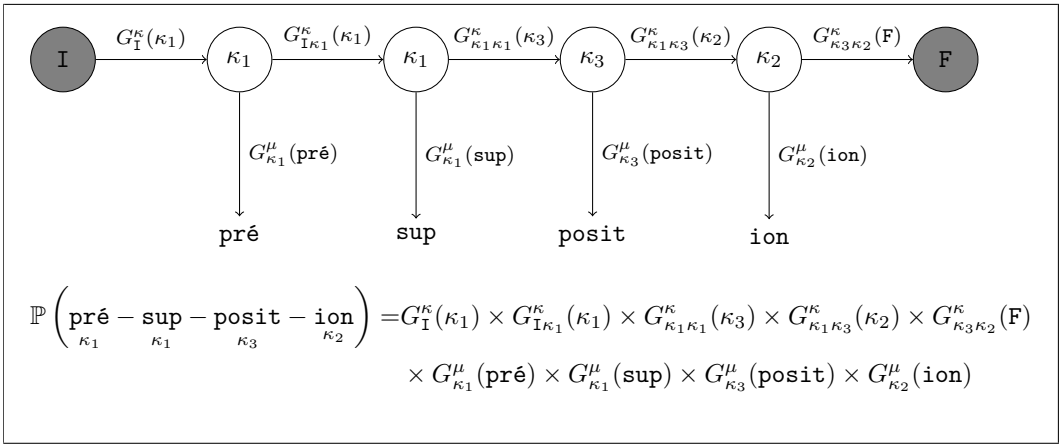


FIGURE 3.8 – Modèle semi-markovien caché (d'ordre 3) de génération de formes de mots : exemple d'analyse d'un mot ainsi que sa probabilité.

d'engendrer une séquence de classes $\boldsymbol{\kappa} = \underset{=\mathtt{I}}{\kappa_0}\ \kappa_1 \dots \kappa_r\ \underset{=\mathtt{F}}{\kappa_{r+1}}$ est donnée par :

$$\mathbb{P}(\boldsymbol{\kappa}) = \prod_{i=1}^{r+1} G_{\tau_n(\kappa_0 \dots \kappa_{i-1})}(\kappa_i)$$

où τ_n est un opérateur de troncature ne conservant que les $(n-1)$ derniers éléments d'une séquence :

$$\begin{aligned} \tau_n(\kappa_0 \dots \kappa_{i-1}) &= \kappa_0 \dots \kappa_{i-1} && \text{si } i \leq n-1 \\ &= \kappa_{i-n+1} \dots \kappa_{i-1} && \text{sinon} \end{aligned}$$

— Des distributions $\left(G^{\mu}_{\kappa}(\mu)\right)_{\kappa \in K}$ déterminant un *modèle d'émission de morphèmes*. Ce modèle représente quels morphèmes chaque identifiant de classe sera susceptible d'engendrer, et détermine par conséquent la manière dont les morphèmes seront regroupés en différentes classes.

Le scénario génératif est le suivant : une chaîne d'identifiants de classes $\boldsymbol{\kappa} = \underset{=\mathtt{I}}{\kappa_0}\ \kappa_1 \dots \kappa_r\ \underset{=\mathtt{F}}{\kappa_{r+1}}$ est engendrée selon la chaîne de Markov déterminée par le modèle de séquence de classes. Cette chaîne d'identifiants commence par le symbole `I` et se termine dès que le symbole `F` est engendré. Ensuite, chaque identifiant κ_i (i=1 ... r) émet un morphème μ_i selon la distribution $G^{\mu}_{cls_i}(\mu_i)$. La forme de mot générée est la concaténation $\mu_1 \dots \mu_r$.

Ce scénario génératif engendre donc des couples $(\boldsymbol{\kappa}, \boldsymbol{\mu})$ où $\boldsymbol{\kappa}$ est une suite d'identifiants de classes et $\boldsymbol{\mu}$ est une suite de morphèmes. Pour une forme de mot w, nous appelerons une *analyse de* w un couple $(\boldsymbol{\kappa}, \boldsymbol{\mu})$ tel que la concaténation des morphèmes de $\boldsymbol{\mu}$ donne w.

3.3.2 A prioris

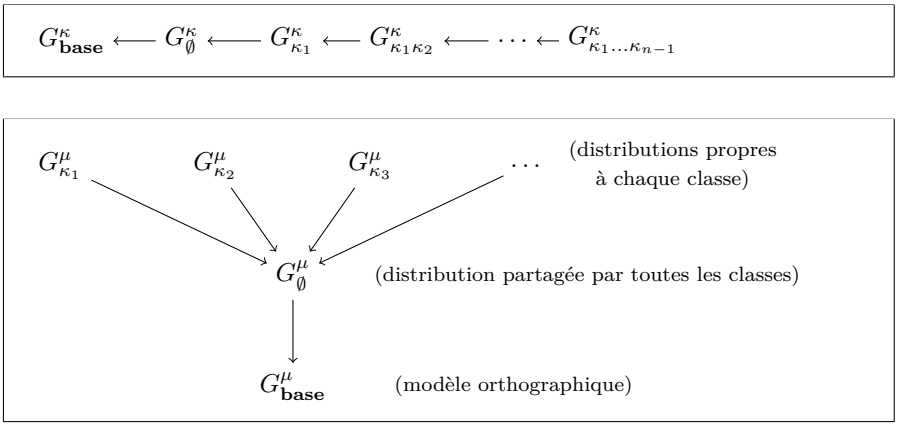


FIGURE 3.9 – Processus de Pitman-Yor hiérarchiques engendrant les modèles de séquence de classes (en haut) et le modèles d'émission de morphèmes (en bas)

Le modèle de séquence de classes et le modèle d'émission de morphèmes sont chacun engendrés par des PYP hiérarchiques dont les structures sont données dans la figure 3.9.

Quelques remarques :

- Les distributions d'émission de morphèmes par classe se replient toutes sur une distribution commune (notée $G^{\mu}_{\emptyset}$). Ainsi, lors de l'échantillonnage des analyses des mots du corpus, chaque segment qui aura été retenu comme morphème potentiel apparaîtra comme étiquette d'au moins une table du CRP associé à $G^{\mu}_{\emptyset}$, et ce quelque soit sa classe. Ceci permet de changer la classe affectée à un segment tout en continuant à le considérer comme un morphème potentiel.
- La distribution de base utilisée pour le modèle de séquence de classes peut être choisie de la manière suivante :
  - Si on ne veut autoriser qu'un nombre fini de classes (c'est-à-dire si K est un ensemble fini), on peut choisir comme $G^{\kappa}_{\mathbf{base}}$ une distribution uniforme sur K.
  - Il est également possible de ne pas limiter le nombre de classes et de laisser le modèle créer autant de classes que nécessaire afin d'obtenir un compromis entre parcimonie du modèle et fidélité aux données. Pour cela, K doit être un ensemble infini, et on peut alors choisir $G^{\kappa}_{\mathbf{base}}$ de sorte qu'elle alloue une masse nulle à tous les singletons [4]

4. Un exemple (un peu artificiel) d'une telle distribution serait par exemple la distribution uniforme sur l'intervalle $[0, 1]$, auquel cas les classes seraient identifiées par des nombres réels entre 0 et 1.

de K. Ainsi, il sera impossible que deux échantillons de $G^{\kappa}_{\mathbf{base}}$ correspondent au même identifiant de classe. Autrement dit, toutes les tables du CRP associé à $G^{\kappa}_{\emptyset}$ seront étiquetées par des identifiants de classe différents, et il y aura donc autant de classes inférées que de tables dans le CRP de $G^{\kappa}_{\emptyset}$.

L'inférence suit la même méthode que pour le modèle n-gramme de morphèmes (section 3.2.3), à ceci près que lors de l'échantillonnage de Gibbs, plutôt que d'échantillonner chaque segmentation conditionnellement aux autres, on échantillonnera chaque *analyse* conditionnellement aux autres. Une fois une analyse ($\boldsymbol{\kappa} = \kappa_0 \dots \kappa_{r+1}, \boldsymbol{\mu} = \mu_1 \dots \mu_r$) échantillonnée, la mettre en cache revient à mettre en cache :

— Toutes les transitions $\tau_n(\kappa_0 \dots \kappa_{i-1}) \to \kappa_i$ pour $i = 1 \dots (r+1)$.

— Toutes les émissions $\kappa_i \to \mu_i$ pour $i = 1 \dots r$.

Il reste donc à détailler comment efficacement échantillonner une analyse étant donné un mot. La méthode que nous présentons est encore une méthode de type *forward-backward*, adaptée à la structure du modèle.

3.3.3 Echantillonnage d'une analyse

Comme dans la section 3.2.3, on considère deux tableaux α et β que l'on remplit par programmation dynamique. α et β sont indexés par des éléments d'un ensemble I, qu'on partitionne en un "intérieur" et en un "bord". Le "bord" correspond aux indices pour lesquels le symbole de classe initiale `I` interviendra dans l'équation de récurrence.

Ici, l'ensemble I considéré est :

$$I = \overbrace{\left\{(i, \kappa_{-n}, \kappa_{-n+1}, \dots, \kappa_{-1}) \mid \kappa_{-n}, \dots, \kappa_{-1} \neq \mathtt{I}, \mathtt{F} \text{ et } n \leq i \leq |w|\right\}}^{=\mathring{I} \text{ (“intérieur” de } I)}$$
$$\cup \underbrace{\left\{(i, \mathtt{I}, \kappa_{-m}, \dots, \kappa_{-1}) | \kappa_{-m}, \dots, \kappa_{-1} \neq \mathtt{I}, \mathtt{F} \text{ et } 0 \leq m \leq n-1 \text{ et } i \geq m\right\}}_{=\partial I \text{ (“bord” de } I)}$$

Pour tout $\mathbf{i} \in I$, notons $P^{\kappa}_{\mathbf{i}}$ la probabilité induite par le CRP associé à la distribution aléatoire $G^{\kappa}_{\mathbf{i}}$. Dé même, pour tout $\kappa \in K \setminus \{\mathtt{I}, \mathtt{F}\}$, notons P^{μ}_{κ} la probabilité induite par le CRP associé à la distribution aléatoire G^{μ}_{κ}.

On définit alors α par les équations de récurrence suivantes :

— si $(i, \kappa_{-n}, \kappa_{-n+1}, \dots, \kappa_{-1}) \in \mathring{I}$: (cas récurrent)

$$\alpha(i, \kappa_{-n}, \kappa_{-n+1}, \dots, \kappa_{-1}) = \sum_{j=n}^{i-1} \sum_{\kappa_{-n-1} \in K \setminus \{\mathtt{I},\mathtt{F}\}} \alpha(j, \kappa_{-n-1}, \dots, \kappa_{-2}) \times P^{\kappa}_{\kappa_{-n} \dots \kappa_{-2}}(\kappa_{-1}) \times P^{\mu}_{\kappa_{-1}}(w^i_j)$$
$$+ \sum_{j=n-1}^{i-1} \alpha(j, \mathtt{I}, \kappa_{-n}, \dots, \kappa_{-2}) \times P^{\kappa}_{\kappa_{-n} \dots \kappa_{-2}}(\kappa_{-1}) \times P^{\mu}_{\kappa_{-1}}(w^i_j)$$

— si $(i, \mathtt{I}, \kappa_{-m}, \ldots, \kappa_{-1}) \in \partial I$: (cas de "bord")

— si $m \geq 1$:

$$\alpha(i, \mathtt{I}, \kappa_{-m}, \ldots, \kappa_{-1}) = \sum_{j=m-1}^{i-1} \alpha(j, \mathtt{I}, \kappa_{-m}, \ldots, \kappa_{-2}) \times P^{\kappa}_{\mathtt{I}\,\kappa_{-m}\ldots\kappa_{-2}}(\kappa_{-1}) \times P^{\mu}_{\kappa_{-1}}(w_j^i)$$

— si $m = 0$:

$$\begin{aligned} \alpha(i, \mathtt{I}) &= 1 \text{ si i=0} \\ &= 0 \text{ sinon} \end{aligned}$$

De même, β est défini par les équations de récurrence suivantes :

— si $(i, \kappa_{-n}, \kappa_{-n+1}, \ldots, \kappa_{-1}) \in \mathring{I}$: (cas récurrent)

$$\begin{aligned} \beta(i, \kappa_{-n}, \kappa_{-n+1}, \ldots, \kappa_{-1}) &= P^{\kappa}_{\kappa_{-n+1}\ldots\kappa_{-1}}(\mathtt{F}) && \text{si i = |w|} \\ &= \sum_{j=i+1}^{|w|} \sum_{\kappa_0 \in K \setminus \{\mathtt{I},\mathtt{F}\}} P^{\kappa}_{\kappa_{-n+1}\ldots\kappa_{-1}}(\kappa_0) \times P^{\mu}_{\kappa_0}(w_i^j) \\ &\qquad \times \beta(j, \kappa_{-n+1}, \ldots, \kappa_0) \text{ si } i < |w| \end{aligned}$$

— si $(i, \mathtt{I}, \kappa_{-m}, \ldots, \kappa_{-1}) \in \partial I$: (cas de "bord")

$$\begin{aligned} \beta(i, \mathtt{I}, \kappa_{-m}, \ldots, \kappa_{-1}) &= P^{\kappa}_{\mathtt{I}\,\kappa_{-m}\ldots\kappa_{-1}}(\mathtt{F}) && \text{si i = |w|} \\ &= \sum_{j=i+1}^{|w|} \sum_{\kappa_0 \in K \setminus \{\mathtt{I},\mathtt{F}\}} P^{\kappa}_{\mathtt{I}\,\kappa_{-m}\ldots\kappa_{-1}}(\kappa_0) \times P^{\mu}_{\kappa_0}(w_i^j) \\ &\qquad \times \beta(j, \kappa_{-m}, \ldots, \kappa_0) \text{ si } i < |w| \text{ et } m = n-1 \\ &= \sum_{j=i+1}^{|w|} \sum_{\kappa_0 \in K \setminus \{\mathtt{I},\mathtt{F}\}} P^{\kappa}_{\mathtt{I}\,\kappa_{-m}\ldots\kappa_{-1}}(\kappa_0) \times P^{\mu}_{\kappa_0}(w_i^j) \\ &\qquad \times \beta(j, \mathtt{I}, \kappa_{-m}, \ldots, \kappa_0) \text{ si } i < |w| \text{ et } m < n-1 \end{aligned}$$

Pour tout $\mathbf{i} = (i, \kappa_{-m}, \ldots, \kappa_{-1}) \in I$, le produit $\alpha(\mathbf{i})\beta(\mathbf{i})$ est la probabilité que w soit engendré, et que la position i soit une limite entre morphèmes, et que $\kappa{-}m, \ldots, \kappa_{-1}$ soient les identifiants de classe précédant le point de coupure situé en i.

L'échantillonnage d'une analyse se déroule comme suit :

1. Calculer les probabilités $\mathbb{P}(i, \kappa) = \alpha(i, \mathtt{I}, \kappa) \cdot \beta(i, \mathtt{I}, \kappa)$ pour $i = 1 \ldots |w|$. Puis normaliser ces probabilités, et échantillonner un premier point de coupure i_1 et un premier identifiant de classe κ_1.
2. De manière générale, en supposant que des points de coupure $0 = i_0 < i_1 < \ldots < i_r < |w|$ ont déjà été placés, et que chaque segment $w_{i_{s-1}}^{i_s}$ a été affecté à une classe κ_s, le prochain point de coupure et la prochaine classe sont échantillonnés :

— En calculant et en renormalisant les probabilités $\mathbb{P}(i, \kappa) = \alpha(i, \kappa_{r-n+2}, \ldots, \kappa_r, \kappa) \cdot \beta(i, \kappa_{r-n+2}, \ldots, \kappa_r, \kappa)$ si $r \geq n-1$.

— En calculant et en renormalisant les probabilités $\mathbb{P}(i, \kappa) = \alpha(i, \mathtt{I}, \kappa_1, \ldots, \kappa_r, \kappa) \cdot \beta(i, \mathtt{I}, \kappa_1, \ldots, \kappa_r, \kappa)$ si $r < n-1$.

...et ce jusqu'à placer le dernier point de coupure à la fin du mot, étape à laquelle se termine l'échantillonnage de l'analyse.

Remarque. Le calcul de α et β nécessite, pour chaque case (c'est-à-dire chaque élément de I), un calcul de complexité $\mathcal{O}(|w| \cdot |K|)$ (par abus de notation, on note $|K|$ non pas le nombre de classes potentielles qui peut être infini, mais le nombre de classes inférées parmi l'ensemble des analyses). Le cardinal de I est d'ordre $\mathcal{O}\left(|w| \cdot |K|^{n-1}\right)$. Donc la complexité de l'échantillonnage est $\mathcal{O}\left(|w|^2 \cdot |K|^n\right)$.

3.3.4 Expériences

3.3.4.1 Protocole

Les expériences ont été réalisées sur les mêmes données que dans le chapitre précédent, avec toujours différents *paramètres de lissage* du corpus.

— Pour les PYP du processus hiérarchique associé au modèle de séquence de classes, nous avons fixé le paramètre de décompte à $\delta = 0.1$, et le paramètre de force à $\alpha = 1$ ou 10. Nous n'avons pas jugé nécessaire d'essayer des α plus élevés, étant donné que contrairement aux distributions d'émissions de morphèmes, les distributions de transitions entre classes ne présentent pas une variété très élevée d'évènements (le nombre d'évènements qu'elles peuvent engendrer correspondant au plus au nombre de classes morphologiques).

— Pour les PYP du processus hiérarchique associé aux modèles d'émission de morphèmes, nous avons fixé tous le δ à 0.1, nous avons affectéaffecté à chaque G^{μ}_{κ} un α pris aléatoirement dans $\{10, 100, 1000\}$, et nous avons affecté à $G^{\mu}_{\emptyset}$ (distribution partagée par toutes les classes) un $\alpha = 1000$. La raison de notre choix d'un α aléatoire parmi $\{10, 100, 1000\}$ est que nous souhaitions orienter le modèle vers une distinction entre différents degrés d"'ouverture" des classes, à savoir distinguer entre :

  — Des classes morphologiques "fermées" présentant une faible variété de morphèmes (par exemple : contenant des affixes courants ou des mots-outils), la fermeture d'une classe étant entraînée par le choix d'un α faible.

  — Des classes morphologiques "ouvertes" présentant une plus grande variété (par exemple : des racines, des emprunts à d'autres langues...), l'ouverture d'une classe étant entraînée par le choix d'un α élevé.

Enfin nous avons essayé de fixer le nombre de classes à six classes (figure 3.10) ou au contraire de laisser la liberté au modèle de fixer le nombre de classes (figure 3.11).

3.3.4.2 Résultats quantitatifs

γ : paramètre de *lissage* du corpus α : paramètre de force des PYP associés aux modèles de séquences de classes	$\gamma = 0$	$\gamma = 0.25$	$\gamma = 0.5$	$\gamma = 0.75$	$\gamma = 1$
$\alpha = 1$	P 36.6 R 64.3 F 46.6 S 34.0 N 14240	P 45.3 R 67.1 F 54.1 S 24.4 N 18071	P 53.0 R 67.9 F 59.5 S 15.9 N 23987	P 40.1 R 81.0 F 53.6 S 24.4 N 7565	P 52.7 R 66.1 F 58.6 S 10.4 N 13748
$\alpha = 10$	P 48.4 R 64.9 F 55.5 S 25.0 N 16259	P 48.1 R 68.0 F 56.3 S 23.7 N 18330	P 64.0 R 73.9 F **68.6** S 12.9 N 28379	P 52.0 R 68.0 F 58.9 S 12.4 N 21202	P 67.3 R 53.7 F 59.7 S 3.3 N 20722

FIGURE 3.10 – Résultats des expériences pour le modèle semi-markovien de classes, avec un nombre de classes fixé à six. P = Précision, R = Rappel, F = F-mesure, S = Taux de segmentation, N = Nombre de tables dans le restaurant associé à la distribution partagée. Les quatres premières valeurs sont exprimées en pourcentages. En gras : meilleure F-mesure, en rouge : meilleure précision, en bleu : meilleur rappel, en vert : taux de segmentation le plus élevé, en orange : nombre de tables le plus élevé.

γ : paramètre de *lissage* du corpus α : paramètre de force des PYP associés aux modèles de séquences de classes	$\gamma = 0$	$\gamma = 0.25$	$\gamma = 0.5$	$\gamma = 0.75$	$\gamma = 1$
$\alpha = 1$	P 47.4 R 65.2 F 54.9 S 23.9 N 25058 C 6	P 47.2 R 69.0 F 56.1 S 23.2 N 20435 C 5	P 54.2 R 67.0 F 59.9 S 12.6 N 25532 C 6	P 58.2 R 75.2 F **65.7** S 10.6 N 20114 C 6	P 41.4 R 64.4 F 50.4 S 11.0 N 11222 C 6
$\alpha = 10$	P 44.9 R 48.9 F 46.8 S 10.2 N 26996 C 5	P 57.7 R 54.8 F 56.2 S 7.6 N 23003 C 5	P 54.1 R 57.3 F 55.6 S 11.3 N 13810 C 6	P 32.9 R 59.4 F 42.3 S 20.0 N 6257 C 7	P 35.7 R 50.6 F 41.9 S 15.9 N 4610 C 7

FIGURE 3.11 – Résultats des expériences pour le modèle semi-markovien de classes, avec un nombre de classes potentiellement infini. C = Nombre de classes inférées.

Les résultats quantitatifs montrent le peu de robustesse de la méthode : la performance du modèle est très sensible aux choix du paramètre de lissage γ, et des hyperparamètres des PYP. Toutefois, les résultats confirment que la performance est améliorée lorsqu'on effectue un compromis entre une présentation des exemples par types ou par occurences (les meilleurs résultats sont obtenus avec $\gamma = 0.5$ lorsque le nombre de classes est fixé, et avec $\gamma = 0.75$ lorsqu'il ne l'est

pas). La meilleure F-mesure est obtenue en fixant le nombre de classes, mais globalement, les performances sont comparables selon que l'on fixe le nombre de classes ou non.

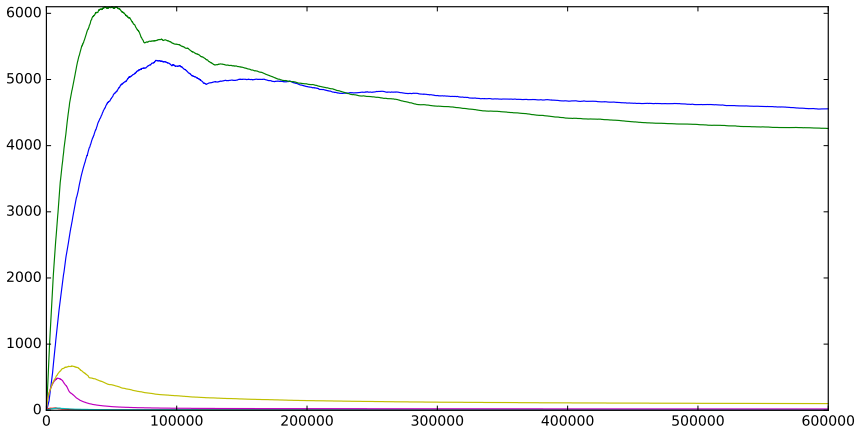


FIGURE 3.12 – Evolution du nombre de morphèmes affectés à chaque classe durant l'inférence. En abscisse : le nombre d'itérations. En ordonnée : le nombre de morphèmes. Chaque courbe correspond à une classe.

La figure 3.13 montre l'évolution du taux de segmentation et de la qualité du modèle au cours de l'inférence. Dans la figure 3.12, nous avons représenté l'évolution de la population de chaque classe au cours de l'inférence (pour l'expérience ayant obtenu la meilleure F-mesure). Nous constatons que cette évolution se fait en deux phases :

— Dans une première phase (correspondant à peu près à la première *epoch*), le nombre de morphèmes de chaque classe augmente brusquement, et une divergence nette entre classes ouvertes et classes fermées apparaît. En effet, à la fin de la première *epoch*, le modèle doit être capable d'expliquer l'ensemble des données, ce qui explique la prolifération de morphèmes durant cette phase.

— A la fin de cette première phase, le modèle dispose de suffisamment de morphèmes pour expliquer les données, mais le lexique de morphèmes n'est pas encore optimisé. Dans une deuxième phase, le modèle cherche à améliorer son lexique de morphèmes à l'aune du critère de parcimonie imposé par les PYP : ceci explique pourquoi au cours de cette deuxième phase, on observe une diminution puis une stabilisation du nombre de morphèmes.

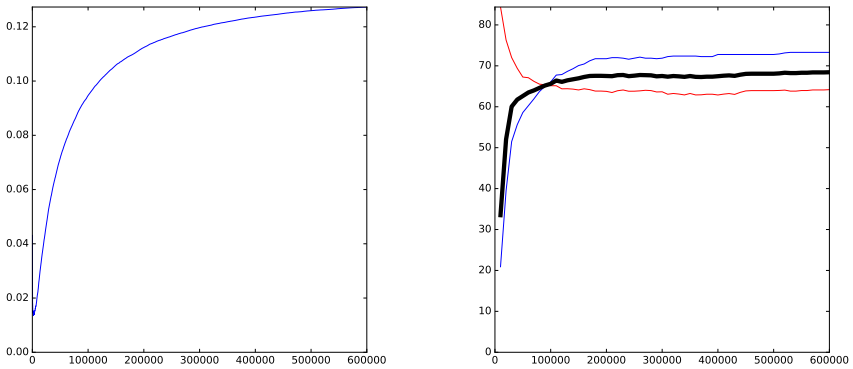


FIGURE 3.13 – Evolution du taux de segmentation (à gauche) et de la précision/rappel/F-mesure (à droite). La précision est en rouge, le rappel en bleu, la F-mesure en gras.

3.3.4.3 Morphotactique induite

Dans la figure 3.14, nous avons représenté la morphotactique induite par le modèle ayant obtenu la meilleure F-mesure. Bien que le nombre de classes dans ce modèle était fixé à six, les deux dernières classes apparaissaient trop rarement pour que nous ayons jugé intéressant de les représenter. La morphotactique induite est loin de correspondre à celle que pourrait produire un linguiste : les classes sont hétérogènes, difficilement interprétables, et contiennent beaucoup de "faux morphèmes". Toutefois, on peut distinguer quelques tendances générales. Tout d'abord, affecter des α différents selon les classes à bien entraîné une divergence entre classes fermées (correspondant aux $\alpha = 100$) et classes ouvertes (correspondant aux $\alpha = 1000$). De plus, dans une certaine mesure, on peut observer une cohérence dans les classes inférées :

— La classe κ_1 contient principalement des mots-outils et des racines.

— La classe κ_2 contient principalement des mots-outils, des racines et des préfixes (`in-, de-, re-, pre-, dis-, ex-, co-`) ainsi que des morphèmes qui peuvent à la fois apparaître comme mots-outils isolés ou comme préfixes (`over, out, with, up, side, off, some, after,...`)

— La classe κ_3 contient des suffixes fréquents (`-s, -ed, -ing, 's, -y, -an, -ness`)

— La classe κ_4 est difficilement interprétable : on y trouve quelques suffixes (`-en, -ment, -able`) mais aussi des racines courtes et des faux morphèmes.

Conclusion

Les *processus de Pitman-Yor hiérarchiques* engendrent des familles de distributions conditionnelles liées par des dépendances modélisées par un *arbre de repli*. Ainsi, pour un modèle n-gramme de morphèmes, pour modéliser le fait que deux contextes $\mu_1\mu_2\dots\mu_{n-1}$ et $\mu'_1\mu_2\dots\mu_{n-1}$ tendent à être suivis par des mots similaires, on peut faire se replier les deux distributions conditionnelles associées à ces contextes vers une même distribution de base associée au contexte $\mu_2\dots\mu_{n-1}$. Le fait de se replier vers des distributions de plus en plus agnostiques entraîne de plus un lissage des distributions *a posteriori*, ce qui contribue à améliorer la généralisation aux formes de mots absentes des données. Il est également possible d'obtenir des analyses morphologiques où les morphèmes sont regroupés en classes, en couplant un modèle de séquence de classes morphologiques à un modèle de génération de morphèmes par classe, en laissant libre cours au modèle pour adapter le nombre de classes à la complexité des données. Pour ces deux modèles, les deux points cruciaux pour l'inférence sont : le maintien d'un CRP par contexte pour calculer les distributions *a posteriori*, et l'utilisation d'une stratégie de programmation dynamique adéquate pour l'échantillonnage des segmentations. Les performances du modèle *n-gramme* sont généralement meilleures que celles du modèle de classes. Néanmoins ce dernier est intéressant du point de vue de la représentation morphologique plus riche et structurée qu'il permet d'obtenir : il y a une relative cohérence au sein des classes obtenues, d'autant plus qu'il est possible d'introduire un biais vers la distinction entre classes fermées et ouvertes en imposant des paramètres de force très différents aux distributions modélisant les différentes classes. Toutefois, les analyses produites par le modèle de classes restent loin d'être satisfaisantes du point de vue linguistique, notamment en raison du cadre entièrement non-supervisé que nous nous sommes imposé.

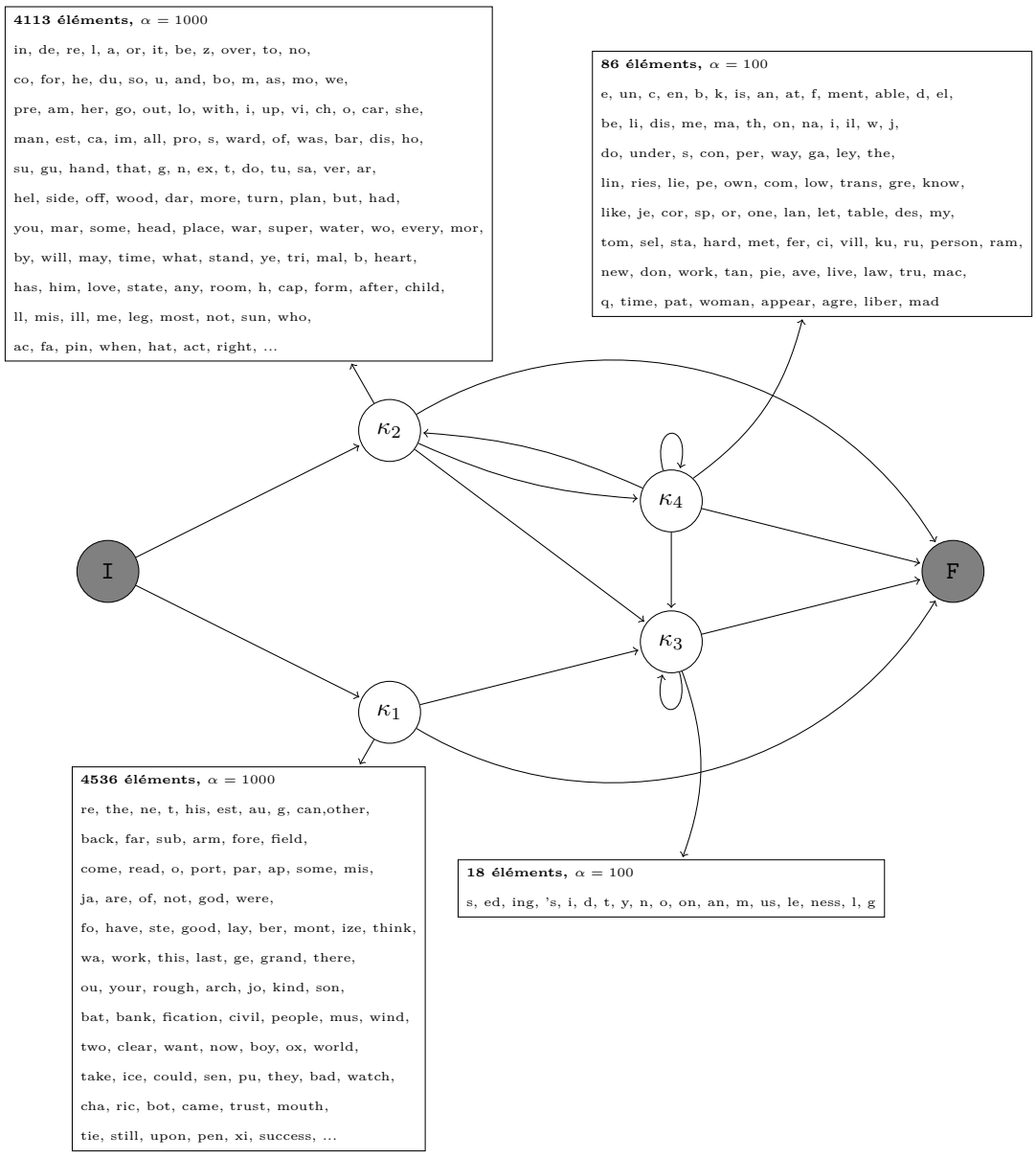


FIGURE 3.14 – Exemple de morphotactique apprise. Seuls les 80% des évènements les plus fréquents (transitions et émissions) sont représentés. Les morphèmes sont listés par fréquence décroissante.

Chapitre 4

Choix des hyperparamètres

Nous abordons dans ce chapitre la question du choix des hyperparamètres dans des hiérarchies de processus de Dirichlet ou de Pitman-Yor. Dans les chapitres précédents, nous avons réalisé des expériences en fixant différentes valeurs d'hyperparamètres, et nous avons vu que leur choix exerçait une influence non négligeable sur la qualité des segmentations produites. Nous souhaiterions disposer d'une méthode de segmentation robuste, c'est-à-dire ne nécessitant pas d'effectuer de choix arbitraires d'hyperparamètres influençant le résultat. Une solution possible serait d'utiliser des *données de développement* pour ajuster les hyperparamètres, mais ceci nécessiterait d'une part de disposer de données segmentées (et donc de renoncer au cadre non-supervisé), d'autre part de lancer un grand nombre de processus de segmentation (un processus de segmentation par jeu d'hyperparamètres) et donc un temps de calcul important. Nous proposons une autre solution, qui consiste à considérer les hyperparamètres comme des variables supplémentaires du modèle, soumises elles-mêmes à un a priori (*hyperprior*), et dont les valeurs seront échantillonnées conjointement aux segmentations.

4.1 Approche bayésienne de l'inférence d'hyperparamètres

4.1.1 *Hyperpriors*

Dans un premier temps (avant de considérer le cas des processus hiérarchiques), considérons le cas d'un processus de Pitman-Yor de distribution de base G_0 fixée. Plutôt que de fixer ses hyperparamètres α et δ, nous les considérons comme des variables aléatoires indépendantes engendrées par des distributions de densités notées respectivement $f_\alpha : \mathbb{R}_+^* \to \mathbb{R}_+$ et $f_\delta : [0,1] \to \mathbb{R}_+$. Ces distributions joueront le rôle d'a prioris sur les hyperparamètres, et nous les appelerons *hyperpriors*.

On considère alors un scénario génératif en trois étapes :

1. $\alpha > 0$ et $\delta \in [0, 1]$ sont engendrés : $\alpha \sim f_\alpha d\alpha$ et $\delta \sim f_\delta d\delta$.
2. Une distribution G est engendrée : $G \sim PYP(\alpha, \delta, G_0)$.
3. La distribution G engendre un jeu d'observations $\mathbf{e} = (e_1, \ldots, e_n) : e_i \sim G$.

Ce scénario génératif définit une distribution jointe sur α, δ, G et $\mathbf{e}$ donnée par :

$$\begin{aligned} d\mathbb{P}(\alpha, \delta, G, \mathbf{e}) = & f_\alpha(\alpha)d\alpha \cdot f_\delta(\delta)d\delta && \text{(a prioris sur } \alpha \text{ et } \delta) \\ & \cdot d\mathbb{P}(G|\alpha, \delta) && \text{(a priori de Pitman-Yor sur } G) \\ & \cdot \prod_{e_i \in \mathbf{e}} G(e_i) && \text{(probabilité de génération des observations)} \end{aligned}$$

Remarque. On utilisera par exemple :

— Un a priori exponentiel pour α, de sorte à permettre toutes les valeurs réels positives possibles tout en favorisant des valeurs faibles (ce qui traduit une exigence de parcimonie sur α) :
$\alpha \sim Exp(A) \quad f_\alpha(\alpha)d\alpha = Ae^{-A\cdot\alpha}d\alpha$

— Un a priori sur δ donné par une loi bêta :
$\delta \sim Beta(D_1, D_2) \quad f_\delta(\delta)d\delta = \dfrac{\delta^{D_1-1}(1-\delta)^{D_2-1}}{B(D_1, D_2)}d\delta$
en prenant D_1 proche de 0 et D_2 proche de 1 de sorte l'a priori favorise des valeurs de δ proches de 0 (rappelons que l'espérance d'une telle loi vaut $\dfrac{D_1}{D_1 + D_2}$). Ceci traduit une exigence de parcimonie sur δ, puisque des valeurs de δ proches de 0 rendent la création de nouvelles tables dans les CRP plus difficile.

4.1.2 *A posteriori* sur les hyperparamètres

Inférer la valeur des hyperparamètres selon ce scénario consiste alors, étant donné un jeu d'observations $\mathbf{e}$, à échantillonner la distribution *a posteriori* $d\mathbb{P}(\alpha, \delta \mid \mathbf{e})$, qui peut se réexprimer de la manière suivante

$$\begin{aligned} d\mathbb{P}(\alpha, \delta \mid \mathbf{e}) &= \frac{1}{Z(\mathbf{e})} d\mathbb{P}(\alpha, \delta, \mathbf{e}) \\ &= \frac{1}{Z(\mathbf{e})} \int_G d\mathbb{P}(\alpha, \delta, G, \mathbf{e}) \\ &= \frac{1}{Z(\mathbf{e})} \cdot f_\alpha(\alpha)d\alpha \cdot f_\delta(\delta)d\delta \cdot \underbrace{\int_G \left(\prod_{e_i \in \mathbf{e}} G(e_i) \right) d\mathbb{P}(G \mid \alpha, \delta)}_{(\star)} \end{aligned}$$

où $Z(\mathbf{e}) = \iint_{\alpha,\delta} d\mathbb{P}(\alpha, \delta, \mathbf{e})\ d\delta\ d\alpha$ est une certaine constante de normalisation ne dépendant que de $\mathbf{e}$.

Le terme $(\star)$ peut s'approcher en mettant en cache la suite d'observations $\mathbf{e}$ dans un restaurant chinois, et en calculant la probabilité (définie par le processus de restaurant chinois) de l'état du restaurant obtenu. Cette probabilité vaut :

$$
\begin{aligned}
&\overbrace{\frac{(\alpha+0\cdot\delta)G_0(l_1)}{0+\alpha}}^{\text{(ouverture de la première table)}}\ \overbrace{\times\frac{1-\delta}{1+\alpha}\times\frac{2-\delta}{2+\alpha}\times\ldots\times\frac{n_1-1-\delta}{n_1-1+\alpha}}^{\text{(remplissage de la première table)}}\\
\times\ &\overbrace{\frac{(\alpha+1\cdot\delta)G_0(l_2)}{n_1+\alpha}}^{\text{(ouverture de la deuxième table)}}\ \overbrace{\times\frac{1-\delta}{n_1+1+\alpha}\times\ldots\times\frac{n_2-1-\delta}{n_1+n_2-1+\alpha}}^{\text{(remplissage de la deuxième table)}}\\
&\qquad\times\ldots\\
&\times\ \overbrace{\frac{(\alpha+(k-1)\cdot\delta)G_0(l_k)}{n_1+\ldots+n_{k-1}+\alpha}}^{\text{(ouverture de la k-ème table)}}\ \overbrace{\times\frac{1-\delta}{n_1+\ldots+n_{k-1}+1}\times\ldots\times\frac{n_k-1-\delta}{n_1+\ldots+n_{k-1}+n_k-1+\alpha}}^{\text{(remplissage de la k-ème table)}}\\
&\qquad\times\ldots\\
&\times\ \overbrace{\frac{(\alpha+(K-1)\cdot\delta)G_0(l_K)}{n_1+\ldots+n_{K-1}+\alpha}}^{\text{(ouverture de la dernière table)}}\ \overbrace{\times\frac{1-\delta}{n_1+\ldots+n_{K-1}+1}\times\ldots\times\frac{n_K-1-\delta}{n_1+\ldots+n_{K-1}+n_K-1+\alpha}}^{\text{(remplissage de la dernière table)}}
\end{aligned}
$$

et peut se réexprimer comme :

$$
(\star)=\underbrace{\prod_{t=0}^{N-1}\frac{1}{t+\alpha}}_{\text{(arrivées de clients)}}\times\underbrace{\prod_{k=0}^{K-1}(\alpha+k\cdot\delta)}_{\text{(ouvertures de tables)}}\times\underbrace{\prod_{k=1}^{K}G_0(l_k)}_{\text{(étiquetages de tables)}}\times\prod_{k=1}^{K}\left(\underbrace{\prod_{i=1}^{n_k-1}(i-\delta)}_{\text{remplissage de la k-ème table}}\right)
$$

ou, à l'aide de la fonction spéciale Γ, et compte tenu de l'égalité $\frac{\Gamma(n+x)}{\Gamma(x)}=\prod_{k=0}^{n-1}(x+k)$:

$$
(\star)=\frac{\Gamma(\alpha)}{\Gamma(\alpha+N)}\times\frac{\Gamma\left(\frac{\alpha}{\delta}+K\right)}{\Gamma\left(\frac{\alpha}{\delta}\right)}\delta^K\times\prod_{k=1}^{K}G_0(l_k)\times\prod_{k=1}^{K}\frac{\Gamma(n_k-\delta)}{\Gamma(1-\delta)}
$$

D'où l'expression de la densité *a posteriori* du couple (α,δ) :

$$
d\mathbb{P}(\alpha,\delta\mid\mathbf{e})=\underbrace{\frac{\cdot\prod_{k=1}^{K}G_0(l_k)}{Z(\mathbf{e})}}_{\text{terme constant}}\cdot f_\alpha(\alpha)\cdot f_\delta(\delta)\cdot\underbrace{\frac{\Gamma(\alpha)}{\Gamma(\alpha+N)}\cdot\frac{\Gamma\left(\frac{\alpha}{\delta}+K\right)}{\Gamma\left(\frac{\alpha}{\delta}\right)}\delta^K\cdot\prod_{k=1}^{K}\frac{\Gamma(n_k-\delta)}{\Gamma(1-\delta)}}_{=g(\alpha,\delta)}\cdot d\alpha d\delta
$$

Nous souhaitons échantillonner cet a posteriori sans avoir à calculer le terme constant, car il n'existe pas de moyen simple de calculer $Z(\mathbf{e})$. Dans ce qui suit,

nous ne considérerons donc que la *densité non-normalisée* $f_\alpha(\alpha)f_\delta(\delta)g(\alpha,\delta)$ et nous décrirons une méthode de Monte-Carlo appropriée (*slice sampling*) pour produire des échantillons approchés de cette densité.

4.1.3 Cas des processus hiérarchiques

4.1.3.1 Sans partage d'hyperparamètres

Considèrons à présent un processus de Pitman-Yor hiérarchique donnant lieu à une famille de distributions aléatoires $(G_c)_{c\in\mathcal{C}}$. Dans un premier temps, considérons le cas où chaque G_c est engendré par un PYP possédant son propre jeu d'hyperparamètres, c'est-à-dire que :

$$\forall c \in \mathcal{C}, G_c \sim PYP\left(\alpha_c, \delta_c, G_{\beta(c)}\right)$$

où β est l'arbre de repli, et où la famille $(\alpha_c,\delta_c)_{c\in\mathcal{C}}$ constitue l'ensemble des hyperparamètres du processus hiérarchique.

De plus, pour chaque $c \in \mathcal{C}$, α_c et δ_c sont soumis à des *hyperpriors* notés respectivement $f_{\alpha_c}d\alpha_c$ et $f_{\delta_c}d\delta_c$.

L'ensemble des observations $\mathbf{e}$ contient des observations *en contexte*, c'est-à-dire de la forme $c \to x$. On note $\mathbf{e}_c$ le sous-ensemble de $\mathbf{e}$ correspondant aux observations dont le contexte est c. Chaque G_c de la hiérarchie est représenté par un restaurant chinois dans lequel les observations de $\mathbf{e}_c$ ont été mises en cache, en se propageant éventuellement vers les descendants de G_c le long de l'arbre de repli.

Le but est alors d'échantillonner la distribution conditionnelle :

$$d\mathbb{P}\left((\alpha_c)_{c\in\mathcal{C}},(\delta_c)_{c\in\mathcal{C}} \mid \mathbf{e}\right) = \frac{1}{Z(\mathbf{e})}\prod_{\substack{c\in\mathcal{C}\\ \mathbf{e}_c\neq\emptyset}} f_{\alpha_c}(\alpha_c)f_{\delta_c}(\delta_c)\cdot g_c(\alpha_c,\delta_c)\cdot d\alpha_c d\delta_c$$

où $g_c(\alpha_c,\delta_c)$ est la probabilité de l'état du restaurant chinois associé à G_c, c'est-à-dire, comme à la section précédente :

$$g_c(\alpha_c,\delta_c) = \frac{\Gamma(\alpha_c)}{\Gamma(\alpha_c+N(c))}\cdot\frac{\Gamma\left(\frac{\alpha_c}{\delta_c}+K(c)\right)}{\Gamma\left(\frac{\alpha_c}{\delta_c}\right)}\delta_c^{K(c)}\cdot\prod_{k=1}^{K(c)}\frac{\Gamma(n_k(c)-\delta_c)}{\Gamma(1-\delta_c)}$$

Il suffit donc, pour chaque contexte $c \in \mathcal{C}$ correspondant à au moins une observation, d'échantillonner le couple (α_c,δ_c) selon la densité non-normalisée $f_{\alpha_c}(\alpha_c)f_{\delta_c}(\delta_c)\cdot g_c(\alpha_c,\delta_c)\cdot d\alpha_c d\delta_c$.

4.1.3.2 Avec partage d'hyperparamètres

En pratique, on n'utilisera jamais un jeu d'hyperparamètres par contexte : ceci nécessiterait un nombre trop élevé d'hyperparamètres. On fera plutôt en sorte que des contextes "similaires" partagent leurs hyperparamètres. Nous proposons de formaliser ce partage en introduisant les objets suivants :

— Un ensemble $\mathcal{H}$ sur lequel les hyperparamètres seront indexés : les hyperparamètres du processus seront représentés par une famille $(\alpha_h, \delta_h)_{h\in\mathcal{H}}$. α_h et δ_h seront soumis à des *hyperpriors* respectifs f_{α_h} et f_{delta_h}.

— Une *table de hachage* $H : \mathcal{C} \to \mathcal{H}$ représentant la manière dont le partage est effectué : G_{c_1} et G_{c_2} partageront les mêmes hyperparamètres dès lors que $H(c_1) = H(c_2)$.

Exemple 15. Si $\mathcal{H} = \mathcal{C}$ et H est l'application identité alors on se situe dans le cas du paragraphe précédent où il n'y a pas de partage d'hyperparamètres.

Exemple 16. Pour le modèle n-gramme de morphèmes présenté au chapitre précédent, on fera par exemple en sorte que des contextes de même longueur partagent leurs hyperparamètres. Pour regrouper les contextes selon leur longueur, on choisira : $\mathcal{H} = \{0, \dots, n-1\}$ et $H : \left\{\mu_1 \dots \overset{\mathcal{C}\to\mathcal{H}}{\mu_r} \mapsto r\right.$.

Inférer les hyperparamètres consistera alors à échantillonner la distribution conditionnelle :

$$d\mathbb{P}\left((\alpha_h)_{h\in\mathcal{H}}, (\delta_h)_{h\in\mathcal{H}} \mid \mathbf{e}\right) = \frac{1}{Z(\mathbf{e})}\left(\prod_{h\in\mathcal{H}} f_{\alpha_h}(\alpha_h) f_{\delta_h}(\delta_h) d\alpha_h d\delta_h\right)\left(\prod_{\substack{c\in\mathcal{C}\\ \mathbf{e}_c\neq\emptyset}} g_c(\alpha_{h(c)}, \delta_{h(c)})\right)$$

$$= \frac{1}{Z(\mathbf{e})}\prod_{h\in\mathcal{H}}\left(f_{\alpha_h}(\alpha_h) f_{\delta_h}(\delta_h) \underbrace{\prod_{\substack{c\in\mathcal{C} \mid h(c)=h\\ \mathbf{e}_c\neq\emptyset}} g_c(\alpha_h, \delta_h)}_{=g_h(\alpha_h,\delta_h)}\right) d\alpha_h d\delta_h$$

et donc, pour tout $h \in \mathcal{H}$, il s'agira d'échantillonner le couple (α_h, δ_h) selon la densité non-normalisée $f_{\alpha_h}(\alpha_h) f_{\delta_h}(\delta_h) g_h(\alpha_h, \delta_h)$.

4.2 Aspects algorithmiques

4.2.1 Calcul des *a posteriori*

On se situe dans le cas de la section précédente, celui d'un PYP hiérarchique dont les hyperparamètres sont donnés par les familles $(\alpha_h)_{h\in\mathcal{H}}$ et $(\delta_h)_{h\in\mathcal{H}}$. Nous effectuerons tous les calculs en domaine logarithmique afin d'éviter les *underflow* de nombres flottants (étant donné que toutes les probabilités intervenant dans le calcul sont extrêmement faibles).

L'*a posteriori* s'écrit :

$$d\mathbb{P}\left((\alpha_h)_{h\in\mathcal{H}}, (\delta_h)_{h\in\mathcal{H}} \mid \mathbf{e}\right) = \frac{1}{Z(\mathbf{e})} p\left((\alpha_h)_{h\in\mathcal{H}}, (\delta_h)_{h\in\mathcal{H}}\right) \prod_{h\in\mathcal{H}} d\alpha_h d\delta_h$$

et s'exprime, en domaine logarithmique et à une constante près (puisqu'on considèrera une densité non-normalisée), par la log-densité :

$$\log p\left((\alpha_h)_{h\in\mathcal{H}},(\delta_h)_{h\in\mathcal{H}}\right) = \sum_{h\in\mathcal{H}} \log f_{\alpha_h}(\alpha_h) + \log f_{\delta_h}(\delta_h) + \log g_h(\alpha_h,\delta_h)$$

Il s'agit donc de rééchantillonner (α_h, δ_h) selon la log-densité $\log f_{\alpha_h}(\alpha_h) + \log f_{\delta_h}(\delta_h) + \log g_h(\alpha_h, \delta_h)$.

Calculons le terme $\log g_h(\alpha_h, \delta_h)$ qui quantifie la différence (imposée par les observations) entre a posteriori et a priori :

$$\log g_h(\alpha_h,\delta_h) = \sum_{\substack{c\in\mathcal{C} \mid h(c)=h \\ \mathbf{e}_c\neq\emptyset}} \log g_c(\alpha_h,\delta_h)$$

$$\begin{aligned}
\log g_h(\alpha_h,\delta_h) = \sum_{\substack{c\in\mathcal{C} \mid h(c)=h \\ \mathbf{e}_c\neq\emptyset}} & \log\frac{\Gamma(\alpha_c)}{\Gamma(\alpha_c/\delta_c)} && (1)\\
& - \log\Gamma(\alpha_c + N(c)) && (2)\\
& + K(c)\log\frac{\delta_c}{\Gamma(1-\delta_c)} + \log\Gamma\left(\frac{\alpha_c}{\delta_c} + K(c)\right) && (3)\\
& + \sum_{k=1}^{K(c)} \log\Gamma\left(n_k(c) - \delta_c\right) && (4)
\end{aligned}$$

Calculer cette somme en sommant sur les $c \in \mathcal{C}$ partageant le jeu d'hyperparamètres (α_h, δ_h) sera très coûteux pour deux raisons :

— Il y aura typiquement un très grand nombre de contextes partageant ces hyperparamètres. Par exemple, dans un modèle trigramme de morphèmes où les contextes de même longueur partagent leurs hyperparamètres, la somme comprendra un terme par contexte de longueur deux, c'est-à-dire par bigramme de morphèmes observable dans les segmentations.

— Le calcul du terme (4) nécessite de sommer sur toutes les tables d'un restaurant. Or le nombre de tables peut devenir très élevé. Par exemple, pour un modèle unigramme de morphèmes, il y aura au moins autant de tables que de types de morphèmes inférés. Par exemple, sur un corpus anglais, nous avons constaté que le restaurant chinois associé à la distribution unigramme contenait 32306 tables.

4.2.1.1 "Sommation par tranches"

Notons $R(h) = \{c \in \mathcal{C} \mid h(c) = h \text{ et } \mathbf{e}_c \neq \emptyset\}$ l'ensemble des contextes associés au jeu d'hyperparamètres (α_h, δ_h).

La solution que nous proposons découle des observations suivantes :

— Pour calculer une somme $\sum_{c\in R(h)} f(N(c))$ où $f(N(c))$ est une fonction (quelconque) du nombre de clients dans le restaurant associé à c, on peut regrouper ensemble les restaurants possédant le même nombre de clients :

$$\sum_{c\in R(h)} f(N(c)) = \sum_{n\in\mathbb{N}} f(n)\cdot \underbrace{\#\left\{c\in R(h)\mid N(c)=n\right\}}_{=\mathbf{cr}(h,n)}$$

$\mathbf{cr}(h,n)$ désigne le nombre de restaurants associés à (α_h,δ_h) possédant exactement n clients [1].

— Pour calculer une somme $\sum_{c\in R(h)} f(K(c))$ où $f(K(c))$ est une fonction (quelconque) du nombre de tables dans le restaurant associé à c, on peut regrouper ensemble les restaurants possédant le même nombre de tables :

$$\sum_{c\in R(h)} f(K(c)) = \sum_{k\in\mathbb{N}} f(k)\cdot \underbrace{\#\left\{c\in R(h)\mid K(c)=k\right\}}_{=\mathbf{tr}(h,k)}$$

$\mathbf{tr}(h,k)$ désigne le nombre de restaurants associés à (α_h,δ_h) possédant exactement k tables [2].

— Pour calculer une somme $\sum_{c\in R(h)}\sum_{k=1}^{K(c)} f(n_k(c))$ où $f(n_k(c))$ est une fonction (quelconque) du nombre de clients à la k-ème table du restaurant associé à c, on peut, sur l'ensemble des tables de tous les restaurants de $R(h)$, regrouper ensemble les tables possédant le même nombre de clients :

$$\sum_{c\in R(h)}\sum_{k=1}^{K(c)} f(n_k(c)) = \sum_{n\in\mathbb{N}} f(n)\cdot \underbrace{\#\left\{(k,c)\mid c\in R(h) \text{ et } 1\leq k\leq K(c) \text{ et } n_k(c)=n\right\}}_{=\mathbf{ct}(h,n)}$$

$\mathbf{ct}(h,n)$ désigne le nombre de tables possédant exactement n clients [3], sur l'ensemble des tables des restaurants associés à (α_h,δ_h).

Compte tenu de ces remarques et en notant $\mathbf{r}(h)=\#R(h)$ le nombre de restaurants associés à (α_h,δ_h), le terme $\log g_h(\alpha_h,\delta_h)$ se réexprime alors comme :

$$\begin{aligned}
\log g_h(\alpha_h,\delta_h) =& \mathbf{r}(h)\cdot\log\frac{\Gamma(\alpha_c)}{\Gamma(\alpha_c/\delta_c)} && (1)\\
&-\sum_{n\in\mathbb{N}}\mathbf{cr}(h,n)\cdot\log\Gamma(\alpha_c+n) && (2)\\
&+\sum_{k\in\mathbb{N}}\mathbf{tr}(h,k)\cdot\left(k\cdot\log\frac{\delta_c}{\Gamma(1-\delta_c)}+\log\Gamma\left(\frac{\alpha_c}{\delta_c}+k\right)\right) && (3)\\
&+\sum_{n\in\mathbb{N}}\mathbf{ct}(h,n)\cdot\log\Gamma\left(n-\delta_c\right) && (4)
\end{aligned}$$

1. On utilise l'abbréviation **cr** pour "clients-restaurants"
2. On utilise l'abbréviation **tr** pour "tables-restaurants"
3. On utilise l'abbréviation **ct** pour "clients-tables"

La raison pour laquelle cette nouvelle manière de sommer accélère le calcul est la suivante : en général, les compteurs $\mathbf{cr}(h, n)$, $\mathbf{tr}(h, k)$ et $\mathbf{ct}(h, n)$ seront non nuls que pour un petit nombre de valeurs, et bien que les sommations se font sur $\mathbb{N}$, en réalité elles ne se feront que sur un sous-ensemble de petite taille de $\mathbb{N}$. Ceci est une conséquence du comportement d"'enrichissement des riches" des CRP, à savoir que :

— Peu de tables recevront un grand nombre de clients. Comme elles reçoivent un grand nombre de clients, leur nombre de clients peuvent (potentiellement) prendre beaucoup de valeurs différentes. Mais comme il ne s'agit que de peu de tables, de fait le nombre de valeurs différentes que leurs nombres de clients prendront sera faible.

— Beaucoup de tables recevront un faible nombre de clients. Le nombre de clients de ces tables prendra donc peu de valeurs différentes (car toutes ces valeurs seront faibles). Par exemple, une majorité de tables aura moins de trois clients. Avec notre "sommation par tranches", cela ne fera que trois termes à sommer (alors qu'en employant la "sommation naïve", cela ferait un terme par table).

— Ces deux considérations expliquent pourquoi le nombre de $n \in \mathbb{N}$ tels que $\mathbf{ct}(h, n) \neq 0$, c'est-à-dire le nombre de termes dans la somme (4), sera relativement faible. Mais des considérations analogues peuvent être faites pour **cr** et **tr**.

Expérimentalement, nous avons vérifié que cette "sommation par tranches" est environ 500 fois plus rapide que la "sommation naïve".

4.2.1.2 Approximation de la fonction Gamma

Nous utilisons l'approximation de Nemes [46] :

$$\log \Gamma(x) \simeq \frac{1}{2}\left(\log 2\pi - \log x\right) + x\left(\log\left(x + \frac{1}{12x - \frac{1}{10x}}\right) - 1\right)$$

car l'approximation donnée par formule de Stirling peut être défaillante pour des petites valeurs de x $(0 < x < 1)$.

4.2.1.3 Maintien des compteurs

Pour calculer efficacement les compteurs **cr**, **tr**, **ct**, il suffit par exemple de les initialiser à zéro, puis de les mettre à jour à chaque fois qu'une observation est mise en cache.

Plus précisément, lorsqu'une observation $c \to x$ est mise en cache dans un restaurant (à $N(c)$ clients, $K(c)$ tables et $\forall k, n_k(c)$ clients à la k-ème table), les mises à jour à effectuer sont :

— Mise à jour de **cr** :
  — décrémenter $\mathbf{cr}(h(c), N(c))$
  — incrémenter $\mathbf{cr}(h(c), N(c) + 1)$

— Mise à jour de **tr** : (seulement si la mise en cache de $c \to x$ entraîne l'ouverture d'une table)
  — décrémenter $\mathbf{tr}(h(c), K(c))$
  — incrémenter $\mathbf{tr}(h(c), K(c)+1)$
— Mise à jour de **ct** : (en notant k l'indice de la table à laquelle le client va s'asseoir)
  — décrémenter $\mathbf{ct}(h(c), n_k(c))$
  — incrémenter $\mathbf{ct}(h(c), n_k(c)+1)$

4.2.2 Echantillonnage par tranches

L'*échantillonnage par tranches* (*slice sampling*) est une méthode de Monte Carlo permettant de produire des échantillons à partir d'une densité non-normalisée, c'est-à-dire : étant donnée une fonction $g : \mathbb{R} \to \mathbb{R}_+$, produire des échantillons de la distribution $d\mathbb{P}(x) = \dfrac{g(x)\,dx}{\int_{\mathbb{R}} g(t)\,dt}$.

Cette méthode découle de la constatation suivante : produire des échantillons x distribués selon $d\mathbb{P}(x)$ revient à échantillonner un couple (x, y) *uniformément* sur l'ensemble $S_g = \{(x, y) \in \mathbb{R} \times \mathbb{R}_+ \mid 0 \leq y \leq g(x)\}$ (c'est-à-dire sur la région du plan délimitée par l'axe des abscisses et le graphe de g), puis à "oublier" la coordonnée y.

4.2.2.1 Méthode de rejet

La *méthode de rejet* est une première application de cette idée : si g est nulle hors d'un intervalle borné $[a, b]$ et bornée par une constante M sur $[a, b]$, il est possible de produire des échantillons uniformes sur S_g en produisant des échantillons uniformes sur le rectangle $[a, b] \times [0, M]$ et en rejetant ceux tombant hors de S_g.

Cette méthode peut se généraliser à un intervalle I non borné si l'on dispose d'une distribution de densité h pour laquelle on sait produire des échantillons, et vérifiant $\forall x \in I, g(x) \leq M \cdot h(x)$ pour une certaine constante M. En effet, dans cette situation, la région S_g est incluse dans la région $S_{M \cdot h}$. On sait de plus échantillonner uniformément sur la région $S_{M \cdot h}$: il suffit pour cela de produire un échantillon $x \sim h(x)dx$ puis d'échantillonner y uniformément sur $[0, M \cdot h(x)]$. Pour échantillonner uniformément sur S_g, il suffit alors d'échantillonner uniformément sur $S_{M \cdot h}$ et de rejeter les échantillons tombant hors de S_g. La méthode de rejet sera d'autant plus efficace que le nombre de rejets sera faible, c'est-à-dire lorsque la région $S_{M \cdot h}$ sera une bonne approximation de S_g.

4.2.2.2 Méthode par tranches

L'échantillonnage par tranches est une méthode ne nécessitant pas de recours à une densité auxiliaire h telle que $\exists M, S_g \subset S_{M \cdot h}$. Cette méthode nous sera

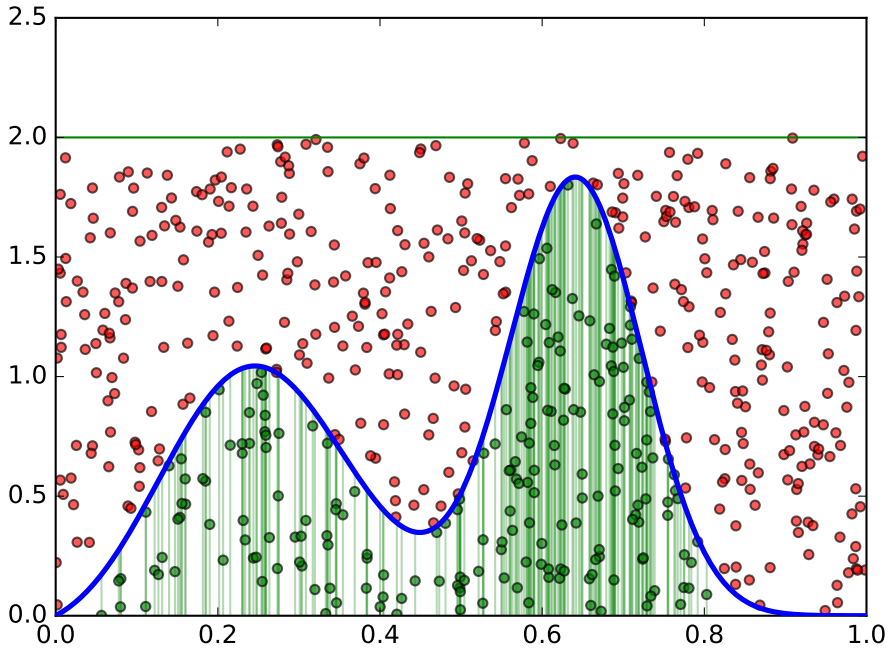


FIGURE 4.1 – Méthode de rejet. Les points verts sont les points acceptés, les points rouges sont les points rejetés.

utile ici car il semble difficile de trouver une telle densité auxiliaire pour les densités *a posteriori* sur les hyperparamètres α et δ.

L'idée est qu'il est possible de produire des échantillons uniformes sur S_g par *échantillonnage de Gibbs*, en échantillonnant alternativement une coordonnée conditionnellement à l'autre. Ainsi, en partant d'un point initial $x_0 \in \mathbb{R}$, on peut effectuer la suite d'échantillonnages suivante :

$$\begin{aligned} y_0 &\sim \mathbb{P}(y \mid x_0) \\ x_1 &\sim \mathbb{P}(x \mid y_0) \\ y_1 &\sim \mathbb{P}(y \mid x_1) \\ x_2 &\sim \mathbb{P}(x \mid y_1) \\ &\dots \\ x_n &\sim \mathbb{P}(x \mid y_{n-1}) \\ y_n &\sim \mathbb{P}(y \mid x_n) \\ &\dots \end{aligned}$$

qui (selon la théorie de l'échantillonnage de Gibbs) convergera vers un échantillon uniforme sur S_g.

— $\mathbb{P}(y \mid x_\star)$ est une distribution uniforme sur l'ensemble $\{y \leq 0 \mid (x_\star, y) \in S_g\} = [0, g(x_\star)]$: pour l'échantillonner, il suffit de produire un échantillon uniforme sur le segment $[0, g(x_\star)]$.

— $\mathbb{P}(x \mid y_\star)$ est une distribution uniforme sur l'ensemble $S_g(y_\star) = \{x \in \mathbb{R} \mid (x, y_\star) \in S_g\}$ (autrement dit, la trace de S_g sur la droite d'ordonnée $y_\star$). Produire un échantillon uniforme sur $S_g(y_\star)$ est plus subtil car cet ensemble n'est pas forcément un intervalle et peut être "fait de plusieurs morceaux". En revanche il est facile de vérifier si un point appartient à cet ensemble. On va donc appliquer une méthode de rejet consistant à :

1. Trouver un intervalle J contenant $S_g(y_\star)$.
2. Echantillonner uniformément sur J.
3. Rejeter les échantillons tombant hors de $S_g(y_\star)$.

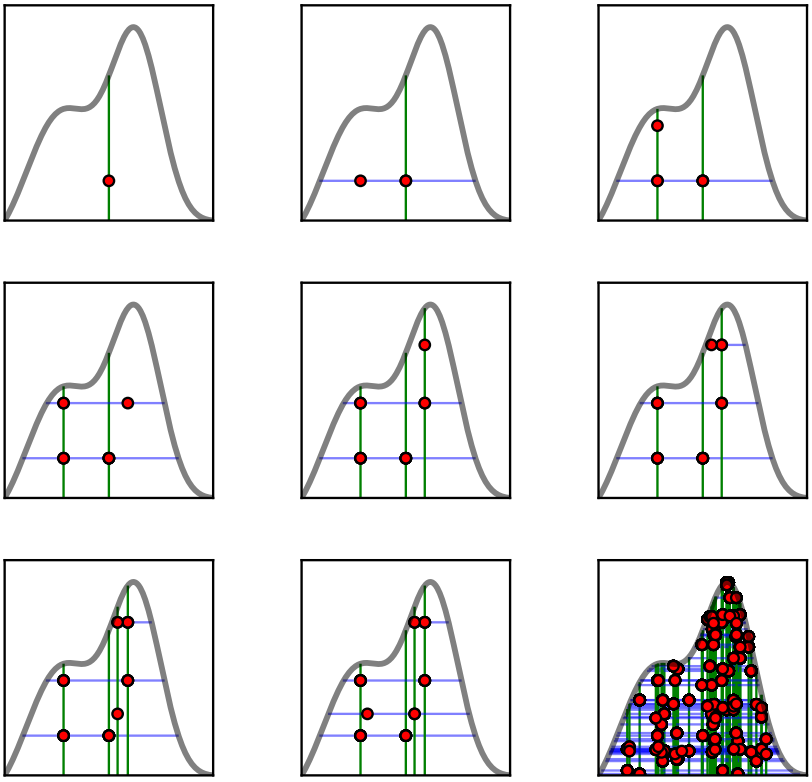


FIGURE 4.2 – Echantillonnage par tranches : premières itérations et 100ème itération.

Il reste à détailler l'étape 1 de ce qui précède, à savoir comment obtenir l'intervalle J. Une manière de procéder dans le cas où la densité g est *unimodale*

(possède un unique maximum local qui est son maximum global) est la stratégie suivante (nécessitant de spécifier un *paramètre de largeur* w) :

1. Partir d'un $x_\star \in S_g(y_\star)$ quelconque (par exemple le dernier échantillon de x dans la chaîne de Gibbs) et d'un intervalle $[a, b]$ de largeur w contenant $x_\star$.
2. Si $a \in S_g(y_\star)$, élargir l'intervalle à gauche : $a \leftarrow a - w$.
3. Si $b \in S_g(y_\star)$, élargir l'intervalle à droite : $b \leftarrow b + w$.
4. Répéter les deux étapes précédentes jusqu'à ce que les deux extrémités de l'intervalle soient hors de $S_g(y_\star)$.

Lorsque la densité est unimodale, la tranche $S_g(y_\star)$ est un intervalle, ce qui garantit que la stratégie ci-dessus retournera bien un intervalle $J = [a, b]$ contenant $S_g(y_\star)$.

Remarque. La méthode d'échantillonnage par tranches telle que présentée ci-dessus est une méthode pour produire des échantillons d'une variable aléatoire réelle. Or, ce qui nous intéresse ici est de produire des échantillons d'un *couple* de variables : le couple (α, δ). Pour cela, on aura recours à un échantillonnage de Gibbs, c'est-à-dire qu'on alternera entre échantillonner $\mathbb{P}(\alpha|\delta, \mathbf{e})$ et $\mathbb{P}(\delta|\alpha, \mathbf{e})$. Ce sont toutes deux des distributions à une variable réelle, et chacune peut donc être échantillonnée à l'aide de la méthode par tranches.

4.3 Expériences

Les résultats présentés ci-dessous ont été obtenus sur les mêmes données qu'au chapitre 3, en utilisant un paramètre de lissage de $\gamma = 0.5$. Le modèle orthographique est un modèle trigramme de caractères engendré par un processus de Pitman-Yor hiérarchique dont l'arbre de repli est donné par la troncature des contextes (contrairement aux expériences réalisées au chapitre 3 où le modèle orthographique était un modèle trigramme fixe, appris sur le corpus préalablement aux segmentations, ce qui explique l'écart des résultats par rapport au chapitre précédent).

4.3.1 Contraintes de longueur

Nous avons observé dans nos expériences préliminaires que l'inférence des hyperparamètres menait à des modèles dégénérés qui entraînaient soit une segmentation excessive (morphèmes de trop petite longueur, trop de morphèmes par mots), soit une segmentation insuffisante (morphèmes trop longs, pas assez de morphèmes par mot).

Afin d'y remédier, une première solution serait d'imposer un *hyperprior* plus contraignant sur les hyperparamètres. Mais l'inférence automatique des hyperparamètres perd alors son intérêt, puisqu'elle ne fait alors que déplacer le problème : plutôt que de se poser la question du choix des hyperparamètres, on se pose la question du choix des *hyperpriors*.

Nous avons envisagé la solution suivante : contraindre le modèle à ne pas sous-segmenter ou sur-segmenter les formes de mots, en lui imposant une forme d'*a priori* sur la longueur moyenne des morphèmes à inférer. Cet *a priori* constitue une forme de connaissance experte sur la langue traitée, mais une forme suffisamment faible pour que l'apprentissage puisse continuer à être qualifié de non-supervisé.

L'*a priori* est injecté au niveau du modèle orthographique : on reprend le modèle orthographique bayésien présenté à la section 3.2.4, en le modifiant de sorte à contraindre la longueur des chaînes de caractères qu'il génère.

Nous avons expérimenté deux modifications du modèle orthographique :

1. Imposer une longueur l suivant une loi géométrique d'espérance λ, où λ est un *a priori* représentant la longueur moyenne des morphèmes de la langue :

$$l \sim Geom(1/\lambda) \qquad \mathbb{P}(l) = \left(1 - \frac{1}{\lambda}\right)^{l-1} \frac{1}{\lambda}$$

2. Imposer une longueur l suivant une loi de Poisson d'espérance λ, où λ est elle-même engendrée par une loi Gamma de paramètres α_l, β_l :

$$\lambda \sim Gamma(\alpha_l, \beta_l) \qquad \mathbb{P}(\lambda) = \frac{\lambda^{\alpha-1}\beta^{\alpha}e^{-\beta\lambda}}{\Gamma(\alpha)}$$

$$l \sim Poisson(\lambda) \qquad \mathbb{P}(l \mid \lambda) = \frac{\lambda^l}{l!}e^{-\lambda}$$

   $Poisson(\lambda)$ représente alors un *a priori* sur la longueur d'un morphème, et $Gamma(\alpha_l, \beta_l)$ joue le rôle d'un *hyperprior* (*a priori* sur un *a priori*). Le fait que la famille de distributions *Gamma* soit conjuguée aux distributions de Poisson facilite le calcul de l'*a posteriori* sur l'hyperparamètre λ.

Une fois la longueur l engendrée, un mot est généré en remplissant l "emplacements" à l'aide du modèle $p-gramme$ défini par la hiérarchie $(H_c)_{c\in\mathcal{C}}$ du modèle orthographique. Comme la fin du mot est fixée d'office par la longueur l, il n'est pas nécessaire de prendre un compte de symbole spécial de fin de mot (que nous avions noté `f`) :

$$G_{\text{ortho}}(c_1 \dots c_l) = \mathbb{P}(l) \times H_{\tau_p(\emptyset)}(c_1) H_{\tau_p(c_1)}(c_2) H_{\tau_p(c_1c_2)}(c_3) \dots H_{\tau_p(c_1 \dots c_{l-1})}(c_l)$$

Dans le cas 2 ci-dessus, où l'hyperparamètre de longueur λ est soumis à un *hyperprior* bayésien, l'*a posteriori* sur λ conditionnellement à un ensemble de morphèmes inféré M est mis à jour selon la formule suivante, issue de la propriété de conjugaison de $Gamma(\alpha_l, \beta_l)$ par rapport à une distribution de Poisson :

$$\mathbb{P}(\lambda \mid M) \sim Gamma\left(\alpha_l + \#M, \beta_l + \sum_{\mu\in M} \text{longueur}(\mu)\right)$$

		Sans rééchantillonnage		Avec rééchantillonnage	
	Contrainte de longueur	Classes	Sans classes	Classes	Sans classes
	Pas de contrainte de longueur	P 39.2 R 72.8 F 50.9 S 27.1	P 55.7 R 65.7 F 60.3 S 15.3	P 20.8 R 96.3 F 34.2 S 72.8	P 38.7 R 59.5 F 46.9 S 27.2
$\alpha = 10$	Contrainte géométrique	P 41.0 R 74.6 F 52.9 S 26.3	P 22.5 R 89.7 F 36.0 S 65.1	P 43.6 R 71.6 F 54.2 S 23.6	P 35.4 R 59.6 F 44.4 S 29.6
	Contrainte de Poisson	P 42.2 R 71.5 F 53.1 S 24.8	P 23.0 R 76.3 F 35.4 S 56.4	P 52.0 R 72.1 F 60.4 S 19.7	P 66.9 R 67.6 F **67.3** S 15.2
	Pas de contrainte de longueur	P 40.1 R 80.9 F 53.6 S 29.5	P 30.4 R 80.1 F 44.1 S 41.9	P 34.9 R 66.6 F 45.8 S 28.0	P 28.7 R 61.7 F 39.2 S 38.3
$\alpha = 100$	Contrainte géométrique	P 43.9 R 73.8 F 55.1 S 24.2	P 30.1 R 78.8 F 43.5 S 40.3	P 38.7 R 80.5 F 52.3 S 32.2	P 26.3 R 75.4 F 39.0 S 50.7
	Contrainte de Poisson	P 44.7 R 73.2 F 55.5 S 24.7	P 34.2 R 69.0 F 45.7 S 32.6	P 42.7 R 76.3 F 54.8 S 26.9	P 48.5 R 64.0 F 55.2 S 21.6

FIGURE 4.3 – Résultats des expériences. P : Précision (la plus élevée est en rouge), R : Rappel (le plus élevé est en bleu), F : F-mesure (la plus élevée est en gras), S : Taux de segmentation (en pourcentages, le plus élevé est en vert). Le modèle orthographique est défini par un processus de Pitman-Yor hiérarchique.

4.3.2 Résultats

— Lorsque les hyperparamètres sont rééchantillonnés sans imposer de contrainte de longueur, les performances sont faibles, et dans la plupart des cas, on obtient des modèles dégénérés avec une précision trop faible et un taux de segmentation trop élevé. Le rappel le plus élevé est obtenu dans cette situation, mais il n'est évidemment pas le signe d'une bonne qualité de segmentations, puisqu'il est obtenu au prix d'une précision et d'une F-mesure extrêmement faibles.

— En revanche, les meilleurs résultats (à la fois avec un modèle de classes et sans classes) sont obtenus en rééchantillonnant les hyperparamètres et en imposant une contrainte de Poisson. De manière générale, la contrainte de Poisson donne de meilleurs résultats que la contrainte géométrique.

— Sans rééchantillonnage d'hyperparamètres, l'utilisation de contraintes de longueur ne semble pas particulièrement apporter de gain de performances.

Nous avons également réalisé des expériences en utilisant un modèle ortho-

		Sans rééchantillonnage		Avec rééchantillonnage	
	Contrainte de longueur	Classes	Sans classes	Classes	Sans classes
	Pas de contrainte de longueur	P 42.8 R 68.5 F 52.7 S 23.2	P 52.9 R 56.8 F 54.8 S 17.1	P 23.9 R 75.9 F 36.3 S 37.6	P 21.3 R 90.1 F 34.5 S 43.5
$\alpha = 10$	Contrainte géométrique	P 42.3 R 69.9 F 52.7 S 22.9	P 54.8 R 56.4 F 55.6 S 15.8	P 23.8 R 72.5 F 35.8 S 35.6	P 21.5 R 91.2 F 34.8 S 41.6
	Contrainte de Poisson	P 44.1 R 76.0 F 55.8 S 22.8	P 53.2 R 55.5 F 54.3 S 16.1	P 22.9 R 74.5 F 35.0 S 39.3	P 19.9 R 77.0 F 31.6 S 43.0
	Pas de contrainte de longueur	P 42.8 R 66.5 F 52.1 S 21.8	P 51.1 R 64.5 F 57.0 S 20.4	P 21.3 R 74.7 F 33.2 S 41.2	P 20.0 R 79.4 F 32.0 S 47.4
$\alpha = 100$	Contrainte géométrique	P 47.5 R 68.5 F 56.1 S 22.1	P 59.2 R 61.3 F **60.2** S 14.9	P 21.4 R 77.3 F 33.5 S 41.8	P 20.9 R 82.5 F 33.4 S 47.7
	Contrainte de Poisson	P 42.4 R 64.6 F 51.2 S 22.1	P 49.2 R 59.2 F 53.7 S 20.1	P 21.4 R 75.7 F 33.4 S 40.9	P 20.3 R 83.4 F 32.6 S 46.5

FIGURE 4.4 – Résultats des expériences. P : Précision (la plus élevée est en rouge), R : Rappel (le plus élevé est en bleu), F : F-mesure (la plus élevée est en gras), S : Taux de segmentation (en pourcentages, le plus élevé est en vert). Le modèle orthographique est un simple modèle trigramme de caractères préalablement appris sur les données.

graphique consistant en un simple modèle trigramme de caractères appris sur les données avant l'inférence des segmentations (figure 4.4). Les résultats obtenus par ce modèle sont très différents de ceux obtenus avec un modèle orthographique non-paramétrique : le fait de rééchantillonner les hyperparamètres entraîne une segmentation excessive, ce qui a pour conséquence des performances bien moindres qu'en fixant les hyperparamètres.

Nous avons également réalisé des expériences sur les données en langue turque du *MorphoChallenge 2005*. Celles-ci consistent en une liste de mots rencontrés dans un ensemble de publications collectées sur le Web et dans des articles de journaux, et comprennent 16 619 455 occurrences de mots pour 582 923 types de mots. Les données d'évaluation sont une liste de 774 mots segmentés par un segmenteur morphologique turc conçu par des linguistes de l'université Bogaziçi, à base d'automates finis. Nous avons utilisé un modèle orthographique non-paramétrique avec une contrainte de Poisson. Lorsque nous autorisions un nombre de classes potentiellement infini, nous avons constaté que le nombre de

classes inférées devenait trop élevé, entraînant un temps de calcul excessif dans l'algorithme *forward-backward* pour produire des segmentations. C'est pour cela que dans nos expériences avec des classes, nous avons borné le nombre de classes à six. Les résultats sont compilés dans la figure 4.5.

	Sans rééchantillonnage		Avec rééchantillonnage	
	Classes	Sans classes	Classes	Sans classes
$\alpha = 1$	P 46.8 R 66.9 F 55.0 S 30.7	P 46.6 R 54.7 F 50.3 S 29.3	P 46.6 R 54.7 F 50.3 S 29.3	P 47.4 R 46.3 F 46.8 S 26.1
$\alpha = 10$	P 35.5 R 75.3 F 48.3 S 51.2	P 45.3 R 46.6 F 46.0 S 26.2	P 56.1 R 65.7 F **60.5** S 22.8	P 72.2 R 35.7 F 47.8 S 11.0

FIGURE 4.5 – Résultats des expériences sur un corpus de langue turque. P : Précision (la plus élevée est en rouge), R : Rappel (le plus élevé est en bleu), F : F-mesure (la plus élevée est en gras), S : Taux de segmentation (en pourcentages, le plus élevé est en vert). Le modèle orthographique est un simple modèle trigramme non-paramétrique.

Conclusion

Comme stratégie de choix d'hyperparamètres pour les modèles présentés au chapitre précédent, plutôt que celle, classique mais coûteuse, de chercher à optimiser la performance du modèle sur des *données de développement*, il est possible de choisir ces hyperparamètres de manière bayésienne, en les soumettant à leur tour à un *a priori* probabiliste. Ainsi par exemple pour un modèle *n-gramme*, tous les contextes de longueur donnée pourront partager un paramètre de force et de décompte communs, et ces deux paramètres seront périodiquement rééchantillonnés en faisant un compromis "bayésien" entre leur *vraisemblance* et l'*a priori* auquel ils sont soumis. Ceci nécessitant une méthode approchée d'échantillonnage, celle que nous avons adopté est l'échantillonnage par tranches, qui consiste, étant donnée une densité de probabilité, à échantillonner uniformément un point entre son graphe et l'axe des abscisses. Enfin, la performance des modèles peut être améliorée par l'introduction d'*a prioris* linguistiques sous la forme d'hyperparamètres supplémentaires. Ainsi, ajouter différentes contraintes probabilistes sur la longueur des morphèmes aide à éviter d'obtenir des modèles dégénérés.

Conclusion

Au premier chapitre de cette thèse, en dressant un aperçu des différentes approches de traitement de la morphologie, nous avons constaté qu'un grand nombre de ces méthodes faisaient appel à des heuristiques relativement arbitraires pour déterminer les morphèmes. Nous avions pour but de décrire des méthodes qui, bien que restant dans un cadre non-supervisé, se dispensaient de ces heuristiques en adoptant une approche fondée sur le principe de longueur de description minimale. Ce dernier consiste à rechercher un modèle pour lequel la somme de la longueur de description du modèle et de la longueur de description des données à l'aide du modèle est minimale. Ce principe est apparu dans les méthodes que nous avons présentés sous une forme bayésienne probabiliste, où la longueur de description du modèle était traduite par un a priori de Pitman-Yor, et où la longueur de description des données à l'aide du modèle était donnée par la probabilité des données selon un scénario génératif.

Nous avons proposé un cadre mathématique formel pour décrire les processus de Pitman-Yor ainsi que leur version hiérarchique. Nous avons notamment vu qu'une hiérarchie de processus de Pitman-Yor pouvait se formaliser comme la donnée d'un ensemble de contextes, d'un arbre de repli et d'un *adapteur* (distribution sur des distributions) pour chaque contexte. Un processus de Pitman-Yor engendre des distributions aléatoires sur un ensemble infini, et il est par conséquent impossible de représenter explicitement ces distributions aléatoires sur une machine, mais nous avons décrit comment marginaliser ces distributions aléatoires à l'aide de processus de restaurant chinois.

Nous avons étudié deux modèles de séquences de morphèmes soumis à un a priori de Pitman-Yor. D'une part, un modèle n-gramme modélisant les séquences selon des dépendances markoviennes. D'autre part, un modèle semi-markovien caché (SHMM) permettant de regrouper les morphèmes en classes morphologiques. Les expériences ont montré que ces deux modèles produisaient des segmentations morphologiques de relativement bonne qualité (vu leur caractère non-supervisé). Les performances du modèle n-gramme sont généralement meilleures. Mais l'intêret de la méthode SHMM est qu'elle produit un modèle plus lisible et des informations plus riches du point de vue linguistique : les morphèmes sont regroupés en classes, et les successions de ces classes sont modélisées par un automate fini. En particulier, il était intéressant de voir que sans a priori relatif à la forme et aux séquences de classes (par exemple, sans imposer

qu'il y ait une classe de préfixes suivie par une classe de racines puis par une classe de suffixes), le modèle SHMM parvenait sur la langue anglaise à découvrir par lui-même des classes de préfixes, des classes de suffixes et des classes de racines, et à établir une distinction entre des classes fermées de morphèmes (peu peuplées mais très usitées) et des classes ouvertes (très peuplées mais par des morphèmes plus rares). Néanmoins, le cadre entièrement non-supervisé fait que les classes découvertes sont très bruitées et restent relativement insatisfaisantes comme description de la morphotactique. Une direction intéressante pour des travaux ultérieurs serait de voir comment l'ajout d'a prioris relativement faibles amélioreraient la qualité des segmentations et de la morphotactique induite. Il serait par exemple possible de :

— Utiliser un *modèle orthographique* différent pour chaque classe, et voir comment il serait possible de capturer les différences entre la forme des affixes et la forme des racines dans ces modèles orthographiques. On pourrait par exemple utiliser pour les affixes un modèle orthographique n'autorisant qu'une ou deux syllabes.

— Introduire des a prioris au niveau du modèle du séquence de classes relatifs au caractère "ouvert" ou "fermé" de chaque classe. Par exemple, imposer que tout mot contienne au moins un morphème appartenant à une classe ouverte (une manière de guider le modèle vers la découverte d'au moins une racine par mot lors de l'inférence).

— "Bootstrapper" l'inférence et la distinction des classes en fournissant dès le début une liste de préfixes ou de suffixes typiques.

Nous avons ensuite discuté du choix des hyperparamètres dans les processus de Pitman-Yor hiérarchiques, les hyperparamètres régulant le degré de parcimonie des distributions engendrés par ces processus. Nous avons proposé une stratégie de rééchantillonnage de ces hyperparamètres selon un *a posteriori* bayésien, en cumulant de l'*échantillonnage par tranches* et une méthode de décompte appropriée. Cette méthode consiste à maintenir des compteurs de clients par restaurants, de clients par tables et de tables par restaurants, et accélère considérablement le calcul de l'a posteriori. Nous avons ensuite mis expérimentalement en évidence que le rééchantillonnage des hyperparamètres conduisait à des modèles dégénérés incapables de maintenir les morphèmes inférés à une échelle intermédiaire entre le caractère et le mot. Nous avons montré que l'introduction d'un a priori faible sur la longueur des morphèmes au niveau du modèle orthographique permettait de remédier à cette dégénérescence.

De nombreuses pistes restent à explorer dans des travaux ultérieurs. On pourrait considérer d'autres structures plus riches dans nos modèles :

— Nous nous sommes limités à des modèles markoviens. Le pendant bayésien non-paramétrique des *grammaires hors contexte* sont les *adaptor grammars*. Il aurait été intéressant de voir dans quelle mesure le SHMM avec un nombre potentiellement infini de classes aurait pu se généraliser à celles-ci, c'est-à-dire concevoir un modèle de grammaire hors contexte avec un

a priori non-paramétrique, et un nombre potentiellement infini de *non-terminaux*.

— De même, nous n'avons pas établi de distinction nette entre segmentation morphologique et tokénisation dans nos modèles. Autrement dit, nos modèles pourraient aussi bien servir à segmenter des mots en morphèmes que des phrases en mots. A la rigueur, il suffirait de jouer sur le niveau de granularité de la segmentation à l'aide des contraintes de longueur introduites à la fin du chapitre 4. Il serait intéressant de voir commment une distinction plus qualitative entre le niveau lexical et le niveau morphologique pourrait être introduite dans une méthode non-supervisée. On pourrait par exemple une méthode réalisant à la fois de la tokénisation et de la segmentation morphologique, basée sur une hiérarchie à deux niveaux, où :

  — Au niveau inférieur de la hiérarchie, un modèle n-gramme de morphèmes engendre des mots comme concaténation de séquences de morphèmes (à l'image des modèles que nous avons présenté).

  — Au niveau supérieur de la hiérarchie, un modèle n-gramme de mots engendre des phrases comme concaténation de séquences de mots, en utilisant comme distribution de base (pour engendrer un mot) le modèle du niveau inférieur.

Une autre direction pour des travaux ultérieurs serait d'évaluer l'utilité des segmentations produites en amont d'autres tâches de traitement des langues (étiquetage syntaxique, traduction automatique...), notamment dans quelle mesure elles aident à résoudre le problème du sur-apprentissage et de la parcimonie des données. En particulier, il serait intéressant de voir dans quelle mesure les segmentations améliorant la performance d'autres tâches en aval coïncident avec les segmentations pertinentes du point de vue linguistique.

Enfin, de par l'essor des réseaux de neurones, les méthodes s'appuyant sur des représentations d'unités linguistiques par des vecteurs de flottants sont de plus en plus utilisés en traitement des langues. Les réseaux de neurones récurrents et convolutionnels offrent notamment une solution au problème du traitement des mots *hors-vocabulaire* (OOV), puisqu'il a été montré qu'ils sont capables de produire de bonnes représentation d'un mot à partir des caractères le composant. Pouvoir travailler à l'échelle du caractère rend donc le problème de la segmentation morphologique caduc dans une certaine mesure. Toutefois, de même qu'on dispose de méthodes décomposant un mot en concaténation de morphèmes, on pourrait concevoir des méthodes décomposant une représentation vectorielle d'un mot en somme de représentations vectorielles des morphèmes le constituant.

Bibliographie

[1] Alfred V. Aho and Jeffrey D. Ullman. *The Theory of Parsing, Translation and Compiling*, volume 1. Prentice-Hall, Englewood Cliffs, NJ, 1972.

[2] American Psychological Association. *Publications Manual*. American Psychological Association, Washington, DC, 1983.

[3] Farah Benarmara, Nabil Hatout, Philippe Muller, and Sylwia Ozdowska, editors. *Actes de TALN 2007 (Traitement automatique des langues naturelles)*, Toulouse, June 2007. ATALA, IRIT.

[4] Delphine Bernhard. Unsupervised morphological segmentation based on segment predictability and word segments alignment. In *Proceedings of 2nd Pascal Challenges Workshop*, pages 19–24, 2006.

[5] Delphine Bernhard. Apprentissage non supervisé de familles morphologiques par classification ascendante hiérarchique. In Benarmara et al. [3], pages 367–376.

[6] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3 :993–1022, 2003.

[7] Michael R. Brent, Sreerama K. Murthy, and Andrew Lundberg. Discovering morphemic suffixes a case study in MDL induction. In *In Fifth International Workshop on AI and Statistics,Ft*, pages 264–271, 1995.

[8] Peter F. Brown, John Cocke, Stephen Della Pietra, Vincent J. Della Pietra, Frederick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Roossin. A statistical approach to machine translation. *Computational Linguistics*, 16(2) :79–85, 1990.

[9] Franck Burlot and François Yvon. Unsupervised learning of morphology in the ussr. In *Proceedings of the 13th International Conference on Statistical Analysis of Textual Data (JADT)*, 2016.

[10] Ashok K. Chandra, Dexter C. Kozen, and Larry J. Stockmeyer. Alternation. *Journal of the Association for Computing Machinery*, 28(1) :114–133, 1981.

[11] Vincent Claveau. Unsupervised and semi-supervised morphological analysis for information retrieval in the biomedical domain. In *Proceedings of COLING 2012*, pages 629–646, Mumbai, India, December 2012. The COLING 2012 Organizing Committee.

[12] Mathias Creutz and Krista Lagus. Unsupervised discovery of morphemes. In *Proceedings of the ACL-02 workshop on Morphological and phonological learning-Volume 6*, pages 21–30. Association for Computational Linguistics, 2002.

[13] Mathias Creutz and Krista Lagus. Induction of a simple morphology for highly-inflecting languages. In *Proceedings of the 7th Meeting of the ACL Special Interest Group in Computational Phonology : Current Themes in Computational Phonology and Morphology*, pages 43–51. Association for Computational Linguistics, 2004.

[14] Mathias Creutz and Krista Lagus. Inducing the morphological lexicon of a natural language from unannotated text. In *In Proceedings of the International and Interdisciplinary Conference on Adaptive Knowledge Representation and Reasoning*, 2005.

[15] Mathias Creutz and Krista Lagus. Unsupervised models for morpheme segmentation and morphology learning. *ACM Trans. Speech Lang. Process.*, 4(1) :3 :1–3 :34, February 2007.

[16] Markus Dreyer and Jason Eisner. Discovering morphological paradigms from plain text using a dirichlet process mixture model. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 616–627. Association for Computational Linguistics, 2011.

[17] Association for Computing Machinery. In *Computing Reviews*, volume 24, pages 503–512. 1983.

[18] Stella Frank. Learning the hyperparameters to learn morphology. In *Proc. of 5th Workshop on Cognitive Aspects of Computational Language Learning (CogACLL)@ EACL*, pages 14–18, 2014.

[19] John Goldsmith. Unsupervised learning of the morphology of a natural language. *Computational Linguistics*, 27(2) :153–198, 2001.

[20] S. Goldwater, T. L. Griffiths, and M. Johnson. Interpolating between types and tokens by estimating power-law generators. In *Advances in Neural Information Processing Systems 18*, 2006.

[21] Sharon Goldwater, Thomas L. Griffiths, and Mark Johnson. A Bayesian framework for word segmentation : Exploring the effects of context. *Cognition*, 112(1) :21–54, 2009.

[22] Sharon Goldwater, Thomas L Griffiths, and Mark Johnson. Producing power-law distributions and damping word frequencies with two-stage language models. *Journal of Machine Learning Research*, 12(Jul) :2335–2382, 2011.

[23] Sharon Goldwater and David McClosky. Improving statistical MT through morphological analysis. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 676–683, Vancouver, British Columbia, Canada, October 2005. Association for Computational Linguistics.

[24] Sharon J Goldwater and Mark Johnson. *Nonparametric Bayesian Models of Lexican Acquisition*. Brown University, 2007.

[25] Stig-Arne Grönroos, Sami Virpioja, Peter Smit, Mikko Kurimo, et al. Morfessor flatcat : An hmm-based method for unsupervised and semi-supervised learning of morphology. In *COLING*, pages 1177–1185, 2014.

[26] Dan Gusfield. *Algorithms on Strings, Trees and Sequences*. Cambridge University Press, Cambridge, UK, 1997.

[27] Margaret A Hafer and Stephen F Weiss. Word segmentation by letter successor varieties. *Information storage and retrieval*, 10(11) :371–385, 1974.

[28] Zellig S Harris. Methods in structural linguistics. 1951.

[29] Zellig S. Harris. From phoneme to morpheme. *Language*, 31(2) :190–222, 1955.

[30] Zellig S Harris. From phoneme to morpheme. In *Papers in Structural and Transformational Linguistics*, pages 32–67. Springer, 1970.

[31] Thomas Hofmann. Unsupervised learning by probabilistic latent semantic analysis. *Mach. Learn.*, 42(1-2) :177–196, 2001.

[32] Institut iazykoznaniia (Akademiia SSSR) and Nikolai Dmitrievich Andreev. *Statistiko-kombinatornoe modelirovanie iazykov*. 1965.

[33] Mark Johnson. Unsupervised word segmentation for sesotho using adaptor grammars. In *Proceedings of the Tenth Meeting of ACL Special Interest Group on Computational Morphology and Phonology*, pages 20–27. Association for Computational Linguistics, 2008.

[34] Mark Johnson and Sharon Goldwater. Improving nonparameteric bayesian inference : experiments on unsupervised word segmentation with adaptor grammars. In *Proceedings of Human Language Technologies : The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 317–325. Association for Computational Linguistics, 2009.

[35] Mark Johnson, Thomas L Griffiths, and Sharon Goldwater. Adaptor grammars : A framework for specifying compositional nonparametric bayesian models. In *Advances in neural information processing systems*, pages 641–648, 2007.

[36] Samarth Keshava and Emily Pitler. A simpler, intuitive approach to morpheme induction. In *Proceedings of 2nd Pascal Challenges Workshop*, pages 31–35, 2006.

[37] Oskar Kohonen, Sami Virpioja, and Mikaela Klami. Allomorfessor : Towards unsupervised morpheme analysis. In *CLEF*, volume 8, pages 975–982. Springer, 2008.

[38] Andrei N Kolmogorov. Three approaches to the quantitative definition ofinformation'. *Problems of information transmission*, 1(1) :1–7, 1965.

[39] Young-Suk Lee. Morphological analysis for statistical machine translation. In Daniel Marcu Susan Dumais and Salim Roukos, editors, *HLT-NAACL*

2004 : Short Papers, pages 57–60, Boston, Massachusetts, USA, May 2 - May 7 2004. Association for Computational Linguistics.

[40] S.E. Levinson. Continuously variable duration hidden Markov models for automatic speech recognition. *Computer Speech and Language*, 1(1) :29 – 45, 1986.

[41] David J. C. MacKay. *Information Theory, Inference & Learning Algorithms*. Cambridge University Press, New York, NY, USA, 2002.

[42] Daichi Mochihashi, Takeshi Yamada, and Naonori Ueda. Bayesian unsupervised word segmentation with nested pitman-yor language modeling. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP : Volume 1-Volume 1*, pages 100–108. Association for Computational Linguistics, 2009.

[43] Christian Monson, Jaime Carbonell, Alon Lavie, and Lori Levin. Paramor : Finding paradigms across morphology. In *Workshop of the Cross-Language Evaluation Forum for European Languages*, pages 900–907. Springer, 2007.

[44] Karthik Narasimhan, Regina Barzilay, and Tommi Jaakkola. An unsupervised method for uncovering morphological chains. *Transactions of the Association for Computational Linguistics*, 3 :157–167, 2015.

[45] Radford M Neal. Slice sampling. *Annals of statistics*, pages 705–741, 2003.

[46] Gergő Nemes. New asymptotic expansion for the gamma function. *Archiv der Mathematik*, 95(2) :161–169, 2010.

[47] Jim Pitman and Marc Yor. The two-parameter Poisson-Dirichlet distribution derived from a stable subordinator. *Ann. Probab.*, 25(2) :855–900, 1997.

[48] Hoifung Poon, Colin Cherry, and Kristina Toutanova. Unsupervised morphological segmentation with log-linear models. In *Proceedings of Human Language Technologies : The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 209–217, June 2009.

[49] Lawrence R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. In *Proceedings of the IEEE*, pages 257–286, 1989.

[50] Jorma Rissanen. Modeling by shortest data description. *Automatica*, 14(5) :465–471, 1978.

[51] Jorma Rissanen. *Stochastic complexity in statistical inquiry*. World Scientific, 1989.

[52] Teemu Ruokolainen, Oskar Kohonen, Sami Virpioja, and Mikko Kurimo. Supervised morphological segmentation in a low-resource learning setting using conditional random fields. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 29–37, 2013.

[53] Patrick Schone and Daniel Jurafsky. Knowledge-free induction of morphology using latent semantic analysis. In *Proceedings of the 2nd workshop on Learning language in logic and the 4th conference on Computational natural language learning-Volume 7*, pages 67–72. Association for Computational Linguistics, 2000.

[54] Kairit Sirts and Sharon Goldwater. Minimally-supervised morphological segmentation using adaptor grammars. *Transactions of the Association for Computational Linguistics*, 1 :255–266, 2013.

[55] Benjamin Snyder and Regina Barzilay. Unsupervised multilingual learning for morphological segmentation. In *Proceedings of ACL-08 : HLT*, pages 737–745, Columbus, Ohio, June 2008. Association for Computational Linguistics.

[56] Y. W. Teh. A hierarchical Bayesian language model based on Pitman-Yor processes. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 985–992, 2006.

[57] Yee Whye Teh and Michael I Jordan. Hierarchical bayesian nonparametric models with applications. *Bayesian nonparametrics*, 1, 2010.

[58] Kei Uchiumi, Hiroshi Tsukahara, and Daichi Mochihashi. Inducing word and part-of-speech with Pitman-Yor hidden semi-Markov models. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1 : Long Papers)*, pages 1774–1782, Beijing, China, July 2015. Association for Computational Linguistics.

[59] Dimitra Vergyri, Katrin Kirchhoff, Kevin Duh, and Andreas Stolcke. Morphology-based language modeling for Arabic speech recognition. In *In Proc. of ICSLP*, pages 2245–2248, 2004.

[60] Bing Xiang, Kham Nguyen, Long Nguyen, R. Schwartz, and J. Makhoul. Morphological decomposition for Arabic broadcast news transcription. In *In Proc. of ICASSP*, volume 1, pages I–I, May 2006.

[61] David Yarowsky and Richard Wicentowski. Minimally supervised morphological analysis by multimodal alignment. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 207–216. Association for Computational Linguistics, 2000.

université PARIS-SACLAY
ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre : Apprentissage non-supervisé de la morphologie des langues à l'aide de modèle bayésiens non-paramétriques

Mots clés : Apprentissage machine, Traitement des langues, Statistiques bayésiennes, Morphologie

Résumé : Un problème central contribuant à la grande difficulté du traitement du langage naturel par des méthodes statistiques est celui de la parcimonie des données, à savoir le fait que dans un corpus d'apprentissage donné, la plupart des évènements linguistiques n'ont qu'un nombre d'occurrences assez faible, et que par ailleurs un nombre infini d'évènements permis par une langue n'apparaitront nulle part dans le corpus.
Les modèles neuronaux ont déjà contribué à partiellement résoudre le problème de la parcimonie en inférant des représentations continues de mots. Ces représentations continues permettent de structurer le lexique en induisant une notion de similarité sémantique ou syntaxique entre les mots.
Toutefois, les modèles neuronaux actuellement les plus répandus n'offrent qu'une solution partielle au problème de la parcimonie, notamment par le fait que ceux-ci nécessitent une représentation distribuée pour chaque mot du vocabulaire, mais sont incapables d'attribuer une représentation à des mots hors vocabulaire.
Ce problème est particulièrement marqué dans des langues morphologiquement riches, ou des processus de formation de mots complexes mènent à une prolifération des formes de mots possibles, et à un faible coincidence entre le lexique observé lors de l'entrainement d'un modèle, et le lexique observé lors de son déploiement. Aujourd'hui, l'anglais n'est plus la langue majoritairement utilisée sur le Web, et concevoir des systèmes de traduction automatique pouvant appréhender des langues dont la morphologie est très éloignée des langues ouest-européennes est un enjeu important.
L'objectif de cette thèse est de développer de nouveaux modèles capables d'inférer de manière non-supervisée les processus de formation de mots sous-jacents au lexique observé, afin de pouvoir de pouvoir produire des analyses morphologiques de nouvelles formes de mots non observées lors de l'entraînement.

Title : Unsupervised learning of natural langage morphology using non-parametric bayesian models

Keywords : Machine learning, Natural langage processing, Bayesian statistics, Morphology

Abstract : A crucial issue in statistical natural language processing is the issue of sparsity, namely the fact that in a given learning corpus, most linguistic events have low occurrence frequencies, and that an infinite number of structures allowed by a language will not be observed in the corpus.
Neural models have already contributed to solving this issue by inferring continuous word representations. These continuous representations allow to structure the lexicon by inducing semantic or syntactic similarity between words.
However, current neural models only partially solve the sparsity issue, due to the fact that they require a vectorial representation for every word in the lexicon, but are unable to infer sensible representations for unseen words.
This issue is especially present in morphologically rich languages, where word formation processes yield a proliferation of possible word forms, and little overlap between the lexicon observed during model training, and the lexicon encountered during its use. Today, several languages are used on the Web besides English, and engineering translation systems that can handle morphologies that are very different from western European languages has become a major stake.
The goal of this thesis is to develop new statistical models that are able to infer in an unsupervised fashion the word formation processes underlying an observed lexicon, in order to produce morphological analyses of new unseen word forms.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France