Université de Montréal

Désambiguisation de Sens par modèles de contextes et son application à la Recherche d'Information

par
Bernard Brosseau-Villeneuve

Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

Mémoire présenté à la Faculté des arts et des sciences
en vue de l'obtention du grade de Maître ès sciences (M.Sc.)
en informatique

Décembre, 2010

Université de Montréal
Faculté des arts et des sciences

Ce mémoire intitulé:

Désambiguisation de Sens par modèles de contextes et son application à la Recherche d'Information

présenté par:

Bernard Brosseau-Villeneuve

a été évalué par un jury composé des personnes suivantes:

Michel Boyer,	président-rapporteur
Jian-Yun Nie,	directeur de recherche
Guy Lapalme,	membre du jury

Mémoire accepté le: . . . . . .

RÉSUMÉ

Il est connu que les problèmes d'ambiguïté de la langue ont un effet néfaste sur les résultats des systèmes de Recherche d'Information (RI). Toutefois, les efforts de recherche visant à intégrer des techniques de Désambiguisation de Sens (DS) à la RI n'ont pas porté fruit. La plupart des études sur le sujet obtiennent effectivement des résultats négatifs ou peu convaincants. De plus, des investigations basées sur l'ajout d'ambiguïté artificielle concluent qu'il faudrait une très haute précision de désambiguation pour arriver à un effet positif. Ce mémoire vise à développer de nouvelles approches plus performantes et efficaces, se concentrant sur l'utilisation de statistiques de cooccurrence afin de construire des modèles de contexte. Ces modèles pourront ensuite servir à effectuer une discrimination de sens entre une requête et les documents d'une collection.

Dans ce mémoire à deux parties, nous ferons tout d'abord une investigation de la force de la relation entre un mot et les mots présents dans son contexte, proposant une méthode d'apprentissage du poids d'un mot de contexte en fonction de sa distance du mot modélisé dans le document. Cette méthode repose sur l'idée que des modèles de contextes faits à partir d'échantillons aléatoires de mots en contexte devraient être similaires. Des expériences en anglais et en japonais montrent que la force de relation en fonction de la distance suit généralement une loi de puissance négative. Les poids résultant des expériences sont ensuite utilisés dans la construction de systèmes de DS Bayes Naïfs. Des évaluations de ces systèmes sur les données de l'atelier Semeval en anglais pour la tâche *Semeval-2007 English Lexical Sample*, puis en japonais pour la tâche *Semeval-2010 Japanese WSD*, montrent que les systèmes ont des résultats comparables à l'état de l'art, bien qu'ils soient bien plus légers, et ne dépendent pas d'outils ou de ressources linguistiques.

La deuxième partie de ce mémoire vise à adapter les méthodes développées à des applications de Recherche d'Information. Ces applications ont la difficulté additionnelle de ne pas pouvoir dépendre de données créées manuellement. Nous proposons donc des modèles de contextes à variables latentes basés sur l'*Allocation Dirichlet Latente* (LDA). Ceux-ci seront combinés à la méthodes de vraisemblance de requête par modèles de

langue. En évaluant le système résultant sur trois collections de la conférence TREC (*Text REtrieval Conference*), nous observons une amélioration proportionnelle moyenne de 12% du MAP et 23% du $GMAP$. Les gains se font surtout sur les requêtes difficiles, augmentant la stabilité des résultats. Ces expériences seraient la première application positive de techniques de DS sur des tâches de RI standard.

Mot clés: informatique, traitement des langues naturelles, recherche d'information, modèles de contexte, désambiguisation de sens, contexte de mots, TAL, RI, IR, DS.

ABSTRACT

It is known that the ambiguity present in natural language has a negative effect on Information Retrieval (IR) systems effectiveness. However, up to now, the efforts made to integrate Word Sense Disambiguation (WSD) techniques in IR systems have not been successful. Past studies end up with either poor or unconvincing results. Furthermore, investigations based on the addition of artificial ambiguity shows that a very high disambiguation accuracy would be needed in order to observe gains. This thesis has for objective to develop efficient and effective approaches for WSD, using co-occurrence statistics in order to build context models. Such models could then be used in order to do a word sense discrimination between a query and documents of a collection.

In this two-part thesis, we will start by investigating the principle of strength of relation between a word and the words present in its context, proposing an approach to learn a function mapping word distance to count weights. This method is based on the idea that context models made from random samples of word in context should be similar. Experiments in English and Japanese shows that the strength of relation roughly follows a negative power law. The weights resulting from the experiments are then used in the construction of Naïve Bayes WSD systems. Evaluations of these systems in English with the Semeval-2007 English Lexical Sample (ELS), and then in Japanese with the Semeval-2010 Japanese WSD (JWSD) tasks shows that the systems have state-of-the-art accuracy even though they are much lighter and don't rely on linguistic tools or resources.

The second part of this thesis aims to adapt the new methods to IR applications. Such applications put heavy constraints on performance and available resources. We thus propose the use of corpus-based latent context models based on *Latent Dirichlet Allocation* (LDA). The models are combined with the query likelihood Language Model (LM) approach for IR. Evaluating the systems on three collections from the *Text REtrieval Conference* (TREC), we observe average proportional improvement in the range of 12% in MAP and 23% in $GMAP$. We then observe that the gains are mostly made on hard queries, augmenting the robustness of the results. To our knowledge, these ex-

periments are the first positive application of WSD techniques on standard IR tasks.

Keywords Computer Science, Natural Language Processing, Information Retrieval, Context Models, Word Sense Disambiguation, Word Context, CS, NLP, IR, WSD.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES

BI	Besoin d'Information
DS	Désambiguisation de Sens
ELS	*English Lexical Sample*
$GMAP$	MAPgéométrique
HAL	Analogie d'Hyperespace au Langage
LDA	Allocation Dirichlet Latente
LM	Modèles de langues (*Language Models*)
LSA	Analyse Sémantique Latente
LSI	Indexation Sémantique Latente
MAP	Précision Moyenne Moyenne (*Mean Average Precison*)
ML	Vraisemblance Maximale (*Maximum Likelihood*)
NB	Naïve Bayes
pLSI	Indexage Sémantique Latent probabiliste
RI	Recherche d'Information
TAL	Traitement Automatique de la Langue

À Léo Paul.

REMERCIEMENTS

Je voudrais tout d'abord remercier mon directeur Jian-Yun Nie, qui a su mener à terme cette collaboration internationale. Un grand merci aussi à Noriko Kando, co-directrice informelle de ce travail.

Je tiens aussi à remercier les gens du RALI, les excellents professeurs Philippe Langlais, Guy Lapalme, et Elliott Macklovitch, et mes collègues, en particulier Jacques Bergeron, Florian Boudin, Fabrizio Gotti, Fadila Hadouche, Stéphane Huet, Pierre-Paul Monty, Alexandre Patry, Jacques Steinlin et Mehdi Yousfi-Monod. Bien que je n'aie été présent au RALI qu'une partie de mes études, grâce à vous j'ai trouvé au RALI un environnement chaleureux et stimulant.

Merci à Satoko Fujisawa, Daisuke Ishikawa et prof. Atsuhiro Takasu du NII pour leurs bons conseils. Merci au Ministère de l'Éducation, de la Culture, des Sports, des Sciences et de la Technologie Japonais pour le financement de ces travaux.

Finalement, merci à ma famille, Jean-Pierre, Marianne, Mina, Sophie et Thérèse, pour leur support. De toute évidence ce travail est aussi le vôtre.

CHAPITRE 1

INTRODUCTION

1.1 Motivation

La Recherche d'Information (RI) est le domaine scientifique s'intéressant au problème de présenter des informations pertinentes en réponse à la requête d'un utilisateur. Bien que le domaine puise ses origines dans les systèmes d'information (bibliothécaires), de nos jours, il a maturé en une science pluridisciplinaire. En effet, comme les données retrouvées sont le plus souvent sous forme textuelle, leur traitement a de fortes relations avec le Traitement des Langues Naturelles (TAL). Les documents eux-mêmes sont semi-structurés, et forment une collection dont la gestion a des considérations connexes aux bases de données. La notion de pertinence d'un document à une requête est un concept subtil, qui touche à la philosophie et la psycholinguistique. L'explosion d'information causée par l'internet depuis les années 90 a donné une importance nouvelle au domaine ; la recherche Web fait maintenant partie de la vie de tous les jours et c'est sans doute maintenant la tâche la plus importante du TAL.

La RI a toutefois des considérations bien différentes des autres tâches du TAL. On peut tout d'abord noter l'envergure des données traitées par les systèmes d'information. Par exemple, un index Web contient des milliards de documents et est en continuel changement. De plus, les utilisateurs de ces systèmes s'attendent à avoir des résultats robustes, basés sur des millions de documents, en l'espace de dixièmes de seconde. Ceci pose des fortes contraintes d'espace et de performance, et limite effectivement la profondeur du traitement qu'on peut effectuer sur les documents. Ainsi, à des fins de performance et de stabilité, les engins de recherche modernes utilisent très peu les propriétés linguistiques du texte, et se limitent à des représentations en *sac-de-mots*, où un document est représenté par l'ensemble des comptes des mots qu'il contient, indépendamment de leur ordre.

Une autre considération de la RI est de modéliser le besoin d'information (BI) de

l'utilisateur. Ceci est souvent difficile étant donné que les requêtes sont souvent très courtes. En effet, la recherche Web aurait en moyenne 2.4 mots clés par requête [56]. Des requêtes si courtes sont naturellement très ambiguës, et cette ambiguïté nuit aux résultats de recherche [14, 22, 47–49, 51]. Par exemple, une requête simple comme "*Orange*" peut référer, entre autres, à un fruit, une couleur et une compagnie de télécommunication. Si on arrive à déterminer quel sens a le mot dans la requête, on peut espérer éliminer les documents ayant un sens différent des résultats. Ce procédé se nomme la désambiguisation de sens (DS).

Un des moyens d'effectuer une DS est d'observer le contexte dans lequel un mot apparaît. Par exemple, si on détecte que le mot *orange* est un adjectif dans une phrase, on comprend qu'il a le sens *couleur*. De la même manière, si *orange* est précédé de l'article *une*, c'est alors probablement le *fruit*. Si le mot *contrat* est présent aux alentours d'*orange*, c'est alors probablement une référence à la compagnie de télécommunication.

Jusqu'à ces jours, les tentatives d'application d'une DS à la RI ne sont toujours pas arrivées à motiver son utilisation. La plupart des études arrivent à des effets négatifs, et les quelques-unes qui ont des résultats positifs ont des problèmes de méthodologie, ou effectuent des évaluations trop limitées pour être convaincantes. Le problème est donc effectivement ignoré dans les engins modernes, et on compense indirectement avec des méthodes heuristiques comme l'utilisation d'index positionnels, ou par une expansion de la requête par des mots additionnels maximisant la cohésion de ceux-ci. Si malgré l'utilisation de ces techniques l'ambiguïté reste problématique, les utilisateurs ne peuvent que reformuler leur requête, ou ajouter des mots clés.

1.2 Objectifs

Faisant état des recherches précédentes, il est évident que la DS appliquée à la RI est un problème difficile. Nous ne nous attaquerons donc pas directement à des applications en RI, mais étudierons d'abord la relation entre un mot et des attributs de son contexte. Avec les contraintes qui se posent aux engins de recherche, pour que nos techniques puissent se concrétiser dans des engins à grande échelle, il faut que notre approche soit

très efficace et performante.

Nous ferons donc l'investigation de modèles de contexte simples utilisant uniquement des statistiques d'occurrence des attributs des mots du contexte. En particulier, il est important de bien définir les attributs à utiliser, et de leur attribuer un poids approprié. De nos jours, la plupart des approches limitent le contexte à une courte "fenêtre" de mots aux alentours d'un *mot cible*. Ceci comporte quelques problèmes : (1) on ignore les mots hors de cette fenêtre malgré qu'on puisse s'attendre à ce qu'ils aient une (petite) importance. (2) les mots présents à l'intérieur de la fenêtre ont souvent des poids uniformes, peu importe leur distance ou leur relation avec le mot cible. Pour remédier à cette situation, nous proposerons une méthode pour apprendre les poids d'un mot de contexte en fonction de sa distance au mot cible. Cette approche est testée sur des tâches de DS de Semeval, un atelier d'évaluation sémantique. Sur des tâches en anglais et en japonais, nous obtenons des résultats compétitifs avec l'état de l'art, avec toutefois des méthodes beaucoup plus légères ne nécessitant pas l'utilisation de ressources ou outils externes.

Nous ferons ensuite une investigation sur l'adaptation des méthodes développées à la RI, en combinant l'approche de la *vraisemblance de la requête* avec l'usage de modèles de contexte à variables latentes. Ces modèles sont appris automatiquement sur les documents de la collection avec une adaptation du procédé *Latent Dirichlet Allocation* (LDA) [4]. Nous effectuerons des tests avec des données d'évaluation de la *Text REtrieval Conference* (TREC). Bien que l'adaptation soit très simple, on observe des augmentations proportionnelles d'en moyenne 12% du MAP et 23% du $GMAP$. Nous procéderons à une analyse en profondeur des résultats. Ceux-ci montrent que les gains se font surtout sur les requêtes difficiles et ainsi augmentent la stabilité de recherche.

1.3 Organisation du mémoire

Dans le chapitre 2, nous ferons une revue des concepts et approches utilisés en désambiguisation de sens. Nous présenterons ensuite une méthode d'apprentissage des poids des mots de contexte selon leur distance du mot cible. Cette approche sera ensuite utilisée dans la construction de systèmes de DS Bayes Naïfs très performants. Ceux-

ci seront testés sur des tâches supervisées de l'atelier Semeval en anglais et en japonais. Nous terminerons le chapitre avec une discussion des résultats et présenterons des points de recherche futurs.

Le chapitre 3 contient une courte revue du domaine de la Recherche d'Information. Nous présenterons les concepts principaux, les moyens d'évaluation des systèmes et l'approche des modèles de langue pour la RI.

Le chapitre 4 fait un survol de la recherche sur les applications de DS pour la RI. Nous présenterons ensuite une adaptation des systèmes supervisés construits précédemment vers des systèmes non-supervisés à variables latentes pour la RI. Ceux-ci seront ensuite testés sur des collections du TREC. Suivra une discussion sur les résultats et des points de recherche futurs.

Nous conclurons dans le chapitre 5 avec une synthèse des contributions de ce mémoire et une discussion sur les avenues de recherche ouvertes par celui-ci.

CHAPITRE 2

DÉSAMBIGUISATION DE SENS

2.1 Introduction

La Désambiguisation de Sens (DS) est un procédé par lequel on identifie le sens d'une occurrence d'un mot. Ce problème est important pour remédier à des problèmes d'ambiguïté sémantique et a des implications dans des tâches du Traitement des Langues Naturelles (TAL) telles que la Traduction Automatique (TA) et la Recherche d'Information (RI). Par exemple, si un système de RI reçoit une requête contenant un mot ambigu tel que "Orange", de meilleurs résultats pourraient être obtenus si le système arrivait à déterminer le sens d'orange (ex : *orange/fruit*, *orange/couleur*, *orange/compagnie-téléphonique*, etc.) dans la requête et les documents.

L'approche générale de la DS est d'exploiter des éléments de contexte, comme la structure syntaxique ou les mots dans les environs du *mot cible*, pour construire des modèles de contexte. L'intuition derrière cette approche est que les sens du mot ont de plus grandes chances de cooccurrer avec certains mots de contexte ou des structures de phrases particulières. Par exemple, le sens *orange/fruit* a plus de chances d'avoir le mot *jus* dans son entourage ou d'être au pluriel, *orange/couleur* a plus de chance d'être un adjectif, et *orange/compagnie-téléphonique* a plus de chances d'avoir un "O" majuscule.

Les modules de DS emploient des attributs de différentes formes. Ceux-ci incluent les types lexicaux, des *sacs-de-mots* (occurrences sans position), des collocations locales (occurrences avec position), des relations syntaxiques et des attributs de modèles de sujets. Des ressources externes telles que des dictionnaires et des thésaurus sont aussi utilisées pour ajouter des connaissances aux modèles. Néanmoins, beaucoup de ces attributs nécessitent l'usage d'outils ou ressources sophistiqués, qui peuvent ne pas être disponibles ou assez performants pour des applications à large échelle. Des attributs tels que des statistiques de cooccurrence sont plus simples à obtenir. Ceux-ci sont utilisés dans pratiquement toutes les applications de la DS, sous l'hypothèse de base que le sens

d'un mot peut être défini par les mots qui l'entourent dans le texte. Nous nommerons ceux-ci *mots de contexte*. Dans la plupart des études où ils sont utilisés, on les accumule dans une "fenêtre" de texte d'une taille fixe aux alentours du mot cible. Toutes les occurrences de mots de contexte se voient habituellement attribuer la même importance. Ainsi, un mot qui est à 5 mots du mot cible sera considéré aussi important qu'un autre mot de contexte à distance 1. On assume donc implicitement que tout mot de la fenêtre a la même capacité de désambiguisation, ce qui est contre-intuitif. On pourrait plutôt penser qu'il est plus probable qu'un mot rapproché ait une plus grande importance qu'un mot éloigné.

Par exemple, dans la phrase "Le gouvernement a créé un fonds spécial pour supporter les gens affectés par l'effondrement du barrage de la rivière Mississippi", les mots "gouvernement" et "spécial" ont un beaucoup plus grand impact sur le sens de "fonds" que les autre mots de la phrase. Si le facteur de la distance du mot est ignoré, et que tous les mots sont utilisés avec la même importance, il serait possible que le sens du mot "fonds" soit faussement classifié dans le sens "extrémité" à cause de la présence des mots "rivière" et "barrage".

Ce principe peut être incorporé en utilisant une fonction dégénérescente – une fonction monotone décroissante mais positive de la distance au mot cible. Il existe beaucoup de telles fonctions : linéaire, exponentielle, logarithmique, etc. Il est toutefois difficile de sélectionner la fonction qui peut prendre en compte correctement l'impact d'un mot de contexte sur le mot cible. On ne peut se fier sur la littérature pour ce faire, car il n'existe pas d'investigation complète concernant les fonctions dégénérescentes en DS, malgré que ce soit un point fondamental.

Dans cette étude, nous suggérons que la fonction idéale dépend de la langue utilisée, et même du genre de document qui constitue la collection. En effet, des langues différentes ont souvent des structures particulières qui peuvent nécessiter des fonctions différentes. Aussi, on pourrait s'attendre à ce qu'une collection dans un domaine particulier (ex : médecine) ait besoin d'une fonction différente d'un autre domaine (ex : romans). On ne peut donc pas définir une fonction universelle. Nous proposons donc dans cette étude d'apprendre la fonction dégénérescente optimale avec des méthodes non-supervisées. De

cette manière, nous espérons définir la fonction qui convient le mieux à une collection de textes, et de retirer le besoin de la déterminer manuellement.

L'idée principale derrière cette approche est que, pour le même mot cible, des modèles de contextes faits à partir de deux ensembles d'échantillons devraient être similaires. Ainsi, la fonction dégénérescente qui maximise la similarité de tels modèles devrait être sélectionnée. En suivant cette idée, nous pouvons effectuer une descente de gradient pour maximiser une mesure de similarité en fonction de poids assignés à chaque distance. Il faut noter que nous n'assumons pas que les modèles de contexte ainsi construits correspondent à un sens unique. Ceux-ci impliquent l'ambiguïté des mots, et ainsi tous les sens sont inclus dans le même modèle. Ceci n'affecte pas notre proposition de base : étant donné que les deux modèles de contexte devraient être également ambigus, leur similarité devrait tout de même être la plus haute possible.

Ces prémisses nous permettront de trouver la fonction dégénérescente optimale pour une collection de textes. Les poids résultants peuvent ensuite être utilisés pour construire des modèles de contexte pour chaque occurrence de mot, ce que nous utiliserons dans des classifieurs Bayes Naïfs (*Naïve Bayes* – NB). Les systèmes résultants seront ensuite testés et comparés à plusieurs fonctions définies manuellement, en utilisant deux ensembles de données de DS : nous ferons tout d'abord quelques expériences en anglais avec les données de la tâche Semeval-2007 *English Lexical Sample* (ELS), puis en japonais avec une participation à la tâche Semeval-2010 *Japanese WSD* (JWSD).

Le reste de ce chapitre est organisé de la manière suivante. La section 2.2 présente l'état de l'art en DS, les attributs et ressources utilisés. La section 2.3 présente nos modèles de contexte et se concentre sur le problème d'apprendre une fonction dégénérescente pour donner des poids aux mots de contextes en fonction de leur distance. Dans la section 2.4 ceux-ci seront combinés avec d'autres attributs pour construire des classifieurs supervisés. Dans la section 2.5 et 2.6, nous évaluerons ces classifieurs avec des tâches de Semeval. Nous conclurons en section 2.7 et 2.8 avec une discussion et des points de recherche futurs.

2.2 L'état de l'art en DS

L'ambiguïté des mots cause des problèmes dans beaucoup de tâches (traduction automatique, recherche d'information, etc.). Par exemple, dans la phrase "He is suffering from an unknown affection", le mot "affection" en anglais est ambigu et peut signifier ou un sentiment ou une maladie. En français, il y aurait effectivement deux traductions pour ce mot : "affection" et "maladie". Il est important de lever l'ambiguïté afin de choisir la bonne traduction. On pourrait appliquer une DS dans de telles situation. L'objectif de base est de déterminer quel sens a un mot, étant donné le contexte ou celui-ci apparaît. Dans cet exemple particulier, nous pourrions utiliser la présence du mot "suffering" pour conclure que le sens utilisé est le sens "maladie".

L'ambiguïté des mots est un problème fondamental lorsqu'on traite avec des langues naturelles. La DS a donc été longuement étudiée et plusieurs approches ont été proposées. En faire un survol complet n'est pas l'objectif de ce mémoire ; nous discuterons seulement des considérations principales et des études pertinentes. Pour des revues complètes du domaine, le lecteur peut consulter [19] et [33].

2.2.1 Considérations principales

En général, les méthodes de DS commencent par déterminer quels sont les sens d'un mot et comment ceux-ci sont représentés. Une méthode très fréquente pour déterminer un *inventaire de sens* est d'en faire une liste. Par exemple, pour le mot "orange", on pourrait décider que l'inventaire est : {couleur, fruit, compagnie-de-télécommunication}. On constate qu'il est discutable de distinguer le fruit et la couleur parce qu'ils sont très liés ; on nomme cette considération la *granularité* de l'inventaire. De plus, on peut se demander s'il est correct d'inclure un sens pour la compagnie de télécommunication, quand il y a d'autres compagnies moins connues qui se nomment aussi "Orange", et qu'il existe même une ville de France du même nom ; cette considération se nomme la *couverture* de l'inventaire.

Ces deux problèmes sont très liés à ceux qu'ont les lexicographes quand ils doivent créer une entrée de dictionnaire. Sur ce point, il est intéressant de citer Kilgarriff [20] :

“Un lexicographe n’est pas seulement un lexicographe avec une limite de temps, mais aussi un lexicographe avec une limite de pages” (*Not only is a lexicographer ’a lexicologist with a deadline’ (Fillmore, 1988) but also a lexicologist with a page limit.*) On voit donc que souvent, l’inventaire de sens est créé de manière à couvrir les sens “importants” en “quelques entrées”. Il est donc normal que des sens clairement définis ne soient pas subdivisés, ou absents car ils sont peu fréquents.

En informatique, on considère qu’un inventaire de sens ayant la plus grande couverture possible est désirable. La couverture n’est ainsi limitée que par les ressources à notre disposition. La granularité des inventaires est toutefois problématique. Il est en effet difficile pour les humains de s’entendre avec précision quand les sens sont très fins [18]. Cette difficulté s’applique aussi aux systèmes, dont les performances souffrent quand les sens sont fins.

Quand un inventaire de sens est établi, on peut ensuite se demander comment représenter un sens et comment faire le lien entre les représentations des sens et les mots dans le texte. On extrait ainsi des attributs du contexte du mot. Si on a des instances étiquetées, on peut aussi amalgamer ces instances pour avoir un modèle par sens, et en faire des classifieurs. Plus simplement, on peut aussi faire des distances entre les représentations de sens, ce qu’on nomme *discrimination de sens*.

2.2.2 Utiliser des attributs linguistiques

Les études passées font l’usage de plusieurs types d’attributs linguistiques. Le système NUS-ML [10] est un système supervisé représentatif de l’état de l’art. Dans cette section, nous décrirons ce système en détails, parce qu’il utilise la plupart des attributs proposés dans les autres études. C’était aussi le système le plus efficace de la tâche Semeval-2007 ELS sur laquelle nous allons effectuer des expériences.

L’architecture de base de ce système est basée sur un système préalable [24] par les mêmes auteurs. Ce système utilisait des classifieurs Machines à Vecteur de Support (Support Vector Machines – SVM), apprenant à discriminer entre le sens en utilisant plusieurs types d’attributs linguistiques. Pour leur participation à Semeval-2007, les auteurs proposent l’addition d’attributs issus de modèles de sujets (*topic models*). Dans des expé-

riences préalables cet ensemble d'attributs (sauf ceux des modèles de sujets) ont eu des meilleurs résultats avec des classifieurs SVM que les classifieurs à entropie maximale et les classifieurs Bayes Naïfs (NB). L'intégration des nouveaux attributs de sujets est toutefois difficile avec des classifieurs SVM. En effet, ceux-ci ne peuvent utiliser la nature probabiliste des modèles de sujets, car ils exigent que les instances de mots à classifier aient la même représentation que les instances d'entraînement. Pour ce faire, il faudrait estimer une inférence pour chaque instance, et utiliser les proportions des sujets inférés comme attributs. La classification d'une nouvelle instance utilisant ces attributs serait donc basée sur une information dégradée, moins précise. De plus, comme dans ce cas il y a beaucoup d'attributs mais très peu de données d'entraînement, pour assurer une robustesse de classification, il faudrait utiliser un SVM à noyau linéaire. Ceci résulte en un modèle qui est un simple hyperplan discriminateur, imposant l'indépendance des attributs. Cette indépendance est contraire aux objectifs des modèles de sujets, qui réussissent à combiner plusieurs sujets simultanément.

Il est donc difficile d'utiliser les modèles de sujets dans le cadre des classifieurs SVM. En remplacement, les créateurs du système NUS-ML ont choisi d'utiliser des classifieurs NB. Leur système utilise les attributs suivants :

Types lexicaux On enregistre le type lexical des mots à chaque position dans une fenêtre de taille 3, incluant le mot cible. Comme nous avons vu avec notre exemple *orange*, les types lexicaux sont clairement reliés au sens des mots.

Sac-de-mots Dans une fenêtre de taille 3, tous les mots dans la fenêtre se voient attribuer un poids identique et sont utilisés pour former un vecteur de contexte ignorant la position des mots. Comme mentionné dans la section 2.1, donner des poids uniformes aux mots de la fenêtre et ignorer complètement les mots plus éloignés est contre-intuitif. Ce type d'attributs sac-de-mots est utilisé dans toutes les approches de DS, dès les travaux de Lesk [26].

Collocations locales Dans ce cas, "collocation" n'a pas le sens habituel. Par collocation, on réfère simplement à une paire de mots formée à une position pré-déterminée. L'idée derrière ces attributs est l'idée courante d'*un sens par collocation* [68]. Les

positions utilisées dans [10] sont les unigrammes (collocations avec le mot cible) aux positions -2, -1, 1, 2, les bigrammes à (-2,-1), (1,2) et les saute-grammes (*skip-grams*) aux positions (-3,-1), (-2,1), (-1,1), (-1,2), (1,3). Comme nous pouvons voir, celles-ci sont symétriques entre ce qui vient avant et après. La sélection de ces paires a été proposée dans des travaux préalables [35, 36].

Relations syntaxiques Les attributs précédents proviennent de ce qu'on nomme le *contexte local.* Un moyen d'aller plus loin dans le contexte est d'utiliser des relations syntaxiques. Dans ce système, certains attributs sont obtenus en utilisant un ensemble de patrons lexicaux. Par exemple, si le mot cible est un nom, on va chercher le *mot tête* (mot principal d'une phrase nominale) et on extrait la forme du mot, son type lexical, et une variable Booléenne représentant si le mot est au-dessus ou au-dessous du mot cible dans l'arbre syntaxique. L'ensemble des patrons et attributs utilisés dépend du type de mot (nom, verbe, adjectif, etc.). Dans ce système, il n'est pas clair si le choix des attributs dépend du type lexical issu d'un étiquetage automatique, ou de l'utilisation du type du mot tel que spécifié dans les données de la tâche (ex : area-n, begin-v, etc.). Le classifieur pourrait donc avoir été construit en utilisant des informations qui ne seraient pas disponibles dans des vraies applications. En utilisant le type lexical résultant d'une analyse automatique, il serait difficile d'identifier quel ensemble d'attributs on devrait utiliser, car certains mots de la tâche peuvent prendre plusieurs types lexicaux (ex : *end*, *express*, *grant*). Dans le cadre de cette tâche, l'ensemble des patrons et attributs ont étés sélectionnés par validation croisée, et on a probablement évité les attributs problématiques sur l'ensemble de ces mots. Pour terminer, il faut noter que ces attributs nécessitent l'utilisation d'analyseurs lexicaux, ce qui est problématique pour des tâches comme la RI étant donné l'envergure des ensembles de données utilisés.

Attributs de sujets Comme les données étiquetées sont souvent limitées, il arrive souvent que des mots de contexte d'un nouvel échantillon ne soient pas observés dans les données d'entraînement. Ces mots auront des probabilités instables, ce qui nuit à la classification. Comme les modèles contiennent souvent des synonymes ou des formulations différentes de ces mots, ces erreurs dépendent du choix du

vocabulaire utilisé et seraient réglées par une reformulation. On nomme ainsi ce problème la *non-concordance du vocabulaire*. Pour réduire ce problème, on peut faire l'usage de modèles de sujets. L'allocation Dirichlet latente (*Latent Dirichlet Allocation* – LDA) [4] (discuté en détail dans la section 4.3.2) est un modèle à variables latentes, qui apprend de manière non-supervisée un ensemble de sujets (distribution de mots). Utilisant ces sujets, on peut représenter un document par une mixture à basse dimensionalité, donnant une probabilité plus haute à des mots connexes à ceux contenus dans le document. Pour la DS, en "étendant" les documents avec des modèles de sujets on peut stabiliser les probabilités des modèles et ainsi réduire la non-concordance du vocabulaire. Dans NUS-ML, on calcule un ensemble de sujets sur un grand corpus. On infère ensuite des mixtures de ceux-ci pour les documents des données d'entraînement. La proportion des allocations des mots de contexte dans une fenêtre de taille 3 autour des mots cibles est utilisée pour calculer une mixture de sujets pour chaque sens. On ajoute au score la probabilité des mots de l'échantillon (dans une fenêtre de taille 3) selon la mixture de sujets du sens.

On peut observer quelques problèmes avec l'usage des attributs spécifiés ci-haut. Tout d'abord, les paires de mots utilisées ont l'objectif de couvrir toutes les collocations possibles dans une fenêtre de 3. Néanmoins, on constate que beaucoup de ces paires ne forment pas des collocations sensées. Étant donné la fine division de ces attributs, ceux-ci pourraient introduire du bruit néfaste à la classification. Un autre aspect manquant est un poids différent de ces couples selon leur distance. Intuitivement, une paire de mots plus proche aurait une plus grande chance de former une collocation réelle qu'une paire plus éloignée. Une explication de ces attributs serait qu'en regardant bien les paires, on voit que les mots proches sont présents dans plus de paires, et donc celles-ci ont l'effet d'une fonction de poids ad-hoc. Notre méthode vise justement à investiguer ces considérations.

Une observation à propos de l'applicabilité de ce système est que l'utilisation d'outils linguistiques sophistiqués et autres ressources pour la DS ne sont pas toujours disponibles dans la pratique (ex : outil d'étiquetage lexical dans beaucoup de langues). Même

quand ceux-ci sont disponibles, leur précision, couverture et fiabilité sont parfois discutables. En effet, comme la plupart de ces outils sont entraînés sur des corpus annotés, ceux-ci auront fort probablement une précision réduite sur des collections différentes des données d'entraînement. Le Web, qui est la collection la plus importante pour la RI, est particulièrement problématique étant donné sa taille et la diversité des documents qu'elle contient.

De plus, le temps de traitement que nécessitent ces outils est souvent trop long pour motiver leur utilisation pour des applications réelles. Par exemple, le calcul de modèles de sujets en utilisant LDA est connu pour être très lourd. Toutefois, bien que les bienfaits de leur utilisation aient étés démontrés [65], ils ne sont toujours pas utilisés pour des applications à grande échelle comme la RI.

Étant donné les considérations énoncées ci-haut, nous tournons vers l'usage exclusif d'attributs de cooccurrence de mots de contexte avec le mot cible.

2.2.3 Méthodes connexes

Dans cette section, nous décrirons quelques approches qui sont en vigueur dans le domaine.

2.2.3.1 Méthodes à base de mémoire

Parmi les méthodes supervisées, autre que les classifieurs qui font une agrégation des instances d'une classe, il y a aussi l'usage de méthodes dites à base de mémoire. Celles-ci consistent à garder les instances entières en mémoire, et calculent des scores de similarité entre un élément et l'ensemble des mots. Ce faisant, on peut définir un voisinage d'instances et observer leur classe avec des classifieurs k plus proches voisins (kPPV) [35]. Un des problèmes rencontrés avec de tels classifieurs est que comme les instances sont éparses, on doit donc en avoir une quantité immense, ou bien les "augmenter" en utilisant des sources de connaissance [26]. La mesure de similarité utilisée est libre de choix par l'utilisateur ; elle est souvent une mesure simple comme la fonction cosinus ou la distance d'Hamming.

L'usage de tels classifieurs était très fréquent en 2010 dans le cadre de la tâche de DS translinguistique de Semeval [25]. Dans cette tâche, le sens d'un mot est exprimé par sa traduction dans plusieurs langues européennes. L'idée n'est pas nouvelle [8], mais elle résout des problèmes reliés à l'étiquetage des données et aux inventaires de sens. Comme il n'y a pas qu'une seule traduction possible pour une instance de mot, les mots sont présélectionnés et les traductions sont groupées afin de faciliter l'annotation. Les groupes de traductions ne sont toutefois pas donnés aux participants. Sans une unique cible de traduction, la plupart des participants ont délaissé les méthodes supervisées habituelles comme SVM et NB, et sont passés à des méthodes à mémoire.

Ces méthodes ont l'avantage d'avoir un faible biais, mais comme le calcul du score implique l'utilisation de tous les instances de la collection, ces méthodes ne sont pas applicables à des problèmes à grande échelle tels que la RI.

2.2.3.2 Modèles de contextes pondérés

L'idée générale derrière les modèles de contexte est d'utiliser les mots qui cooccurrent avec le mot cible, pour aider à spécifier leur sens. Quand de tels modèles sont construits pour un mot cible à désambiguïser, on peut le comparer avec le modèle de contexte de chaque sens afin de sélectionner le plus proche.

Derrière cette idée générale, il reste toujours le problème important de déterminer comment construire le modèle. Comme décrit précédemment, la méthode de base utilise une fenêtre de taille fixe autour du mot cible où chaque occurrence compte pour 1. En d'autres termes, on suppose que tous les mots de contexte dans la fenêtre ont la même importance par rapport au sens du mot cible. Comme discuté précédemment, ceci est contre-intuitif. En réalité, nous pouvons nous attendre à ce que les mots de contexte rapprochés aient un plus grand impact sur le sens du mot cible que les mots plus distants. Cette observation est la motivation pour les approches qui utilisent une fonction dégénérescente pour réduire l'importance (ou poids) des mots de contextes en fonction de leur distance au mot cible.

Plusieurs fonctions dégénérescentes ont été proposées dans différentes tâches du TAL. Dans [13], on propose l'utilisation d'une fonction dégénérescente exponentielle

pour modéliser la force de dépendance entre deux mots, dans le cadre de la RI. La force de dépendance entre deux mots x et y est définie comme étant $e^{-\delta(Dist(x,y)-1)}PMI(x,y)$. Dans cette équation, $Dist(x,y)$ est la distance moyenne entre les occurrences des deux mots dans la collection. $PMI(x,y)$ est l'Information Mutuelle Ponctuelle entre les deux mots ($\log \frac{p(x,y)}{p(x)p(y)}$), où les probabilités sont basées sur des statistiques de cooccurrence dans une fenêtre fixe. Cette mesure est utilisée pour maximiser la cohésion d'un ensemble de mots de traduction pour une requête en RI translinguistique (RITL).

Dans [37], il est proposé de représenter un sens par le centroïde des vecteurs de contexte de ses instances étiquettées. Les composantes de ces vecteurs sont limitées à des statistiques de cooccurrences avec les mots de contexte. Pour améliorer la qualité de la représentation, avant la normalisation, les fréquences des mots sont multipliées par plusieurs facteurs. Un de ces facteurs consiste à diviser la fréquence par la racine carrée de la distance moyenne des occurrences, selon un principe que les auteurs nomment la "densité locale". Une amélioration de 3% de la précision de désambiguation est observée par l'addition de ce facteur. Comme nous pouvons nous y attendre, vu que dans la plupart des cas il n'y a qu'une instance d'un mot par échantillon, ceci est équivalent à appliquer une loi de puissance $d^{-\delta}$ avec un paramètre $\delta = 0.5$ sur les occurrences de contexte.

Dans [54], on utilise l'Analogie d'Hyperespace au Langage (*Hyperspace Analog to Language – HAL*), qui utilise une fenêtre glissante de taille fixe, afin de déterminer des statistiques de cooccurrence de mots. Un mot de contexte plus proche se trouvera dans plus de fenêtres qu'un mot de contexte plus éloigné. Ceci est équivalent à appliquer une fonction dégénérescente linéaire aux mots de contexte car le poids d'un mot de contexte est défini par $N - d$, où N est la taille de la fenêtre, et d est la distance du mot. Les vecteurs de contexte définis de cette manière ont été utilisés afin d'estimer la similarité entre les mots. L'utilisation des similarités résultantes pour l'expansion de requête pour la RI a eu un effet positif[1].

Dans une étude plus récente [28], on propose l'utilisation de modèles de langues positionnels pour la RI, en distribuant le poids des occurrences de mots à leurs positions environnantes. Pour ce faire on calcule des comptes propagés $c'(w,i) = \sum_j c(w,j)k(dist(i,j))$, où $c(w,j)$ est le compte pour un mot w à une position j dans le document, et k étant une

fonction de noyau dégénérescente. En utilisant ceci, on peut déterminer des scores pour un mot à toutes les positions du document, et combiner ces scores en utilisant des stratégies heuristiques. Les expériences montrent que de telles fonctions peuvent améliorer l'efficacité de la recherche. Comme il n'était pas clair quelle fonction de noyau fonctionnerait le mieux, les auteurs ont fait des expérimentations essayant plusieurs types de fonctions : Gaussienne ($e^{-\delta d^2}$), Linéaire ($\max\{0, 1-\delta d\}$), Cosine ($\cos(\delta d)$ jusqu'à $d = \frac{\pi}{2\delta}$), et Circulaire ($\sqrt{1-\delta d^2}$ jusqu'à $d = \delta^{-\frac{1}{2}}$). De celles-ci, la fonction Gaussienne a donné le meilleur résultat.

Comme nous le voyons, toutes ces études utilisent une fonction ayant une forme prédéfinie. Toutefois, ces fonctions n'ont pas été comparées à fond pour une même tâche. Il n'est pas clair quelle fonction fonctionnerait le mieux dans le cadre de la DS, car aucune de ces études ne s'est penchée sur ce point. Comme elles ne sont pas utilisées en DS, il n'est pas clair de voir si de telles fonctions peuvent mener à des améliorations comparativement à de simples fonctions uniformes. Plutôt que de définir une fonction manuellement, nous allons tenter d'apprendre les poids optimaux à partir des données. Dans la prochaine section, nous décrirons nos modèles proposés, ainsi que la méthode non-supervisée pour calculer des poids pour les mots de contexte en fonction de leur distance.

2.3 Apprendre le modèle de contexte optimal

2.3.1 Modèles de contexte

Nous proposons l'usage de modèles de contexte simples composés d'une distribution unigramme des mots qui occurrent dans les alentours du mot cible. Nous assumons que des mots de contextes à des distances différentes ont des importances différentes, que nous appellerons compte dans le modèle de contexte. Comme la distance dans la fenêtre est un entier, on peut définir ces poids comme un vecteur, soit ω_i le compte que prend une occurrence de mot quand il est à la distance i. Cette notation est très simple, mais généralise toutes les fonctions possibles. Par exemple, une fonction uniforme dans une fenêtre de taille s peut être définie par $\omega_i = 1$ si $1 \leq i \leq s$, 0 sinon. Nous décrirons plus

tard la manière de décrire cette fonction ω_i.

Si nous assumons avoir une fonction ω_i, étant donné un ensemble d'entraînement du mot cible (possiblement pour un sens spécifique), nous pouvons décrire son modèle de contexte de la manière suivante. Soit W un ensemble de fenêtres contenant le mot cible t , et soit $c_{W,i,x}$ un nombre d'occurrences du mot de contexte x à la distance i dans W. Nous pouvons alors définir le modèle de contexte de vraisemblance maximale (*maximum likelihood* – ML) comme étant :

$$P_{ML,W}(x) = \frac{\sum_i \omega_i c_{W,i,x}}{\sum_{i,x'} \omega_i c_{W,i,x'}}$$

Une telle distribution de probabilités forme un modèle de contexte pour le mot cible. Comme cette formulation souffre du problème de probabilité zéro [52], il faut donner une partie de la masse de probabilité à des mots qui ne sont pas observés. On nomme "lissage" l'opération consistant à modifier une distribution pour la rendre plus réaliste. Dans cet ouvrage, nous proposons l'usage de lissage Dirichlet utilisant les probabilités de collection $P(x|\mathcal{C})$ comme a priori :

$$P_{Dir,W}(x) = \frac{\sum_i \omega_i c_{W,i,x} + \mu_W P(x|\mathcal{C})}{\sum_{i,x'} \omega_i c_{W,i,x'} + \mu_W}$$

Cette manipulation permet de donner une masse non-nulle à toutes les observations et réduit les problèmes de données éparses.

2.3.2 Estimation d'un a priori Dirichlet approprié

Le pseudo-compte μ_W contrôle la force de l'a priori, et on suppose habituellement que c'est une constante définie manuellement. Pour que notre modèle suive les données, nous estimons automatiquement ce pseudo-compte de façon à maximiser la vraisemblance retire-un (*leave-one-out*) :

$$\mathcal{L}_{-1}(\mu_W) = \sum_i \sum_{x \in V} \omega_i c_{W,i,x} \log P_{Dir,W-\omega_i}(x)$$

$$P_{Dir,W-\omega_i}(x) = \frac{\sum_j \omega_j c_{W,j,x} - \omega_i + \mu_W P(x|\mathcal{C})}{\sum_{j,x'} \omega_j c_{W,j,x'} - \omega_i + \mu_W}$$

où V est le vocabulaire. $P_{Dir,W-\omega_i}(x)$ est la probabilité estimée en retirant une occurrence de x ayant un poids ω_i. En retirant une occurrence et calculant sa vraisemblance, on peut estimer la vraisemblance des données futures. En choisissant le pseudo-compte qui maximise cette valeur, on peut appliquer un lissage approprié à la précision des données accumulées. Comme nous pouvons le voir, la formule ci-dessus a été adaptée afin de permettre des comptes non-uniformes aux occurrences. Nous utilisons la méthode de descente de gradient de type Newton :

$$\mu^{(0)} = 1 \qquad \mu^{(k+1)} = \mu^{(k)} - \frac{\mathcal{L}'_{-1}(\mu^{(k)})}{\mathcal{L}''_{-1}(\mu^{(k)})} \qquad \text{jusqu'à} \quad \left|\mu^{(k+1)} - \mu^{(k)}\right| < \epsilon$$

Ceci résulte en les dérivées suivantes :

Soit $C_{W,x} = \sum_j \omega_j c_{W,j,x}, \quad C_W = \sum_x C_{W,x}$,

$$\begin{aligned}
\mathcal{L}'_{-1}(\mu) &= \sum_i \sum_{x \in V} \omega_i c_{W,i,x} \left[\left(\frac{P(x|\mathcal{C})}{C_{W,x} - \omega_i + \mu_W P(x|\mathcal{C})} \right) - \left(\frac{1}{C_W - \omega_i + \mu_W} \right) \right] \\
\mathcal{L}''_{-1}(\mu) &= -\sum_i \sum_{x \in V} \omega_i c_{W,i,x} \left[\left(\frac{P(x|\mathcal{C})}{C_{W,x} - \omega_i + \mu_W P(x|\mathcal{C})} \right)^2 - \left(\frac{1}{C_W - \omega_i + \mu_W} \right)^2 \right]
\end{aligned}$$

Utiliser cette méthode d'estimation enlève le besoin d'un paramètre additionnel. De plus, le pseudo-compte estimé s'ajuste automatiquement à l'envergure des nombres dans ω.

2.3.3 Évaluer la similarité

L'idée de base derrière notre approche est très simple. Si nous assumons que le modèle de contexte tel que défini ci-dessus est une représentation du sens d'un mot, alors si deux mots a et b sont synonymes, les modèles de contexte faits à partir des ensembles de fenêtres A et B, ceux-ci contenant des échantillons des deux mots, devraient être similaires. Ainsi, commençant avec des poids uniformes, nous pouvons alors aller vers la maximisation d'une mesure de similarité de distribution sur ω. Si T est un ensemble

contenant la totalité des fenêtres pour notre mot cible t, nous pouvons définir A et B comme une partition aléatoire de T en deux ensembles. On peut noter que même si le mot cible t est polysémique, si les échantillons sont pris aléatoirement d'un même corpus, la mixture des sens devrait être similaire dans ces deux ensembles, et cette mixture représenterait l'amalgame des sens du mot.

Pour nos tests initiaux nous avons essayé des mesures de similarité populaires. La première de celles-ci est la *divergence Kullback-Leibler* (KL), aussi connue sous le nom d'entropie relative. C'est une mesure de divergence fondée sur la théorie de l'information, indiquant la quantité de bits additionnels moyenne nécessaire pour utiliser un encodage P, quand la distribution réelle est Q. On la définit par :

$$\begin{aligned} D_{KL}(P,Q) &= \sum_x P(x) \log \frac{P(x)}{Q(x)} \\ &= -\sum_x P(x) \log Q(x) + \sum_x P(x) \log P(x) \\ &= H(P,Q) - H(P) \end{aligned}$$

Comme nous pouvons le voir, c'est équivalent à la différence entre l'entropie croisée $H(P,Q)$, et l'entropie de$H(P)$. À cause du logarithme utilisé dans l'équation de l'entropie croisée, la seconde distribution Q doit être lissée.

Une autre mesure populaire que nous avons tenté d'utiliser est le Radius d'Information (*Information Radius* – $IRad$), aussi connu sous le nom de *divergence Jensen-Shannon* (JS), et *différence totale à la moyenne*. C'est la moyenne des divergences KL des deux distributions contre la moyenne des deux distributions :

$$IRad(P,Q) = \frac{1}{2} D_{KL}\left(P, \frac{P+Q}{2}\right) + \frac{1}{2} D_{KL}\left(Q, \frac{P+Q}{2}\right)$$

Un avantage de cette mesure est qu'elle est symétrique et bornée ; elle a une valeur de zéro quand les deux distributions sont identiques, et de un quand elles sont complètement différentes. $1 - IRad(P,Q)$ se comporte comme une cosine, ce qui explique le nom Radius d'Information. Aussi, faire la moyenne des deux distributions implique une forme de lissage.

Malheureusement, ces deux mesures ont des problèmes car l'entropie de base des modèle est annulée. Ces deux mesures ne considèrent ainsi que la similarité des distributions, peu importe l'information qui est contenue dans celles-ci. Par exemple, la divergence KL est minimale (nulle) quand les distributions sont identiques, peu importe les distributions. Ces distributions pourraient aussi bien être celles qui coïncident avec les données (distribution optimale) ou une distribution basée sur des poids uniformes (peu utile). La divergence KL ne différenciera pas ces deux cas. Il faudrait donc en plus de maximiser la similarité, aussi maximiser l'utilité des distributions. Ceci ne peut être fait en minimisant uniquement la divergence KL. L'$IRad$ a le même problème.

Les observations présentées ci-haut nous montrent qu'il faut combiner deux aspects : maximiser la similarité entre les modèles, et le pouvoir expressif de ceux-ci. En d'autres termes, il faut pénaliser les modèles peu expressifs. Nous proposons ainsi de définir cet expressibilité par l'entropie des modèles : plus basse est l'entropie, meilleur est le modèle. La mesure que nous devrions minimiser pourrait donc être :

$$D_{KL}(P, Q) - H(P) = H(P, Q)$$

Minimiser l'entropie croisée est donc suffisante pour assurer des modèles similaires à basse entropie. De plus, nous pouvons stabiliser cette valeur en utilisant les deux directions comme le fait l'$IRad$:

$$H(P, Q) + H(Q, P)$$

Tel qu'expliqué ci-haut, la mesure d'entropie croisée $H(P, Q)$ demande que la deuxième distribution soit lissée pour contourner le problème de probabilité zéro. Nous arrivons donc à la fonction de perte suivante que nous voudrons minimiser :

$$l(\omega) = H(P_{ML,A}, P_{Dir,B}) + H(P_{ML,B}, P_{Dir,A})$$

Voici les dérivées nécessaires à une descente de gradient (W et $T - W$ sont A et B ou l'inverse) :

$$\begin{aligned}
\frac{\partial H\left(P_{ML,W}, P_{Dir,(T-W)}\right)}{\partial \alpha_i} &= -\sum_{x \in V} \Bigg[\frac{\partial P_{ML,W}(x)}{\partial \alpha_i} \log P_{Dir,(T-W)}(x) + \\
&\qquad \frac{\partial P_{Dir,(T-W)}(x)}{\partial \alpha_i} \times \frac{P_{ML,W}(x)}{P_{Dir,(T-W)}(x)} \Bigg] \\
\frac{\partial P_{ML,W}(x)}{\partial \alpha_i} &= \frac{c_{W,i,x} - P_{ML,W}(x) c_{W,i}}{\sum_j \alpha_j c_{W,j}} \\
\frac{\partial P_{Dir,W}(x)}{\partial \alpha_i} &= \frac{c_{W,i,x} - P_{Dir,W}(x) c_{W,i}}{\sum_j \alpha_j c_{W,j} + \mu_W}
\end{aligned}$$

Maintenant, le lecteur peut voir que l'optimal de cette fonction (ou des autres mesures ci-haut) dépend de la quantité de comptes dans les modèles. Par exemple, avoir peu d'instances favorise des poids uniformes. Au contraire, avec plus d'instances on peut arriver à une plus basse entropie croisée en assignant la plupart du poids vers les positions plus proches. Dans le cas le plus extrême, nous obtiendrons une solution dégénérée avec tout le poids à la distance 1. Le problème de savoir quelle quantité de données utiliser n'est pas facilement résolu. Le problème est encore plus difficile dans notre cas étant donné que nous utiliserons un corpus externe dans nos expériences. Nous pourrions tenter de réduire le problème en ajoutant des a priori sur les paramètres, ou de manière équivalente faire une régularisation, mais la quantité d'instance aura encore un effet, et ceci laisserait encore le problème de choisir la force des a priori ou régularisation.

En remplacement, nous proposons d'utiliser beaucoup d'instances, mais utiliser un arrêt rapide (*early stop*), sachant que les meilleurs poids se trouvent sur le chemin de descente de gradient, entre les poids uniformes et la solution dégénérée. Une façon facile de choisir un point d'arrêt serait d'évaluer les poids sur les données de tâche, à mesure que l'apprentissage avance. Nous avons toutefois trouvé que ceci était lourd et peu commode. En pratique nous avons trouvé que sous l'hypothèse que les mots éloignés devraient avoir une distance proche de zéro, nous pourrions arrêter l'algorithme quand des poids vont sous une petite valeur. Il n'est pas clair ce qu'une telle valeur devrait être, mais les valeurs que nous avons choisies en regardant les graphes des poids ont bien fonctionné.

Pour nos expériences, nous avons utilisé la descente de gradient stochastique : on

choisit un mot au hasard (ambigu ou non) à partir du corpus, on calcule son gradient, on fait un petit pas de gradient, et le procédé est répété. L'algorithme 2.1 est un pseudo-code de ce procédé. Le procédé ci-haut est appliqué sur une collection en anglais, plus une collection japonaise, avec $\eta = \epsilon = 0.001$. Dans des tests additionnels, nous observons qu'avoir un pseudo-compte optimal pour chaque mot lors de l'algorithme n'est pas crucial, et qu'avec un $\mu = 1000$ manuel, les poids se redimensionnent pour donner un a priori adéquat. Dans les sections suivantes nous décrirons les fonctions de poids résultantes dans le contexte d'expériences de DS.

2.4 Classifieurs pour la DS supervisée

Comme nous utilisons les même systèmes pour les tâches en anglais et en japonais, nous les expliquerons avant d'entrer dans les détails des tâches. Nous avons décidé d'utiliser des classifieurs NB pour les raisons suivantes.

Premièrement, malgré que des classifieurs SVM ont été montrés supérieurs aux NB dans le passé [24], l'intégration de plusieurs types d'attributs, comme les attributs de modèles de sujets de NUS-ML (section 2.2.2), n'est pas chose facile. Dans notre cas, l'utilisation de comptes réels était problématique car ils ont l'effet de rendre les données plus éparses. Comme un lissage ne peut être appliqué aux classifieurs par hyperplans comme SVM, nous avons utilisé NB.

Deuxièmement, les classifieurs NB peuvent être adaptés aux méthodes non-supervisées. Pour des tâches à large échelle, on ne peut s'attendre à ce qu'il existe des données étiquetées, ou même des inventaires de sens valides. Même des efforts tels que SemCor [32], qui contiennent les deux, seraient confrontés à des domaines différents, et souffriraient d'une couverture insuffisante lors d'applications à grande échelle comme la RI. Comme les classifieurs NB assignent des probabilités aux échantillons à classifier, ils sont compatibles avec les méthodes non-supervisées. Par exemple, une approche non-supervisée très simple serait d'utiliser l'algorithme *k-moyennes*, en commençant avec k classes assignées aléatoirement. À chaque itération, on pourrait calculer les probabilités des classes pour chaque échantillon, ou bien garder la plus probable pour une classifi-

Algorithm 2.1 ApprendPoids($\mathcal{C}, \eta, \epsilon$)

$\omega \leftarrow 1^k$
repeat
  $T \leftarrow$ {Prend les fenêtres du prochain mot}
  $(A, B) \leftarrow$ PartitionAléatoire(T)
  for W in A, B **do**
    $P_{ML,W} \leftarrow$ ConstruitML(W,ω)
    $\mu_W \leftarrow$ CalculePseudoCompte(W,$\mathcal{C}$)
    $P_{Dir,W} \leftarrow$ ConstruitDir($P_{ML,W}, \mu_W, \mathcal{C}$)
  end for
  $grad \leftarrow \nabla H(P_{ML,A}, P_{Dir,B}) +$
    $\nabla H(P_{ML,B}, P_{Dir,A})$
  $\omega \leftarrow \omega - \eta \frac{grad}{\|grad\|}$
until {critère d'arrêt atteint}
return $\omega / \max\{\omega_i\}$

cation stricte, ou encore utiliser les probabilités pour un algorithme EM. Si le sens des mots est lié à leur contexte, on devrait arriver à produire des groupes sémantiquement corrects.

Finalement, en travaillant avec des ensembles de données de plus en plus grands, la performance du procédé de classification devient un facteur important. Les classifieurs NB sont très attrayants sur ce point, comme ils fonctionnent avec un simple ensemble de comptes éparses. Les autres approches par hyperplans comme SVM sont plus lourdes car elles conservent un poids pour chaque attribut (plusieurs selon l'approche de gestion des classes multiples). Les méthodes les plus difficilement adaptables sont les méthodes dites "à mémoire", car elles conservent les échantillons individuels, et calculent des similarités avec tous les échantillons étiquetés, ce qui les rend trop lourdes pour des usages à grande échelle.

2.4.1 Attributs utilisés

Comme expliqué en Section 2.2.2, l'usage d'outils comme les étiqueteurs lexicaux et analyseurs syntaxiques, et des ressources externes comme des dictionnaires, thésaurus et ontologies sont prohibitifs pour des applications à grande échelle à cause de considérations de performance et couverture. Nous utiliserons donc des attributs alternatifs : les

formes du mot cible, le modèle de contexte et les mots outils locaux.

Formes du mot cible Les formes variées du mot cible ont des relations avec la distribution des sens. Certaines formes ne sont pas ambiguës, certaines sont liées avec le type lexical, lui-même lié au sens du mot. Nous l'utilisons dans un modèle de langue séparé car si on le traitait avec une troncature ou une lemmatisation comme les mots de contexte, on perdrait les détails de sa forme. Il forme une variable multinomiale différente des autres mots, et nécessite ainsi un lissage différent. Pour le lisser, on peut utiliser des a priori Dirichlet comme une distribution uniforme, ou un sous-ensemble des probabilités de collection contenant les formes possibles du mot cible.

Dans nos classifieurs NB, le mot cible de l'instance w avec la classe de sens S ont la probabilité suivante :

$$P_{Cible}(w|S) = \frac{|cible(w) \in S| + \mu_{Cible} P(cible(w)|aprioriUtilisé)}{|S| + \mu_{Cible}}$$

Mots outils locaux Les mots outils (ex : *le*, *et*, *à*...) n'ont pas de sens en tant que tels, mais ont des relations syntaxiques avec les mots qui les voisinent. Ces relations indiquent des composantes du type lexical, et aussi du rôle syntaxique du mot dans la phrase. Ainsi, comme on peut dire que les mots de contenu sont des attributs sémantiques, les mots outils seraient à la fois des attributs lexicaux et syntaxiques. Toutefois, comme la quasi-totalité de leurs instances ne sont pas liées au mot cible, ils ne sont pas utiles dans les modèles de contexte. On peut toutefois s'attendre à ce que ceux qui sont directement voisins du mot cible soient liés à celui-ci. De plus, il est important de remarquer que les mots outils ont de différentes implications selon s'ils occurrent avant ou après le mot. Par exemple, le fait que "un" soit présent avant "vers" dans "un vers" indique que vers est un nom. S'il était après, comme dans "vers un", ce serait alors probablement une préposition. Nous avons donc avantage à faire la différence entre un mot-outil étant avant ou après le mot cible. Celui-ci nous informe du type lexical, mais ne se limite pas à cette fonction, car il implique le rôle syntaxique dans la phrase. Dans l'exemple précédent,

on comprend donc que “un vers” désigne probablement le sens “assemblage de mots”, et “vers un” désigne le sens “direction”. Dans notre approche, nous utilisons les mots outils consécutifs qui occurrent directement avant et après le mot cible, arrêtant de les ajouter lorsqu'on tombe sur un mot de contenu. Par exemple, pour l'anglais : “... put it *in the* bank *for* safety *in* ...” (mots outils en italique, mot cible souligné) résulte en les attributs {in-avant, the-avant, for-après}. Nous ne considérons pas la distance des mots outils comme étant pertinente, et ceux-ci sont considérés indépendants pour simplifier le tout. Comme la présence d'un mot-outil n'implique pas nécessairement l'absence des autres, nous considérons chaque paire mot-outil/direction comme une variable Bernouilli avec une valeur de 1 quand le mot est présent, et 0 sinon. On va ainsi utiliser l'absence du mot dans le calcul du score. Comme ces attributs sont très fréquents, ils devraient être bien modélisés. On peut donc leur appliquer un simple lissage de Laplace, consistant à ajouter une observation artificelle à chaque attribut observable. La probabilité qu'un des attributs soit vrai ou faux est donc :

$$P(x = 1|S) = \frac{|w' \in S \ : \ x \in w'| + 1}{|S| + 2}$$

$$P(x = 0|S) = (1 - p(x = 1|S))$$

Le score de ces attributs pour une instance w étant donné le sens S est ainsi :

$$P_{Outils}(w|S) = \prod_{x \in attributsOutils} \begin{cases} P(x = 1|S) & \text{si } x \in w \\ P(x = 0|S) & \text{sinon} \end{cases}$$

Attributs de modèle de contexte Ces attributs sont tels que définis dans la section 2.3.1.

Nous définissons leur probabilité pour l'échantillon w et la classe S comme étant :

$$P_{Con}(w|S) = \prod_{x \in contexte(w)} P_{Con}(x|S)^{\omega_{dist(x)}}$$

Dans l'équation, $dist(x)$ est la distance de fenêtre d'un mot de contexte particulier.

La fonction de poids est aussi utilisée dans le score étant donné que l'échantillon à classifier suit la même distribution que le reste des données. Il faut ici noter que, la probabilité convertie en log-prob de ces attributs peut être vue comme la log-vraisemblance du modèle de contexte ML fait uniquement de l'échantillon à classifier, contre le modèle de contexte lissé des données d'entraînement. Ceci est identique à la mesure de similarité que nous avons utilisée pour l'apprentissage des poids.

2.4.2 Combiner les attributs

Le score final d'un échantillon w pour la classe S est :

$$Score(w, S) = P(S)P_{Cible}(w|S)^{\lambda_{Cible}}P_{Outil}(w|S)^{\lambda_{Outil}}P_{Con}(w|S)^{\lambda_{Con}}$$

avec $P(S)$ étant l'a priori de sens, et P_{Cible}, P_{Outil}, P_{Con} étant tels que définis ci-haut. Les paramètres λ_{Cible}, λ_{Outil} et λ_{Con} sont utilisés pour contrôler l'impact des attributs. Ceux-ci peuvent être fixés à 1 quand le lissage est défini manuellement, car le lissage sera choisi pour maximiser les résultats, ce qui a pour effet de contrôler l'impact des attributs.

Pour des classifieurs NB, il est pratique courante d'appliquer plus ou moins de lissage sur des attributs pour contrôler leur impact. L'usage courant de lissage Laplace dans les classifieurs NB a l'effet de sur-lisser les attributs afin de contourner les problèmes de dépendance entre les attributs. Ceci n'est pas idéal, par exemple, lorsque deux attributs sont très dépendants (ex : deux fois le même attribut). De tels attributs devraient avoir leur impact (log-prob) réduit (à la moitié si identiques) pour compenser cette dépendance, mais il n'est pas clair comment ceci peut être fait en contrôlant le lissage. En guise de remplacement, nous proposons d'utiliser un lissage Dirichlet basé sur les méthodes d'estimation décrites dans la section 2.3.2. Nos expériences montrent que quand assez d'occurrences de mots sont dans le modèle, l'estimation devient assez fiable pour être utilisée. Comme la valeur estimée fixe les probabilités résultantes, pour contrôler l'impact nous multiplions les log-probs avec les paramètres λ_{Cible}, λ_{Outil} and λ_{Con} choisis

par validation croisée.

2.5 Expériences sur Semeval-2007 English Lexical Sample

Semeval (précédemment nommé Senseval) est un atelier d'évaluation sémantique tenu à tous les trois ans. Nous ferons des expériences en utilisant les données de la tâche de 2007 nommée *English Lexical Sample* (ELS) [42]. La tâche consiste à construire des classifieurs de DS maximisant la précision de classifications sur des données de test retirées. Les détails des données sont les suivants :

- Les échantillons de mots sont extraits du Wall Street Journal (WSJ) Treebank de 1 million de mots.
- Il y a 100 mots (65 verbes et 35 noms), choisis pour leur polysémie et le nombre d'instances étiquetées.
- Les données originales sont annotées utilisant des inventaires de sens de WordNet, mais comme ceux-ci sont trop fins pour obtenir un bon accord des annotateurs, les organisateurs ont utilisé les groupes de sens de WordNet définis dans le projet OntoNotes [18]. Ceci résulte en un inventaire plutôt grossier, ayant en moyenne 3.6 sens par mot.
- Le nombre d'instances étiquetées est affiché dans le tableau 2.2. L'accord moyen des annotateurs serait au delà de 90%.

begin-v	**attempt-v**
1 start, have an initial point	1 try to do something
2 initiate an undertaking	2 the act of surmounting, climbing, descending
3 make a locution, speak	
condition-n	**future-n**
1 Event – state	1 time to come
2 stipulation	2 Linguistic – verb tense
3 consideration	3 commodities
4 Science – not the control	4 personal time to come

Tableau 2.1 – Exemples d'inventaires de sens de OntoNotes

	Entraînement	Test	Total
Verbes	8988	2292	11280
Noms	13293	2559	15852
Total	22281	4851	

Tableau 2.2 – Nombre d'instances dans les données Semeval-2007 ELS

2.5.1 Systèmes de base

Nous utiliserons plusieurs systèmes de base afin de comparer notre système. Un système *Sens le Plus Fréquent* (SPF) assigne toujours le sens le plus probable trouvé dans les données d'entraînement. Il indique le minimum théorique pour la précision de classification. Si le score d'un système SPF change beaucoup entre les données d'entraînement et de test, cela peut indiquer que les deux ensembles sont de nature différente.

Bien que nos classifieurs sont des NB, pour fin de comparaison, nous avons aussi utilisé un système SVM utilisant les mêmes attributs que ceux décrits ci-haut. Nous avons utilisé les librairies SVM-light avec des noyaux linéaires, et le script de recherche en grille pour sélectionner automatiquement les paramètres de régularisation. De nos essais, nous avons vu que les classifieurs SVM ont beaucoup de difficultés à utiliser des données de compte de mots. Nous avons essayé plusieurs formes pour celles-ci comme de normaliser les comptes d'un échantillon, utiliser une échelle logarithmique, etc. Mais en fin de compte, peu importe leur forme, beaucoup de mots sont observés si peu souvent que l'envergure de leurs attributs est mal estimée. Les meilleurs résultats sont donc obtenus avec des attributs Booléens indiquant la présence d'un mot dans une fenêtre fixe de taille 3.

Pour nos systèmes NB, nous utilisons plusieurs fonctions de poids utilisées dans les études précédentes :

Uniforme : $\omega_i = 1$ si $1 \leq i \leq \delta$, 0 sinon, où δ est la taille de la fenêtre.

Linéaire : $\omega_i = \max\{0, 1 - (i-1)\delta\}$, où δ contrôle le taux de dégénérescence.

Gaussienne-0 : $\omega_i = e^{-\delta(i^2-1)}$, est une Gaussienne où le centre de la courbe est à la distance 0. δ contrôle la variance.

Gaussienne-1 : $\omega_i = e^{-\delta(i-1)^2}$, est une Gaussienne où le centre de la courbe est à la

distance 1. δ contrôle la variance.

Exponentielle : $\omega_i = e^{-(i-1)\delta}$, où δ est le paramètre de l'exponentielle

Nous nommons la courbe issue de notre méthode *Apprise* :

Apprise : ω_i est le poids appris comme démontré précédemment.

Voyant les formes résultantes de l'apprentissage, nous proposons aussi l'ajout d'une fonction suivant une *loi de puissance négative*, qui est définie par :

Puissance négative : $\omega_i = i^{-\delta}$, où δ contrôle le taux de dégénérescence.

Cette fonction a été proposée implicitement dans [37] en tant qu'une division des fréquences d'occurrences par la racine carrée de leur distance moyenne. Les paramètres de ces fonctions ont été sélectionnés par des essais exhaustifs, maximisant la précision de classification sur une validation croisée retire-un des données d'entraînement.

2.5.2 Calculer des poids

Les données fournies dans les tâches Semeval n'étaient pas suffisantes pour calculer des poids pour notre système. Nous avons donc utilisé un corpus externe formé de la collection *Associated Press* 88-90 de la conférence TREC (CD 1 & 2), contenant 242 918 documents. Chaque mot dans cette collection sera considéré comme un mot cible pour fin d'estimation des poids. La collection a été traitée avec le troncateur Porter, et nous avons accumulé des fenêtres complètes de taille 100 pour chaque racine. Nous avons considéré que la taille de 100 était suffisante, comparativement aux fenêtres de taille 3 habituelles. Pour rendre l'opération plus efficace, nous avons limité nos modèles à 1000 instances pour chaque mot cible. Autrement dit, un mot cible ayant plus de 1000 instances dans la collection va avoir ses instances tronquées à 1000. Nous estimons que 1000 instances sont suffisantes pour cette estimation. Nous utilisons un anti-dictionnaire formé des 10 mots les plus fréquents. Ces mots sont remplacés par un symbole spécial pour conserver les distances, fusionnant les instances consécutives en une seule qui n'est pas comptée (ex : "of the" devient "#stop") . Au total, on retrouve 32 650 mots cibles contenant un total de 5 870 604 fenêtres.

Dans la figure 2.1 nous pouvons voir la courbe apprise et les courbes des meilleurs paramètres des systèmes de bases. Toutes les courbes ont plus ou moins la même pente dégénérescente au début, mais la courbe apprise diffère par la présence d'une "longue queue".

La loi de puissance négative résulte en une très bonne approximation de la courbe apprise. C'est une fonction qui n'a jamais été proposée dans les études précédentes. Comme nous le verrons, celle-ci a aussi de très bons résultats.

2.5.3 Résultats

Les résultats des systèmes sont listés dans le tableau 2.3. À la droite, nous avons les systèmes proposés originalement lors de la tenue de la tâche en 2007. Nous décrirons maintenant les paramètres des systèmes. Les mots de contextes ont été traités de la même manière que pour le corpus externe. Le mot cible est utilisé sans troncature, mais on retire les majuscules. Les systèmes NB utilisent une concaténation de la collection *AP* et des données Semeval pour modèle de langue de collection, car les données Semeval ne contiennent que les courts passages contenant les mots cibles. Les mots reliés à ceux-ci ont donc une probabilité plus élevée qu'à l'habitude. L'a priori des classes utilise un décompte absolu de 0.5 pour chaque compte de classe. Les paramètres des systèmes sont listés dans le tableau 2.4. Ceux-ci ont été choisis en maximisant la précision retire-un sur l'ensemble des données.

Les résultats de ces expériences confirment notre intuition que les fonctions dégénérescentes sont bénéfiques pour la DS. Les précisions obtenues se comparent bien à l'état de l'art courant. Toutes sauf la meilleure soumission des 14 systèmes originaux sont surclassées [42]. Un test de rééchantillonage aléatoire de chaque réponse (une direction, sans diviser les égalités, 2 millions de passes) entre *NB puissance négative* et NUS-ML donne une p-valeur de 0.1818. Nous pouvons donc conclure que notre système n'est pas significativement différent de l'état de l'art, même s'il est beaucoup plus léger et adaptable, car il n'utilise pas d'attributs comme les collocations locales, attributs positionnels, types lexicaux et attributs de sujets.

Nous remarquons aussi que la loi de puissance négative fonctionne extrêmement

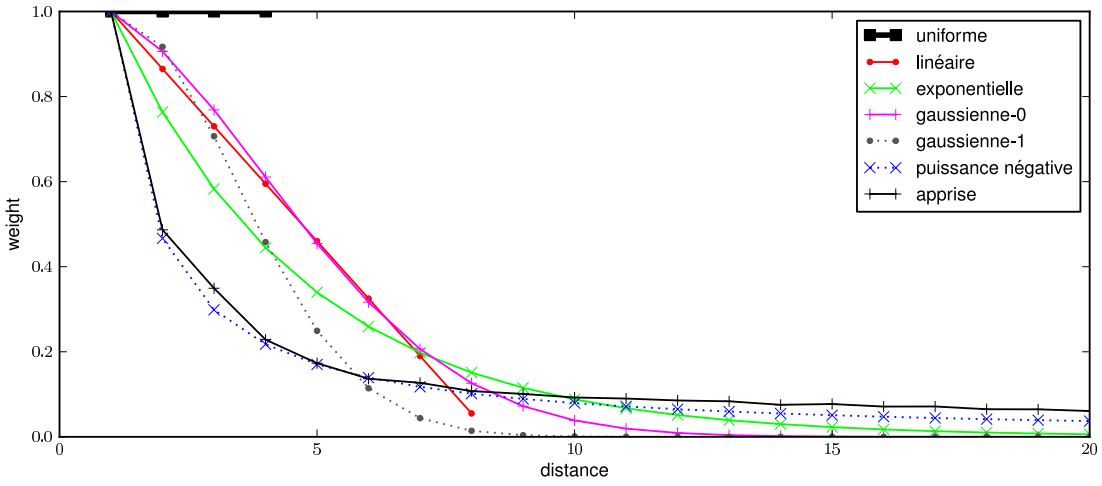


Figure 2.1 – Fonctions de poids en anglais pour les systèmes Semeval-2007 ELS

bien, même un peu mieux que notre fonction, même si la différence n'est pas significative. On pouvait s'attendre à ce résultat étant donné que les courbes sont similaires, et que nous avons utilisé un corpus externe potentiellement différent des données Semeval. Les résultats montrent clairement que la loi de puissance négative est la bonne fonction à prendre si nous avions à en spécifier une manuellement.

2.6 Expériences sur Semeval-2010 Japanese WSD

Nous avons testé la même approche pour l'édition 2010 de Semeval, dans le cadre de la tâche Semeval-2010 *Japanese WSD* (JWSD) [38]. C'est aussi une tâche du type échantillon lexical, très similaire au ELS. Ses détails sont les suivants :

- Les échantillons de mots sont extraits du *Balanced Corpus of Contemporary Written Japanese* (BCCWJ) [29]. Ce corpus, tout comme le *Brown corpus* [23], contient plusieurs types de documents tels que livres, magazines, journaux, "livres blancs", retranscriptions du parlement Japonais, sites Web (blogs et sites de Q&R), etc.
  Les données de la tâche contiennent un sous-ensemble de ceux-ci : "livres blancs" et journaux sont dans les données d'entraînement, les données de test contiennent en plus des sites Web (Q&R).
- Il y a 50 mots (22 noms, 23 verbes, 5 adjectifs), choisis selon l'entropie de leur

Système	Val-Crois. (%)	Ens. Test (%)
SPF	78.66	77.76
NB uniforme	87.47	86.17
SVM	87.25	86.66
NB gauss.-1	88.13	86.97
NB linéaire	88.26	87.03
NB gauss.-0	88.45	87.07
NB exp.	88.94	87.57
NB apprise	89.65	88.25
NB puiss. nég.	**89.69**	**88.31**

Systèmes originaux 2007 (14)

Système	Type	Test set (%)
Tor	Non-supervisé	52.09
Duluth	Non-supervisé	53.78
USYD	SVM	74.27
MFS	ligne de base	74.27
ITC	SVM	79.57
UBC-ZAS	SVD+kPPV	79.92
VUTBR	NB	80.25
OE	NB, SVM	83.81
KU	Semi-supervisé	85.05
USP-IBM-1	ILP	85.15
USP-IBM-2	SVM	85.65
I2R	Supervisé	86.42
UBC-ALM	SVD+kPPV	86.93
NUS-ML	NB	**88.68**

Tableau 2.3 – Précision de DS sur les données Semeval-2007 ELS

distribution de sens. Ceci est la moitié du nombre de mots utilisés dans ELS, mais la proportion des types lexicaux semble plus balancée car on donne moins d'importance aux verbes.

– Pour l'inventaire de sens, on utilise le niveau intermédiaire du dictionnaire Iwanami Kokugo. Ce dictionnaire a des définitions sous forme d'arbres, ayant un niveau sous le niveau intermédiaire. Le niveau le plus bas contient des informations précises sur un des sens, et serait trop fin pour l'usage présent. Les organisa-

Système	poids de fenêtre	P_{Cible}	P_{Outil}	P_{Con}
NB uniforme	$\delta = 4$	Laplace 0.65	$\lambda_{Outil} = 1$	$\lambda_{Con} = 0.7$
SVM	Booléen 3	–	–	–
NB linéaire	$\delta = 0.135$	Laplace 0.85	$\lambda_{Outil} = 1.1$	$\lambda_{Con} = 0.985$
NB gauss.-0	$\delta = 1/2.(3.9)^2$	Laplace 1.7	$\lambda_{Outil} = 1$	$\lambda_{Con} = 0.9$
NB gauss.-1	$\delta = 1/2.(2.4)^2$	Laplace 1	$\lambda_{Outil} = 1$	$\lambda_{Con} = 0.8$
NB exp.	$\delta = 0.27$	Laplace 0.7	$\lambda_{Outil} = 1.1$	$\lambda_{Con} = 1.15$
Nb apprise	–	Laplace 1.35	$\lambda_{Outil} = 1$	$\lambda_{Con} = 1.15$
NB puiss. nég.	$\delta = 1.1$	Laplace 1.4	$\lambda_{Stop} = 1$	$\lambda_{Con} = 1.25$

Tableau 2.4 – Paramètres des systèmes pour Semeval-2007 ELS

teurs soutiennent que cet inventaire a une granularité moyenne, mais dans les faits il y a en moyenne 3.58 sens par mots, ce qui est pratiquement identique avec les données de ELS.

– Dans le cas où aucun sens de l'inventaire ne s'applique à une instance, les annotateurs ont appliqué une étiquette "hors-dictionnaire", qui est ensuite vue comme une classe de sens ordinaire.
– Il y a 50 instances d'entraînement et de test pour chaque mot. Ceci est différent de la tâche ELS sur deux points : (1) pour ELS le nombre d'instances variait en fonction de la fréquence des mots dans le corpus, (2) les données ELS avaient environ quatre fois plus d'instances d'entraînement (en moyenne 222 par mot cible). L'accord des annotateurs sur JWSD avait une métrique Kappa de 0.678.

Comme en japonais il n'y a pas de séparation des mots, les organisateurs ont traité les documents avec l'analyseur morphologique ChaSen et ont corrigé les erreurs manuellement. Ceci fournit aux participants les segmentations, types lexicaux, lectures de mots, et formes de bases pour les mots conjugués. La proportion des instances en fonction du genre de document est listée dans le tableau 2.5. Comme nous pouvons le voir, les domaines sont différents entre les données d'entraînement et de test, la différence principale étant l'ajout de pages Web Q&R dans les données de test. Dans les expériences

genre	Entraînement	Test	Total
livres blancs	345	207	552
Livres	1537	884	2421
Journaux	627	634	1261
Pages Web Q&R	0	775	775

Tableau 2.5 – Nombre d'instances étiquetées par genre de document dans les données de Semeval-2010 JWSD

sur la DS japonaise, nous comparons notre méthode avec les mêmes systèmes de base que pour ELS (Section 2.5.1).

2.6.1 Traitement de texte japonais

Nous avons tenté d'effectuer un traitement similaire en japonais à ce que nous avons fait en anglais, malgré que les langues soient très différentes. Nous ne pouvons pas appliquer une troncature en japonais, donc nous avons utilisé la lemmatisation. Comme nous avions des données d'analyse lexicale, nous avons converti les mots à leur lecture ou forme de base quand elle était présente, et nous avons concaténé les *kanjis* (caractères chinois) présents dans la forme de surface. Ceci est motivé par les aspects suivants de la langue japonaise :

- Le japonais comporte beaucoup d'homophonie, donc utiliser uniquement les lectures de mots donnerait une trop forte amalgamation. Les *kanjis* utilisés sont très liés aux sens des mots, même pour des variations d'un même mot de base. Par exemple, certains verbes ont des différents caractères afin de spécialiser leur sens. Un bon exemple de ceci est le verbe "rencontrer" *au*, qui peut prendre des sens différents selon qu'il signifie rencontrer quelqu'un directement ou par hasard, "rencontrer" un accident ou une mauvaise expérience, ou même "être compatible". Dans certains cas, l'écriture intentionnelle d'un mot en Hiragana ou Katakana plutôt qu'en kanji peut donner un sens plus général ou abstrait au mot.
- En japonais, il est possible qu'un mot ait des orthographes alternatifs. Par exemple, certains mots peuvent être écrits ou bien avec uniquement des *kanji*s, ou encore avec les *kanjis* et des "caractères accompagnateurs" (*okurigana*). Ceux-ci sont des compléments phonétiques ayant pour objectif de désambiguïser la lecture d'un mot. Un exemple de ceci se trouve dans le mot "ventes" *uriage* qui peut être écrit u-ri-a-ge ou u-ri-a-ge (le soulignement indique que la lecture fait partie du caractère). Il est donc utile de faire abstraction de ces différences.
- La présence de lectures différentes pour la même forme de surface peut aussi altérer le sens d'un mot. Par exemple, en japonais, tout comme en anglais ou en français, le mot "marché" a deux sens : un est "un marché", l'endroit où on peut acheter des choses, l'autre est "le Marché", le concept de commerce. Bien que les deux sens s'écrivent de manière identique dans les trois langues, en japonais, la pronon-

ciation du mot est différente selon le sens : la lecture japonaise *ichiba* indique le concept, et la lecture chinoise *shijô* indique l'endroit.

Ainsi, concaténer les *kanjis* séparément de la lecture du mot a pour objectif de conserver certaines distinctions dans certaines formes.

En remplacement d'un anti-dictionnaire, nous utilisons les données lexicales pour retirer toutes les conjugaisons (*setsuzoku-to*, *jodô-shi*, etc.), les "particules" (tous les *jo-shi*), les symboles (blancs, *kigô*, etc.), et les chiffres. Notez qu'un traitement par fréquence aurait pu être plus efficace, mais il n'est pas clair si nous devrions considérer les conjugaisons comme faisant partie du mot ou non, donc nous avons simplement tout retiré. Pour les attributs de mots outils, nous avons utilisé les 10 formes de surface les plus fréquentes, qui sont toutes des particules (*jo-shi*).

2.6.2 Calculer des poids

Comme pour la tâche ELS, les données fournies n'étaient pas suffisantes pour calculer des poids. Nous avons donc utilisé un corpus externe composé d'un an du journal Mainichi (2008), provenant des collections NTCIR-8. Nous avons utilisé Chasen+Unidic pour recréer un format semblable aux données de la tâche. Ceci n'est pas idéal, parce que l'analyseur et ses paramètres pourraient être différents, le résultat automatique n'est pas corrigé comme les données de tâche. Les poids résultants, ainsi que ceux utilisés dans les systèmes sont affichés dans la figure 2.2.

Comme nous pouvons le voir, les poids baissent beaucoup plus vite que dans leur équivalent anglais. La différence peut être expliquée partiellement par un traitement différent du texte. On peut aussi trouver des explications linguistiques à ceci. La structure de la langue japonaise est généralement vue comme étant du type sujet-complément-verbe. L'anglais et le français sont sujet-verbe-complément. L'ordre des éléments de la phrase est aussi plus flexible en japonais ; les sujets sont souvent même omis. Ces facteurs résultent en un contexte local pauvre et beaucoup de dépendances non-locales. La relation d'un mot avec son contexte local est donc réduite. Un exemple concret de ceci se trouve avec les verbes : en anglais et en français, un verbe est souvent entre son sujet et son complément et aura ainsi un poids élevé. En japonais, le verbe est habituellement

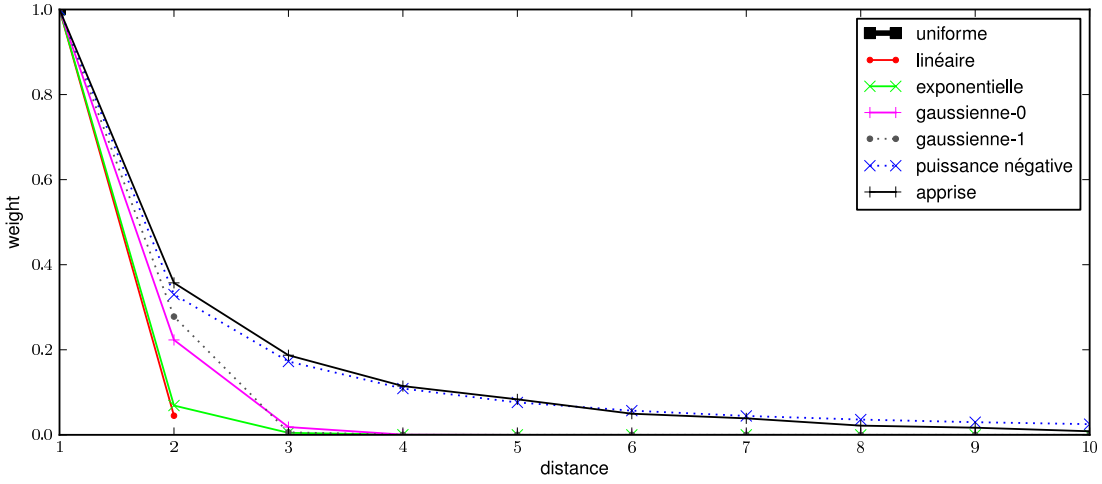


Figure 2.2 – Fonctions de poids en japonais pour les systèmes Semeval-2010 JWSD

à la toute fin de la phrase, et le sujet au tout début. Les deux sont donc très loins et il est ainsi difficile de détecter leur dépendance et leur donner un poids élevé.

2.6.3 Résultats

Les résultats de la tâche sont listés dans le tableau 2.6. À la droite nous avons les résultats formels de la tâche. Nos systèmes sont RALI-1 et RALI-2 [6, 7]. La différence entre les deux sytèmes est que RALI-1 utilise un pseudo-compte constant, et RALI-2 utilise le lissage Dirichlet automatique. Comme nous pouvons le voir, le pseudo-compte estimé a mieux fonctionné que celui déterminé manuellement. Ceci confirme la supériorité du lissage automatique pour suivre les données. La différence entre les scores de gauche et ceux de droite est l'ajout des attributs de mot-outils et un traitement plus simple des mots cibles qui sont mis dans le modèle de contexte. Ceci montre que les occurrences des formes du mot cible sont particulières et sont mieux modélisées séparément. Le système *Baseline* créé par les organisateurs était un système SVM utilisant les attributs suivants :

– Type lexical, et information morphologique détaillée des mots dans une fenêtre de taille 2, incluant le mot cible.
– Attributs sac-de-mots dans une fenêtre de taille 2.

Système	Cross-Val (%)	Ens. test (%)
SPF	75.23	68.96
SVM	83.13	77.44
NB uniforme	82.61	76.32
NB gauss.-1	82.89	76.60
NB linéaire	82.65	76.68
NB gauss.-0	82.85	76.88
NB exp.	82.97	76.88
NB puiss. nég.	83.41	77.28
NB apprise	**83.61**	**77.64**

Systèmes originaux (10)

Système	Ens. test (%)
MSS-2	63.84
MSS-1	64.04
MSS-3	66.04
HIT-1	66.12
JAIST-1	68.64
JAIST-3	72.08
JAIST-2	74.76
Baseline	75.28
RALI-1	75.92
RALI-2	**76.36**

Tableau 2.6 – Précision de DS sur les données Semeval-2010 JWSD

– Relations syntaxiques de base : pour un nom, aller chercher le verbe associé, pour un verbe son sujet.

– Attributs du *Bunrui Goi Hyou* pour les mots de contenu voisins du mot cible.

Comme on peut le constater par les résultats, ce système est très fort, et n'est surclassé que par les nôtres. On peut donc le considérer comme un système participant. Notre système de base SVM a toutefois fonctionné mieux que nos évaluations lors de la participation, grâce à l'ajout des attributs de mots outils. Auparavant, en les retirant nous avons enregistré une précision – la proportion des instances qui sont classifiées correctement – de 74.92, moins que le système *baseline*. L'amélioration est donc beaucoup plus grande avec les classifieurs SVM que les NB. La raison pour ceci est que le SVM peut donner des poids différents aux mots outils individuellement.

En ce qui concerne les autres soumissions originales, les autres équipes ont mis des efforts dans la direction des particularités de ces données, comme la détection des nouveaux sens, et l'utilisation des genres de documents. Il n'est pas surprenant que ces tentatives n'aient pas donné des résultats positifs, étant donné qu'il n'y avait pas de pages Web Q&R dans les données d'entraînement malgré que c'était un des genres les plus fréquents dans les données de test. La détection de nouveau sens est aussi problématique, étant donné que sans utiliser le dictionnaire, il est impossible de différencier les instances de nouveaux sens, des instances de sens du dictionnaire qui ne sont pas observés.

La classe *nouveau sens* était aussi très rare dans les données d'entraînement, et pouvait contenir plusieurs "nouveaux sens" différents.

Tout comme dans les expériences en anglais, on constate que la fonction de poids apprise fonctionne mieux que les méthodes de base. Toutefois, l'amélioration est beaucoup moindre que ce qu'on observe dans le ELS. Ceci peut être expliqué par la baisse rapide des poids pour le japonais. En effet, la quasi-totalité des poids est mise sur la distance 1 ; les mots plus loin n'ont pas un grand impact. Ceci s'applique aussi aux systèmes utilisant des fenêtres fixes. Comme pour les données ELS, nous avons fait des essais exhaustifs pour trouver la taille optimale. Quand pour ELS des fenêtres de taille 3 ou 4 étaient optimales, dans le cas du japonais la performance des systèmes *SVM* et *NB uniforme* baisse de manière consistance quand on augmente la taille de la fenêtre. Le résultats optimaux sont obtenus en utilisant des fenêtres de taille 1.

On peut observer que, tout comme l'ELS, la loi de puissance négative a une forme très similaire à la fonction de poids apprise, et que leurs performances sont aussi similaires. Ceci montre encore que la loi de puissance négative modèle bien l'impact des mots de contexte sur le mot cible.

Les configurations des systèmes sont listées dans le tableau 2.7. Pour ces données, le lissage Dirichlet automatique n'a eu des meilleurs résultats qu'avec *NB appris* et *NB puissance négative*. Une explication pour ceci serait le nombre d'instances étiquetées réduites par rapport au ELS. Comme les autres fonctions ont des poids très réduits pour les mots de contexte, ceci résulterait en une estimation moins juste du pseudo-compte.

2.7 Discussion

Nos expériences montrent que la DS japonaise est plus difficile qu'en anglais, probablement parce qu'il y a plus de dépendances non-locales. Afin de capturer de telles dépendances sans utiliser des analyseurs syntaxiques, nous allons investiguer l'incorporation de patrons simples dans notre méthode d'apprentissage de poids. Comme celle-ci consiste à la base de grouper des occurrences de mots dans des cases, nous n'avons pas à nous limiter à une simple distance de fenêtre, et pouvons définir des cases selon des

Système	poids de fenêtre	P_{Cible}	P_{Outil}	P_{Con}
NB uniforme	$\delta = 1$	Dirichlet 4	$\lambda_{Outil} = 1$	Dirichlet 90
SVM	Booléen 1	–	–	–
NB linéaire	$\delta = 0.955$	Dirichlet 1.3	$\lambda_{Outil} = 0.525$	Dirichlet 70
NB gauss.-0	$\delta = {}^{1}\!/_{2.(1.0)^2}$	Dirichlet 2	$\lambda_{Outil} = 0.5$	Dirichlet 90
NB gauss.-1	$\delta = {}^{1}\!/_{2.(0.625)^2}$	Dirichlet 2.1	$\lambda_{Outil} = 0.5$	Dirichlet 100
NB exp.	$\delta = 2.675$	Dirichlet 0.105	$\lambda_{Outil} = 0.55$	Dirichlet 70
NB apprise	–	Dirichlet 1.4	$\lambda_{Outil} = 0.5$	$\lambda_{Con} = 0.65$
NB puiss. nég.	$\delta = 1.6$	Dirichlet 1.1	$\lambda_{Outil} = 0.6$	$\lambda_{Con} = 0.5$

Tableau 2.7 – Paramètres des systèmes pour Semeval 2010 JWSD

critères plus spécialisés.

Aussi, comme nous avons maintenant une idée de la nature de la fonction de poids optimale, nous pouvons investiguer un des problèmes de la distance de fenêtre : le fait qu'elle augmente uniformément sans prendre en compte le mot rencontré. Nous pensons que certains mots, comme les mots outils, devraient introduire moins de distance que les autres. Au contraire, nous pouvons nous attendre à ce que des formes telles que des virgules, les points, les parenthèses et paragraphes devraient introduire une plus grande distance que les mots réguliers. Nous pourrions donc utiliser un *score de congruence* pour un mot, un indicateur montrant en moyenne de combien ce qui arrive avant le mot diffère de ce qui vient après. Une difficulté liée à calculer une telle mesure est qu'on doit comparer le contexte avant et après, ce qui causerait un grand problème de données éparses. Pour pallier à ce problème, nous pensons pouvoir utiliser des techniques de réduction de dimensionalité comme LDA.

La fonction de poids et les modèle de contexte résultant de notre méthode pourraient avoir plusieurs applications. Les modèles de contexte peuvent être vus comme étant des distributions de probabilités conditionnelles d'un mot étant donné une occurrence du mot cible. De telles probabilités sont souvent utilisées dans des méthodes basées sur les cooccurrences, comme l'expansion de requête en RI.

Comme nous l'avons vu avec la tâche Semeval-2010 DS translinguistique, la traduction d'un mot est très liée à la DS. Les modèles de contexte que nous proposons pourraient donc être utilisés pour améliorer les systèmes de RI translinguistiques ou

la traduction automatique. Les techniques de traduction automatique statistique n'exploitent le contexte d'utilisation d'un mot que de manière très limitée. L'utilisation du contexte des mots pourrait donc amener une grande amélioration.

Finalement, les poids résultant de notre procédé peuvent être vus comme la force espérée de la relation entre deux mots dans un document, en fonction de leur distance. Des considérations de relations de mot dans les documents et requêtes sont des défis dans la recherche courante en RI. La fonction de poids pourrait être facilement intégrée dans un modèle de dépendance pour la RI. Nous planifions une telle intégration dans le futur.

2.8 Conclusion

Dans cette étude, nous avons présenté une méthode basée sur des arguments fondés pour assigner des poids à des mots de contexte dans des modèles sac-de-mots. Des études préalables en RI ont proposé l'usage de fonctions dégénérescentes pour modéliser la relation entre deux mots, mais dans tous les cas les fonctions sont définies manuellement, heuristiquement. Pour investiguer ce que serait la fonction de poids optimale, nous avons proposé une méthode non-supervisée pour calculer des poids pour les mots de contexte en fonction de leur distance au mot cible. L'idée générale était de trouver des poids qui coïncident bien avec les données, de sorte que des modèles de contexte faits pour le même mot mais avec des échantillons différents choisis aléatoirement soient similaires. C'est la première fois que ce principe général est utilisé à ces fins. Nos expériences en DS en anglais et japonais suggèrent que ce principe est valide.

L'observation que la fonction de poids optimale a la forme d'une loi de puissance négative est intéressante, et pourrait aider des investigations futures dans les vraies dépendances entre des mots rapprochés. Ceci sera investigué dans des travaux à venir.

Nous avons aussi proposé l'usage d'attributs alternatifs, en remplacement des collocations locales, et des données d'analyseur syntaxique et étiqueteur lexical. Nous avons trouvé que les formes du mot cible et mots outils voisins de ceux-ci sont très liés à la distribution de sens. Ces attributs, combinés avec les modèles de contexte, et des techniques

d'estimation, résultent en des systèmes de DS légers qui ont toutefois des résultats aussi bons que l'état de l'art.

De nos expériences en anglais et japonais, nous avons aussi montré que la fonction dégénérescente optimale est liée à la langue, bien qu'elle suive le même type de fonction. Pour des langues comme le japonais, où le style d'écriture varie beaucoup en fonction du type de document, cette fonction pourrait aussi dépendre du corpus. Notre approche a l'avantage de s'adapter à de telles différences.

Les méthodes proposées pourraient être adaptées à des tâches de TAL comme la RI et la Traduction Automatique. Ce sont des problèmes que nous investiguerons dans des recherches futures.

CHAPITRE 3

RECHERCHE D'INFORMATION

La Recherche d'Information (RI) est le domaine scientifique qui "s'intéresse à la structure l'analyse, l'organisation, l'entreposage, la recherche et l'accès à l'information" [46]. Ce domaine naît de l'automatisation des techniques de bibliothéconomie par l'utilisation d'ordinateurs. Dès 1945, Vannevar Bush décrivait dans son ouvrage philosophique *"As we may think"* le besoin d'améliorer nos méthodes scientifiques afin de gérer la masse d'information qui devient plus en plus grande. À ces fins, Bush envisage l'utilisation de l'ordinateur pour gérer l'accès à l'information, et ainsi propulser la science dans le futur, inventer un nouveau moyen de penser, littéralement.

Il faut comprendre qu'à cette époque, consulter des références bibliothécaires était très fastidieux. Il fallait alors passer par l'intermédiaire de bibliothécaires spécialistes, qui construisaient des index de documents manuellement. Ceux-ci devaient développer une taxonomie pour classifier les documents, et des stratégies de recherches pour arriver à trouver les documents. La Recherche d'Information visa tout d'abord à automatiser la plupart de ces opérations.

Depuis ce temps, beaucoup de chemin a été fait dans la compréhension du procédé de RI. L'avènement du Web en particulier, a donné une importance nouvelle à la RI. La RI devient de plus en plus variée et traite maintenant avec une gamme de documents multimédia. Toutefois, dans ce mémoire nous allons nous limiter à la recherche de documents textuels. Dans ce chapitre, nous allons faire une revue des concepts principaux liés à la RI et des techniques et modèles employés de nos jours.

3.1 Concepts principaux

La RI peut être décrite par quelques éléments fondamentaux. Tout d'abord, un utilisateur a un *besoin d'information* (BI) qu'il formule en une *requête* soumise au système de RI. Cette requête est le plus souvent sous forme textuelle, étant composée de quelques

mots clés. Le système de RI consulte ensuite l'*index* de sa *collection* de documents, et propose une liste ordonnée de résultats *pertinents*. La pertinence est un concept flou et subjectif. Bien qu'un humain peut, étant donné un BI, juger de la pertinence d'un document, cela reste difficile à formaliser et automatiser. De plus, un système travaillant avec la requête n'a pas une idée claire du BI. En effet, les requêtes sont des vagues approximations du BI, et une même requête pourrait avoir des objectifs différents selon le contexte. On laisse donc au système RI la tâche de définir et modéliser la notion de pertinence, et de nous offrir des résultats suivant le *principe de rang par probabilité* (*probability ranking principle*) [61], selon lequel le moyen optimal d'ordonner les résultats est de les mettre en ordre de probabilité de pertinence espérée.

3.2 Évaluation des systèmes

Bien que la formalisation de la notion de pertinence soit ouverte à la discussion, il est possible de s'en approcher en évaluant les résultats des systèmes. Le domaine de la RI bénéficie d'une forte tradition d'empirisme [44]. On évalue les systèmes par divers critères et mesures. Tout d'abord, on distingue la notion de *performance* (*efficien*cy), et l'*efficacité* (*effectiveness*). La performance représente l'efficacité du traitement en espace et temps ; elle est reliée à des mesures telles que le nombre de requêtes traitées par seconde. L'efficacité représente la capacité de trouver les documents pertinents.

3.2.1 Ensemble de tests

La performance d'un système est facilement évaluable, mais c'est plus difficile pour l'efficacité. Un moyen de comparer deux systèmes est d'utiliser un *ensemble de test*s. Les systèmes comparés doivent utiliser les mêmes données :

- Une *collection* de documents est préparée à l'avance.
- Des *requêtes* et une explication de leur besoin d'information sont définis.
- Des *jugements* de pertinence sont construits par des annotateurs humains pour chaque requête.

La difficulté la plus importante lors de la création de telles collections est de produire des jugements de pertinence de qualité. La majorité des ensembles de tests sont créés lors de “compétitions” telles que celles organisées par le *Text REtrieval Conference* (TREC). Dans ces tâches, on définit au préalable la collection de tests et les requêtes, puis les équipes participantes soumettent des résultats de recherche. On puise (*pooling*) ensuite les premiers résultats (typiquement 100) de chaque liste soumise et les annotateurs évaluent leur pertinence. Les résultats qui ne sont pas dans ces puits sont ensuite considérés comme non-pertinents. Ainsi, il est important d'être confiant d'avoir trouvé la quasi-totalité des documents pertinents en utilisant une profondeur de puits (*pool depth*) suffisante, et en s'assurant qu'il y ait une bonne variété de systèmes dans l'ensemble original. Si ces conditions sont atteintes, les systèmes pourront être comparés par des mesures utilisant les résultats de recherche.

Des collections de tests peuvent être utilisées après la compétition, mais il faut être prudent. Des jugements insuffisants entraînent ce qu'on nomme le biais de puits (*pool bias*), qui se produit quant un système utilise des jugements post-compétition, mais que beaucoup des documents qu'il suggère ne font pas partie du puits original. Ces documents, qui sont potentiellement pertinents, sont alors considérés comme non-pertinents, ce qui a l'effet de pénaliser le score de ce système.

3.2.2 Mesures d'évaluation populaires

Les mesures les plus simples pour évaluer l'efficacité d'un système sont la *précision* et le *rappel*. La *précision* est la proportion des documents soumis qui sont pertinents :

$$\text{précision} = \frac{|\{\text{documents pertinents}\} \bigcap \{\text{documents trouvés}\}|}{|\text{documents trouvés}|}$$

Le *rappel* est la proportion des documents pertinents qui se retrouvent dans les résultats soumis.

$$\text{rappel} = \frac{|\{\text{documents pertinents}\} \bigcap \{\text{documents trouvés}\}|}{|\text{documents pertinents}|}$$

Une mesure qui unit ces deux statistiques est la mesure F [61], qui est la moyenne harmonique pondérée de la précision et du rappel :

$$F_\beta = \frac{(1+\beta^2)\text{précision} \cdot \text{rappel}}{\beta^2 \cdot \text{précision} + \text{rappel}}$$

Le paramètre β contrôle l'importance accordée à la précision par rapport au rappel, et est souvent 1.0 pour une importance égale. La précision et le rappel ont typiquement une relation inverse : il est facile d'avoir une haute précision quand on ne propose que les documents qu'on estime hautement pertinents, et il est aussi possible d'avoir un rappel parfait en proposant tous les documents.

Comme en RI la quasi-totalité des documents ne sont pas pertinents, ces mesures ne sont pas très utiles. On peut toutefois les calculer à chaque point de la liste de résultats pour les combiner en des mesures plus utiles. Voici les plus populaires :

La *Précision à X* (typiquement 10) est un indicateur utile pour évaluer les premiers résultats de la recherche. En effet, particulièrement en recherche Web, les utilisateurs regardent rarement plus loin que les quelques premiers résultats. On peut donc accorder une grande importance à la précision initiale en utilisant cette mesure. Soit $pert_i$ un nombre entre 0 et 1 indiquant le degré de pertinence du document à la position i, la précision à X est :

$$P@X = \frac{1}{X}\sum_{i=1}^{X} pert_i$$

Le *rappel à X* est la proportion des documents pertinents trouvés après X documents :

$$R@X = \frac{\sum_{i=1}^{X} pert_i}{|\text{documents pertinents}|}$$

On peut aussi s'intéresser à la *précision à un degré de rappel*, celle-ci est la précision à la position de la liste de résultats où on atteint un degré de rappel particulier. Cette mesure est pertinente car elle fait abstraction du nombre de documents pertinents de la requête. On peut donc en faire la moyenne pour plusieurs requêtes, et même en faire un graphe qu'on nomme la courbe précision-rappel. Cette relation est généralement décroissante, mais comme la précision augmente quand on trouve un document pertinent, il peut être

utile d'interpoler les valeurs à la plus haute valeur subséquente dans le graphe. De plus, pour simplifier la compréhension, il est utile de limiter les valeurs à quelques degrés de rappel. Des incréments de 0.1 entre 0.0 et 1.0 sont très utilisés et on nomme ce graphe la *précision moyenne interpolée à 11 points*.

La *précision moyenne* (*average precision* – $AveP$) est la moyenne des précisions aux positions de la liste où se trouvent des documents pertinents. Si $|R|$ est le nombre de documents pertinents, alors :

$$AveP = \frac{1}{|R|} \sum_i P@(i) pert_i$$

L'idée derrière cette mesure est que la valeur d'un document est proportionnelle à l'espérance de trouver un document pertinent à un point de la liste. En effet, si on trouve beaucoup de documents erronés, l'utilisateur aura peu tendance à vouloir continuer à regarder les résultats. Pour obtenir une évaluation globale du système, on peut faire la moyenne de cette valeur pour toutes les requêtes, ce qui donne la *précision moyenne moyenne* (*Mean Average Precision* – MAP).

Le $nDCG$ (*normalized Discounted Cumulative Gain*) est une autre mesure à un chiffre, qui se voit comme étant une amélioration au MAP. On le définit de la manière suivante :

$$DCG = \sum_i \frac{2^{pert_i} - 1}{\log_2(1+i)}$$

$$nDCG = \frac{DCG}{IDCG}$$

Ici, DCG est le score des résultats de la requête. $IDCG$ est le DCG idéal obtenu si on avait les documents en ordre décroissant de pertinence. On utilise le $IDCD$ pour normaliser les scores. La différence principale au MAP est que, comme les utilisateurs ne regardent habituellement que les premiers résultats, on limite l'importance des résultats dans les positions avancées de la liste par une échelle logarithmique. De plus, en mettant la valeur de pertinence en exposant, on accorde une plus grande importance aux documents les plus pertinents. Cette mesure, bien qu'elle comporte des éléments arbitraires,

gagne en popularité de nos jours [45].

Nous avons présenté les mesures les plus populaires dans la section précédente. À celles-ci, nous ajouterons une dernière métrique jugée pertinente.

3.2.2.1 Aggrégations par moyenne géométrique

Le MAP *géométrique* ($GMAP$) [64] est une métrique d'évaluation qui a été inventée dans le contexte de la tâche *robust* de la conférence TREC . Celle-ci provient de la constatation que les métriques d'évaluations standard comme le MAP ou $nDCG$ donnent une importance égale aux améliorations peu importe la difficulté des requêtes. Un exemple concret de ceci est le suivant : si nous avons un gain en $AveP$ de 0.0 à 0.1 pour une requête difficile, et une réduction de 0.6 à 0.5 dans une autre requête facile, il n'y aura aucun changement en MAP.

On peut toutefois comprendre qu'il est plus important d'augmenter une requête où on peine à trouver des documents, quitte à réduire une requête pour laquelle on en trouve déjà beaucoup. Ainsi, face à cet illogisme de la moyenne arithmétique, on propose l'utilisation de la moyenne géométrique : le produit de n éléments mis à la puissance $1/n$. On peut aussi exprimer cette valeur comme l'exponentiation de la moyenne des logarithmes des valeurs. Dans les deux formulations, on contate toutefois qu'une valeur nulle donne une valeur invalide (log ou racine de zéro). Dans le cas où nous faisons la moyenne des scores des requêtes, ceci sera fréquent (aucun document trouvé). En théorie, si on ne limitait pas le nombre de documents proposés, il serait impossible d'avoir des valeurs nulles ou parfaites, et ainsi pour avoir des scores plus corrects, il faudrait faire un lissage des scores. Toutefois, comme nos collections, listes de documents et jugements de pertinence sont limités, il n'est pas clair comment un tel lissage devrait être fait. Il est donc pratique courante de simplement ajouter une petite quantité ϵ au score pour éviter un logarithme ou une racine infinie, puis de la retirer sur le résultat final.

Dans le cas d'une aggrégation de scores $AveP$, comme le programme *trec_eval* calcule les métriques à quatre décimales, on ajoute souvent $\epsilon = 0.00001$, à toutes $AveP$. Si

Q est l'ensemble des requêtes évaluées, le $GMAP$ est défini par :

$$\begin{aligned} GMAP &= \prod_{q \in Q} \left[AveP(q) + \epsilon \right]^{1/|Q|} - \epsilon \\ &= \exp\left\{ \frac{1}{|Q|} \sum_{q \in Q} \ln\left(AveP(q) + \epsilon \right) \right\} - \epsilon \end{aligned}$$

Cette métrique est à notre avis beaucoup plus réaliste que le MAP. Malheureusement il est difficile de convaincre les gens d'évaluer leurs systèmes avec d'autres métriques, quand les gains faits en utilisant des technique répandues – par exemple le *feedback aveugle* (*blind feedback*) – ont généralement des effets néfastes sur une autre métrique.

En général, on considère que les aggrégations utilisant la moyenne arithmétique évaluent la quantité totale de documents trouvés, et celles utilisant la moyenne géométrique évaluent la *robustesse* de recherche. Il est intéressant de noter que la moyenne géométrique peut être appliquée pour agréger toute métrique bornée, et peut ainsi être utilisée pour la $P@10$, $nDCG$ et même la précision à un point de rappel. Lors de l'usage d'une telle métrique, nous mettront "G" avant son nom (ex : $GMAP$, $GnDCG$, etc.)

3.2.3 Note sur l'évaluation

Normalement, quand on compare deux systèmes, il faut aussi montrer que les différences ne sont pas dues au hasard par des tests de signifiance statistique de leur différence. Aussi, veuillez noter que ces métriques n'évaluent que l'efficacité des résultats et ignorent l'interraction avec l'utilisateur du système, qui est tout de même un facteur important.

3.3 Index et requête

Les systèmes de RI devant être capables de fonctionner avec des collections contenant des milliards de documents, il est impossible de calculer des scores pour tous les documents pour chaque requête. Une solution à ce problème est d'utiliser un index inversé. Quand un index régulier liste les mots contenus dans un document, un index

inversé fait le contraire et liste les documents contenant un mot. Un système de RI construit donc un index par unité d'indexation, et celui-ci a une entrée pour chaque document la contenant. Cette entrée peut contenir des données additionnelles, comme la fréquence du mot, une liste des positions des occurrences du mot dans le document, dans quel champ du document (titre, corps, etc.) se trouve le mot, etc.

3.3.1 Unité d'indexation

Les unités d'indexation sont les éléments d'index auxquels on associe un document. Les requêtes doivent avoir des unités compatibles avec celles des collections. Ces unités sont le plus souvent un seul mot, quoi que des études aient été faites sur l'usage de n-grammes (n mots consécutifs) [55], bitermes (deux mots sans ordre) [57], ou autres schémas tels que des *triplets* (un mot, un idiome de relation, et son mot attaché). Ces formes d'index sophistiqués n'ont, de nos jours, pas démontré une utilité motivant les ressources additionnelles. Plutôt qu'indexer un n-gramme, on peut alternativement indexer les positions des mots et introduire des statistiques de proximité entre les mots clés dans les modèles, ce qui a un effet similaire. Malgré ces alternatives, de nos jours, les index sont encore le plus souvent basés sur un simple mot.

3.3.2 Qu'est-ce qu'un mot ?

On pourrait penser à tort que le concept de mot, ou plus précisément *unité lexicale* (*token*), se limite à une série de lettres entre des espaces ou ponctuations. En français, anglais, ou dans d'autres langues européennes, cette définition peut être correcte. C'est toutefois problématique pour les langues ayant une morphologie riche (langues scandinaves, néerlandais, hollandais, allemand, arabe, etc.) pour lesquelles on trouve des mots composés très longs. Il est une bonne pratique de séparer ces mots. L'opération est similaire à séparer des mots ayant des traits d'union en français.

Le problème de définir ce qu'est un mot est encore plus difficile pour les langues asiatiques. En effet, des langues telles que le chinois, le japonais et le coréen n'emploient pas de séparateurs de mots. On nomme *segmentation* la tâche de séparer une

série de caractères en une série d'unités lexicales. La segmentation est le plus souvent faite en utilisant des outils d'analyse syntaxique, mais il existe des ambiguïtés de segmentation, et il est souvent difficile de déterminer si une série de caractères devrait être un ou plusieurs mots.

3.3.3 Troncature et lemmatisation

Un des traitements le plus souvent effectué sur les unités lexicales est la lexémisation (ou troncature – *stemming*). Cette opération convertit un mot en lexème ou racine (*stem*), composé des premiers caractères du mot. Par exemple, les mots *"international"* et *"internationalement"* pourraient être amalgamés (*conflated*) en un seul mot *"internation*". En effet, on pourrait s'attendre à ce qu'un utilisateur cherchant *"international"* veuille aussi les résultats pour *"internationalement"*. La troncature a donc pour effet de réduire le vocabulaire de la collection, et en même temps le *problème de non-correspondance du vocabulaire* (*vocabulary mismatch*) entre la requête et les documents.

La lemmatisation est une opération semblable. Plutôt que de retirer des suffixes, on utilise des principes morphologiques pour convertir le mot en une forme de base. Par exemple, *finissent* serait converti à l'infinitif *finir*, plutôt qu'une racine comme *fin*. On voit qu'une telle distinction *fin*/*finir* a peu d'utilité en pratique, et que généralement, la troncature fonctionne mieux que la lemmatisation [17].

3.4 Modèles de langues pour la RI

Dans ce mémoire, nous utiliserons le paradigme des *Modèles de Langue* (LM) pour la RI [11, 40]. Cette approche est relativement récente, et gagne en popularité parce qu'elle offre des bonnes performances tout en ayant peu de paramètres. De plus, elle est basée sur des principes probabilistes Bayesiens, ce qui est très attrayant comparé aux approches heuristiques présentes dans les modèles de type espace vectoriel [46]. De plus, l'approche est facile à adapter à nos besoins.

3.4.1 Modèle de langue

Un modèle de langue (LM) est un outil provenant du TAL [52] assignant une probabilité à une série de mots $P(w_{1..k})$. Celui-ci est construit en comptant les occurrences de séries de n mots (n-grammes), et on assume une hypothèse Markovienne de cette longueur (on assume que les mots ne dépendent que des $n-1$ mots précédents). Les longueurs de n les plus courantes sont 1 (unigrammes), 2 (bigrammes) et 3 (trigrammes). Ainsi, la probabilité d'une série de mots $w_{1..k}$ est formulée par

$$p(w_{1..k}) \quad = \quad \prod_i p(w_i | w_{i-n+1\,..\,i-1})$$

Un problème fondamental rencontré avec les LM est qu'il est difficile d'estimer les probabilités des événements rares. En effet, on dit de la fréquence d'occurrence des mots qu'elle suit la *loi de Zipf* (nommé selon le linguiste Américain George Kingsley Zipf), qui spécifie que les mots, mis en rang de fréquence décroissante r, suivent généralement une relation $1/r$. Ceci implique que la majorité des mots sont très rares et ne seront que très peu observés dans les corpus. Ces éléments auront donc une probabilité nulle ou instable, et ainsi pour une série de mots assez longue, on aura une probabilité nulle. Il faut donc assigner une partie de la masse de probabilités aux événements non observés.

3.4.2 Lissage des documents

L'opération d'affecter une masse de probabilités aux événements non observés se nomme le lissage. Dans des systèmes de RI, étant donné la taille des collections et le fait qu'on ne doit produire un score stable avec quelques mots clés, et quelques observations dans un document, on se limite presque exclusivement à des modèles unigrammes. Quand on dispose d'un document de longueur limitée, on veut ainsi compléter son LM en le lissant avec un LM de collection. En effet, la collection ayant une envergure beaucoup plus grande qu'un document, elle offre des probabilités unigrammes beaucoup plus solides pour les mots rares. Les méthodes de lissage les plus utilisées sont les suivantes :

Le lissage *Jellinek-Mercer* (JM) est une simple interpolation linéaire :

$$P_{JM}(x) = \lambda P_{ML}(x|D) + (1 - \lambda)P(x|\mathcal{C})$$

Le lissage *Dirichlet* s'ajuste à la longueur du document :

$$P_{Dir}(x) = \frac{|D|P_{ML}(x|D) + \mu P(x|\mathcal{C})}{|D| + \mu}$$

On incrémente le compte pour un mot x par μ fois la probabilité de ce mot dans la collection. Le fondement de cette méthode est que la sélection d'un mot est une variable aléatoire multinomiale. On peut appliquer un a priori à cette variable sous la forme d'une Dirichlet. Ceci résulte en une forme fermée qui est l'équivalent d'ajouter $\mu P(x|\mathcal{C})$ pseudo-observations au modèle de langue à vraisemblance maximale. Ces observations auront une importance différente selon la longueur du document.

Le *décompte absolu* (absolute discounting) retire δ unités (entre 0 et 1) de compte à tous les mots du document et les remplace par des comptes de la collection :

$$P_{AD}(x) = \frac{\max\{|D|P_{ML}(x|D) - \delta, 0\} + |V_D|\delta P(x|\mathcal{C})}{|D|}$$

Dans cette équation, V_D est le vocabulaire du document. L'idée derrière cette méthode est que les mots ayant un petit nombre d'occurrences dans un document sont très rares, et ainsi ceux qui sont observés sont souvent sur-estimés.

3.4.3 Vraisemblance de la requête

L'approche de base des modèles de langue pour la RI se nomme le *modèle de vraisemblance de la requête*. L'idée derrière celle-ci est qu'un utilisateur peut formuler sa requête en s'imaginant quels sont les termes qu'on pourrait s'attendre à retrouver dans les documents pertinents. Formellement, on assume donc que chaque document est un échantillon de langage, et que les documents pertinents sont ceux qui génèrent la requête avec la plus grande vraisemblance. Ainsi, pour une requête q, la probabilité d'un document d est :

$$p(d|q) = \frac{p(q|d)p(d)}{p(q)}$$

La probabilité de la requête est constante pour tous les documents et peut donc être ignorée :

$$p(d|q) \propto p(q|d)p(d)$$

Et ainsi, sous l'hypothèse d'indépendance des mots clés :

$$p(d|q) \propto p(d) \prod_i P(q_i|D)$$

La probabilité a priori $p(d)$ est souvent considérée comme uniforme, et peut ainsi être ignorée. Elle peut toutefois contenir toutes sortes d'informations comme la considération de la longueur du document [2, 39] ou des indicateurs du potentiel de pertinence du document [5, 21].

Les études sur le lissage utilisé pour ces modèles [69, 70] montrent que pour ces modèles le décompte absolu est meilleur que Jellinek-Mercer, et que Dirichlet est meilleur que le décompte absolu. Il est toutefois possible de combiner le lissage Dirichlet et Jellinek-Mercer pour obtenir de meilleurs résultats.

Une hypothèse développée par Zhai et Lafferty [69] serait que le lissage ne sert pas qu'à estimer des probabilités réalistes pour les mots des documents, mais qu'il gère aussi des éléments de "modélisation de requête" (*query modeling*). Par exemple, on peut estimer un pseudo-compte Dirichlet automatiquement à partir de la collection, mais ça donne des résultats moindres qu'une constante sélectionnée. Utiliser un pseudo-compte constant mais plus gros, ou faire un deuxième lissage Jellinek-Mercer en plus du Dirichlet améliore la gestion des poids des mots.

Des études récentes [2, 53] mettent en doute cette théorie. Selon celles-ci, la différence entre les formes de lissage serait plutôt causée par l'effet de pénaliser ou non les documents plus courts, qui auraient tendance à être moins pertinents. On voit donc que comme le paradigme de modèles de langues est relativement récent, il reste beaucoup à apprendre pour arriver à une compréhension plus complète de la relation entre les

modèles de langues et la notion de pertinence.

CHAPITRE 4

DÉSAMBIGUISATION DE SENS EN RECHERCHE D'INFORMATION

L'utilisation de techniques de Désambiguisation de Sens (DS) n'a pas, de nos jours, mené à des améliorations solides dans des tâches du TAL [33]. C'est aussi le cas pour la Recherche d'Information (RI), malgré que la question ait été abordée à plusieurs reprises [8, 14, 22, 47, 49, 62, 66]. Dans ce chapitre, nous commencerons par faire un parcours des efforts passés, puis nous présenterons une adaptation du système proposé dans la section 2.4 de ce mémoire pour des applications de RI. La différence principale à cet effet est que l'application précédente était une approche supervisée assignant des étiquettes de sens à des mots en contexte. Pour une utilisation en RI, nous n'avons pas accès à un inventaire de sens, ni à des données étiquetées suffisantes. Nous allons donc passer à une approche non-supervisée. À ces fins, nous allons utiliser le même ensemble d'attributs que ceux utilisés dans les expériences supervisées, mais en remplacement des aggrégations représentant un sens, nous apprendrons les tendances principales du contexte par *allocation Dirichlet latente*. Ceci réduit l'ensemble des fenêtres d'un mot dans un document en une représentation à basse dimensionalité. Cette représentation sera ensuite utilisée pour faire une *discrimination de sens* avec le contexte des mots de requête. Le score de discrimination sera ensuite combiné avec l'approche de la vraisemblance de requête.

Ce chapitre est organisé de la manière suivante. Dans la section 4.1, nous ferons une revue des efforts faits afin d'intégrer des techniques de DS en RI. Dans la section 4.2, nous résumons les conclusions des études précédentes, et proposons une liste de points importants. Dans la section 4.3 nous décrirons la construction de modèles de contextes latents, et leur intégration dans un système de RI. Dans la section 4.4, nous effecturons des expériences sur des collections de la conférence TREC, puis discuterons les résultats. Nous conclurons en section 4.5 et présenterons des points de recherche futurs.

4.1 Recherche sur la DS en RI

La recherche sur la DS dans le contexte de la RI commence dès les débuts de la RI. Avec l'apparition du Web, le problème a toutefois pris une nouvelle importance. Les premières approches étudiées étaient proches des techniques de DS standard, et n'ont pas eu des résultats positifs. On comprend donc que les techniques doivent être plus adaptées à la tâche de la RI. Nous verrons donc dans cette section une sélection d'études qui, sans réussir à valider les avantages de la DS, permettent d'obtenir une perspective sur la problématique de l'ambiguïté dans les documents et les requêtes.

4.1.1 Krovetz & Croft (1992) [22]

Cet article propose tout d'abord qu'il y aurait deux types d'ambiguïtés : syntaxique et sémantique. Ces deux catégories sont analogues à la différence entre homonymie et polysémie. Bien qu'il soit difficile de déterminer automatiquement la nature d'homonymie ou de polysémie d'un mot, l'homonymie serait moins problématique pour les applications de RI, étant donné que ce serait la superposition de deux mots différents. Les deux sens ne partageant pas un noyau sémantique, ils devraient se trouver dans des contextes différents, et ainsi seraient désambiguïsés implicitement par la cooccurrence de plusieurs mots clés. Le deuxième type d'ambiguïté de mot, la polysémie, comporte un lien concret entre les sens, et est donc plus abstrait et serait donc plus difficile à analyser.

On discute ensuite la forme que devrait prendre la représentation du sens. À cet effet, les auteurs proposent l'utilisation d'un dictionnaire. Ceux-ci sont utilisés dans l'idée d'*experts de mot* qui était populaire à l'époque. Dès lors, on note toutefois des problèmes de couverture des dictionnaires, qui seraient, plutôt qu'un problème de concordance du vocabulaire, un problème de couverture de sens d'un mot et de cooccurrence entre les définitions et les mots de contexte.

Les expériences contenues dans cet article ont peu d'utilité dans un contexte moderne étant donné les méthodes très heuristiques utilisées. Toutefois, une expérience observant la distribution des sens de mot montre que le premier sens du dictionnaire est souvent le plus fréquent, et qu'il prend une très grande quantité des occurrences. Ceci est répété

dans plusieurs articles subséquents, sous l'idée de l'existence d'un "sens dominant".

4.1.2 Sanderson (1994) [47]

Cet article effectue une investigation sur l'effet qu'a l'ambiguïté sur les performances de RI. La méthode utilisée est de contrôler la quantité d'ambiguïté présente dans la collection par l'introduction de "pseudo-mots". Par exemple, on peut fusionner le mot "banane" et "kalashnikov" en une seule unité d'indexation. Le mot résultant devient ambigu, et la réduction de précision serait semblable au phénomène d'ambiguïté qui apparaît naturellement. On pourrait donc estimer les gains obtenus avec une désambiguation parfaite. L'approche est due à Yarowsky [68] mais n'était pas utilisée dans le contexte de la RI.

La conclusion de cette étude est que l'ambiguïté telle qu'ajoutée par les pseudo-mots a peu d'effet sur les résultats de recherche. Avec un peu de perspective, on comprend toutefois que les mots fusionnés sont sélectionés aléatoirement, et ainsi tendent à être peu reliés. Quand plus d'un mot est présent dans la requête, la cooccurrence de ces mots tend vers le sens correct, et ainsi cette ambiguïté est du type d'homonymie, automatiquement réglée quand il y a plusieurs mots dans la requête. Un deuxième problème avec l'approche, est que, si les pseudo-mots sont sélectionnés aléatoirement dans le vocabulaire en ignorant la fréquence des mots, la nature Zipfienne du langage a pour effet que la fusion a peu d'impact. En effet, les mots fréquents seront beaucoup présent dans les mots de requête, mais si on choisit des mots fusionnés aléatoirement dans le vocabulaire, ceux-ci seront fort probablement peu fréquents, et auront donc peu d'impact sur les résultats de recherche [49]. Gaustad [14] montre ensuite que l'idée d'utiliser des pseudo-mots est donc non-fondée, et ceci se concrétise lorsqu'on évalue les différences de performance de systèmes de DS quand on compare les mots réellement ambigus contre les pseudo-mots.

4.1.3 Shütze & Pedersen (1995, 1998) [50, 51]

La méthode proposée dans cet article consiste à définir un vecteur de contexte pour chaque occurrence, puis à effectuer un groupage des occurrences. Les mots du vecteur de contexte sont pondérés par tf-idf ($tf \log(N/df)$ où tf est la fréquence du mot, N est le nombre de documents de la collection, et df le nombre de documents contenant le terme), le groupage est de type *buckshot* par rapport au cosinus entre les instances et les centroïdes des groupes. L'affectation d'une instance de sens est du type dur. L'avantage de cette approche est qu'elle ne dépend pas de ressources externes. Elle est donc applicable à toutes les langues et ne souffre pas de problèmes de couverture.

Cette expérience serait la première application positive de DS pour la RI. On constate une augmentation relative de la précision de 14% en combinant le score de mots et le score de sens, mais l'étude a une portée limitée car les résultats ne portent que sur 25 sujets d'une petite collection (sujets 51-75 sur la collection WSJ de TREC). Le fait qu'il n'y ait pas eu d'études subséquentes montrerait que ces résultats sont largement surestimés. Sanderson [48] explique que ces résultats positifs sont dus à l'usage de requêtes étendues qui contiennent des dizaines de mots, et non seulement le champ titre comme on le retrouve habituellement. Ceci démontre toutefois le potentiel qu'aurait une désambiguation de haute qualité.

4.1.4 Gonzalo et al. (1998) [15]

Cet article apporte l'idée originale de l'existence du corpus annoté SemCor [32]. SemCor est un sous-ensemble du Brown corpus ayant été manuellement étiqueté avec WordNet comme inventaire de sens. En l'utilisant, on peut simuler le cas où on aurait une désambiguisation parfaite. Les auteurs ont donc créé des jugements de pertinence pour des requêtes soumises à la collection SemCor. L'idée est bonne, mais l'application faite est non-standard quant à la RI. On coupe les documents en "chunks cohérents" (171 fragments/documents), qu'on résume en quelques phrases (22 mots en moyenne). Les résumés sont annotés sémantiquement avec les mêmes étiquettes sémantiques qu'on retrouve dans SemCor. L'expérience effectuée est ensuite de considérer les résumés comme étant

une requête, et de retrouver le fragment original.

Les conclusions de cette recherche seraient qu'indexer avec des sens pris de WordNet améliorerait la qualité des résultats de 29% relative en comparaison avec système SMART. Toutefois, il faut mettre ces résultats en perspective car cette étude a plusieurs problèmes. (1) Il n'y a que 171 documents, ce qui réduit de beaucoup les possibilités de faux-positifs. (2) Les résumés/requêtes étant faits après la lecture du passage, ils auront une très haute concordance de vocabulaire. (3) Les résumés ayant beaucoup de mots sont trop bien définis en comparaison avec des requêtes réelles.

4.1.5 Stokoe, Oakes et Tait (2003) [59]

Les auteurs se basent sur la littérature précédente pour résumer la problématique de la manière suivante :

- La distribution de sens comporte un sens dominant à la fois dans les documents et les requêtes, ce qui explique les bonnes performances des systèmes qui n'utilisent pas la DS.
- Lorsque l'inventaire de sens est très fin, les erreurs de désambiguisation ont un effet très négatif.
- Pour améliorer l'efficacité de la recherche, il faut minimiser les impacts négatifs des désambiguations erronées.
- Les conclusions de Sanderson, à l'effet qu'une précision de désambiguisation d'au moins 90% serait nécessaire pour voir des améliorations, sont discutables.

En se basant sur ces principes, les auteurs proposent un système de DS basé sur SemCor. L'implémentation du système n'est pas précisée en détail, mais on mentionne l'utilisation des mots fréquents dans une fenêtre limitée à la phrase courante, des attributs de mots positionnels dans le voisinage du mot cible. On mentionne aussi utiliser les informations de fréquences présentes dans WordNet pour contrôler l'impact de la désambiguation. Ainsi, sans des évidences suffisantes, les poids seront identiques aux poids de base n'utilisant pas WordNet.

L'approche est testée sur les sujets 451-550 de TREC sur la collection WT10G. Les résultats donnent un MAP de 3.40% pour la recherche standard, et 5.04% avec la désa-

mbiguation. Ceci est une augmentation de 1.64% en termes absolus, et 48.24% relatifs. Cette étude montrerait donc que la DS aurait un effet bénéfique. Toutefois, on peut s'interroger sur les scores des systèmes qui sont anormalement bas, car on pourrait s'attendre à un MAP d'environ 20% sur ces données avec un système de base. Avec un score si bas, peu importe ce que nous ferons il sera difficile de réduire le score, et ainsi toute tentative aura un effet positif. Ceci ne peut donc être considéré comme une preuve que la DS fonctionne.

4.1.6 Kim et al (2004) [9]

Les auteurs de cet article se basent sur les résultats observés à Semeval, qui montreraient que des inventaires de sens plus généraux seraient plus appropriés pour des applications de RI. Ils proposent donc d'utiliser WordNet pour générer des "sens noyaux", étant une des 25 catégories les plus générales dans la hiérarchie de sens de WordNet. Ces sens sont extrêmement larges, et ainsi on peut les assigner avec une grande stabilité en observant les occurrences de mots aux alentours. Chaque instance d'un mot dans un document peut être assignée à un sens différent, si ceci est le cas, on considère que tous ces sens sont satisfaits. Pour les expériences, on utilise les sujets 351-450 de TREC. Les auteurs utilisent trois formules de score plus simples que le BM25 habituel [43]. Dans les cas où on constate des scores inférieurs avec les formules de base, on observe des améliorations en utilisant la DS. Toutefois, on n'observe pas d'amélioration avec la meilleure des trois formules de base, qui est proche de la formule habituelle.

Ainsi, comme on n'a pas d'améliorations à comparer avec un système de base acceptable, on ne peut pas conclure que la DS a un effet positif. Au mieux, celle-ci compense quand on utilise une mauvaise formule.

4.1.7 Stokoe (2005) [58]

Cette étude porte sur l'investigation de la différence entre l'homonymie et la polysémie. On adapte la méthode de pseudo-mots développée par Sanderson pour pouvoir simuler le phénomène de polysémie. Pour générer de l'homonymie, on fusionne des

mots n'ayant pas de relation (sélection aléatoire). Pour générer de la polysémie, on utilise WordNet pour prendre des mots connexes. Pour chaque mot des requêtes de test (501-550 de TREC), et pour chaque entrée de sens pour ce mot, on parcourt les hyperonymes en montant la hiérarchie jusqu'à ce qu'on trouve un autre mot dans cette catégorie. Les instances de ces mots sont fusionnés, résultant en un phénomène qui serait semblable à la polysémie.

Des expériences ajoutant des pseudo-mots de type homonymie confirment qu'ils ont un impact sur l'efficacité de la recherche : le MAP change de 13.34% vers 11.45% (-14% relative), la P@10 de 25.83% à 22.08% (-16% relative). Les pseudo-mots de type polysémique ont un impact plus fort : 18.75% de MAP (-28.71% relative) et 18.75% de P@10 (-27.41% relative). Ceci indiquerait donc que la polysémie a un plus grand effet sur l'efficacité de la recherche. Il est toutefois difficile de comparer les résultats, surtout si on ne peut comparer la quantité de mots rendus ambigus pour les deux expériences. On peut donc au mieux conclure que l'ambiguïté a un effet plus fort qu'on aurait pu imaginer auparavant.

4.2 Principes importants

La littérature sur la DS en RI montre que la tâche est difficile. Les études montrant des effets positifs ont malheureusement des problèmes de méthodologie ou ont une portée insuffisante. De celles-ci on peut toutefois avoir une vue d'ensemble des points importants en ce qui concerne la RI :

- Les mots ambigus ont souvent une distribution de sens où un sens dominant prend la plupart des instances. Ce phénomène est présent à la fois dans les documents et les requêtes. L'ambiguïté affecte donc une minorité de requêtes.
- Il y aurait une différence entre l'ambiguïté causée par la polysémie d'un mot et l'homonymie. L'homonymie serait moins problématique que la polysémie parce que comme les sens n'ont pas un lien abstrait, on peut s'attendre à une plus grande facilité à distinguer les sens.
- Comme les mots de la requête sont souvent cohérents entre eux, leur cooccurrence

dans un document effectue une désambiguation implicite. Cette désambiguisation serait moins efficace pour une ambiguïté de type polysémique.

- Une haute précision de désambiguation est nécessaire pour obtenir un effet positif de la désambiguisation. Le problème est que, bien que les documents offrent suffisamment de contexte pour assigner un sens à un mot, la requête n'offre que très peu d'informations, et il est donc difficile de cerner le sens correct du mot de la requête.

En ce qui concerne la définition d'un sens, nous pouvons ajouter les points suivants :

- Les ressources créées manuellement telles que WordNet ou SemCor sont rarement disponibles. On constate en effet qu'aucune étude de la DS pour RI unilingue n'existe en dehors de l'anglais.
- Les ressources existantes ont des problèmes de couverture, surtout en ce qui concerne la complétude des inventaires des sens. Les inventaires ne couvrent souvent que les sens les plus "importants" malgré que d'autres sont clairement définis [20]. Les inventaires manuels manquent donc d'empirisme et sont sujet à des biais tels que culturels, ou de fréquence.
- La granularité des inventaires est problématique. Il est connu dans la littérature qu'un inventaire de sens "fin" résulte en une précision de désambiguisation faible ; le phénomène touche aussi les humains, qui ont un taux d'accord plus bas lorsqu'un mot a beaucoup de sens potentiels.

Aussi comme les applications de RI sont à très grande échelle, on ajoute les contraintes suivantes :

- L'inventaire de sens doit être généré automatiquement et s'ajuster à la collection utilisée.
- Les méthodes doivent être très efficaces à la fois au moment de l'indexation et à l'évaluation des requêtes.
- Plutôt qu'un index dur affectant un sens à un mot, il est préférable d'effectuer une discrimination de sens entre la requête et les documents. La force de celle-ci devra s'effectuer en fonction des évidences à la fois dans la requête et dans les documents.

Nous proposerons une approche avec ces points en tête.

4.3 Constriction de requête par modèles de contexte latents

L'approche présentée dans cette section est une adaptation du système de DS présenté à la section 2.4 de ce mémoire. Ce système consiste à générer une représentation à dimensionalité réduite du contexte d'occurrences d'un mot dans un document. Ce contexte réduit pourra représenter les tendances générales du contexte, et on pourra ensuite calculer des probabilités de génération du contexte de la requête selon celui d'un document. Cette probabilité sera ensuite intégrée dans le modèle de vraisemblance de requête présenté en section 3.4.3. L'effet obtenu est un resserrement du contexte de la recherche, qui a l'effet de raffiner le besoin d'information, et ainsi d'enlever une partie des problèmes d'ambiguïté.

4.3.1 Modèles supervisés

Pour rappeler l'approche développée pour les systèmes supervisés, nous utilisions les attributs suivants :

- Les formes du mot cible forment une variable multinomiale. Comme ceux-ci sont beaucoup observés, un lissage simple ajoute-un est suffisant.
- Les mots du contexte, avec un poids en fonction de leur distance du mot cible. Nous avons vu avec les expériences de DS que la fonction de poids idéale est très similaire à une loi de puissance négative. Ceux-ci forment une deuxième multinomiale qu'on peut lisser avec les probabilités au niveau de la collection.
- La présence/absence des mots outils voisins du mot cible, avec des attributs différents selon si ceux-ci sont avant ou après le mot cible. Nous les traitons comme des variables Bernouilli : une multinomiale à deux états. Tout comme les mots cibles, comme les mots outils sont observés très souvent un lissage ajoute-un est suffisant.

Pous contrôler la force d'impact de chaque type d'attribut, plutôt que de contrôler la quantité de lissage, on propose des techniques d'estimation des a priori, puis on contrôle

l'impact des attributs en multiplier les log-probs résultante par un facteur par type d'attribut. Ceci aide à gérer les problèmes de dépendances entre attributs.

Pour appliquer ces classifieurs à la RI, où nous ne disposons pas d'inventaire de sens ou de données annotées, nous devons passer à des approches non-supervisées. À ces fins, nous proposons l'utilisation de l'allocation Dirichlet latente (LDA) [4]

4.3.2 Latent Dirichlet Allocation (LDA)

Des modèles latents tels que LDA sont utilisés depuis très longtemps en RI. Des modèles comme LDA sont souvent nommés *modèles de sujets*, car ils permettent de "résumer" un document par une mixture de sujets – des distributions de mots connexes. Comme nous faisons une application en RI, il est nécessaire d'introduire les précurseurs les plus importants de LDA.

Le premier modèle latent pour la RI se nomme l'*indexage sémantique latent* (*Latent Semantic Indexing – LSI*) [12], aussi connu sous le nom de *Latent Semantic Analysis*. Celui-ci utilise des techniques d'*analyse en composantes principales* ou, de manière équivalente, de *décomposition en valeurs singulières*. En utilisant des principes géométriques, on effectue une approximation du vecteur des mots d'un document par un ensemble réduits de vecteurs. Ces vecteurs représentent les tendances principales des documents de la collection. On peut ensuite utiliser ce "résumé" afin d'évaluer la similarité d'un document avec une requête, par exemple par cosinus. Cette approche est très heuristique, et nécessite souvent l'utilisation de poids tf-idf. La capacité du système est contrôlée par le nombre de vecteurs conservés.

Plus tard, Hoffman propose le LSI probabiliste – pLSI [16], aussi connu sous le nom de *modèle d'aspect*. Celui-ci vise à corriger l'approche heuristique du LSI en représentant les documents par une mixture d'un ensemble de sujets. Chaque sujet est une distribution de mots reliés et ceux-ci sont appris par un algorithme EM. C'est une approche à vraisemblance maximale, et ainsi souffre de problèmes de sur-apprentissage. À ces fins, Hoffman propose l'utilisation d'un contrôle de la capacité par tempérance, évaluant les sujets sur un ensemble externe.

Quelques années après pLSI fut proposée l'allocation Dirichlet latente (LDA) [4].

Celle-ci vise à corriger des défauts de pLSI par l'utilisation d'une approche Bayesienne. Comme on ajoute un a priori à chaque paramètre du modèle, celui-ci arrive à ajuster la force d'inférence en fonction des données observées, et on évite les problèmes de surapprentissage. Il en résulte un modèle ayant un procédé génératif complet, ce qui permet une inférence correcte sur des nouveaux documents.

Un problème que nous avons en tentant d'utiliser LDA pour modéliser le contexte est que celui-ci fait l'apprentissage de la mixture d'une seule multinomiale, quand nous avons plusieurs multinomiales dans notre système supervisé. Nous allons donc simplement fusionner tous les types d'attributs, et maximiser la vraisemblance de toutes les observations.

4.3.2.1 Procédé génératif

LDA postule qu'un document est composé d'une mixture d'un nombre fini de "sujets" (*topics*). Chacun de ces sujets est une distribution multinomiale de mots, et la mixture de ceux-ci suit une distribution Dirichlet. Pour générer un document, on a besoin des paramètres suivants :

- α est un vecteur des pseudo-comptes qui sont les hyperparamètres d'une distribution Dirichlet. On suppose habituellement que ces pseudo-comptes sont uniformes, et donc α est un seul nombre.
- β_{kw} est la probabilité du mot w dans le sujet k.
- On peut optionnellement ajouter du lissage à β en lui ajoutant un a priori Dirichlet contrôlé par un paramètre η.

Ayant ces deux informations, LDA propose qu'un document $d = \{w_{1..N}\}$ (ensemble de mot non-ordonnée) dans un corpus $\mathcal{C}$ est généré de la manière suivante :

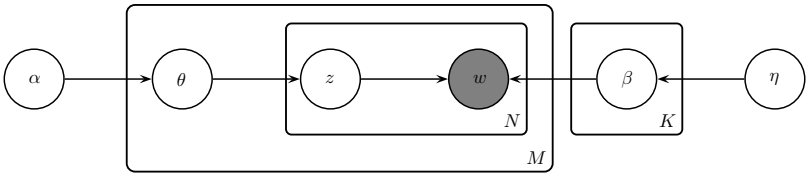


Figure 4.1 – Représentation graphique de LDA lissé

1. Choisir une longueur de document $N \sim$ Poisson(ξ)
2. Choisir une mixture globale de sujets pour le document : $\theta \sim$ Dir(α)
3. Pour chaque mot $n = 1..N$
   (a) Choisir un sujet pour le mot n : $z_n \sim$ Multinomiale(θ)
   (b) Choisir le mot w_n à partir de z_n et β : $w_n \sim p(\blacksquare \mid z_n, \beta)$

En pratique on peut ignorer la spécification de la longueur de document.

4.3.3 Points importants pour l'adaptation

Avant de discuter l'adaptation de LDA à nos fins, nous notons les points importants suivants :

- Un modèle LDA utilisant un a priori uniforme pour la distribution de sujets fera l'allocation afin que chaque sujet couvre une partie quasi-égale des contextes du mot. Le nombre de sujets associés à un sens sera ainsi proportionnel à sa fréquence. Comme le sens dominant d'un mot sera donc couvert par plus d'un sujet, il faudra plus de sujets que le nombre de sens réels. Il est intéressant de noter qu'un degré de finesse d'un sens lié à la quantité de données nous donne un inventaire de sens adaptatif. Ceci est à notre avis la solution idéale au problème de l'inventaire de sens. De plus, même si un mot n'est pas ambigu, il sera utilisé dans une variété de contextes qu'il peut être utile de modéliser. Notre adaptation ne sera donc pas exclusivement une application de DS, mais l'application générale des contextes des mots.
- Comme le modèle de RI calcule un score au niveau du document, toutes les observations d'un mot dans un document devraient être amalgamées. Ceci a l'avantage de nous donner plus d'informations pour l'inférence d'un document.
- Comme les utilisateurs de systèmes RI ne mettent que rarement des mots outils entre les mots clés, ceux-ci devraient avoir une plus petite importance dans les modèles de contexte que ce que nous avions dans les expériences supervisées. Le point le plus important est que si un mot-outil n'est pas présent dans une requête, nous ne devrions pas supposer qu'il devrait aussi être absent dans le document.

– Ceci pose la question de se demander à quel point une requête est différente d'un mot en contexte de document. En particulier, les mots du contexte de la requête suivent-ils la même relation qu'on observe dans les expériences ? Sur ce point, nous prenons l'hypothèse que quand la requête est courte, les mots clés sont fortement reliés, et quand la requête est longue, ceux-ci sont semblables aux mots en contexte des documents. Comme quand la requête est courte, les mots ne sont jamais loin, et ainsi les poids de la fonction de distance seront relativement forts. L'utilisation de la courbe de poids à la fois dans les documents et les requêtes serait donc acceptable.
– Les données étiquetées pour les expériences précédentes avaient l'effet de réduire automatiquement l'impact des attributs qui n'étaient pas liés au sens du mot. Par exemple, si un mot-outil n'est pas lié au sens d'un mot, avec assez d'échantillons, celui-ci aura des probabilités plus ou moins uniformes dans les modèles de langues pour chaque sens. Toutefois, si nous effectuons un apprentissage non-supervisé, des tendances non-importantes pourraient dominer les sujets. Par exemple, si notre liste de mots outils comprend 10 mots clés, nous aurons 40 états (10 x (avant/après) x (présent/absent)) et ainsi 20 observations par fenêtre (présent ou non), ce qui est environ quatre fois plus que le compte total des mots d'un contexte. Normalement, si ces mots n'aident pas à prédire les autres, ceux-ci ne devraient pas être différemment distribués selon les sujets, mais ceci reste à confirmer.

4.3.4 Apprentissage des sujets

La tâche effectuée par LDA est d'effectuer le procédé inverse de son procédé génératif, soit d'apprendre les valeurs de α et β à partir des documents observés. Pour ce faire, on maximise la probabilité à posteriori des données et des paramètres. Comme la probabilité d'un mot selon les sujets est une somme pondérée de ceux-ci, le problème n'a pas une forme conjuguée rendant possible une solution analytique. On doit donc utiliser une approche approximative pour calculer la probabilité des sujets $P(\theta|d)$ étant donné un document d. Plusieurs algorithmes existent à ces fins. Toutefois, comme nous utilisons des attributs à comptes réels, nous utiliserons une approche déterministe qu'on nomme

l'inférence variationnelle Bayesienne EM (VBEM).

4.3.4.1 Inférence variationnelle Bayesienne (VBEM)

Cette approche est celle proposée par Blei dans l'article initial de LDA. Elle consiste en un algorithme EM variationnel, dont l'étape E estime les probabilités postérieures de θ en maximisant la vraisemblance des données par l'entremise d'une classe de distribution simplifiée. L'algorithme de l'inférence variationnelle (algorithme 4.1) est très intuitif.

Algorithm 4.1 InferenceVariationnelle(c,α,β)

// c_i est le "compte" d'observations de l'attribut i dans le document
// ϕ_{ik} est l'assignation courante de l'attribut i au sujet k.
// γ_k est le nombre total de comptes affectés au sujet k, c'est un pseudo-compte qui exprime la distribution de $\theta \sim Dir(\gamma)$
$\phi_{ik} := \frac{1}{K}$ pour tous i, k
repeat
  $\gamma_k := \alpha_k + \sum_i c_i \phi_{ik}$ pour tous k
  for chaque attribut i observé dans le document **do**
    for $k = 1$ to K **do**
      $\phi_{ik} := \exp\left(\Psi\left(\lambda_{ki}\right) - \Psi\left(\sum_{i'} \lambda_{ki'}\right) + \Psi\left(\gamma_k\right)\right)$
    end for
    {*Normaliser ϕ_i pour qu'il somme à un*}
  end for
until convergence de la vraisemblance

return ϕ, γ

Comme on peut le voir, on commence avec une affectation uniforme des sujets. On alterne ensuite entre l'estimation de la mixture globale du document, et l'estimation des affectations des mots. L'affectation des attributs se fait par :

$$\phi_{ik} \propto \exp\left(\mathbb{E}\left[\log \beta_{ki} \mid \lambda_k\right]\right) \exp\left(\mathbb{E}\left[\log \theta_i \mid \gamma\right]\right)$$

$$\mathbb{E}\left[\log \beta_{ki} \mid \lambda_k\right] = \Psi(\lambda_{ki}) - \Psi(\sum_{i'} \lambda_{ki'})$$

$$\mathbb{E}\left[\log \theta_i \mid \gamma\right] = \Psi(\gamma_i) - \Psi(\sum_{i'} \gamma_{i'})$$

où $\Psi(x) = \log(\Gamma(x))' = \frac{\Gamma(x)'}{\Gamma(x)}$ est la fonction *digamma*, gamma (Γ) étant la généralisation à des nombres réels de la fonction factorielle. $\Psi(x)$ peut être estimé par une approximation polynomiale de Taylor. La preuve montrant que cette valeur maximise l'espérance est inclue dans l'article de Blei, mais on peut la comprendre comme l'espérance de la log-vraisemblance des allocations en fonction de la mixture de sujets. En regardant l'algorithme 4.1, on pourrait penser que cet algorithme est erronné car il ignore les probabilités de base des mots, et n'utilise uniquement que les différences de celles-ci dans les sujets. Il faut toutefois comprendre qu'une fois les probabilités converties en log-probabilité, la probabilité de base des mots, qui pourrait être exprimée par un facteur multipliant la probabilité du mot, peut être sortie du logarithme, et est ainsi une constante. Pour maximiser la vraisemblance d'un document, celle-ci n'est donc pas importante.

Maintenant concernant l'adaptation, l'algorithme 4.1 utilise une estimation lissée de β qui considère celui-ci comme une variable aléatoire (tout comme α). Le lissage est donc introduit par un a priori Dirichlet. Blei propose d'ajouter un a priori uniforme de force η à β, mais comme nous avons un meilleur a priori avec les probabilités de collection, nous utiliserons les valeurs suivantes :

$$\lambda_{kw} = \sum_{d \in \mathcal{D}} c_{dw} \phi_{dwk} + \begin{cases} \eta_{Cible} & \text{Si } w\text{est un mot cible} \\ \eta_{Outil} & \text{Si } w\text{est un mot outil} \\ \eta_{Con} P(w|\mathcal{C}) & \text{Sinon} \end{cases}$$

$$\mathbb{E}\left[\beta_{kw} \mid \lambda_{kw}\right] = \frac{\lambda_{kw}}{\sum_{w'} \lambda_{kw'}}$$

$$\mathbb{E}\left[\log \beta_{kw} \mid \lambda_{kw}\right] = \exp\left\{ \Psi(\lambda_{kw}) - \Psi(\sum_{w'} \lambda_{kw'}) \right\}$$

Ce qui donne ensuite l'équation de mise à jour :

$$\phi_{ik} \propto \exp\left\{ \left[\Psi(\lambda_{kw}) - \Psi(\sum_{w'} \lambda_{kw'}) \right] + \Psi(\gamma_i) \right\} \tag{4.1}$$

Dans notre cas, comme la somme des λ utilise des comptes de mots qu'on n'observe pas dans le modèle, il est utile de séparer les comptes de lissage des comptes assignés :

$$\begin{aligned}\sum_w \lambda_{kw} &= \left[\sum_w \sum_{d \in \mathcal{D}} c_{dw} \phi^{\star}_{dwk}\right] + \left[\, |V_{Cible}|\, \eta_{Cible} + |V_{Outil}|\, \eta_{Outil} + \eta_{Con} \right] \\ &= C_k + H\end{aligned}$$

$$\phi_{ik} \propto \exp\left\{[\Psi(\lambda_{kw}) - \Psi(C_k + H)] + \Psi(\gamma_i)\right\}$$

Les Ψ de gauche ne dépendent pas de l'inférence d'un document particulier, et peuvent donc être calculés une seule fois par phase M. Comme nous utilisons plusieurs a priori différents, nous n'utilisons pas d'estimation pour H, mais α peut être estimé par une méthode de Newton rapide (incluse dans le papier de Blei).

4.3.5 Construction des modèles

Nous commençons par supposer que lors de l'indexation des documents, on conserve les fenêtres de chaque occurrence de mot. Une fenêtre de taille limitée devrait être suffisante ; pour nos expériences nous avons limité les fenêtres à 20 mots de chaque côté. Ceci donne une utilisation d'espace disque qui est potentiellement 41 fois la taille de la collection de documents. On peut toutefois motiver cet espace par d'autres usages de ces fenêtres, par exemple afin de générer des résumés (*snippets*). Si on considère que le modèle de contexte est constant et calculé au préalable, il ne sera pas nécessaire de conserver les fenêtres après l'indexage.

Il est aussi utile de conserver des statistiques de fréquence des mots dans le contexte du mot cible. Lors de la conversion en format LDA, celles-ci permettront de filtrer le vocabulaire, fixant une borne inférieure du nombre d'occurrences (*cutout*), ou mettre une borne supérieure à la taille du vocabulaire des contextes. Pour nos expériences, nous avons utilisé une borne inférieure de 10 occurrences sur les attributs sans tenir compte de leur distance.

La conversion en comptes se fait avec les traitements suivants :

– Le *mot cible* est traité légèrement avec un retrait des majuscules et du "s" final si

présent. On s'attend à ce que l'utilisateur ne désire pas que de telles différences de formes aient un impact sur les résultats de recherche. Toutefois, il est possible que des améliorations additionnelles soient observées sans ce traitement (ex : chercher "aid" vs "AIDS"). Les mot cibles ont des identificateurs numériques différents des autres mots de contextes, même s'ils ont la même forme.

– Les *mots outils* ont les majuscules retirées, et sont différenciés selon qu'ils sont observés avant ou après le mot cible. Nous rappellerons que des mots outils consécutifs avant ou après le mot cible sont considérés comme indépendants, et sont ajoutés en ordre de distance jusqu'à ce qu'on observe un mot "de contenu" (non-objet). Quand on arrive à la fin d'un champ du document sans observer de mot de contenu, on ne peut pas déterminer que le mot est absent, et on n'ajoute donc pas d'attribut pour ce mot. Ceci peut se produire le plus souvent dans des champs courts tels que le titre. On pourrait aussi généraliser ceci aux requêtes, mais dans notre cas nous n'utiliserons pas l'absence des mots cibles dans les requêtes.
– Les *mots de contenu* sont convertis en racines avec le troncateur Porter, et sont ajoutés avec un poids en fonction de leur distance. Tout comme pour les expériences supervisées, les mots cibles consécutifs comptent pour une unité de distance. Pour modéliser la force de la relation entre les mots de contexte et le mot cible, nous utilisons une loi de puissance négative de 1.1.

Comme dans nos expériences de DS sur les données Semeval, on contrôle l'impact de chaque type d'attribut en multipliant leur log-prob. Ceci peut être fait dans LDA en multipliant tous les comptes par des facteurs ω^{LDA}_{Cible}, ω^{LDA}_{Outil}, ω^{LDA}_{Con}. En tant que lissage des sujets, on applique des a priori uniformes de η_{Cible} et η_{Outil} pour les mots cibles et les mots-outils. Pour les mots de contexte, on utilise les probabilités de collections

type d'attribut	Poids dans LDA ω^{LDA}	Lissage η	Poids dans la requête $\omega^{requête}$
mot cible	1.0	1.0 uniforme	1.0
mot outils	0.1	0.1 uniforme	0.1
mots de contenu (distance i)	$1.25 \cdot i^{-1.1}\mathbb{I}_{i\leq 20}$	1250 collection	$1.25 \cdot i^{-1.1}$

Tableau 4.1 – Poids des attributs pour les expériences de RI

dans a priori Dirichlet de force η_{Con}. Les valeurs de ces paramètres (tableau 4.1) sont identiques à celles utilisées dans les expériences en anglais sur les données Semeval, sauf pour les mots outils qui ont un poids réduit. En effet, nous observons qu'avec un poids de 1 comme dans les expériences, ces attributs compteraient pour la majorité des comptes du modèle, et ainsi auraient un beaucoup trop grand impact sur les modèles.

Nous utilisons une version modifiée de lda-c, l'implémentation de base de LDA faite par Blei. Ce système utilise un algorithme EM variationnel Bayesien (VBEM). Cet algorithme est connu pour être très lent car il nécessite le calcul de la fonction *digamma* (la dérivée du logarithme de la fonction gamma) une fois par sujet pour chaque phase M de l'inférence. Des alternatives plus efficaces faites pour le LDA traditionnel sont le *Collapsed Gibbs Sampling* (CGS) [41] et *le Collapsed Variational Bayes* (CVB) [60]. Toutefois, ces deux méthodes nécessitent le retrait du mot dont on recalcule l'allocation. Ceci se fait très rapidement quand toutes les occurrences de mots ont un compte uniforme car la réaffectation est identique pour toutes les occurrences. Toutefois comme dans notre cas nous avons des comptes flottants, il faudrait faire une réaffectation indépendante pour chaque occurrence, ce qui réduirait beaucoup l'avantage de performance de ces algorithmes. De plus, le calcul de digamma est beaucoup moins problématique dans notre cas comme nous utilisons beaucoup moins de sujets. Dans cette situation, VMEM pourrait bien être le plus rapide des algorithmes disponibles.

Pour nos expériences, nous utilisons les paramètres suivants : 10 sujets, $\alpha = 0.1$ initialement, mais nous utilisons l'évaluation automatique. Les critères de convergence par défaut sont utilisés. Nous avons utilisé l'approche standard d'initialisation des sujets avec 50 documents sélectionnés aléatoirement (*seeded*). Cette initialisation a pour objectif d'établir une asymétrie de départ pour les sujets, ceux-ci vont ensuite s'ajuster pour suivre les tendances principales des contextes. Idéalement, ceux-ci ne devraient pas avoir d'impact sur les sujets finaux, mais des fois ceux-ci peuvent mener à des optimum locaux différents. Comme nous avons beaucoup de modèles à construire, nous n'avons pas fait plusieurs passes de sélection de ces initialisations.

4.3.5.1 Intégration dans le score

Ayant construit un modèle de contexte pour chaque mot w de la collection, nous disposons maintenant des informations suivantes :

- Un ensemble de sujets $Z_{w,k}, k = 1..K$ représentés par une matrice β_w.
- Les allocations de sujets pour chaque document. Celles-ci sont représentées par $\gamma_{w,d,k}, k = 1..K$ qui sont les paramètres d'une Dirichlet exprimant l'a posteriori $\theta_{w,d}$ des sujets pour le contexte du document d. Si un document ne contient pas le mot w, ceux-ci sont identiques à α (et donc uniformes). Nous pouvons donc obtenir la probabilité d'un attribut de contexte x du mot w pour un document d par :

$$p(x|w,d) = \sum_k p(x|Z_{w,k})p(Z_{w,k}|d)$$

$$P(Z_{w,k}|d) = \begin{cases} \frac{\gamma_{w,d,k}}{\sum_{k'} \gamma_{w,d,k'}} & w \in d \\ 1/K & \text{sinon} \end{cases}$$

$$P(x|Z_{w,k}) = \beta_{w,k,x}$$

Maintenant, soit une requête $Q = \{q_1, q_2..., q_n\}$. Pour cette requête nous faisons l'extraction des attributs de contexte de chaque mots comme nous l'avons fait pour les documents. Chaque mot clé a donc un ensemble d'attributs $F_q = \{f_1, f_2, ..., f_m\}$ et chacun d'entre eux ont un poids :

$$\omega_f^{requête} = \begin{cases} \omega_{Cible}^{requête} & \text{si } f\text{est une occurence de mot cible} \\ \omega_{Outil}^{requête} & \text{si } f\text{est une occurrence de mot outil} \\ \omega_{Con,dist}^{requête} & \text{si } f\text{est un mot de contenu. } dist\text{est la distance à } q_i \\ 0 & \text{sinon (hors vocabulaire)} \end{cases}$$

Le score du contexte d'un mot de requête q pour un document d est ainsi :

$$\log p(F_q|w,d) = \sum_{f \in F_q} \omega_f^{requête} \log p(f|q,d)$$

Ce score peut être combiné avec une moyenne géométrique pondérée :

$$score(Q|d) = \sum_{q \in Q} [(1 - \lambda) \log p(q|d) + \lambda \log p(F_q|q, d)]$$

où λ contrôle l'impact qu'a le modèle de contexte.

4.3.6 Complexité

Un problème majeur rencontré lors de l'utilisation de modèles de sujets est le temps de calcul, et surtout la taille des modèles. Les outils comme LDA ne sont pas utilisés dans des applications à grande échelle telle que la recherche Web justement à cause de telles considérations. Notre approche contourne toutefois beaucoup de ces problèmes :

La taille des modèles est prohibitive Comme nos modèles sont faits indépendamment pour chaque mot, ils peuvent être créés au besoin selon l'ambiguité du mot ou sa popularité dans les requêtes. On peut aussi assumer qu'un vocabulaire limité est suffisant pour modéliser un mot. Les modèles pourront ainsi avoir une taille réduite, et pourront être contenus sur une seule machine. De plus, nous pouvons envisager que les sujets puissent être correctement modélisés avec une partie des données. Par exemple, on pourrait mettre une borne supérieure sur le nombre de documents utilisés pour le calcul des sujets, et inférer une seule fois les postérieurs pour le reste puis à chaque nouvelle entrée d'index. Les modèles ne sont donc pas problématiques quant au temps de calcul. Finalement, si on conserve les fenêtres des mots, on peut aussi mettre le modèle d'un mot à jour au besoin, selon la vraisemblance des nouvelles entrées d'index.

L'évaluation des requêtes est très lent Les modèles LDA ordinaires appliqués à la RI utilisent des centaines de sujets (800 dans [65]), et ainsi pour évaluer la probabilité d'un mot dans un document il faut faire le produit scalaire entre les probabilités dans les sujets et la mixture de sujets du document. Nos modèles de contextes utilisent moins de sujets, donc l'évaluation d'un attribut est plus rapide. Comparé au temps nécessaire pour accéder à l'index sur disque, ce calcul est suffisamment

performant et de plus, il peut être fait en parallèle.

L'espace disque nécessaire est immense L'index résultant de notre approche contient le compte du mot dans le document et les pseudo-comptes des sujets. Nous jugeons qu'un tel index est acceptable, étant donné que dans la pratique on utilise de bien plus grands index tels que les index positionnels. Les données de fenêtres utilisées pour la génération des modèles sont toutefois très grandes, mais celles-ci ne sont pas utilisées au moment de l'évaluation des requêtes. On peut de plus les utiliser à d'autres fins, comme la génération de *snippets*.

4.4 Expériences et résultats

4.4.1 Données de test

Nos expériences portent sur trois collections de test de la conférence TREC.

La première collection, *ap*, consiste en les sujets 1-150 de la tâche *ad-hoc* de TREC 1-3 sur la collection *Associated Press* 1988-1989 (AP88-89). C'est une collection d'articles de journaux de très petite envergure. Des résultats sur cette collection sont très souvent reportés étant donné que c'est une des premières collections de TREC. Il faut toutefois prendre les résultats sur cette collection avec un grain de sel, car il est dit que celle-ci comporte des problèmes de qualité dans les jugements de pertinence.

La deuxième collection, *robust-04*, est l'ensemble des sujets de la tâche *robust* de TREC 2004 sur la collection constituée des CDs TREC volume 4 et 5 sans les *Congressional Records*. C'est une collection de taille moyenne, contenant 528 155 documents, pour une taille de 1904 MO. Les sujets utilisés sont les sujets 301-450 de la tâche ad-hoc de TREC 6–8 et les sujets 601-700 faits pour *robust* 2003 et 2004. Ceci donne un ensemble de 249 sujets (un sujet n'a pas de document pertinent), ce qui permet d'évaluer solidement la stabilité de la recherche. La tâche met l'accent sur une efficacité constante sur l'ensemble des résultats, et propose ainsi l'usage du $GMAP$ (avec $\epsilon = 0.00001$).

La troisième collection, *wt10g,* utilise les données de la tâche *web* de TREC 2000 et 2001 sujets 451-550 sur la collection WT10G. Cette collection est un échantillon du Web, et a ainsi un contenu plus varié (en forme et fond). La collection contient 1.69

millions de pages en anglais, pour un total de 10 GO, ce qui, à l'époque, était considéré comme une collection de grande taille. Les sujets ont la particularité d'avoir un champ titre extrait de vraies requêtes d'un engin commercial, qui sont ensuite complétés (description et narration) par les organisateurs.

4.4.2 Paramètres

L'évaluation commence par un système de base utilisant la *vraisemblance de requête* (*query likelihood – ql*). Ce score de base utilise un lissage Dirichlet. Les expériences passées montrent qu'un pseudo-compte entre 2000 et 3000 donne habituellement les meilleurs résultats [70]. Nous utilisons donc une valeur intermédiaire de $\mu = 2500$. Pour les trois collections, nous utilisons uniquement le champ *titre*. Nous appliquons l'anti-dictionnaire standard utilisé dans *Lucene*, mais les mots qui ne sont pas utilisés dans la recherche seront tout de même utilisés comme attributs de contexte.

Notre système avec modèles de contexte (*ql+context*) utilise les mêmes scores de base. Les poids donnés aux attributs de contexte sont les mêmes que ceux utilisés dans les évaluations Semeval (voir le tableau 4.1). Le poids λ qui contrôle l'impact des attributs de contexte est sélectionné par validation croisée, à la valeur qui maximise le MAP des autres requêtes de la collection. Nous utilisons $K = 10$ sujets avec une valeur initiale de $\alpha = 0.1$ avec l'estimation automatique à chaque itération. Nous utilisons les paramètres de convergence par défaut de lda-c.

4.4.3 Résultats

Les résultats de recherche sont listés dans le tableau 4.2. Nous utilisons les mesures populaires comme le MAP, $nDCG$ et $P@10$, et aussi des versions utilisant la moyenne géométrique $GMAP$ et $GnDCG$. Nous n'incluons pas une version géométrique de $P@10$ car c'est une mesure instable.

Comme on peut le voir, on observe des gains considérables sur toutes les collections. Comme plus de gains sont faits sur les moyennes géométriques que sur les moyennes arithmétiques, on comprend que plus de gains sont faits sur les requêtes difficiles qui,

Collection	MAP	GMAP	nDCG	GnDCG	P@10	GP@10
ap ql	0.2065	0.0832	0.4523	0.3232	0.4120	0.0548
ap ql+context	$0.2300^{\ddagger}$ (+11.37%)	$0.1037^{\ddagger}$ (+24.30%)	$0.4831^{\ddagger}$ (+6.80%)	$0.3593^{\ddagger}$ (+11.17%)	$0.4370^{\star}$ (+6.67%)	0.0831 (+51.61%)
robust-04 ql	0.2509	0.1415	0.5251	0.4549	0.4249	0.1387
robust-04 ql+context	$0.2782^{\ddagger}$ (+10.88%)	$0.1654^{\ddagger}$ (+16.92%)	0.5541 (+5.52%)	$0.4849^{\ddagger}$ (+6.58%)	$0.4502^{\ddagger}$ (+5.95%)	$0.1705^{\star}$ (+22.87%)
wt10g ql	0.1911	0.0775	0.4597	0.3112	0.2707	0
wt10g ql+context	$0.2057^{\dagger}$ (+7.64%)	$0.0874^{\star}$ (+12.77%)	$0.4726^{\star}$ (+2.80%)	$0.3520^{\star}$ (+13.11%)	$0.2879^{\star}$ (+6.35%)	$0^{\star}$ (+0%)
wt10g-noMeta ql	0.1924	0.0802	0.4623	0.3155	0.2768	0.0285
wt10g-noMeta ql+context	$0.2123^{\ddagger}$ (+10.34%)	$0.0934^{\dagger}$ (+16.55%)	$0.4829^{\ddagger}$ (+4.46%)	$0.3641^{\dagger}$ (+15.39%)	$0.2949^{\dagger}$ (+6.57%)	$0.0278^{\dagger}$ (−2.60%)

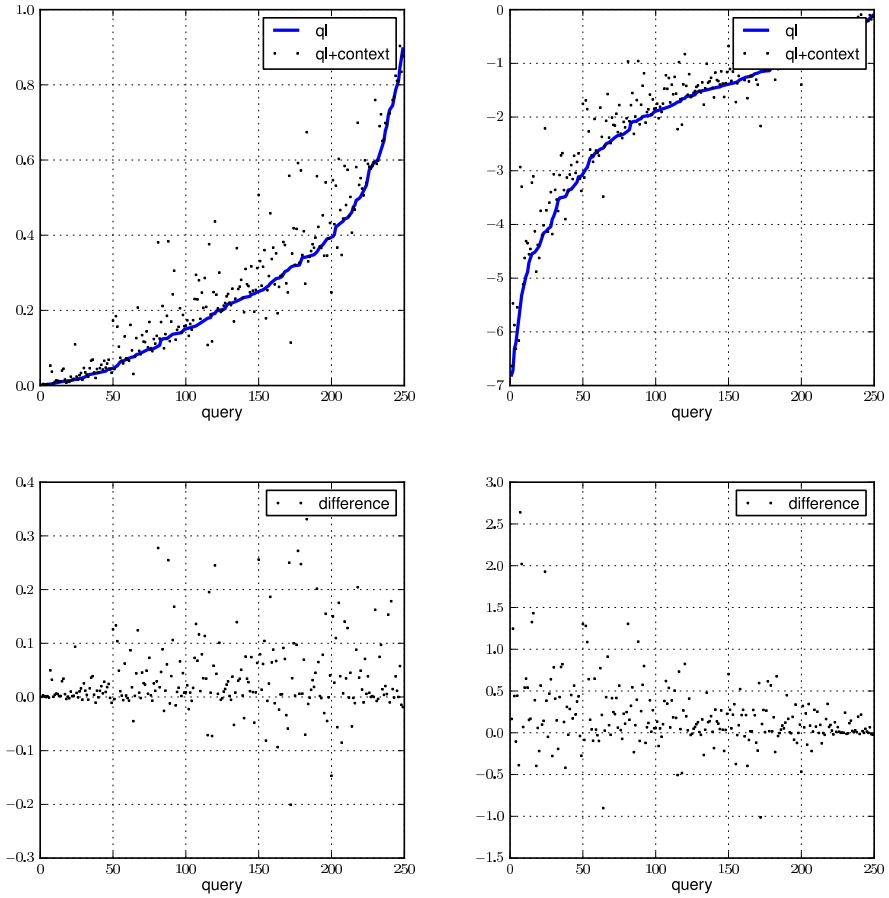
p-valeurs : $^{\star} < 0.05$, $^{\dagger} < 0.01$, $^{\ddagger} < 0.001$ (tests de ré-échantillonage aléatoires à 2 directions, 20M passes)

Tableau 4.2 – Résultats de RI avec et sans modèles de contexte

comme elles ont un bas score, ont plus d'effet sur les moyennes géométriques. On peut confirmer ceci en observant les différences sur l'échelle logarithmique (figure 4.2). La technique aide donc à rendre possible des requêtes difficiles et augmente la robustesse de la recherche. C'est une conclusion à laquelle on pouvait s'attendre lors de la conception de ce système. Les améliorations sont comparables avec celles observées en utilisant un modèle LDA ordinaire [65], quoi que notre système ait un effet bien différent. On pourrait aussi s'attendre à ce que notre système ait de plus grandes améliorations que le modèle de Xing et Croft en ajoutant un expansion de requête ou du feedback de pertinence.

On constate toutefois des gains moindres sur wt10g. En investigant cette différence, nous avons constaté que ceci est dû à la méthode de création des sujets 451-550. Ceux-ci sont extraits de logs d'un engin commercial, et les organisateurs ont imaginé un besoin d'information à partir de ceux-ci. Ceci soulève certaines considérations qui s'appliquent à la pratique réelle des systèmes de RI :

Méta mots clés. Les requêtes contiennent parfois des mots qu'on ne désire pas trouver dans les documents. Par exemple, dans les requêtes 451-550 on trouve des requêtes telles que : "Where can I find information about kappa alpha psi ?", "Where can I



(axes x : sujets en ordre de $AveP$ du système ql, y : AveP échelle linéaire (gauche), logarithmique(droite))

Figure 4.2 – $AveP$ et différences entre le système ql et ql-context sur la collection robust-04

find growth rates for pine trees ?", "Where can I find information on the decade of the 1920's", etc. Des mots tels que "Where can I find information" sont des méta mots clés que le système devrait filtrer, et idéalement utiliser pour comprendre l'intention de l'utilisateur. Si on ne les filtre pas, on peut s'attendre à des effets négatifs sur les résultats de recherche, étant donné que ceux-ci font dériver le "sens" de la requête.

Fautes de frappes et d'orthographe. Les requêtes contiennent souvent des fautes de frappes ou d'orthographe. Par exemple : "Whan did Jackie Robinson appear at his

first game", "natifityscenes", "angyoplast7". De telles erreurs causent souvent des mots improbables, et ainsi l'approche habituelle est de proposer des corrections à l'utilisateur, augmentant la probabilité des mots clés.

Besoins d'informations mal formulés Il est souvent difficile pour les utilisateurs de former leurs requêtes avec quelques mots clés. Ainsi, il arrive que les formulations aient une connotation différente du besoin d'information de l'utilisateur. Dans les requêtes 451-550, nous trouvons des exemples de requêtes et descriptions telles que "orchids : How big is the orchid growing industry in the U.S. ?", "Jennifer Aniston : Find documents that identify movies and/or television programs that Jennifer Aniston has appeared in.". Comme on peut le voir, les besoins d'informations se penchent sur un aspect des requêtes qui est différent de ce qu'on pourrait s'attendre.

Comme dans cette étude nous faisons des expériences dans un contexte simplifié, nous n'avons pas de module pour traiter ces problèmes. Toutefois, nous avons peu de requêtes et ainsi il est possible de les corriger manuellement (tableau 4.3). Nous avons procédé à une seconde évaluation des requêtes de la collection *wt10g*, que nous nommons *wt10g-noMeta*. Dans cette évaluation, nous avons corrigé le premier type d'erreur en enlevant les méta-mots clés. Nous constatons une augmentation de l'efficacité sur le système de base, mais surtout le système avec contexte. Ceci montre que le système avec contexte est sensible à une mauvaise formulation de la requête.

On peut aussi regarder les graphes de précision à 11 points de rappel (figure 4.3). Comme on peut le voir, la précision est plus haute à tous les points de rappel, ce qui indique que les améliorations sont stables. On confirme aussi que le rappel à 1.0 n'est pas augmenté. Ceci est normal étant donné que notre technique n'utilise que les mots de la requête, sans faire une expansion de requête.

4.4.3.1 Différences par requête

Dans les études sur les applications de DS en RI, on prévoit qu'on pourrait avoir une amélioration de quelques degrés de MAP. C'est en effet le cas, avec une amélio-

~~What is a~~ Bengals cat
~~do~~ beavers live in salt water
wha~~e~~n did Jackie Robinson appear at his first game
nativity_scenes
~~information about~~ the Peer Gynt Suite
~~what is~~ the compostion of zirconium
~~where can I find information about~~ kappa alpha psi
~~what did~~ babe ruth do in the 1920's
~~where can i find~~ growth rates for the pine tree
~~where is~~ the Eldorado Casino in Reno
angioplast~~7~~y
~~Where can I find information on the~~ decade of the 1920's

Tableau 4.3 – Corrections sur les sujets 451-550

ration absolue entre 2 et 3% sur nos collections de test. Toutefois, on peut s'attendre à ce que les gains ne soient pas uniformes. Un moyen de visualiser ceci est de faire un graphe des différences de précision entre le système de base et celui utilisant les modèles de contextes. Dans le tableau 4.4, nous avons mis les différences des requêtes en ordre croissant. Le graphe de gauche est dans l'échelle uniforme utilisée dans la moyenne arithmétique du MAP, le graphe de droite montre les différences dans l'échelle logarithmique utilisée dans la moyenne géométrique du $GMAP$. Comme on peut le voir, il y a un effet positif tangible pour environ 60% des requêtes, et un effet négatif sur environ 20%. On voit que les effets sont exponentiellement plus grands aux deux extrémités.

On peut aussi se demander quels sont les sujets pour lesquels la précision varie beaucoup. Dans les tableaux 4.4, 4.5 et 4.6, nous avons listé les cinq plus grandes différences positives et négatives pour les trois collections.

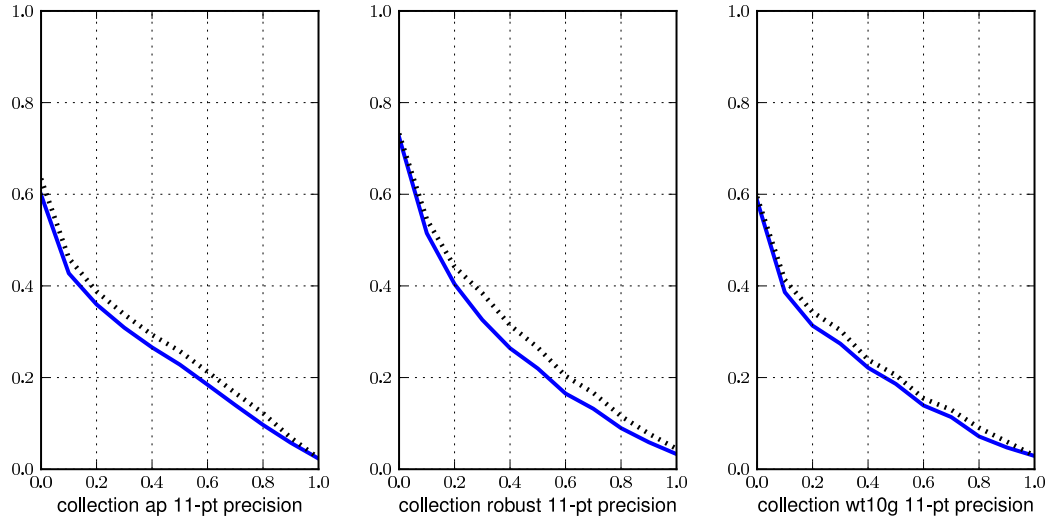
On voit que les plus grand impacts positifs contiennent tous au moins un mot ambigu dans une expression qu'on pourrait décrire comme étant une collocation. Nous pouvions nous attendre à ceci étant donné qu'on utilise le score de base favorisant la présence des mots, et que le score de contexte peut renforcer cet effet. Ainsi l'effet sera peu abstrait pour les plus grandes améliorations. Il est toutefois plus difficile d'expliquer les effets négatifs. En regardant les listes de résultats, on constate que les résultats sont logiques étant donné les titres, mais que ceux-ci sont moins variés. Le système aurait ainsi l'effet inverse de diversifier les résultats de recherche, et augmente les problèmes de non-

	titre		titre
-0.1497	Hostage Taking	-1.562	Find Innovative Companies
-0.1412	Military Coups Detat	-0.887	Demographic Shifts across National Boundaries
-0.1006	Negotiating an End to the Nicaraguan Civil War	-0.824	Hostage Taking
-0.0611	Official Corruption	-0.678	Military Coups Detat
-0.0337	US Budget Deficit	-0.522	Alternative renewable Energy Plant Equipment Installation
	...		...
0.1684	US Political Campaign Financing	1.458	US USSR Arms Control Agreements
0.2149	Attempts to Revive the SALT II Treaty	3.404	Purchasers of Modern Communications Equipment
0.2314	Dumping Charges	3.645	Marketing of Agrochemicals
0.2463	What Backing Does the National Rifle Association Have	4.111	US Army Acquisition of Advanced Weapons Systems
0.3112	Mitsubishi Heavy Industries Ltd	6.509	Black Monday

Tableau 4.4 – Sujets ayant les plus grandes différences de précision sur la collection ap dans l'échelle linéaire (gauche) et l'échelle logarithmique (droite)

	titre		titre
-0.2007	cruise health safety	-1.013	cruise health safety
-0.1467	Turkey Iraq water	-0.902	transportation tunnel disasters
-0.0934	Literary Journalistic Plagiarism	-0.506	Flavr Savr tomato
-0.0850	quilts income	-0.482	obesity medical treatment
-0.0812	New Hydroelectric Projects	-0.465	Turkey Iraq water
	...		...
0.2547	territorial waters dispute	1.325	mainstreaming
0.2558	term limits	1.431	family planning aid
0.2721	lead poisoning children	1.929	price fixing
0.2775	gasoline tax US	2.020	Russian food crisis
0.3311	Industrial Espionage	2.638	World Court

Tableau 4.5 – Sujets ayant les plus grandes différences de précision sur la collection robust-04 dans l'échelle linéaire (gauche) et l'échelle logarithmique (droite)



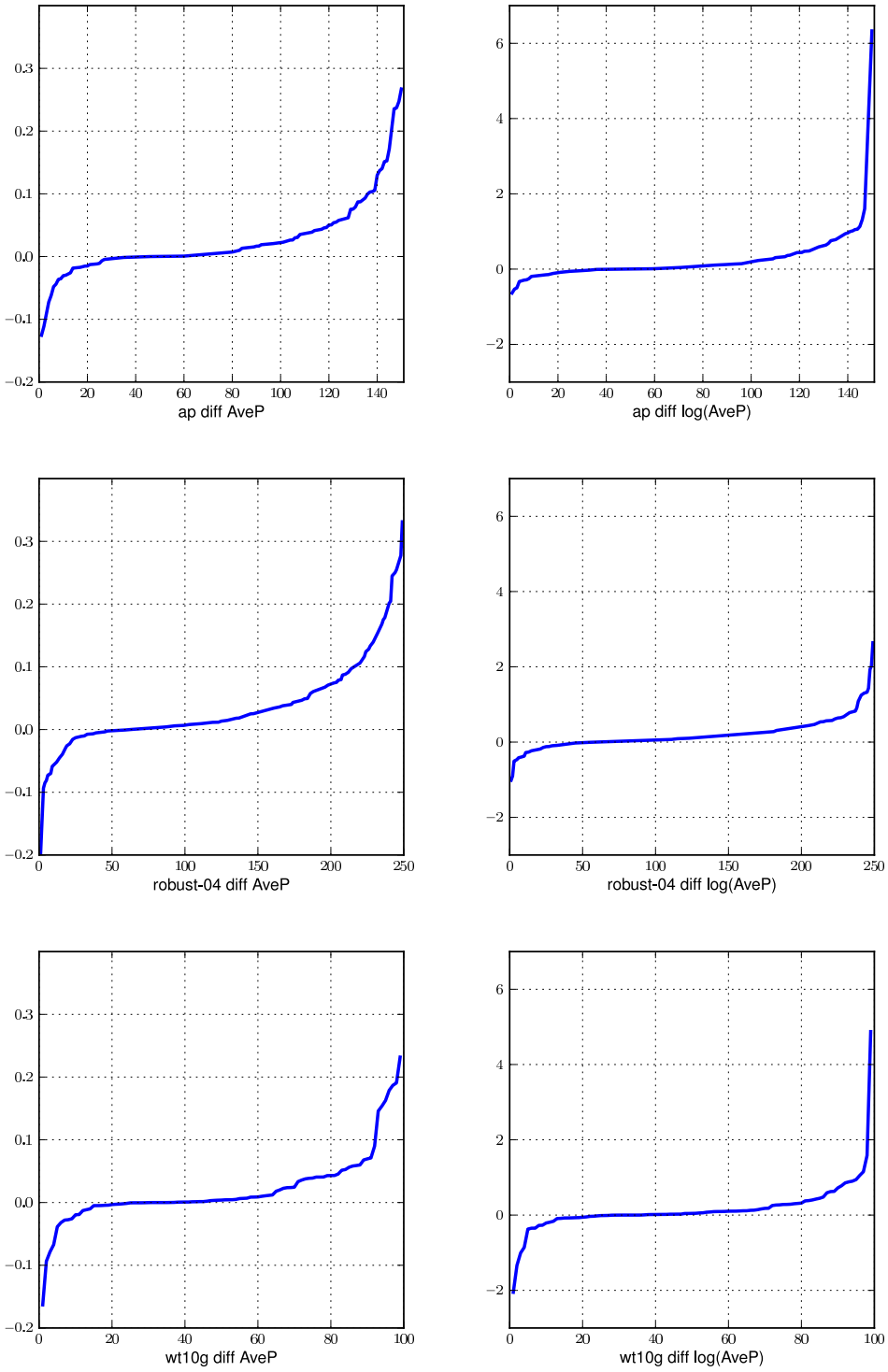
(axes x : rappel, y : précision interpolée)

Figure 4.3 – Précisions à 11 points de rappel pour les collections de test (ligne : ql, pointillé : ql+context)

concordance entre la formulation et le besoin d'information. Par exemple, pour "Hostage Taking", les résultats concernent principalement des prises d'otages dans des prisons, ce qui diffère du besoin d'information qui voulait des prises d'otages pour des fins politiques. Même chose pour "pool cue", qui propose principalement des sites de compagnies, quand on voudrait plutôt des explications historiques. On peut donc conclure qu'une définition correcte du besoin d'information, avec une bonne connotation, est très importante.

4.4.4 Discussion

On constate des améliorations plutôt consistantes en utilisant les modèles de contextes. Il y a toutefois des améliorations qu'on pourrait envisager pour avoir des améliorations additionnelles.



(axes x : requête en ordre de différence, y : différence en précision)

Figure 4.4 – Différences en AveP (gauche) et en log(AveP) (droite) sur la collection ap (haut), robust-04 (centre) et wt10g (bas)

	titre		titre
-0.1631	pool cue	-2.0513	how to erase scar
-0.0936	how to erase scar	-1.3368	retire
-0.0786	does stress cause obesity	-1.0074	does stress cause obesity
-0.0672	how is cancer related to cell reproduction	-0.8575	Royal Carribean Cruise Lines
-0.0386	How does a hygrometer measure the humidity in the atmosphere	-0.3717	radiography what are the risks
	...		...
0.1627	how is water supplied to the mojave desert region	0.9434	Steroids what does it do to your body
0.1788	booker t washington	1.0538	are sun beds safe
0.1868	edmund hillary sir	1.1517	fasting
0.1907	us savings bonds	1.5833	how e mails bennefits businesses
0.2318	j robert oppenheimer	4.8752	Toronto FIlm Awards

Tableau 4.6 – Sujets ayant les plus grandes différences de précision sur la collection wt10g dans l'échelle linéaire (gauche) et l'échelle logarithmique (droite)

4.4.4.1 Qualité des modèles

Les modèles que nous utilisons sont très simples, probablement trop. Tout d'abord, nous n'avons pas utilisé des distinctions de champ. Les documents contiennent souvent des champs titres, corps, et on pourrait supposer que ceux-ci forment des fenêtres séparées avec des importances différentes. Nos modèles ont concaténé les titres et le corps des documents sans faire cette distinction.

Nos modèles de contexte sont construits avec une implémentation basée sur lda-c. Cet outil utilise l'inférence variationnelle Bayesienne EM (VBEM) proposée dans la publication initiale de Blei. Grâce au succès de LDA, d'autres stratégies d'inférences ont été développées depuis. En effet, de nos jours on utilise plutôt des méthodes comme l'échantillonnage Gibbs écroulé (*Collapsed Gibbs Sampling* – CGS) [41] et l'inférence variationnelle Bayesienne écroulée (*Collapsed Variational Bayes* – CVB) [60]. Bien que ces algorithmes perdraient beaucoup de leurs avantages de performance sur nos données, ceux-ci donneraient des meilleurs sujets, et comme ils sont des algorithmes itératifs, ils convergent beaucoup plus rapidement. Pour accélérer notre système, on pourrait utiliser une adaptation de VBEM comme l'inférence variationnelle stochastique [30], qui permettrait d'avoir des bons sujets rapidement pour les premières itérations.

Un deuxième point d'amélioration possible pour le nouveau système est de réduire

l'indépendance des sujets. Après la publication de LDA, des extensions réduisant l'indépendance des sujets ont été proposées. Les modèles de sujets corrélés (*Correlated Topic Models* – CTM) [3] modélisent les corrélations des sujets en remplaçant la Dirichlet par une Gaussienne logistique, et utilisent une matrice de covariance pour modéliser celle-ci. L'allocation Pachinko (*Pachinko Allocation* – PAL) [27] propose un graphe acyclique unidirectionnel (DAG) à plusieurs couches. L'inférence passe donc par des noeuds internes, qui permettent d'ajouter des dépendances entre les sujets. Dans ces deux approches, les dépendances entre les sujets permettent d'avoir des modèles à basse perplexité avec moins de sujets. Comme dans notre approche nous voulons utiliser très peu de sujets, il est important que ceux-ci résument bien le contextes des mots. Ces méthodes seraient donc plus adaptées à nos fins que LDA.

Bien que ces variations de LDA puissent améliorer la qualité des modèles, ça ne change pas le fait que LDA est fait pour utiliser une grande quantité de sujets. On cherche ainsi dans ces modèles à réduire les dépendances entre les sujets, et même pour CTM et PAL, on évite de faire des calculs utilisant des couples de sujets (comme inverser des Hessiennes de covariance pour CTM). Comme nos modèles utilisent peu de sujets, plutôt que de faire de telles simplifications, il serait mieux de créer un nouveau modèle latent, mettant plus d'importance sur les interractions entre les sujets.

4.4.4.2 Compenser le manque d'information des requêtes

Un problème fondamental de toutes les approches de DS pour la RI est le peu d'informations que nous donne la requête. À ces fins, nous avons quelques propositions d'améliorations.

Un premier point d'amélioration serait d'estimer les poids des attributs de contexte. Dans l'implémentation que nous avons utilisée, nous utilisions un poids constant global dans la proportion des poids trouvés par expériences Semeval. Ceux-ci sont appropriés pour le contexte des documents, mais c'est discutable pour les requêtes. On pourrait ainsi envisager qu'utiliser des méthodes d'apprentissage machine pour apprendre des poids en fonction de requêtes pourrait résulter en des améliorations additionnelles. Avec les collections que nous utilisions, nous n'avions pas assez d'informations pour apprendre

de tels poids, mais ce serait possible en utilisant des logs d'usagers. Les informations contenues dans les modèles de contexte seraient une bonne source d'attributs de classification pour déterminer si les documents sont pertinents ou non. Nous avons fait une petite expérience à ces fins. Avec les modèles utilisés dans ce chapitre, nous observons qu'avec une valeur de contrôle λ oracle, on arrive à un MAP de 27% sur la collection *ap*, soit le double des améliorations observées avec un λ constant. Cette borne théorique n'est toutefois pas réaliste étant donné qu'il est normal que certains des sujets baissent de précision étant donné leur besoin d'information mal formulé. Ça montre toutefois qu'on pourrait avoir plus de gains en estimant les poids de contexte en fonction des requêtes.

Un deuxième moyen de compenser le manque d'information serait de permettre à l'utilisateur d'introduire des *mots clés de contexte*. Ceux-ci seraient inclus dans les attributs de contexte, mais sans être des mots clés recherchés. De tels mots clés pourraient être ajoutés au besoin quand les requêtes ont des problèmes sémantiques, et n'auraient pas l'effet de dérive de la requête qui se produit quand ajoute des mots clés dans une requête. On pourrait ainsi réduire le besoin de l'utilisateur de reformuler ses requêtes, tout en rendant possible des requêtes plus précises. L'utilisateur pourrait être assisté en la sélection de tels mots en affichant des mots représentatifs des sujets des mots clés. En les acceptant ou les refusant, on pourrait effectuer ce qu'on pourrait appeler un *feedback sémantique*. Comme on a déjà les modèles de sujets et des méthodes pour inférer des mixture à partir du contexte, de tels algorithmes pourraient être facilement implémentés.

4.4.4.3 Vers des applications d'expansion de requêtes

Comme nous l'avons vu dans les résultats d'expérience, un des problèmes de notre approche est qu'étant donné qu'on utilise les même comptes qu'un index ordinaire, elle n'augmente pas le rappel. À ces fins, bien qu'il soit possible d'appliquer un feedback aveugle pour augmenter le MAP, ces techniques ont tendance à n'améliorer que les requêtes faciles, et réduire la stabilité des résultats. L'approche que nous proposons ayant l'objectif contraire, il serait plutôt préférable de l'inclure dans des techniques d'expansion de requête. C'est en effet ce qui est fait par la plupart des systèmes participants aux tâches *robust* de TREC. Comme nous avons déjà les sujets et des méthodes d'infé-

rences, il faudrait simplement inférer les mixtures pour chaque mot clé en fonction de ses attributs de contexte dans la requête, puis fusionner les sujets résultants pour des probabilités de mots d'expansion. D'une manière plus agressive, il serait possible d'utiliser les modèles de contexte d'un mot pour inférer les autres, et ainsi compenser le manque d'information. Ceci aurait le double avantage d'augmenter le rappel, et aussi d'enlever le besoin de l'index de sujets additionnel. Les modèles de sujets seraient ainsi libres d'être mis à jour continuellement par des méthodes stochastiques, et pourraient aussi être faits à la fois à partir des documents et les requêtes. Tout comme discuté dans le paragraphe précédent, on pourrait proposer des mots de contexte, menant à un *feedback d'expansion*. Les mots acceptés ou refusés pourraient servir pour corriger les inférences.

4.5 Conclusion

Dans ce chapitre nous avons vu une adaptation du système de désambiguisation de sens que nous avons développé pour des tâches supervisées de Semeval. Afin de satisfaire les contraintes imposées par des applications en Recherche d'Information, nous utilisons une réduction de dimensionalité basée sur l'allocation Dirichlet latente. Construisant un modèle de contexte par mots, nous obtenons un index où on ajoute les mixtures de sujets pour une paire mot/document. Ceci contourne beaucoup des problèmes associés à l'application de modèles de sujets à des collections de grande taille. En combinant le score de vraisemblance de requête habituel avec un score de contexte, on observe des augmentations comparables ou meilleures que les applications LDA ordinaires, avec des gains relatifs du MAP de plus de 10%, et de plus de 20% du $GMAP$. On constate que les gains sont d'une nature différente : quand les applications LDA effectuent l'équivalent d'une expansion de document, notre approche a un effet contraignant additionnel qu'on pourrait nommer une *constriction de requête*. Celles-ci mènent à des améliorations touchant surtout les requêtes difficiles, améliorant la stabilité de recherche. Bien que LDA soit un modèle peu approprié pour modéliser une mixture de différent types d'attributs, la méthode montre beaucoup de potentiel pour rendre possible des requêtes très difficiles ou précises.

Les résultats de cette recherche ouvrent beaucoup d'avenues de recherche. Tout d'abord, pour compenser le manque d'informations de la requête, il serait possible d'intégrer une deuxième requête de mots de contexte qui ne seraient pas cherchés directement. Ceci permettrait de corriger des problèmes sémantiques des mots clés, et aussi la connotation de l'information désirée. La création de cette requête de contexte pourrait aussi être assistée en présentant des mots représentatifs des sujets des mots de la requête. Celle-ci aiderait l'utilisateur à comprendre les problèmes sémantiques de sa requête, et en acceptant ou refusant des mots, l'utilisateur pourrait intégrer un *feedback sémantique*.

Une avenue alternative serait d'utiliser les modèles de contexte afin de faire une expansion de requête. On peut en effet voir les modèles de contexte comme une distribution conditionnelle variant en fonction de la mixture des sujets. Nous pourrions ainsi les utiliser pour avoir une expansion adaptée, inférant mutuellement les sujets des mots clés pour avoir des mots d'expansion qui cooccurrent avec l'ensemble des mots, plutôt qu'additionner les probabilités de contexte de chaque mot.

CHAPITRE 5

CONCLUSION

Dans ce mémoire nous avons tout d'abord fait une investigation des tendances de la relation entre un mot et un autre mot qui se trouve dans ses alentours. On pourrait s'attendre à ce que la force de cette relation suive une fonction de la distance du mot, mais il n'est pas évident d'établir quelle fonction est optimale. Il serait aussi envisageable que des langues différentes aient des fonctions différentes, donc nous ne pouvons pas spécifier une fonction manuellement. Nous avons donc proposé d'apprendre automatiquement des poids en fonction de la distance entre deux mots, et de l'utiliser pour faire des modèles de contextes simples composés d'un modèle de langue unigramme. Pour apprendre la fonction de poids nous faisons l'hypothèse suivante : si on assume qu'un modèle de contexte est une représentation du sens du mot, alors des modèles de contexte de mots qui sont synonymes devraient être similaires. Nous utilisons ensuite le fait qu'un mot est synonyme avec lui-même, ou du moins a la même mixture de sens, et établissons ainsi que la fonction de poids idéale est celle qui maximise la similarité entre des modèles de contextes faits d'échantillons aléatoires d'un même mot.

Pour apprendre cette fonction, nous commençons avec une fonction uniforme et effectuons une descente de gradient stochastique, minimisant l'entropie croisée entre deux modèles aléatoires pour un mot. En expérimentant en anglais et en japonais, nous observons que les fonctions résultantes sont différentes d'une langue à l'autre, mais sont toutes deux très semblables à des lois de puissances négatives : les poids baissent initialement très rapidement, puis convergent de plus en plus lentement proches de zéro, formant une *longue queue*.

Les poids résultants ont ensuite été évalués dans le contexte de tâches de désambiguisation de sens de l'atelier Semeval en anglais avec la tâche Semeval-2007 *English Lexical Sample* (ELS), et une participation à la tâche Semeval-2010 *Japanese WSD* (JWSD). Nous avons proposé l'utilisation de modèles Naïve Bayes avec quelques modifications nouvelles. Tout d'abord, nous utilisons les poids appris précédemment sur corpus, et

contrôlons leur impact en évaluant la force de l'a priori Dirichlet par validation croisée, puis redimentionnant les log-probs pour contourner des problèmes de dépendance des attributs. Nous proposons aussi l'utilisation d'attributs alternatifs comme les formes du mot cible et les mots-outils voisins. En combinant les trois types d'attributs nous arrivons à une précision de désambiguisation qui n'est pas significativement différente du meilleur système proposé pour l'ELS, et nous obtenons la première place pour notre participation au JWSD. Des expérimentations additionnelles montrent que dans les deux cas le système qui utilise les poids appris a une plus haute précision que les meilleurs paramètres d'un ensemble de fonctions proposées dans des études précédentes : linéaire, Gaussienne et exponentielle. La fonction de puissance négative, que nous proposons à partir de notre observation des poids, fonctionne toutefois aussi bien que la fonction apprise, et le paramètre optimal donne une courbe pratiquement identique aux poids appris.

Les évidences de la fonction que suit la relation entre deux mots ont beaucoup d'applications pour les méthodes basées sur les cooccurrences. De nos jours, la plupart des systèmes qui utilisent des comptes en contextes, accumulent ceux-ci dans une fenêtre fixe utilisant des poids uniformes. Notre fonction de poids offre une alternative attrayante et bien fondée à ces fins. Par exemple, la traduction d'un terme est un problème très lié à la désambiguisation de sens : en faisant l'alignement de bitextes, nous pourrions en effet attacher un mot de la phrase cible à un contexte de la phrase source, et construire automatiquement des modèles de contextes comme ceux que nous avons utilisés dans nos expériences. Ceux-ci pourraient servir à produire des probabilités de traduction désambiguïsées par le contexte.

Nous avons ensuite procédé à une adaptation de nos modèles de contexte afin de les utiliser dans des tâches de Recherche d'Information. La différence principale entre les deux applications est que nous devons passer à des méthodes non-supervisées pour la représentation du sens. Nous proposons donc d'utiliser l'allocation Dirichlet Latente (LDA) pour faire l'apprentissage des tendances principales des attributs de contextes. Ceci mène à une représentation à dimensionalité réduite qu'on peut introduire dans le score de vraisemblance de requête. Une évaluation du système résultant avec trois collec-

tions de la *Text REtrieval Conference* (TREC) donne une amélioration proportionnelle moyenne de 12% du MAP et 23% du $GMAP$. En investiguant les résultats, on constate que les gains se font surtout sur les requêtes difficiles, augmentant la robustesse de la recherche.

Nous observons que bien que la méthode ait des améliorations de performance comparables aux applications de LDA pour la RI standard, leur effet est très différent. Les applications de LDA ont l'effet de faire une expansion de document, ou plutôt un lissage ajusté au contenu du document. Notre approche a l'effet inverse, et ajoute des contraintes de contexte sur les mots des documents, menant à une *constriction de requête*. En restreignant le sens de ce qui est spécifié dans la requête, nous pouvons espérer rendre possibles des requêtes très précises ou difficiles.

Les points levés par notre étude ouvrent beaucoup d'avenues de recherche. Tout d'abord, l'adaptation que nous avons proposée calcule des sujets maximisant l'ensemble des observations dans le cadre d'une seule mixture de multinomiales. Ceci ignore la nature des attributs, qui sont en fait plusieurs multinomiales différentes. En améliorant la modélisation des attributs, nous pourrions nous attendre à de plus grandes améliorations de l'efficacité de recherche. De plus, après la proposition de LDA, des systèmes réduisant les contraintes d'indépendance des sujets furent proposés. L'utilisation de telles méthodes augmenterait la qualité des modèles.

Un deuxième point d'intérêt serait de développer des méthodes pour corriger le manque d'information sémantique du besoin d'information. Nous pouvons ainsi envisager l'ajout d'une deuxième *requête de contexte* dont les mots ne seraient pas cherchés, mais qui seraient utilisés pour corriger le sens désiré de la requête. Afin d'aider l'utilisateur à formuler des mots de contexte, nous pouvons développer des méthodes de visualisation sémantique de la requête, en présentant des mots des sujets des mots clés. En les refusant ou les acceptant, l'utilisateur pourrait effectuer un *feedback sémantique.*

Une troisième application des modèles de contextes développés serait d'en faire une adaptation pour effectuer une expansion de requête. Nous pouvons en effet considérer que les modèles de contextes sont un continuum de distributions de probabilités conditionnelles. En faisant des inférences des modèles de contextes entre eux, nous pouvons

introduire des probabilités de dépendance entre les mots, et produire des mots d'expansion de meilleure qualité. Nous pourrions nommer le procédé *Constriction-Expansion*.

Ceci conclut ce mémoire portant sur les contextes de mots, beaucoup de recherche s'annonce encore devant nous sur ce sujet passionnant.

5.1 Articles résultants de ces travaux

Les travaux présentés dans ce mémoires ont été préparés en parallèle à des publications dans des conférences et journaux.

Le chapitre 2 de ce mémoire fut tout d'abord publié dans le cadre de la conférence Semeval-2010 avec une participation à la tâche Japanese WSD [6] (court article de 4 pages présenté par une affiche). Une version plus longue contenant des expériences additionnelles fut ensuite publiée dans le cadre de la conférence Coling-2010 [7] (article long de 8 pages avec une présentation orale). L'intégrale du chapitre 2 (en Anglais) est à paraître dans l'édition spéciale "Issue on SemEval-2 Japanese task and Word Sense Disambiguation", du journal "Journal of Natural Language Processing" Japonais sous le titre "Construction of context models for Word Sense Disambiguation", en Juillet 2011.

Le chapitre 4 de ce mémoire fera l'objet d'une soumission à la *Conference on Information and Knowledge Management*(ACM SIGIR 2011), sous la forme d'un article long (10 pages) nommé "Latent context model for Information Retrieval", à paraître en Octobre 2011.

BIBLIOGRAPHIE

[1] Jing Bai, Dawei Song, Peter Bruza, Jian-Yun Nie, and Guihong Cao. Query expansion using term relationships in language models for information retrieval. In *CIKM '05 Proceedings*, pages 688–695, New York, NY, USA, 2005. ACM. ISBN 1-59593-140-6.

[2] Roi Blanco and Alvaro Barreiro. Probabilistic document length priors for language models. In *ECIR'08 : Proceedings of the IR research, 30th European conference on Advances in information retrieval*, pages 394–405, Berlin, Heidelberg, 2008. Springer-Verlag. ISBN 3-540-78645-7, 978-3-540-78645-0.

[3] David M. Blei and John D. Lafferty. Correlated topic models. In *Advances in Neural Information Processing Systems (NIPS)*, 2006.

[4] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3 :993–1022, 2003.

[5] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. In *WWW7 : Proceedings of the seventh international conference on World Wide Web 7*, pages 107–117, Amsterdam, The Netherlands, The Netherlands, 1998. Elsevier Science Publishers B. V.

[6] Bernard Brosseau-Villeneuve, Noriko Kando, and Jian-Yun Nie. Rali : Automatic weighting of text window distances. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 375–378, Uppsala, Sweden, July 2010. Association for Computational Linguistics.

[7] Bernard Brosseau-Villeneuve, Jian-Yun Nie, and Noriko Kando. Towards an optimal weighting of context words based on distance. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 107–115, Beijing, China, August 2010. Coling 2010 Organizing Committee.

[8] Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. Word-sense disambiguation using statistical methods. In *Proceedings of the 29th annual meeting on Association for Computational Linguistics*, pages 264–270, Morristown, NJ, USA, 1991. Association for Computational Linguistics.

[9] Sang bum Kim, Hee cheol Seo, and Hae chang Rim. Information retrieval using word senses : root sense tagging approach. In *SIGIR 2004*, pages 258–265, 2004.

[10] Jun Fu Cai, Wee Sun Lee, and Yee Whye Teh. Nus-ml : improving word sense disambiguation using topic features. In *SemEval '07 Proceedings*, pages 249–252, Morristown, NJ, USA, 2007. Association for Computational Linguistics.

[11] W. Bruce Croft and John Lafferty. *Language Modeling for Information Retrieval*, volume 13. Kluwer Academic Publishers, 2003.

[12] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6) :391–407, 1990.

[13] Jianfeng Gao, Ming Zhou, Jian-Yun Nie, Hongzhao He, and Weijun Chen. Resolving query translation ambiguity using a decaying co-occurrence model and syntactic dependence relations. In *SIGIR '02 Proceedings*, pages 183–190, New York, NY, USA, 2002. ACM. ISBN 1-58113-561-0.

[14] Tanja Gaustad. Statistical corpus-based word sense disambiguation : Pseudowords vs. real ambiguous words. In *Companion Volume to the Proceedings of the 36th annual meeting of the ACL*, pages 61–66, Toulouse, France, 2001. ACL Press.

[15] Julio Gonzalo, Felisa Verdejo, Irina Chugur, and Juan Cigarrin. Indexing with wordnet synsets can improve text retrieval. In *Proceedings of COLING / ACL '98 Workshop on the Usage of WordNet for NLP*, pages 38–44, Montreal, Canada, 1998. ACL Press.

[16] Thomas Hofmann. Probabilistic latent semantic indexing. In *Proceedings of SIGIR '99*, pages 50–57, New York, NY, USA, 1999. ACM. ISBN 1-58113-096-1.

[17] Vera Hollink, Jaap Kamps, Christof Monz, and Maarten De Rijke. Monolingual document retrieval for european languages. *Information Retrieval*, 7(1-2) :33–52, 2004. ISSN 1386-4564.

[18] Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. Ontonotes : the 90% solution. In *NAACL '06 : Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume : Short Papers on XX*, pages 57–60, Morristown, NJ, USA, 2006. Association for Computational Linguistics.

[19] Nancy Ide and Jean Véronis. Introduction to the special issue on word sense disambiguation : the state of the art. *Computational Linguistics*, 24(1) :2–40, 1998. ISSN 0891-2017.

[20] Adam Kilgarriff. I don't believe in word senses. *Computers and the Humanities*, 31(2), 1997.

[21] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46 (5) :604–632, 1999. ISSN 0004-5411.

[22] Robert Krovetz and W. Bruce Croft. Lexical ambiguity and information retrieval. *ACM Trans. Inf. Syst.*, 10 :115–141, April 1992. ISSN 1046-8188.

[23] H. Kucera and W. N. Francis. *Computational Analysis of Present-Day American English*. Brown University Press, Providence, RI, 1967.

[24] Yoong Keok Lee and Hwee Tou Ng. An empirical evaluation of knowledge sources and learning algorithms for word sense disambiguation. In *EMNLP '02 : Proceedings of the ACL-02 conference on Empirical methods in natural language processing*, pages 41–48, Morristown, NJ, USA, 2002. Association for Computational Linguistics.

[25] Els Lefever and Veronique Hoste. Semeval-2010 task 3 : cross-lingual word sense disambiguation. In *DEW '09 : Proceedings of the Workshop on Semantic Evalua-*

tions : Recent Achievements and Future Directions, pages 82–87, Morristown, NJ, USA, 2009. Association for Computational Linguistics. ISBN 978-1-932432-31-2.

[26] M. Lesk. Automatic sense disambiguation using machine readable dictionaries : How to tell a pine cone from an ice cream cone. In *Proceedings of the 5th SIGDOC*, pages 24–26, 1986.

[27] Wei Li and Andrew Mccallum. Pachinko allocation : Dag-structured mixture models of topic correlations. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 577–584, 2006.

[28] Yuanhua Lv and ChengXiang Zhai. Positional language models for information retrieval. In *SIGIR '09 Proceedings*, pages 299–306, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-483-6.

[29] Kikuo Maekawa. Compilation of the balanced corpus of contemporary written japanese in the kotonoha initiative (invited paper). In *ISUC '08 Proceedings*, pages 169–172, Washington, DC, USA, 2008. IEEE Computer Society. ISBN 978-0-7695-3433-6.

[30] Francis Bach Matthew Hoffman, David Blei. Online learning for latent dirichlet allocation. In *Advances in Neural Information Processing Systems (NIPS)*, 2010.

[31] Paul McNamee, Charles Nicholas, and James Mayfield. Addressing morphological variation in alphabetic languages. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '09, pages 75–82, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-483-6.

[32] G.A. Millea, C. Leacock, R. Tengi, and R. T. Bunker. A semantic concordance. In *Proceedings of the ARPA Workshop on Human Language Technology*, pages 303–308, 1993.

[33] Roberto Navigli. Word sense disambiguation : A survey. *ACM Comput. Surv.*, 41 (2) :1–69, 2009. ISSN 0360-0300.

[34] Robert Krovetz Nec and Robert Krovetz. On the importance of word sense disambiguation for information retrieval. In *State of the Art and Future Research. Proceedings of the LREC 2002 Workshop. Las*, 2002.

[35] Hwee Tou Ng. Exemplar-based word sense disambiguation : Some recent improvements. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, pages 208–213, 1997.

[36] Hwee Tou Ng and Hian Beng Lee. Integrating multiple knowledge sources to disambiguate word sense : an exemplar-based approach. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, pages 40–47, Morristown, NJ, USA, 1996. Association for Computational Linguistics.

[37] Jong-Hoon Oh and Key-Sun Choi. Word sense disambiguation using static and dynamic sense vectors. In *Proceedings of the 19th international conference on Computational linguistics - Volume 1*, pages 1–7, Morristown, NJ, USA, 2002. Association for Computational Linguistics.

[38] Manabu Okumura, Kiyoaki Shirai, Kanako Komiya, and Hikaru Yokono. Semeval-2010 task : Japanese wsd. In *SemEval '10 Proceedings*. Association for Computational Linguistics, 2010.

[39] Javier Parapar, David E. Losada, and Álvaro Barreiro. Compression-based document length prior for language models. In *SIGIR '09 : Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 652–653, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-483-6.

[40] Jay M. Ponte and W. Bruce Croft. A language modeling approach to information retrieval. In *SIGIR '98 : Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 275–281, New York, NY, USA, 1998. ACM. ISBN 1-58113-015-5.

[41] Ian Porteous, David Newman, Alexander Ihler, Arthur Asuncion, Padhraic Smyth, and Max Welling. Fast collapsed gibbs sampling for latent dirichlet allocation. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '08, pages 569–577, New York, NY, USA, 2008. ACM.

[42] Sameer S. Pradhan, Edward Loper, Dmitriy Dligach, and Martha Palmer. Semeval-2007 task 17 : English lexical sample, srl and all words. In *SemEval '07 Proceedings*, pages 87–92, Morristown, NJ, USA, 2007. Association for Computational Linguistics.

[43] C S. Robertson, H. Zaragoza, Stephen Robertson, and Hugo Zaragoza. The probabilistic relevance framework : Bm25 and beyond, 2009.

[44] Stephen Robertson. On the history of evaluation in ir. *J. Inf. Sci.*, 34(4) :439–456, 2008. ISSN 0165-5515.

[45] Tetsuya Sakai. On the reliability of information retrieval metrics based on graded relevance. *Inf. Process. Manage.*, 43(2) :531–548, 2007. ISSN 0306-4573.

[46] Gerard. Salton. *Automatic Information Organization and Retrieval.* McGraw Hill Text, 1968. ISBN 0070544859.

[47] Mark Sanderson. Word sense disambiguation and information retrieval. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '94, pages 142–151, New York, NY, USA, 1994. Springer-Verlag New York, Inc. ISBN 0-387-19889-X.

[48] Mark Sanderson. Retrieving with good sense. *Information Retrieval*, 2 :49–69, 2000.

[49] Mark Sanderson and C. J. Van Rijsbergen. The impact on retrieval effectiveness of skewed frequency distributions. *ACM Trans. Inf. Syst.*, 17 :440–465, October 1999. ISSN 1046-8188.

[50] Hinrich Schütze. Automatic word sense discrimination. *Comput. Linguist.*, 24 : 97–123, March 1998. ISSN 0891-2017.

[51] Hinrich Schutze and Jan O. Pedersen. Information retrieval based on word senses. In *In Proceedings of the 4th Annual Symposium on Document Analysis and Information Retrieval*, pages 161–175, 1995.

[52] Advaith Siddharthan. Christopher d. manning and hinrich schutze. foundations of statistical natural language processing. mit press, 2000. isbn 0-262-13360-1. 620 pp. *Nat. Lang. Eng.*, 8(1) :91–92, 2002. ISSN 1351-3249.

[53] Mark D. Smucker and James Allan. An investigation of dirichlet prior smoothing's performance advantage. Technical report, 2005.

[54] D. Song and P. D. Bruza. Towards context sensitive information inference. *Journal of the American Society for Information Science and Technology*, 54(4) :321–334, 2003. ISSN 1532-2882.

[55] Fei Song and W. Bruce Croft. A general language model for information retrieval. In *CIKM '99 : Proceedings of the eighth international conference on Information and knowledge management*, pages 316–321, New York, NY, USA, 1999. ACM. ISBN 1-58113-146-1.

[56] Amanda Spink, Dietmar Wolfram, Major B. J. Jansen, and Tefko Saracevic. Searching the web : the public and their queries. *J. Am. Soc. Inf. Sci. Technol.*, 52(3) : 226–234, 2001. ISSN 1532-2882.

[57] Munirathnam Srikanth and Rohini Srihari. Biterm language models for document retrieval. In *SIGIR '02 : Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 425–426, New York, NY, USA, 2002. ACM. ISBN 1-58113-561-0.

[58] Christopher Stokoe. Differentiating homonymy and polysemy in information retrieval. In *Proceedings of the conference on Human Language Technology and*

Empirical Methods in Natural Language Processing, HLT '05, pages 403–410, Morristown, NJ, USA, 2005. Association for Computational Linguistics.

[59] Christopher Stokoe, Michael P. Oakes, and John Tait. Word sense disambiguation in information retrieval revisited. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, SIGIR '03, pages 159–166, New York, NY, USA, 2003. ACM. ISBN 1-58113-646-3.

[60] Yee Whye Teh, David Newman, and Max Welling. A collapsed variational bayesian inference algorithm for latent dirichlet allocation. In *Advances in Neural Information Processing Systems (NIPS)*, 2006.

[61] C. J. van Rijsbergen. *Information Retrieval*. Butterworths, London, 2 edition, 1979.

[62] Ellen M. Voorhees. Using wordnet to disambiguate word senses for text retrieval. In *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '93, pages 171–180, New York, NY, USA, 1993. ACM. ISBN 0-89791-605-0.

[63] Ellen M. Voorhees. Using wordnet to disambiguate word senses for text retrieval. In *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '93, pages 171–180, New York, NY, USA, 1993. ACM. ISBN 0-89791-605-0.

[64] Ellen M. Voorhees. Overview of the trec 2004 robust retrieval track. In *Proceedings of the Thirteenth Text REtrieval Conference (TREC2004*, page 13, 2004.

[65] Xing Wei and W. Bruce Croft. Lda-based document models for ad-hoc retrieval. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '06, pages 178–185, New York, NY, USA, 2006. ACM. ISBN 1-59593-369-7.

[66] Jason M. Whaley and Javed A. Aslam. An application of word sense disambiguation to information retrieval. Technical report, 1999.

[67] Jinxi Xu and W. Bruce Croft. Corpus-based stemming using cooccurrence of word variants. *ACM Trans. Inf. Syst.*, 16(1) :61–81, 1998. ISSN 1046-8188.

[68] David Yarowsky. One sense per collocation. In *HLT '93 : Proceedings of the workshop on Human Language Technology*, pages 266–271, Morristown, NJ, USA, 1993. Association for Computational Linguistics. ISBN 1-55860-324-7.

[69] ChengXiang Zhai and John Lafferty. Two-stage language models for information retrieval. In *SIGIR '02 : Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 49–56, New York, NY, USA, 2002. ACM. ISBN 1-58113-561-0.

[70] Chengxiang Zhai and John Lafferty. A study of smoothing methods for language models applied to information retrieval. *ACM Trans. Inf. Syst.*, 22(2) :179–214, 2004. ISSN 1046-8188.