Université de Montréal

Modèles de langue en recherche d'information

par
Carmen Alvarez

Département d'informatique et de recherche opérationelle (DIRO)

Mémoire présenté à la Faculté des études supérieures
en vue de l'obtention du grade de M.Sc.
en informatique

Février, 2004

Université de Montréal
Faculté des études supérieures

Ce mémoire intitulé :
Modèles de langue en recherche d'information

présenté par :
Carmen Alvarez

a été évalué par un jury composé des personnes suivantes :

Yoshua Bengio	président-rapporteur
Philippe Langlais	directeur de recherche
Jian-Yun Nie	codirecteur
Guy Lapalme	membre du jury

Mémoire accepté le 29 avril 2004

Résumé

Dans ce travail, nous présentons plusieurs approches aux modèles de langue en recherche d'information. D'abord, nous étudions des techniques de lissage pour le modèle unigramme. Cette approche, comme la plupart des approches traditionnelles à la recherche d'information, se concentre sur les mots simples. L'hypothèse d'indépendance entre mots est une sursimplification du problème de recherche d'information et nous cherchons à améliorer la performance d'un système de recherche d'information en prenant compte des relations entre mots.

Nous étudions l'utilité d'un modèle bigramme pour prendre en compte le contexte des mots, mais cette approche présente deux contraintes indésirables pour la recherche d'information : les mots doivent être adjacents, et l'ordre des mots est important. Nous trouvons qu'un modèle bigramme n'amène qu'une très légère amélioration par rapport au modèle unigramme.

Nous évaluons une autre unité lexicale pour prendre compte des dépendances entre mots : l'affinité lexicale, une paire de mots qui se trouvent proches dans un texte (dans une fenêtre de n mots), sans contrainte sur l'ordre des mots. Nous développons plusieurs approches aux modèles de langue qui incorporent ces affinités lexicales.

Notre première approche consiste à utiliser seulement les affinités lexicales dans le score de pertinence d'un document pour une requête. Ensuite, nous proposons un modèle de langue qui se base principalement sur les paires, et utilise les mots simples dans le lissage des modèles. Finalement, nous présentons une approche où les paires et les mots simples sont combinés dans un modèle de langue, et où leurs contributions relatives au score de pertinence sont plus configurables. Nous observons des améliorations par rapport au modèle unigramme lorsque la contribution des mots simples au score est relativement élevée par rapport à celle des paires.

Mots-clés

modèle de langue, recherche d'information, mots composés, affinités lexicales

Abstract

We present several language modeling approaches for information retrieval. First, we study smoothing techniques for a unigram model. This approach, like a large part of traditional information retrieval approaches, focuses on single words. The hypothesis of independence between words is an oversimplification of the information retrieval problem, and we investigate ways to improve the performance of an information retrieval system by taking the relationships between words into context.

We study the usefulness of a bigram model to account for word context, but this approach presents two undesirable constraints for information retrieval : the words must be adjacent, and word order is important. We find that a bigram model only offers a very slight improvement over the unigram model.

We evaluate another lexical unit to take account of word dependencies : the lexical affinity, a pair of words that occur near each other (separated by at most n words) in a text, without a restriction on word order. We develop several language modeling approaches that incorporate these lexical affinities.

Our first approach consists of using only the lexical affinities in the relevance score of a document for a query. We then propose a language model that is based primarily on word pairs, and uses the single words to smooth the model. Finally, we present an approach in which the pairs and single words are combined in a language model, and in which their relative contributions to the relevance score are more configurable. We observe improvements in comparison with the unigram model when the contribution of the single words to the relevance score is relatively high compared to that of the word pairs.

Keywords

language model, information retrieval, compound terms, lexical affinities

Table des matières

Liste des tableaux

Table des figures

Chapitre 1

Introduction

1.1 La recherche d'information

1.1.1 Un système de recherche d'information

Le but d'un système de recherche d'information est de trouver les documents dans une collection qui peuvent répondre au besoin d'information d'un usager. La recherche d'information comprend trois concepts :

- Un besoin d'information
- Une collection de documents
- La pertinence, ou le degré auquel un document satisfait au besoin d'information

Le besoin d'information est représenté par une requête, qui peut être exprimée dans une langue naturelle (ex : "Trouver les documents qui parlent de l'histoire du Canada") ou dans un format structuré tel qu'une requête booléenne (ex : "histoire ET Canada"). Les documents aussi peuvent être structurés (par exemple, avec plusieurs champs pour l'auteur, la date, et le titre du document), ou sans aucune structure. Une collection peut être spécialisée (par exemple, constituée d'articles médicaux ou de droit), ou générale, comme le Web. Nous décrivons la collection utilisée dans notre travail dans le chapitre 2.

Le figure 1.1 montre les éléments de base d'un système de recherche d'information. Un processus d'indexation est effectué, où le système crée une représentation interne des documents et de la requête. Une fonction ou score de correspondance, $score(d, q)$, mesure la corrélation entre ces représentations. La correspondance peut être binaire (pertinent ou non-pertinent) ou peut mesurer le degré de pertinence. Idéalement, la correspondance entre ces deux représentations, déterminée par le système, doit s'accorder au jugement de pertinence de

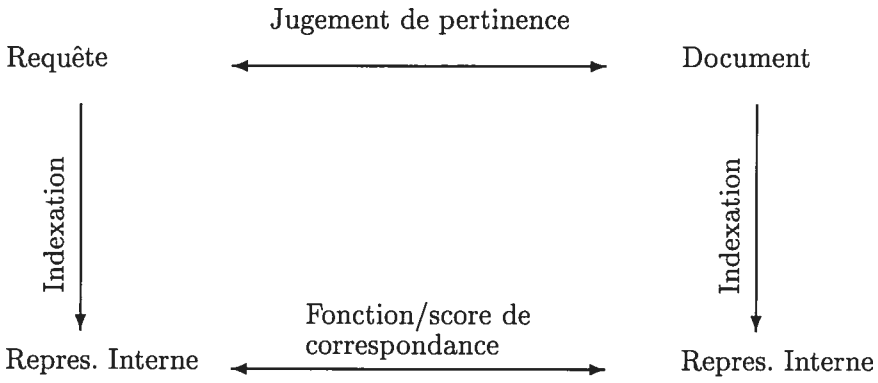

FIG. 1.1 – Éléments de base d'un système de recherche d'information

l'usager. Pour une requête donnée, le système retourne des documents en ordre décroissant du score de pertinence. Ainsi, un système de recherche d'information comprend deux modèles : un modèle d'indexation (qui peut en fait être décomposé en deux modèles : l'un pour l'indexation des documents, l'autre pour les requêtes), et un modèle de pertinence.

1.1.2 Indexation

Idéalement, le résultat de l'indexation serait une représentation des concepts des documents et des requêtes. Cependant, étant donné la complexité de l'analyse sémantique et l'extraction du sens à partir d'un texte, les unités les plus souvent utilisées pour représenter un texte sont les mots simples. Une première représentation consiste à considérer un texte comme l'ensemble des mots le consituant :

$$doc = \{w_1, w_2, \ldots, w_n\} \tag{1.1}$$

Cependant, cette approche ne prend pas en compte l'importance relative des mots dans le document. Une deuxième représentation est :

$$doc = \{(w_1, p_1), (w_2, p_2), \ldots, (w_n, p_n)\} \tag{1.2}$$

où p_i est le poids, ou l'importance, du mot w_i, pour le document. Les mots les plus importants pour un document ont en général deux caractéristiques :

- ils se trouvent fréquemment dans le document.
- ils sont peu fréquents dans la collection et donc servent à distinguer un document dans

la collection.

La combinaison de ces deux caractéristiques permet d'éviter que les mots outils ou "vides" tels que "le" ou "et" aient un poids élevé pour un document à cause de leur fréquence élevée. Le poids $p_{w,d}$ d'un mot w pour un document d comprend alors deux composantes :

- la fréquence du mot dans le document, $tf(w,d)$ (*term frequency*), ou une normalisation telle que la fréquence relative, $\frac{tf(w,d)}{\sum_{w \in d} tf(w,d)}$.
- l'inverse de sa fréquence globale, $idf_w = \log(N/n_w)$ où N est le nombre de documents dans la collection et n_w est le nombre de documents où w apparaît.

Traditionnellement, ces deux facteurs sont combinés selon l'équation 1.3 pour former le poids d'un mot w dans un document d :

$$p_{w,d} = tf(w,d) \times idf_w \tag{1.3}$$

Il existe encore des mots "vides" (sans aucun sens intéressant pour le document) qui peuvent avoir un *idf* relativement bas, comme "auparavant" ou "desquelles". Pour s'assurer que ces mots n'ont pas un poids trop élevé dans un document, une *stopliste* est souvent appliquée. Une *stopliste* est une liste qui contient un ensemble de mots "vides", qui sont éliminés du document à la première étape de l'indexation.

Il est à noter que cette approche d'indexation ne considère pas les relations entre les mots. Bien que cette hypothèse d'indépendance entre mots ne soit pas toujours valide, on la fait normalement pour simplifier l'indexation ainsi que les calculs de pertinence que nous décrivons maintenant.

1.1.3 Approches classiques au modèle de pertinence

Le but de cette section est de présenter de manière synthétique les alternatives les plus classiques à l'approche que nous explorons dans ce mémoire. Nous invitons le lecteur à lire [2] et [4] pour plus d'information sur ces approches.

Le modèle booléen simple

Le modèle booléen a été un des premiers modèles proposés pour la recherche d'information. Ce modèle de base représente les documents comme un ensemble de mots (eq 1.1), sans égard à leur poids relatif. Le processus d'indexation décrit dans la section 1.1.2 est utilisé

pour sélectionner les mots importants, mais le poids ne figure ni dans la représentation des documents et requêtes, ni dans le score de pertinence. Une requête est représentée comme une combinaison de mots et d'opérateurs booléens. Par exemple :

$$q = (w_1 \vee w_2) \wedge \neg w_3$$

où $\vee$, $\wedge$, et $\neg$ sont les opérateurs logiques OU, ET, et NON, respectivement. Le score de pertinence d'un document d pour la requête q est calculé en décomposant la requête et en appliquant les quatre formules suivantes :

$$score(d, w) = \begin{cases} 1 & \text{si } w \in d \\ 0 & \text{sinon} \end{cases}$$

$$score(d, t_1 \wedge t_2) = \begin{cases} 1 & \text{si } score(d, t_1) = 1 \text{ et } score(d, t_2) = 1 \\ 0 & \text{sinon} \end{cases}$$

$$score(d, t_1 \vee t_2) = \begin{cases} 1 & \text{si } score(d, t_1) = 1 \text{ ou } score(d, t_2) = 1 \\ 0 & \text{sinon} \end{cases}$$

$$score(d, \neg t) = \begin{cases} 0 & \text{si } score(d, t) = 1 \\ 1 & \text{sinon} \end{cases}$$

où le terme t est un mot simple w de la requête ou une combinaison de termes t et d'opérateurs logiques. Ce modèle attribue un score binaire, 0 ou 1, aux documents. Ceci présente une certaine limitation, car le système doit retourner les documents dans un ordre arbitraire.

Le modèle booléen flou

Une variante du modèle booléen emploie les poids des mots dans la représentation des documents (eq 1.2) et donne un score de pertinence non-binaire. Les requêtes sont représentées de la même manière que pour le modèle booléen simple, mais le score de pertinence se base

sur les formules suivantes :

$$\begin{aligned} score(d,t) &= p_{w,d} \\ score(d,t_1 \wedge t_2) &= score(d,t_1) \times score(d,t_2) \\ score(d,t_1 \vee t_2) &= score(d,t_1) + score(d,t_2) - score(d,t_1) \times score(d,t_2) \\ score(d,\neg t) &= 1 - score(d,t) \end{aligned} \tag{1.4}$$

Ainsi, les documents peuvent être triés en ordre décroissant du score de pertinence. Il convient de noter que dans ces deux approches, l'utilisateur doit formuler des requêtes structurées.

Le modèle vectoriel

L'approche classique la plus souvent utilisée est le modèle vectoriel. Dans ce modèle, un espace vectoriel de dimension n est défini, où n est le nombre de mots uniques dans la collection. Un document d, tout comme une requête q, est alors représenté par un vecteur de cet espace de dimension n :

$$\begin{aligned} d &= < d_1, d_2, \ldots, d_n > \\ q &= < q_1, q_2, \ldots, q_n > \end{aligned} \tag{1.5}$$

où d_i est le poids du mot i dans le document d et q_i est le poids du mot i dans la requête q. Plusieurs scores de pertinence du document d pour la requête q ont été proposés :

- produit interne

$$score(d,q) = \sum_{i=1}^{n} d_i \times q_i \tag{1.6}$$

- Dice

$$score_{Dice}(d,q) = \frac{2\sum_{i=1}^{n} d_i \times q_i}{\sum_{i=1}^{n} d_i + \sum_{i=1}^{n} q_i} \tag{1.7}$$

- Jaccard

$$score_{Jaccard}(d,q) = \frac{\sum_{i=1}^{n} d_i \times q_i}{\sum_{i=1}^{n} d_i + \sum_{i=1}^{n} q_i - \sum_{i=1}^{n} d_i \times q_i} \tag{1.8}$$

- cosinus

$$score_{cos}(d,q) = \frac{\sum_{i=1}^{n} d_i \times q_i}{\sqrt{\sum_{i=1}^{n} d_i^2}\sqrt{\sum_{i=1}^{n} q_i^2}} \tag{1.9}$$

Le produit interne (eq 1.6) est une mesure de similarité entre deux vecteurs. Les scores Dice (eq 1.7) et Jaccard (eq1.8) sont des normalisations du produit interne, pour que les scores soient entre 0 et 1. Le cosinus (eq 1.9) mesure l'angle entre deux vecteurs et donne aussi un score entre 0 et 1.

Mots composés dans le modèle vectoriel

Nie et Dufort [13] ont démontré une amélioration en performance du modèle vectoriel lorsque des mots composés sont compris dans l'indexation et le score de pertinence. Le gain en performance dépend largement de la façon dont les termes composés sont employés. Si les termes composés sont compris dans le même vecteur que les mots simples, la performance du système est moins bonne. Cependant, une augmentation en performance est possible en construisant deux vecteurs indépendants pour les mots simples et les mots composés, d_{simple} et $d_{compose}$, et en les combinant dans le score avec des poids différents I_{simple} et $I_{compose}$ comme suit :

$$score(d,q) = I_{simple} \times score(d_{simple},q) + I_{compose} \times score(d_{compose},q) \tag{1.10}$$

1.1.4 Fichier inversé

Pour éviter un parcours de la collection complète pour une requête donnée, un fichier inversé est employé, qui liste tous les documents où un mot particulier apparaît. Le format du fichier inversé est comme suit :

$$w_i : (d_1, p_{i,1}), (d_2, p_{i,2}), \ldots (d_n, p_{i,m})$$

où d_j est un document qui contient le mot w_i, et $p_{i,j}$ est le poids du mot i dans le document j. On peut donc limiter la recherche aux documents contenant au moins un mot de la requête.

1.2 Modèles de langue en recherche d'information

L'utilisation de modèles de langue pour la recherche d'information a été originellement proposée par Ponte et Croft [14]. Dans cette section nous présentons d'abord une introduction aux modèles de langue, et puis décrivons comment les modèles de langue peuvent être appliqués à la recherche d'information.

1.2.1 Introduction aux modèles de langue

Un modèle de langue est une distribution sur l'ensemble des phrases d'une langue. Formellement, un modèle de langue estime $p(s)$, la probabilité d'observer la phrase s dans la langue modélisée. Pour que le modèle soit une distribution probabiliste valide, il faut que :

$$\sum_{s} p(s) = 1$$

où la sommation est faite sur l'ensemble des phrases possibles. Plusieurs représentations de la phrase s sont possibles, telles qu'une séquence de caractères, d'unités morphologiques, ou d'étiquettes syntaxiques. Dans le cadre des modèles de langue appliqués en recherche d'information, ainsi que dans de nombreuses autres activités du traitement automatique de la langue (TAL), la représentation standard d'une phrase est une séquence de N mots $w_1^N = w_1, \ldots, w_N$.

Avec cette représentation, nous avons :

$$p(s) = p(w_1^N) = p(w_1, w_2, w_3, \ldots, w_N)$$

Comme la probabilité jointe $p(x, y)$ peut être représentée par une probabilité conditionnelle $p(x, y) = p(y|x)p(x)$, on peut décomposer la distribution jointe $p(s)$ de la façon suivante :

$$\begin{aligned} p(s) &= p(w_N|w_1, w_2, w_3, \ldots, w_{N-1}) \times p(w_1, w_2, w_3, \ldots, w_{N-1}) \\ &= p(w_N|w_1, w_2, w_3, \ldots, w_{N-2}) \times p(w_{N-1}|w_1, w_2, w_3, \ldots, w_{N-2}) \times p(w_1, w_2, w_3, \ldots, w_{N-2}) \\ &= \prod_{i=1}^{N} p(w_i|w_1, \ldots, w_{i-1}) \\ &= \prod_{i=1}^{N} p(w_i|w_1^{i-1}) \end{aligned} \tag{1.11}$$

où $h_i = w_i^{i-1}$ est appelé l'historique de w_i.

Chacune des probabilités $p(w_i|w_i^{i-1})$ constitue un paramètre du modèle de langue. Pour simplifier le calcul des paramètres, on fait une approximation appelée communément un modèle n-gramme, qui consiste à ne garder du contexte conditionnant (l'historique) que les $n-1$ derniers mots qui précèdent le mot w_i à prédire. Ainsi :

$$p(s) \approx \prod_{i=1}^{N} p(w_i|w_{i-n+1}^{i-1}) \tag{1.12}$$

Les modèles les plus employés dans les applications TAL sont les modèles unigrammes (1-gramme), bigrammes (2-gramme), et trigrammes (3-gramme). La majorité des travaux sur l'utilisation de modèles de langue en recherche d'information jusqu'ici a porté sur le modèle 1-gramme, pour des raisons de sous-représentation de données que nous étudions dans ce mémoire.

1.2.2 Estimation des paramètres d'un modèle de langue

Une approche de base pour entraîner les paramètres d'un modèle n-gramme consiste à sélectionner le modèle qui maximise la vraisemblance d'un corpus d'entraînement (le plus grand possible). On peut montrer que ceci revient à l'estimation intuitive par fréquence relative :

$$p_{MLE}(w_i|w_{i-n+1}^{i-1}) = \frac{c(w_{i-n+1}^{i-1}w_i)}{\sum_w c(w_{i-n+1}^{i-1}w)} \tag{1.13}$$

La fréquence relative d'un n-gramme w_{i-n+1}^{i} est le rapport du nombre de fois où le n-gramme a été vu sur le nombre de fois où l'historique w_{i-n+1}^{i-1} a été vu dans le corpus. Pour le cas unigramme, la formule 1.13 prend une forme particulièrement simple :

$$p_{MLE}(w) = \frac{c(w)}{N} \tag{1.14}$$

où $c(w)$ est le nombre de fois où le mot w se trouve dans le corpus, et N est le nombre total des mots du corpus.

1.2.3 Score de pertinence basé sur un modèle de langue

Dans le cadre de la recherche d'information chaque document dans une collection est considéré comme un échantillon d'un langage particulier, et un modèle de langue est entraîné

pour chaque document. Une façon de mesurer la pertinence d'un document pour une requête est par la probabilité de générer la requête étant donné le modèle de langue du document. La probabilité de pertinence $p(d|q)$ d'un document d étant donnée une requête q, se décompose par la règle de Bayes :

$$p(d|q) = \frac{p(q|d) \times p(d)}{p(q)} \tag{1.15}$$

Pour une requête donnée, les documents sont ordonnés en ordre décroissant de cette probabilité. Comme le facteur $p(q)$ est le même pour tous les documents, et donc ne change pas l'ordre, on peut l'enlever du score, et donc :

$$p(d|q) \propto p(q|d) \times p(d) \tag{1.16}$$

En outre, on peut faire l'hypothèse d'une distribution uniforme des documents :

$$p(d) = \frac{1}{|C|} \tag{1.17}$$

où $|C|$ est le nombre de documents dans la collection. Dans une vraie application, certains documents peuvent être à priori favorisés, et $p(d)$ peut être adaptée à l'usager. Cependant, avec la distribution uniforme, le terme $p(d)$ devient constant pour tous les documents et n'intervient plus dans le classement des documents. On a alors :

$$p(d|q) \propto p(q|d) \tag{1.18}$$

ce qui est la probabilité de générer la requête q selon le modèle de langue du document d. Cette probabilité peut aussi être écrite comme $p_d(q)$. Avec un modèle unigramme, le score de pertinence d'un document d étant donnée la requête $q = w_1, \ldots, w_N$ est alors :

$$score(d, q) = p_d(q) = \prod_{i=1}^{N} p_d(w_i) \tag{1.19}$$

Et le score basé sur un modèle bigramme est :

$$\begin{aligned} score(d, q) &= p_d(w_1)p_d(w_2|w_1)p_d(w_3|w_2)\cdots p_d(w_N|w_{N-1}) \\ &= p_d(w_1)\prod_{i=2}^{N} p_d(w_i|w_{i-1}) \end{aligned} \tag{1.20}$$

On peut voir qu'en comparaison avec l'approche classique du modèle vectoriel, l'approche des modèles de langue combine l'indexation et le calcul de pertinence dans un seul modèle. De plus, le concept de *tf* est inhérent au modèle de langue, plutôt qu'incorporé heuristiquement. On peut également remarquer qu'un modèle 1-gramme ne tient pas compte des relations entre les mots, ce qui correspond à l'hypothèse d'indépendance entre les termes, faite dans l'approche de base du modèle vectoriel.

1.2.4 Sous-représentation des données

Le problème majeur inhérent aux modèles de langue est la sous-représentation (ou clair-semance) des données d'entraînement, un document étant habituellement assez court. Ceci pose donc le problème du traitement adéquat des n-grammes inconnus, communément appelé lissage. L'approche de l'estimation par maximum de vraisemblance accorde une probabilité nulle à tous les n-grammes non-vus dans le corpus d'entraînement. Cela pose un problème lorsqu'on veut calculer la probabilité d'une phrase s dont un n-gramme w_{i-n+1}^{i} n'a pas été vu dans le corpus. Même si un n-gramme n'a pas été vu, attribuer une probabilité nulle à toute phrase le contenant constitue sans aucun doute une sous-estimation flagrante de la probabilité d'une telle phrase. Une façon en apparence simple de palier ce problème, dans des applications telles que la reconnaissance de la parole, où un seul modèle global est utilisé, est d'utiliser un corpus le plus grand possible pour entraîner le modèle. Cependant, la sous-représentation de données demeure problématique pour les n-grammes d'ordre supérieur à 1, même avec un corpus de très grande taille. Rosenfeld [15] donne l'exemple de l'entraînement des modèles bigramme et trigramme à partir d'un corpus de journaux, de 38 millions mots. La plupart des trigrammes avaient des comptes très bas, la majorité étaient vus juste une fois dans le corpus. Quand le modèle a été appliqué pour prédire de nouveaux articles, un tiers des trigrammes dans les nouveaux documents étaient inconnus du modèle. Le problème est encore plus délicat en recherche d'information où la taille des documents est fixe et généralement petite. Le défi auquel on s'intéresse ici est d'entrainer un modèle à partir de quelques centaines de mots. Des techniques de lissage qui tentent de limiter ce problème sont discutées en détail dans le chapitre 3.

1.2.5 Approches antérieures aux modèles de langue en recherche d'information

L'approche de Ponte et Croft, 1998

Ponte et Croft [14] ont proposé la première approche aux modèles de langue en recherche d'information. Pour pallier le problème de sous-représentation des données, la probabilité observée du mot w dans le document d, $p_{d_{MLE}}(w)$ est lissée avec la probabilité moyenne du mot w parmi les documents où il apparait, $p_{avg}(w)$. Lorsqu'un mot est inconnu du document, sa probabilité globale dans le corpus est utilisée. Formellement :

$$p_d(w) = \begin{cases} p_{d_{MLE}}(w)^{(1.0-R_{w,d})} \times p_{avg}(w)^{R_{w,d}} & \text{si } p_{d_{MLE}}(w) > 0 \\ p_{corpus}(w) & \text{sinon} \end{cases} \tag{1.21}$$

où $p_{corpus}(w)$ est la probabilité du mot w dans la collection, et $R_{w,d}$, le risque associé à l'utilisation de $p_{avg}(w)$ pour le mot w dans le document d. C'est une fonction de la fréquence du mot w dans le document d et dans le corpus, plutôt qu'un paramètre configurable. Nous décrivons plus en détail l'utilisation des probabilités d'un modèle de corpus dans le chapitre 3. Cette première approche, contrairement à toutes les tentatives qui suivent, est non-paramétrique. C'est à dire, la probabilité $p_d(w)$ se base complètement sur les fréquences observées dans la collection, et il n'est pas nécessaire d'apprendre de paramètre spécifique. Ponte et croft rapportent que cette approche donne de meilleures performances que le modèle vectoriel classique.

L'approche de Song et Croft, 1999

Song et Croft [17] introduisent plusieurs techniques qui améliorent la performance des modèles de langue par rapport à l'approche de Ponte et Croft. Premièrement, les probabilités $p_d(w)$ sont lissées avec un lissage Good-Turing. Ensuite, la probabilité du mot w dans le document d est interpolé avec la probabilité du mot dans le corpus :

$$p_{d_{interp}}(w) = \lambda_d p_d(w) + (1 - \lambda_d) p_{corpus}(w) \tag{1.22}$$

Le modèle unigramme est aussi interpolé avec un modèle bigramme :

$$p_d(w_2|w_1) = \lambda_2 p_{d_{MLE}}(w_2|w_1) + (1 - \lambda_2) p_d(w_2) \tag{1.23}$$

En plus d'une amélioration du modèle unigramme (équation 1.22) par rapport à l'approche de Ponte et Croft (équation 1.21), Song et Croft observent une augmentation en précision avec l'utilisation du modèle bigramme. Les valeurs optimales des paramètres λ_d et λ_2 sont déterminées empiriquement.

Un modèle de pertinence, Lavrenko et Croft, 2001

Lavrenko et Croft [10] incorporent les modèles de langue $p_d(w)$ dans un modèle probabiliste classique. L'approche probabiliste classique comprend un modèle de pertinence $p(w|R)$, qui estime la probabilité que le mot w aparaît dans un document pertinent. Similairement, la probabilité que le mot apparaît dans un document non-pertinent, $p(w|N)$, est modélisée. Les documents pour une requête sont triés selon les chances que le document d soit pertinent :

$$\frac{p(d|R)}{p(d|N)} \approx \prod_{w \in d} \frac{p(w|R)}{p(w|N)} \tag{1.24}$$

Le défi principal consiste à construire le modèle de pertinence $p(w|R)$, la probabilité qu'un mot w soit pertinent pour la requête, sans un corpus d'entraînement des documents pertinents. Lavrenko et Croft utilisent des modèles de langue des documents ainsi qu'un modèle de corpus pour estimer le modèle de pertinence, $p(w|R)$.

Specifiquement, la probabilité qu'un mot w soit pertinent pour la requête q est la probabilité de voir w étant donnés les mots de la requête $q_1, \ldots q_k$:

$$p(w|R) \approx p(w|q_i, \ldots, q_k) = \frac{p(w, q_1, \ldots, q_k)}{p(q_1, \ldots, q_k)} \tag{1.25}$$

et la probabilité jointe de voir le mot w avec les mots de la requête $q_1, \ldots, q_k$ se base sur les probabilités des mots dans les documents ou ils apparaissent, selon les modèles de langue $p_d(w)$. La performance du modèle de pertinence a dépassé celle du modèle vectoriel ainsi qu'une implantation du modèle décrit par Song et Croft [17].

L'importance d'un terme dans la requête, Hiemstra, 2002

À l'instar de Song et Croft [17], Hiemstra [7] propose une interpolation entre le modèle du document $p_d(w)$ et le modèle du corpus $p_{corpus}(w)$. Plutôt que de déterminer les contributions relatives des modèles de manière empirique, et de manière fixe pour l'ensemble de documents, il introduit un coefficient λ_i qui estime l'importance d'un mot w_i de la requête. Les coefficients

sont entrainés pour chaque mot d'une requête ; la probabilité interpolée est quant à elle exprimée par :

$$p_{d_{interp}}(w_i) = \lambda_i p_d(w_i) + (1 - \lambda_i) p_{corpus}(w_i) \tag{1.26}$$

Les mots qui se trouvent dans une stopliste ont une importance nulle ($\lambda_i = 0$), tandis que les mots obligatoires tels que précisés par l'usager (par exemple, indiqués avec un "+" dans la requête) ont une importance maximale ($\lambda_i = 1$). Pour les autres mots, un algorithme EM est utilisé pour apprendre automatiquement leur importance. *Relevance feedback*, une technique communement employée, où les premiers n documents retrouvés par le système sont supposés pertinents afin d'ajuster des paramètres de la requête, est employée pour créer un ensemble de documents pertinents utilisés par l'algorithme d'entraînement des λ_i. Hiemstra rapporte une amélioration en performance par rapport au modèle probabiliste classique qui incorpore aussi *relevance feedback*.

L'introduction du modèle de requête, Zhai et Lafferty, 2001

Zhai et Lafferty [9] introduisent le concept d'un modèle de requête, $p_q(w)$. Deux processus génératifs sont considérés. Un usager $\mathcal{U}$ choisit un modèle de requête θ_q selon une distribution $p(\theta_q|\mathcal{U})$, et la requête est générée à partir de ce modèle selon $p(w|\theta_q)$. Similairement, chaque document d dans la collection est généré par un modèle θ_d, choisi de la collection $\mathcal{S}$ selon la distribution $p(\theta_d|\mathcal{S})$. Le système trie les documents pour une requête selon une fonction de risque $\mathrm{R}(d;q)$ en choisissant le document d comme réponse à la requête q.

Zhai et Lafferty proposent deux approches générales pour mesurer le risque $\mathrm{R}(d;q)$ pour les modèles θ_d et θ_q et une valeur binaire de pertinence R :

- une fonction basée uniquement sur la pertinence R :

$$\mathrm{R}(d;q) \propto -p(R = 1|q,d) \tag{1.27}$$

- une fonction de distance entre les deux modèles :

$$\mathrm{R}(d;q) \propto \Delta(\theta_q, \theta_d) \tag{1.28}$$

  Une implantation possible de cette fonction Δ est le cosinus ou produit interne des deux vecteurs, du modèle vectoriel.

Les documents sont ordonnés en ordre décroissant du risque. Formellement :

$$score(d, q) = -\mathrm{R}(d; q) \tag{1.29}$$

La deuxième approche a été implantée, la fonction de distance Δ étant la *Kullback-Leibler divergence* entre deux modèles unigramme θ_q et θ_d :

$$\Delta(\theta_q, \theta_d) = \sum_w p(w|\theta_q) \log \frac{p(w|\theta_q)}{p(w|\theta_d)} \tag{1.30}$$

La performance avec cette approche a été comparable à l'approche classique du modèle vectoriel.

Modèles de traduction

Jin et al. [6] et Lafferty et Berger [1],[8] présentent l'utilisation d'un modèle de traduction en recherche d'information. L'idée générale derrière ces approches est qu'un document est un échantillon d'un langage verbeux, et une requête est un échantillon d'un langage concis. Un modèle de traduction est entraîné qui prédit la probabilité de traduire un document vers une requête. Les documents pour une requête sont ordonnés selon la probabilité que la requête soit une traduction du document. Berger et Lafferty observent une hausse en précision par rapport au modèle vectoriel, et Jin et al. mesurent une amélioration par rapport à un modèle de langue implanté selon l'approche de Song et Croft (équation 1.22).

1.3 Méthode d'évaluation d'un système

L'évaluation d'un système de recherche d'information consiste à mesurer le taux de satisfaction de ses utilisateurs. Cependant, pour les phases d'investigation et développement, sans la disponibilité d'évaluations d'usagers, on dispose typiquement des trois éléments suivants pour l'évaluation du système :

- une collection de documents
- un ensemble de requêtes
- une liste de documents pertinents pour chaque requête, produite par des juges humains

Il est à noter que l'ensemble de requêtes et les jugements de pertinence sont coûteux à construire. Normalement les jugements de pertinence viennent d'un système qui a été déjà implanté et évalué, où les juges ont indiqué les documents pertinents retrouvés par le système

rang du document	pertinent ?	précision	rappel
1		0	0
2	√	1/2	1/3
3	√	2/3	2/3
4		2/4	2/3
5		2/5	2/3

TAB. 1.1 – Exemple des calculs de précision et rappel

pour un ensemble de requêtes. À partir de ces trois éléments, nous pouvons mesurer des taux comme ceux que nous décrivons maintenant.

Précision et rappel

Deux métriques évaluent habituellement la performance d'un système : la précision et le rappel. La précision mesure le pourcentage de documents retrouvés qui sont pertinents, tandis que le rappel mesure le pourcentage de documents pertinents, selon la liste de référence, que le système a trouvé. Formellement :

$$\text{précision} = \frac{\#\text{ de documents pertinents retrouvés}}{\#\text{ de documents retrouvés}} \tag{1.31}$$

$$\text{rappel} = \frac{\#\text{ de documents pertinents retrouvés}}{\#\text{ de documents pertinents}} \tag{1.32}$$

Le tableau 1.1 illustre les calculs de précision et rappel pour les premiers 5 documents retrouvés pour une requête en particulier, pour laquelle la collection contient 3 documents pertinents. Le premier document retrouvé n'est pas jugé pertinent, et donc la précision et le rappel pour ce document sont nuls. Le deuxième document retrouvé par le système est pertinent. La précision pour les deux premiers documents est alors 1 document pertinent retrouvé divisé par 2 documents retrouvés. Le rappel est 1 document pertinent retrouvé divisé par 3 documents de la collection qui ont été jugés pertinents.

Courbe de précision vs. rappel

Une façon d'évaluer un système est de tracer une courbe de précision vs. rappel. Pour une requête donnée, on calcule la précision à plusieurs points de rappel, entre 0% et 100%. Puis, pour chaque point de rappel, on prend la moyenne de la précision pour l'ensemble de requêtes. Le système parfait trouverait seulement les documents retrouvés, avec une précision et rappel de 100%. Cependant, en pratique, la précision d'un système diminue au fur et à mesure que

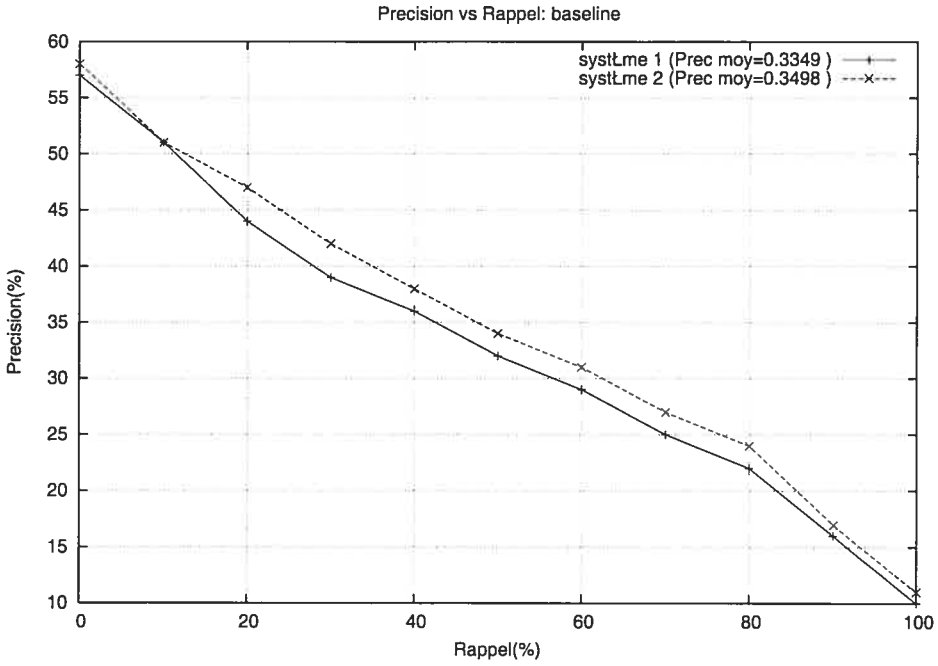

FIG. 1.2 – Courbe de précision vs. rappel

le rappel augmente. Le figure 1.2 est un exemple d'une courbe typique de précision vs rappel, qui compare la performance globale de deux systèmes.

Polarisation Souvent, les courbes sont ajustées avec une modification appelée interpolation. Pour deux points de rappel, i et j, $i < j$, si la précision au point i est inférieur à celle au point j, on dit que la précision interpolée à i = la précision à j. Formellement :

$$p'_i = \max(p_i, p_j), \forall i < j$$

où p'_i est la précision interpolée au point de rappel i, et p_i est la vraie précision au point de rappel i. Cette interpolation est discutable, mais constitue une pratique courante en évaluation de systèmes de recherche d'information.

Précision moyenne

Pour faciliter la comparaison de systèmes, surtout quand il y a plusieurs systèmes ou si les courbes croisent, une valeur simple qui représente la performance du système est utile. La précision moyenne sur 11 points $(0.0, \ldots, 1.0)$ ou 10 points $(0.1 \ldots, 1.0)$ de rappel sont deux métriques standards. Dans notre travail, nous utilisons la précision moyenne sur 11 points pour évaluer toutes nos expériences, ainsi que le système de référence, décrit dans la section

2.3.

1.4 TREC

TREC (Text REtrieval Conference) est une conférence annuelle organisée par le National Institute of Standards and Technology (NIST). Le but des conférences TREC est de promouvoir la recherche dans le domaine de la recherche d'information. À chaque conférence TREC, un ensemble de collections de test est disponible, chaque collection contenant un corpus de documents et un ensemble de requêtes. Une liste de jugements de pertinence (les documents pertinents pour chaque requête) sera produite après que chaque groupe participant ait soumis ses résultats pour chaque requête. Le cadre de l'évaluation permet aux laboratoires participants de comparer leurs systèmes de recherche d'information en vue de les améliorer. Le corpus de documents, les requêtes, et les jugements de pertinence que nous avons utilisés pour ce travail viennent des conférences TREC, et sont décrits dans le chapitre 2.

1.5 Notre travail

Dans ce travail, nous étudions d'abord les approches de modèles de langue unigramme et bigramme pour la recherche d'information. Dans le chapitre 3 nous présentons plusieurs méthodes de lissage pour les modèles unigramme, et plusieurs techniques d'intégration d'un bigramme.

Ensuite, dans le chapitre 4, nous introduisons des approches qui incorporent des paires des mots dans des modèles de langue, sans les contraintes sur l'ordre des mots ou sur l'adjacence que le modèle bigramme présente. Ces paires, ou affinités lexicales, sont des paires de mots qui se trouvent dans une fenêtre de n mots, et leur ordre n'est pas pris en compte.

Nos premières techniques qui incorporent les affinités lexicales se basent uniquement sur les paires. Ensuite nous présentons un modèle de langue qui se base principalement sur les affinités lexicales, et utilise les mots simples lors du lissage. Finalement, nous présentons un modèle de langue qui consulte toujours les mots simples et les paires de mots.

Chapitre 2

Le corpus utilisé dans ce projet

2.1 Le corpus

La collection utilisée dans ce projet comprend 78 321 documents, de la collection TREC AP90 (Disk3, Associated Press newswire, 1990). Les documents sont des articles de journaux de 1990, en anglais. Ils sont structurés en format SGML, avec plusieurs champs (tous les documents ne possèdent pas tous les mêmes champs). Voir la figure 2.1 pour un tel document. Nous avons utilisé dans ce travail les champs suivants :

- head : le titre du document
- byline : l'auteur / les auteurs du document
- note : quelques commentaires sur le document, typiquement 1-2 phrases
- text : le contenu du document

D'autres champs comprennent le numéro du document ou le nom du fichier, qui ne seraient pas très utiles pour une recherche basée sur des mots clés. Les documents ont été soumis à un pré-traîtement :

- Tous les mots ont été transformés en minuscule
- 571 mots outils et "vides" (tels que "your", "be", et "nevertheless") ont été enlevés du corpus. Cette stopliste provient du système SMART (décrit dans la section 2.3).
- Une lemmatisation morphologique (ex : "imposing" est transformé en "impose", et "strains" devient "strain") a été appliqué aux documents.

La figure 2.2 illustre le texte considéré après les étapes de pré-traitement appliquées au document de la figure 2.1. On voit que le pré-traitement n'est pas parfait et peut profiter des améliorations. En particulier, le mot "veterinarians" n'a pas été transformé à sa racine "veterinarian" et le token "'s", qui n'a aucune importance, a été gardé. Aussi, des mots

<DOC>
<DOCNO> AP900202-0100 </DOCNO>
<FILEID>AP-NR-02-02-90 1224EST</FILEID>
<FIRST>r a AM-DruggedPets 02-02 0441</FIRST>
<SECOND>AM-Drugged Pets,0456</SECOND>
<HEAD>Veterinarians Treating More Pets for Drug Overdose</HEAD>
<DATELINE>ATLANTA (AP) </DATELINE>
<TEXT>
If the nation needs another sign that the drug problem is getting worse, veterinarians have one. Animal doctors in the Atlanta area say they are treating an increasing number of patients who are disoriented, having seizures, in a coma or undergoing cardiac arrest because they ate their owner's drugs.
...
</TEXT>
</DOC>

FIG. 2.1 – Exemple d'un document (AP900202-100) avant traitement

<doc>
<docno> ap900202-0100 </docno>
<fileid> ap-nr-02-02-90 1224est </fileid>
<first> am-druggedpets 02-02 0441 </first>
<second> am-drugged pets,0456 </second>
<head> veterinarians treat pet drug overdose </head>
<dateline> atlanta (ap) </dateline>
<text>
nation another sign drug problem worse , veterinarians . animal doctor atlanta area treat increasing number patient disorient , seizure , coma undergo cardiac arrest eat owner 's drug .
...
</text>
</doc>

FIG. 2.2 – Le document de la figure 2.1 après les traitements

tels que "no", "not" et "never" qui peuvent changer le sens d'un mot ou phrase, et donc la pertinence d'un document pour une requête, sont compris dans le stopliste.

2.1.1 La taille des documents, avant et après traitement

Le tableau 2.1 présente des statistiques sur les mots simples de la collection globale, et le tableau 2.2 montre des statistiques sur les paires de mots dans la collection, où une paire comprend deux mots qui se trouvent dans une fenêtre d'au plus cinq mots, dans la même phrase. Le chapitre 4 élabore la définition et l'utilisation de paires de mots dans le

Mots simples	
# mots	19 359 407
# mots uniques	266 770
# hapax (mots avec fréq=1)	121 880
Fréq moy. des mots	73

TAB. 2.1 – Statistiques du corpus : mots simples

Paires	
# paires	68 886 121
# paires uniques	17 076 783
# hapax (paires avec fréq=1)	10 243 292
Fréq moy. des paires	4

TAB. 2.2 – Statistiques du corpus : paires de mots

Les mots les plus fréquents		
rang	mot	fréquence
1	"year"	109 285
2	"state"	85 380
3	"percent"	75 350
4	"million"	69 767
5	"government"	66 105

TAB. 2.3 – Les mots les plus fréquents dans la collection

Les paires les plus fréquentes		
rang	paire	fréquence
1	"associate press"	39 538
2	"press writer"	33 189
3	"associate writer"	33 118
4	"state united"	27 871
5	"soviet union"	13 336

TAB. 2.4 – Les paires les plus fréquentes dans la collection

contexte de la recherche d'information. Le tableau 2.5 montre le nombre de mots et paires par document. Tous les chiffres s'appliquent aux documents après lemmatisation et l'utilisation de la stopliste, sauf autrement indiqué. On observe qu'environ 46% des mots simples et 60% des paires sont des hapax, des entités n'apparaissant qu'une fois dans la collection. On peut remarquer aussi qu'après le pré-traitement, la taille moyenne des documents tombe de 462 mots (264 mots uniques) à 247 mots (165 mots uniques), ce qui signifie que 47% des mots (38% des mots uniques) ont été enlevés des documents. La figure 2.3 illustre la distribution des documents selon leur taille (comptée en mots), avant et après pré-traitement. Avant le pré-traitement, la distribution des documents de taille 100-600 mots est approximativement uniforme, et la plage de variation de la taille des documents est entre 100 et 800 mots pour plus de 90% des documents. Après le traitement, plus que 90% des documents ont moins de 500 mots, avec la plus grande partie de la distribution ayant entre 100 et 400 mots.

Les tableaux 2.3 et 2.4 montrent les mots et paires les plus fréquents dans le corpus.

2.2 Les requêtes

53 requêtes ont été utilisées pour toutes les expériences. Les requêtes sont les "cross language topics in English" de TREC-6 (requêtes 1-25) et TREC-7 (requêtes 26-53). Les requêtes sont structurées en format SGML, avec trois champs :

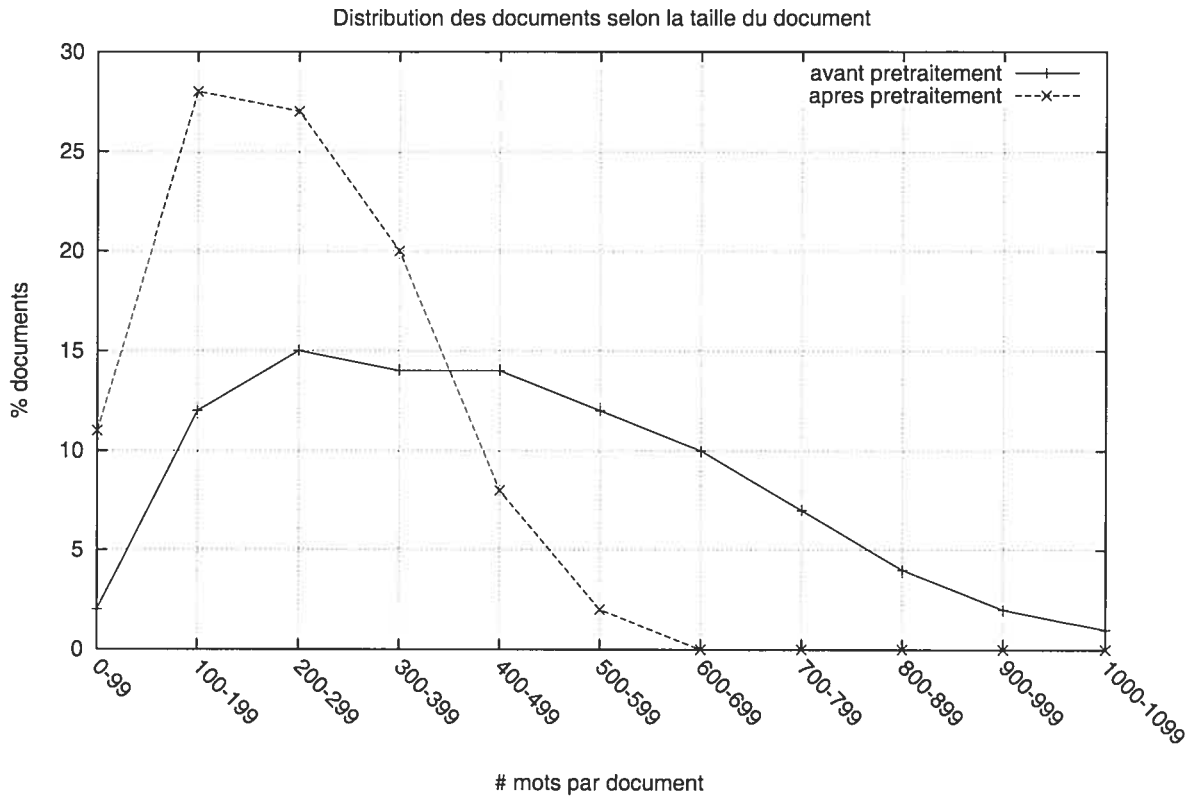

FIG. 2.3 – Distribution des documents selon le nombre de mots par document

	Min	Max	Moyen	Écart-type
Mots simples				
# mots avant lemmatisation/stopliste	4	1993	462	243
# mots uniques avant lemmatisation/stopliste	4	1474	264	124
# mots après pré-traitement	0	1574	247	127
# mots uniques après pré-traitement	0	1281	165	80

Paires			
	Min	Max	Moyen
# paires	0	6842	880
# paires uniques	0	6296	218

TAB. 2.5 – Chiffres par document : le nombre de mots et paires par document, avant et après pré-traitement. "Min" indique le document le plus petit, "Max" le document le plus grand, et "Moyen" la taille moyenne des documents dans la collection.

<top>
<num> Number : CL25
<E-title> European foreign laborers
<E-desc> Description :
Has the demand for foreign labor in Germany and Switzerland remained constant or changed since the fall of the Berlin wall ?
<E-narr> Narrative :
A relevant document should contain information on the change in job opportunities, working conditions and attitudes towards foreign labor in Germany and Switzerland since the fall of the Berlin wall which brought in an influx of cheap East European labor.
</top>

FIG. 2.4 – Exemple d'une requête

<top>
<title> european foreign laborers
<desc> demand foreign labor germany switzerland remain constant fall berlin wall ?
<top>

FIG. 2.5 – Exemple de la requête de la figure 2.4 après traitement

- titre
- description
- narrative

Le même traitement (lemmatisation et utilisation d'une stopliste) sur les documents est appliqué aux requêtes. Pour chaque expérience, deux tests ont été effectués : un test avec le champ "titre" et l'autre avec les deux champs "titre" et "description". La figure 2.4 montre une requête de la collection TREC-6 avant pré-traitement, et la figure 2.5 montre la même requête après la lemmatisation et l'utilisation de la stopliste.

Le tableau 2.6 montre la taille des requêtes avant et après pré-traitement. Les requêtes titres ne contiennent que de mots uniques. Le pré-traitement a un effet plus prononcé pour les requêtes plus longues, qui contiennent normalement plus de mots vides que les requêtes plus concises.

Le moteur de recherche AlltheWeb[1] fournit une page qui présente les 10 dernières requêtes faites au moteur, sur lesquelles aucun pré-traitement n'a été effectué. Nous avons consulté cette page et calculé, sur un ensemble de 1000 requêtes, une longueur moyenne de 2.4 mots par requête (avec une gamme de 1 à 10 mots). Les requêtes sur le web sont donc plutôt courtes, même en comparaison avec les requêtes titres utilisées dans notre travail.

[1] http ://www.alltheweb.com/recentqueries

	titre			titre&desc		
	Min	Max	Moyen	Min	Max	Moyen
# mots avant pré-traitement	1	8	3.2	5	46	15.8
# mots uniques avant pré-traitement	1	8	3.2	4	34	12.5
# mots après pré-traitement	1	5	2.5	4	26	9.3
# mots uniques après pré-traitement	1	5	2.5	3	19	6.9

TAB. 2.6 – Chiffres par requête

2.3 Système de référence (baseline)

Afin de disposer de performances de référence (baseline dans la suite) par rapport auxquelles comparer nos expériences, nous avons fait usage du système SMART. L'implantation du modèle vectoriel du système SMART a été utilisée pour les expériences du baseline. Le système SMART fournit une architecture flexible pour des expérimentations en recherche d'information, ainsi qu'une implantation par défaut du modèle vectoriel. Pour notre travail, SMART (ainsi que le système développé pour les expériences de ce projet) a été configuré pour retourner les 1000 premiers documents jugés pertinents par le système, pour chaque requête. Notre système sort les documents retrouvés dans le format utilisé par le programme d'évaluation de SMART, et le programme d'évaluation de SMART est utilisé pour toutes les expériences. Nous rappelons que les mêmes pré-traitements ont été effectués dans les deux cas, ce qui autorise une comparaison directe des résultats.

Chapitre 3

Modèles n-gramme en recherche d'information

Dans ce chapitre, nous étudions plusieurs techniques de lissage pour un modèle 1-gramme en recherche d'information et présentons la précision moyenne que nous avons obtenue pour chaque approche. Ensuite, nous présentons plusieurs approches à l'utilisation d'un modèle 2-gramme dans le score de pertinence d'un document pour une requête, ainsi que les résultats que nous avons obtenus avec le modèle 2-gramme.

3.1 Lissage des modèles n-gramme

Plusieurs techniques de lissage ont été proposées pour limiter le problème de sous-représentation de données. Le nom "lissage" prend son sens si l'on considère que l'opération consiste à redistribuer une proportion de la masse de probabilité attribuée par les modèles aux n-grammes vus à l'entraînement à ceux non vus. Les techniques de lissage combinent normalement un modèle n-gramme avec des modèles d'ordres inférieurs, qui souffrent moins du problème de sous-représentation de données. Il existe deux approches générales pour combiner plusieurs modèles, nommément la combinaison backoff et la combinaison interpolée. Nous décrivons maintenant l'essence de ces deux approches.

3.1.1 Lissage backoff

Dans le cas du backoff, un modèle d'ordre inférieur est consulté lorsqu'un n-gramme est inconnu du modèle ; ce que nous pouvons formuler par la récurrence suivante :

$$p_{bo}(w_i|w_{i-n+1}^{i-1}) = \lambda_2 \times p_{MLE}(w_i|w_{i-n+1}^{i-1}) + \theta_{w_{i-n+1}^{i}} \alpha_{w_{i-n+2}} \times p_{bo}(w_i|w_{i-n+2}^{i-1})$$

où

$$\theta_{w_{i-n+1}^{i}} = \begin{cases} 0 & \text{si } p_{MLE}(w_{i-n+1}^{i}) > 0 \\ 1 & \text{sinon} \end{cases} \tag{3.1}$$

La récursivité arrête par exemple avec le modèle 1-gramme :

$$p_{bo}(w) = p(w) \tag{3.2}$$

Dans le cas d'un modèle bigramme, la probabilité backoff est donnée par :

$$p_{bo}(w_i|w_{i-1}) = \lambda_2 \times p_{MLE}(w_i|w_{i-1}) + \theta_{w_{i-1}w_i} \alpha_{w_{i-1}} \times p(w_i) \tag{3.3}$$

Pour que le modèle backoff soit une distribution valide, il faut que les probabilités $p_{bo}(w_i|w_{i-1})$ somment à 1 sur l'ensemble des w_i pour tout w_{i-1}, et la valeur de α_{w-i} est donc calculé selon la valeur de λ_2.

$$\sum_{w_i} p_{bo}(w_i|w_{i-1}) = \sum_{w_i} \lambda_2 \times p_{MLE}(w_i|w_{i-1}) + \sum_{w_i} \theta_{w_{i-1}w_i} \alpha_{w_{i-1}} \times p(w_i) = 1 \tag{3.4}$$

Comme $\sum_{w_i} p_{MLE}(w_i|w_{i-1}) = 1$, nous avons :

$$\lambda_2 + \alpha_{w_{i-1}} \sum_{w_i} \theta_{w_{i-1}wi} p(w) = 1$$

et donc :

$$\alpha_{w_{i-1}} = \frac{1 - \lambda_2}{\sum_{w_i} \theta_{w_{i-1}w_i} p(w_i)} \tag{3.5}$$

3.1.2 Lissage interpolé

En ce qui concerne un modèle interpolé, les modèles d'ordres inférieurs sont toujours consultés lors du calcul, que l'on peut décrire par la récurrence :

$$p_{interp}(w_i|w_{i-n+1}^{i-1}) = \lambda_2 \times p_{MLE}(w_i|w_{i-n+1}^{i-1}) + (1 - \lambda_2) \times p_{interp}(w_i|w_{i-n+2}^{i-1}) \tag{3.6}$$

Là encore, on peut arrêter la récursivité avec un modèle 1-gramme, soit $p_{interp}(w) = p(w)$.

Selon Chen et Goodman [2], l'approche d'interpolation est plus adaptée quand les comptes des n-grammes sont faibles. Dans les sections 3.4 et 4.3 nous décrivons comment nous avons appliqué et évalué ces deux concepts de lissage. Pour une étude en profondeur des techniques de lissage et une comparaison de leur performance, voir [2] et [4].

3.2 Lissage du modèle 1-gramme

Le problème de sous-représentation de données, même dans le cas d'un modèle 1-gramme, devient plus sévère dans le cadre de la recherche d'information, où les documents sont en général très petits (de l'ordre de 200 mots en comparaison avec les 38 millions de mots utilisés dans les expériences décrites par exemple par Rosenfeld [15]). Si un document ne contient pas un mot de la requête, il est certainement trop sévère de lui attribuer une pertinence nulle.

3.2.1 Une probabilité globale pour les mots inconnus

Pour un modèle unigramme, l'approche de lissage la plus simple consiste à ajouter un mot inconnu "*unk*" au vocabulaire, de lui attribuer une probabilité fixe $p(unk)$ (la même probabilité pour tous les modèles des documents), et de renormaliser les probabilités des mots vus dans chaque document. Intuitivement, $p(unk)$ devrait être plus basse que la probabilité du mot le moins fréquent dans tous les documents du corpus, pour éviter qu'un mot inconnu dans un document ait une probabilité plus élevée qu'un mot vu dans un autre document. La probabilité d'un mot w vu dans un document d serait alors ajustée par :

$$p_{d_{fixe\_collection}}(w) = p_{d_{MLE}}(w) \times (1 - p(unk)) \tag{3.7}$$

Nous pouvons facilement vérifier que notre nouveau modèle répond à la contrainte stochastique, c'est à dire, que les probabilités de tous les mots somment à 1 :

$$\begin{aligned} p(unk) + \sum_{w \neq unk} p_{d_{fixe\_collection}}(w) &= p(unk) + \sum_{w \neq unk} p_{d_{MLE}}(w) \times (1 - p(unk)) \\ &= p(unk) + (1 - p(unk)) \sum_{w \neq unk} p_{d_{MLE}}(w) \\ &= p(unk) + 1 - p(unk) \\ &= 1. \end{aligned}$$

3.2.2 Une probabilité par document pour les mots inconnus

Une autre approche consiste à attribuer une probabilité $p_{d_{fixe\_doc}}(unk)$, par document, qui se base sur la probabilité la plus faible du document. Formellement :

$$p_{d_{fixe\_doc}}(unk) = \lambda_{min} \min_{w \in d} p_{d_{MLE}}(w) \tag{3.8}$$

où λ_{min} définit la masse de probabilité à attribuer à $p_d(unk)$ par rapport à la probabilité du mot le moins fréquent dans le document d. La probabilité lissée d'un mot connu devient alors :

$$p_{d_{fixe\_doc}}(w) = p_{d_{MLE}}(w) \times (1 - p_{d_{fixe\_doc}}(unk)) \tag{3.9}$$

Le problème principal avec cette approche est que la probabilité $p_{d_1 fixe\_doc}(unk)$ pour un document d_1 pourrait être plus grand que les probabilités des mots vus dans un autre document d_2, si le document d_1 est beaucoup plus petit que le document d_2. L'exemple suivant illustre le problème :

Soit la requête q = "famine sudan", et les deux documents dont les caractéristiques sont décrites dans le tableau 3.1.

Le modèle avec une probabilité $p_{d_{fixe\_doc}}(unk)$ par document donne un score plus élevé au document non-pertinent d_1 qu'au document pertinent d_2, parce que la probabilité $p_{d_1 fixe\_doc}(unk)$ est plus grande que les probabilités des mots vus dans le document d_2.

document	nombre de mots	description
d_1	10	non-pertinent, ne contient ni "famine" ni "sudan"
d_2	94	pertinent, contient "famine" et "sudan"

	$p_{d_{fixe\_doc}}(unk)$ par document				$p(unk)$ fixe pour tout document d			
document	p(famine)	p(sudan)	p(unk)	p(q|d)	p(famine)	p(sudan)	p(unk)	p(q|d)
d_1	-	-	0.091	0.0083	-	-	0.0001	10E-8
d_2	0.015	0.055	-	0.0009	0.015	0.055	-	0.0009

TAB. 3.1 – $p(unk)$ globale versus $p_d(unk)$ par document

3.2.3 Lissage avec un modèle de corpus

Les approches que nous avons décrites accordent toutes les deux une probabilité fixe (soit par document, soit par collection) à tous les mots inconnus. En réalité, tous les mots inconnus ne devraient pas nécessairement avoir la même probabilité. Considérons l'exemple d'un corpus de documents dans le domaine du droit, et une requête avec les deux mots "loi" et "immigration". Dans un tel corpus, le mot "immigration" est plus discriminant entre les documents que le mot "loi", et donc le terme manquant "immigration" ne devrait pas recevoir la même probabilité que le terme manquant "loi". Les approches classiques de recherche d'information comme le modèle vectoriel résolvent ce problème par la prise en compte du facteur *idf* dans l'indexation.

Dans le cadre d'un modèle de langue, une solution est de lisser les probabilités avec un modèle de corpus. Un modèle de corpus est entraîné pour maximiser la vraisemblance de la collection complète de documents, selon l'équation 1.14, si l'on fait l'hypothèse que la collection est assez grande et contient tous les mots simples qui seront utilisés dans toutes les requêtes.

Lissage interpolé

Une façon d'incorporer le modèle de corpus aux modèles de document est d'utiliser une probabilité interpolée entre les modèles du document et du corpus :

$$p_{d_{corpus\_interp}}(w) = \lambda_d p_{d_{MLE}}(w) + (1 - \lambda_d) p_{corpus_{MLE}}(w)) \tag{3.10}$$

L'exemple suivant montre comment le lissage avec le modèle du corpus tient compte du pouvoir de discrimination des mots manquants :

document	$p(loi)$	$p(immigration)$
d_1	0.001	0.001
d_2	-	0.001
d_3	0.001	-
corpus	0.0001	0.00001

Pour la requête "loi immigration" dans un corpus du domaine du droit, le document d_1 qui contient les deux mots est à priori le plus pertinent. De même, le document d_2 est intuitivement plus pertinent, car il contient le mot plus discriminant, "immigration". d_2 ne contient pas le terme "loi", mais comme "loi" n'est pas discriminant, ce document ne devrait pas être pénalisé autant que d_3.

Si on emploie le lissage de l'équation 3.10, avec λ_d=0.4 pour illustration, les calculs de pertinence pour les trois documents sont :

$$\begin{aligned}
p_{d1}(q) &= [0.4(0.001) + 0.6(0.0001)] \times [0.4(0.001) + 0.6(0.00001)] = 0.00046 \times 0.000406 \\
&= 1.9E-7 \\
p_{d2}(q) &= [0 + 0.6(0.0001)] \times [0.4(0.001) + 0.6(0.00001)] = 0.00006 \times 0.000406 = 2.4E-8 \\
p_{d3}(q) &= [0.4(0.001) + 0.6(0.0001)] \times [0 + 0.6(0.00001)] = 0.00046 \times 0.000006 = 2.8E-9
\end{aligned}$$

On voit ici que les documents sont ordonnés en accord avec notre intuition.

Il reste cependant à trouver la valeur optimale de λ_d. Dans le cas extrême où $\lambda_d = 1$, on revient au problème où un document recevra un score nul s'il manque un terme de la requête. À l'autre extrémité, si $\lambda_d = 0$, seulement les probabilités du corpus seraient utilisées et aucune discrimination ne serait faite entre les documents. Une façon heuristique pour déterminer les contributions relatives des modèles du document et du corpus consiste à établir une contrainte où la contribution optimale du modèle discriminant, le modèle du document, est supérieure à celle du modèle du corpus, pour les mots vus dans le document :

$$\begin{aligned}
\lambda_d p_{d_{MLE}}(w) &> (1 - \lambda_d) p_{corpus}(w), \forall w \in d \\
\lambda_d p_{d_{MLE}}(w) &> p_{corpus}(w) - \lambda_d p_{corpus}(w) \\
\lambda_d \left(p_{d_{MLE}}(w) + p_{corpus}(w)\right) &> p_{corpus}(w)
\end{aligned}$$

Cette contrainte nous semble intuitive, car si nous avons une estimation non-nulle du modèle du document, cette estimation est plus spécifique que l'estimation du corpus, et donc à utiliser en priorité. Nous avons alors :

$$\lambda_d > \frac{p_{corpus}(w)}{p_{d_{MLE}}(w) + p_{corpus}(w)}, \forall w \in d \tag{3.11}$$

pour tout mot w vu dans le document d, et pour tout document d, pour entraîner une valeur globale de λ_d pour la collection. Cependant, d'autres approches pour déterminer les contributions optimales des deux modèles sont aussi possibles. Dans nos travaux, la valeur optimale de λ_d est déterminée de manière empirique en vérifiant la contraint formulée par l'équation 3.11.

Lissage backoff

Une autre façon d'incorporer le modèle du corpus est avec une approche backoff, où les probabilités du corpus sont utilisées seulement si un mot n'existe pas dans le document :

$$p_{d_{corpus\_bo}}(w) = \lambda_d p_{d_{MLE}}(w) + \theta_w \alpha \times p_{corpus_{MLE}}(w))$$

où

$$\theta_w = \begin{cases} 1 & \text{lorsque } p_{d_{MLE}}(w) = 0 \\ 0 & \text{sinon} \end{cases} \tag{3.12}$$

Il faut déterminer α en fonction de λ_d, en respectant la contrainte stochastique :

$$\begin{aligned} \sum_w p_{d_{corpus\_bo}}(w) &= \sum_w \lambda_d p_{d_{MLE}}(w) + \sum_w \theta_w \alpha \times p_{corpus_{MLE}}(w)) = 1 \\ 1 &= \lambda_d + \alpha \sum_w \theta_w p_{corpus_{MLE}}(w) \end{aligned}$$

et alors :

$$\alpha = \frac{1 - \lambda_d}{\sum_w \theta_w p_{corpus_{MLE}}(w)} = \frac{1 - \lambda_d}{1 - \sum_{w \in d} p_{corpus_{MLE}}(w)} \tag{3.13}$$

Pour un λ_d donné et un mot inconnu, l'approche backoff attribue une probabilité plus élevée que l'approche interpolée, au mot inconnu, car $\alpha > (1 - \lambda_d)$. Il suit que, pour que la contribution des mots vus soit supérieure à celle des mots inconnus, la valeur optimale de λ_d pour l'approche backoff doit être plus grande que dans l'approche interpolée.

Lisser le modèle de corpus

Un problème avec l'incorporation du modèle de corpus est qu'il est possible qu'un mot qui se trouve très fréquemment dans le corpus mais qui est absent dans un document peut contribuer plus au score de pertinence que les mots vus dans le modèle du document. Une façon de contourner ceci est d'utiliser une stopliste comme nous l'avons décrit dans la section 1.1.3.

Il est également possible qu'une requête contienne des mots qui ne se trouvent pas dans le corpus. Dans ce cas, un lissage doit être appliqué au modèle du corpus, de la même manière que pour les modèles de document (voir l'équation 3.7). On accorde une probabilité aux mots inconnus du corpus, $p_{corpus}(unk)$, et la probabilité d'un mot vu dans le corpus devient :

$$p_{corpus\_fixe}(w) = p_{corpus_{MLE}}(w) \times (1 - p_{corpus}(unk)) \tag{3.14}$$

Alors, la probabilité interpolée entre le modèle du document d et le modèle du corpus devient :

$$p_{d_{corpus\_interp\_fixe}}(w) = \lambda_d p_{d_{MLE}}(w) + (1 - \lambda_d) p_{corpus\_fixe}(w)) \tag{3.15}$$

Et la probabilité du mot avec l'approche backoff devient :

$$p_{d_{corpus\_bo\_fixe}}(w) = \lambda_d p_{d_{MLE}}(w) + \theta_w \alpha \times p_{corpus\_fixe}(w)) \tag{3.16}$$

3.3 Expériences avec un modèle unigramme

Nous avons testé la performance de l'utilisation d'un modèle unigramme pour la recherche d'information. Les premiers 1000 documents retrouvés par le système sont gardés et ordonnés en ordre décroissant du score calculé par l'équation 1.19, et les techniques de lissage que nous avons décrites pour $p_d(w)$ ont été testées. Le tableau 3.2 montre la précision obtenue avec ces approches :

- $p_{MLE}(w)$ (eq 1.14), aucun lissage
- $p_{d_{fixe\_collection}}(w)$, avec une probabilité fixe $p(unk)$ accordée aux mots inconnus (eq 3.7)
- $p_{d_{fixe\_doc}}(w)$, avec une probabilité $p_{d_{fixe\_doc}}(unk)$ par document (eq 3.9), avec plusieurs valeurs de λ_{min}.

Nous rapportons les résultats des approches d'intégration du modèle de corpus dans le tableau 3.3 :

	test	équation de $p_d(w)$	titres	titres et desc.
	baseline		0.3349	0.3498
1	$p_{fixe\_collection}(unk)$=10E-3, $\forall$ doc d	3.7	0.3536	0.3464
2	$p_{fixe\_collection}(unk)$=10E-4, $\forall$ doc d	3.7	0.3574	0.3440
3	$p_{fixe\_collection}(unk)$=10E-5, $\forall$ doc d	3.7	0.3639	0.3226
4	$p_{fixe\_collection}(unk)$=10E-6, $\forall$ doc d	3.7	0.3637	0.2980
5	$p_{fixe\_collection}(unk)$=10E-7, $\forall$ doc d	3.7	0.3637	0.2982
6	$p_{d_{fixe\_doc}}(unk)$ par document,λ_{min}=0.001	3.9	0.3544	0.3012
7	$p_{d_{fixe\_doc}}(unk)$ par document,λ_{min}=0.00001	3.9	0.3546	0.2839
8	$p_{d_{fixe\_doc}}(unk)$ par document,λ_{min}=0.01	3.9	0.3396	0.2732
9	$p_{d_{fixe\_doc}}(unk)$ par document,λ_{min}=0.1	3.9	0.2833	0.1343
10	aucun lissage,$p_{d_{MLE}}$	1.14	0.2707	0.0412
11	$p_{fixe\_collection}(unk)$=0.01, $\forall$ doc d	3.7	0.1633	0.1210
12	$p_{d_{fixe\_doc}}(unk)$ par document,λ_{min}=1	3.9	0.1114	0.0075

TAB. 3.2 – Précision moyenne avec les modèles 1-gramme testés

		interpolé			backoff		
	test	équation	titre	titre&dsc	équation	titre	titre&dsc
1	p_{corpus_fixe}, (λ_d=0.2)	3.15	0.4067	0.4275	3.16	0.3834	0.4091
2	p_{corpus_fixe}, (λ_d=0.4)	3.15	0.4053	0.4241	3.16	0.4052	0.4245
3	p_{corpus_fixe}, (λ_d=0.6)	3.15	0.4071	0.4139	3.16	0.4099	0.4148
4	$p_{corpus_{MLE}}$, (λ_d=0.1)	3.10	0.3799	0.3592	3.12	0.3633	0.3255
5	$p_{corpus_{MLE}}$, (λ_d=0.2)	3.10	0.4064	0.3783	3.12	0.3833	0.3629
6	$p_{corpus_{MLE}}$, (λ_d=0.4)	3.10	0.4050	0.3764	3.12	0.4051	0.3781
7	$p_{corpus_{MLE}}$, (λ_d=0.6)	3.10	0.4068	0.3663	3.12	0.4099	0.3683
8	$p_{corpus_{MLE}}$, (λ_d=0.8)	3.10	0.3949	0.3461	3.12	0.3967	0.3475
	baseline		0.3349	0.3498		0.3349	0.3498

TAB. 3.3 – Précision moyenne avec l'intégration du modèle du corpus

- $p_{d_{corpus\_interp}}(w)$, un lissage interpolé des modèles de document et de corpus, avec plusieurs valeurs de λ_d (eq 3.10)
- $p_{d_{corpus\_bo}}(w)$, un lissage backoff des modèles de document et de corpus, avec plusieurs valeurs de λ_d (eq 3.12)
- $p_{d_{corpus\_interp\_fixe}}(w)$, une interpolation entre les modèles de document et corpus avec une probabilité fixe $p_{corpus}(unk)$ pour les mots inconnus dans le corpus. (eq 3.15)
- $p_{d_{corpus\_interp\_fixe}}(w)$, un lissage backoff avec une probabilité fixe $p_{corpus}(unk)$ pour les mots inconnus dans le corpus. (eq 3.16)

Nous présentons maintenant des commentaires généraux sur les performances des approches évaluées. Ensuite, une discussion sur les approches de lissage fixe est donné, suivie par une analyse des approches qui intègrent un modèle de corpus.

3.3.1 Comparaison des techniques de lissage

À l'exception de quelques lissages de $p_d(unk)$ par document ou $p(unk)$ fixe pour tous documents, toutes les techniques de lissage amènent un gain en précision moyenne, par rapport à l'approche où aucun lissage n'est effectué (tab. 3.2, ligne 10). Les gains sont compris entre 5% (avec $p_d(unk)$ par document, tab. 3.2, ligne 9) et 50% (l'interpolation avec le modèle de corpus avec $p_{corpus}(unk)$ fixe, tab. 3.3, ligne 3) pour les requêtes titres et entre 225% (tab. 3.2, ligne 9) et 938% (tab. 3.3, ligne 1) pour les mêmes techniques et les requêtes titres et descriptions, selon le calcul de gain suivant :

$$gain(a,b) = \frac{(\text{précision moyenne})_a - (\text{précision moyenne})_b}{(\text{précision moyenne})_b} \times 100 \qquad (3.17)$$

où le gain est l'amélioration en performance apportée par la technique de lissage a, par rapport à la technique de lissage b.

On observe que le lissage de $p_d(unk) = \min_{w \in d} p_d(w)$ (tab. 3.2, ligne 12) par document diminue la performance. Cela est expliqué par le fait que la probabilité d'un mot inconnu dans un document court est supérieure aux probabilités des mots vus dans un document long ; situation dont nous avons déjà discuté. Utiliser une probabilité $p_d(unk)$ plus faible, ou attribuer une probabilité fixe à un mot inconnu, $p(unk)$, à travers tous les documents, résout ce problème. Les lissages d'intégration du modèle du corpus, qui tiennent compte du pouvoir de discrimination des mots manquants, accroissent encore la précision, et permettent au système de dépasser le baseline, que nous avons décrit dans la section 2.3.

3.3.2 Un lissage fixe (tableau 3.2)

Sans aucun lissage (ligne 10), ou avec le lissage de $p_d(unk) = \min_{w \in d} p_d(w)$ (ligne 12) par document, la précision mesurée avec les requêtes de titres et descriptions est inférieure de 1%, par rapport à une précision de 27% sans aucun lissage (ligne 10), et 11% pour le discount $p_d(unk)$ (ligne 12) pour les requêtes de 2-3 mots. Au fur et à mesure que la longueur de la requête augmente, la probabilité qu'un document pertinent ne contienne pas un mot dans la requête augmente. Alors, la probabilité qu'un document pertinent reçoive un score nul si aucun lissage n'est utilisé (ligne 10) est plus elevée pour les requêtes longues. Le problème de mots inconnus est donc plus sévère pour les requêtes longues.

Une probabilité trop élevée pour les mots inconnus

Pour les requêtes courtes, les résultats ne sont pas très sensibles à la valeur de $p(unk)$ fixe pour tous documents, si $p(unk) < 0.001$. Pour les requêtes titres et descriptions, la précision a une plage de variation de 5% (de 30% à 35%) en précision moyenne, dépendant de la valeur de $p(unk)$. Cependant, peu importe la longueur de la requête, si la probabilité attribuée aux mots inconnus est trop élevée (par exemple, 0.01, ligne 8), la performance diminue. Comme la longueur moyenne des documents dans cette collection est environ 200 mots, la probabilité la plus faible par document, en moyenne, est environ $1/200 = 0.005$. Si $p(unk) = 0.01$, plus de poids est accordé aux mots inconnus qu'aux mots vus dans le document.

Le même phénomène est observé si la probabilité $p_d(unk)$ varie selon le document. Si on attribue la probabilité la plus faible dans le document à $p_d(unk)$ ($\lambda_{min} = 1$, ligne 12), la performance diminue par rapport à aucun lissage. Là aussi, trop de poids relatif est alloué aux mots inconnus. La précision augmente si $p_d(unk)$ est plutôt faible par rapport aux mots vus, avec une valeur optimale de $\lambda_{min} = 0.001$.

Lissage fixe : requêtes courtes versus requêtes longues

Avec un lissage fixe $p(unk)$, soit par document, soit global, le modèle unigramme dépasse le baseline avec les requêtes titres, mais pas avec les requêtes titres et description. Comme nous l'avons décrit, le problème de mots inconnus est plus sévère pour les requêtes longues. Pour les requêtes titres qui sont habituellement plutôt courtes, il est rare que plus d'un mot soit absent dans un document pertinent. Attribuer une probabilité fixe $p(unk)$ à un mot de la requête ne présente pas le même problème qu'on observe dans les requêtes titres et descriptions qui sont plus longues : que si plusieurs mots de la requête sont absents dans le document, ils ne doivent pas contribuer au score avec le même poids (comme dans notre exemple de la requête "immigration loi" et un corpus du domaine de droit). L'intégration du modèle du corpus résout ce problème pour les requêtes titres et descriptions, comme nous avons décrit dans la section 3.2.3.

3.3.3 L'intégration du modèle du corpus (tableau 3.3)

Les approches qui intègrent le modèle de corpus dans le lissage donnent de meilleurs résultats par rapport à un lissage fixe pour les mots inconnus. La valeur de $p_{corpus}(unk) = 10E - 9$ a été choisie pour être plus basse que la probabilité du mot le moins fréquent du corpus, soit 10E-7.

Modèle du corpus non-lissé

En ce qui concerne la valeur de λ_d dans les lissages interpolés et backoff, sans une probabilité fixe $p_{corpus}(unk)$ (lignes 4 à 8), la précision atteinte varie entre 38% et 41% pour les requêtes titres, et a aussi une plage de variation de 3% (entre 35% et 38%) en précision moyenne, pour les requêtes titres et descriptions, pour des valeurs de λ_d entre 0.1 et 0.8, pour l'approche d'interpolation. L'approche backoff est légèrement plus sensible aux valeurs de λ_d avec des plages de variation de 5%, entre 36% et 41%, et entre 33% et 38% pour les requêtes titres, et titres et descriptions, respectivement.

Modèle du corpus lissé

L'utilisation d'une probabilité fixe pour les mots non vus dans le corpus, $p_{corpus}(unk)$, a un effet positif et cet effet est plus important sur les requêtes longues, dû à la possibilité élevée d'avoir des mots inconnus dans les documents pertinents. Employer un $p_{corpus}(unk)$ permet d'augmenter la précision moyenne de 38% à 43% sur les requêtes titres et descriptions (avec $\lambda_d = 0.2$, lignes 5 et 1) tandis que l'amélioration amenée par $p_{corpus}(unk)$ n'augmente la précision que de 0.0003%, (de 40.68% jusqu'à 40.71%), pour les requêtes titres (avec $\lambda_d = 0.6$, lignes 7 et 3). Ces résultats ne sont pas très sensibles à la valeur de λ_d. Pour l'approche d'interpolation, les plages de variation de précision pour les valeurs de λ_d entre 0.2 et 0.6 (lignes 1 à 3) sont de 0.18% et 1.4% pour les requêtes titres, et titres et descriptions, respectivement. Cette gamme est légèrement plus élevée avec l'approche backoff, avec des plages de variation de 2.6% et 1.5% pour les requêtes titres, et titres et descriptions, respectivement.

Interpolation vs backoff

En général, les approches interpolation et backoff donnent des résultats comparables. Nous pouvons remarquer cependant que lorsque l'influence du corpus est grande (λ_d est petit), la précision moyenne obtenue par l'approche backoff est inférieure à celle de l'interpolation. Ceci est dû, comme nous l'avons mentionné dans la section 3.2.3, au fait que la probabilité allouée aux mots inconnus est plus grande avec l'approche backoff qu'avec le lissage d'interpolation, pour un λ_d donné. Alors pour les valeurs petites de λ_d, une contribution trop élevée des mots inconnus dans le score est plus probable avec le lissage backoff.

3.3.4 Les meilleures approches du modèle 1-gramme

Les techniques de lissage qui donnent les meilleures performances sont celles qui combinent les probabilités du document et du corpus, en particulier avec un modèle du corpus lissé avec une probabilité fixe $p_{corpus}(unk)$, pour prendre en compte les mots non vus dans le corpus (l'équation 3.15). Nous décrivons dans la prochaine nos tentatives pour améliorer encore la performance à l'aide d'un modèle bigramme.

3.4 Expériences avec un modèle bigramme

Nous avons testé l'efficacité de l'utilisation d'un modèle bigramme pour le score de pertinence (équation 1.20). Les approches de lissage backoff (équation 3.1) et interpolation (équation 3.6) ont été évaluées. Nous avons investigué les améliorations possibles avec un modèle bigramme, par rapport aux trois approches du modèle unigramme : un lissage fixe pour les mots inconnus ($p_{d_{fixe\_collection}}(w)$, équation 3.7), et les deux approches d'intégration du modèle de corpus (interpolation ($p_{d_{corpus\_interp\_fixe}}(w)$, équation 3.15) et backoff ($p_{d_{corpus\_bo\_fixe}}(w)$, équation 3.16)). Les meilleurs paramètres, selon les expériences de la section 3.3, ont été utilisés pour les modèles unigrammes. L'approche d'intégration avec le modèle de corpus, backoff (équation 3.12) ou interpolation (équation 3.10), correspond à l'approche d'intégration des modèles 1-gramme et 2-gramme, backoff (équation 3.1) ou interpolation (équation 3.6). Spécifiquement, les formules combinées testées sont :

- Un modèle bigramme où le modèle unigramme est lissé avec $p(unk)$ fixe :
  - INTERPFIXE : Interpolation avec le modèle 2-gramme :

$$p_{interp}(w_i|w_{i-1}) = \lambda_2 \times p_{d_{MLE}}(w_i|w_{i-1}) + (1-\lambda_2) \times p_{d_{fixe\_collection}}(w_i) \qquad (3.18)$$

  - BOFIXE : Backoff avec le modèle 2-gram :

$$p_{bo}(w_i|w_{i-1}) = \lambda_2 \times p_{d_{MLE}}(w_i|w_{i-1}) + \theta_{w_{i-1}w_i}\alpha_{w_{i-1}} \times p_{d_{fixe\_collection}}(w_i) \qquad (3.19)$$

- INTERP2 : Le lissage interpolé, avec deux interpolations : une interpolation du modèle unigramme avec le modèle de corpus, et une interpolation entre les modèles 1-gramme

et 2-gramme :

$$p_{interp}(w_i|w_{i-1}) = \lambda_2 \times p_{d_{MLE}}(w_i|w_{i-1}) + (1-\lambda_2) \times [\lambda_d p_{d_{MLE}}(w_i) + (1-\lambda_d) p_{corpus\_fixe}(w_i)] \quad (3.20)$$

- BO2 : Le lissage backoff : un lissage backoff du modèle unigramme avec le modèle de corpus, et un lissage backoff entre les modèles 1-gramme et 2-gramme :

$$p_{bo}(w_i|w_{i-1}) = \lambda_2 \times p_{d_{MLE}}(w_i|w_{i-1}) + \theta_{w_{i-1}w_i} \alpha_{w_{i-1}} \times [\lambda_d p_{d_{MLE}}(w_i) + \theta_w \alpha p_{corpus\_fixe}(w_i)] \quad (3.21)$$

Les résultats de ces approches sont décrits dans les tableaux 3.5 et 3.6 et discutés dans les sections qui suivent.

3.4.1 Résultats des approches d'interpolation

On observe que pour les approches d'interpolation (INTERPFIXE et INTERP2, dans le tableau 3.5), la précision moyenne obtenue est équivalente à celle du modèle 1-gramme lorsque $\lambda_2 = 0$. De plus, on peut voir que les approches d'interpolation se comportent mieux avec des valeurs relativement petites de λ_2. Les meilleurs résultats sont obtenus avec λ_2=0.01 pour l'approche INTERPFIXE et λ_2=0.0001 pour l'approche INTERP2.

Les contributions relatives des bigrammes et mots simples au score de pertinence

Afin de comprendre l'importance des contributions relatives des bigrammes et des mots simples, considérons un document typique de 200 mots. Si un mot typique se trouve dans le document 1 à 5 fois, la probabilité du mot sera entre $\frac{1}{200}$ et $\frac{5}{200}$, soit de 0.005 à 0.025. Le mot peut se trouver toujours dans le même contexte, ou dans jusqu'à 5 contextes différents. Une probabilité typique pour un bigramme $p(w_i|w_{i-1})$ serait alors entre $\frac{1}{5}$ et 1, soit de 0.2 à 1. Les probabilités des bigrammes sont alors considérablement plus élevées que celles des mots simples.

Une contribution trop élevée des bigrammes

Peu importe la façon dont les modèles 1-gramme et 2-gramme sont intégrés (interpolation ou backoff), une contribution trop élevée des bigrammes amène une perte de précision. Le tableau 3.4 illustre le problème. Ici, le document d_1 contient les trois mots individuels de

	d_1	d_2
$p(w_1)$	0.01	0.01
$p(w_2)$	0.01	0.01
$p(w_3)$	0.01	$p(unk) = 0.001$
$p(w_2\|w_1)$	0	0.8
$p(w_3\|w_2)$	0	0

	$\lambda_2 = 0.1$		$\lambda_2 = 0.6$	
	d_1	d_2	d_1	d_2
$p(w_1)$	0.01	0.01	0.01	0.01
$\lambda_2 p(w_2\|w_1) + (1-\lambda_2)p(w_2)$	0.009	0.089	0.004	0.484
$\lambda_2 p(w_3\|w_2) + (1-\lambda_2)p(w_3)$	0.009	0.0009	0.004	0.0004
score(d,q)	8.1E-7	8.01E-7	1.6E-7	1.9E-6

TAB. 3.4 – Exemple de deux documents avec 3 mots simples et 2 bigrammes, et l'effet de λ_2

la requête, mais ne contient pas les bigrammes de la requête. Un deuxième document d_2 ne contient que deux des trois mots, mais ces deux mots sont dans le même contexte (bigramme) que dans la requête. Il n'est pas complètement évident dans cet exemple de déterminer lequel de ces deux documents est le plus pertinent pour la requête, mais on peut voir comment la valeur de λ_2 peut causer une différence entre les scores des deux documents trop élevée. Avec une valeur de λ_2=0.1, les deux documents ont des scores presque égaux. Par contre, si la contribution des bigrammes augmente à λ_2=0.6, le document d_2 a un score 8 fois supérieur au score de celui du document d_1.

Les résultats

L'approche INTERPFIXE amène une amélioration marginale au modèle unigramme lissé avec un $p(unk)$ fixe (l'équation 3.7), soit une augmentation de précision moyenne de 0.53% (de 35.36% à 35.89%) pour les requêtes titres, et une augmentation de 0.80% (de 34.64% à 35.44%) pour les requêtes titres et descriptions. L'amélioration obtenue par l'approche INTERP2, par rapport au modèle unigramme lissé avec le corpus (l'équation 3.15), est encore moins significative, avec un gain de 0.01% (de 40.71% à 40.72%) pour les requêtes titres, et 0.02% (de 42.75% à 42.77%) pour les requêtes titres et descriptions.

3.4.2 Résultats des approches backoff

Les approches backoff testées (dans le tableau 3.6) n'ont pas dépassé en performance les modèles unigrammes. La meilleure précision moyenne réalisée par l'approche BOFIXE est 33.39% pour les requêtes titres (soit 2.0% inférieur au 35.36% du modèle 1-gramme) et

	INTERPFIXE $p(unk) = 10E-3$		INTERP2 $\lambda_d = 0.6$	INTERP2 $\lambda_d = 0.2$
λ_2	titre	titre&desc	titre	titre&desc
modèle 1-gramme	0.3536	0.3464	0.4071	0.4275
0	0.3536	0.3464	0.4071	0.4275
0.0001	0.3536	0.3464	0.4072	0.4277
0.001	0.3545	0.3473	0.4070	0.4271
0.01	0.3589	0.3544	0.4067	0.4183
0.1	0.3408	0.3390	0.3930	0.4117

TAB. 3.5 – Précision moyenne avec le modèle bigramme, approches d'interpolation

33.35% pour les requêtes titres et descriptions (soit 1.3% inférieur au 34.64% du modèle 1-gramme). Pour l'approche BO2, la perte de précision est de 1.9% pour les requêtes titres (de 40.99% à 39.13%) et de 1.5% pour les requêtes titres et descriptions (de 42.45% à 40.98%).

Une contribution trop faible des bigrammes

Nous avons déjà discuté de la perte de précision qui se produit lorsque la contribution des bigrammes λ_2 est trop élevée. De l'autre côté, pour l'approche backoff, une contribution trop petite des bigrammes présente un autre problème qui ne s'applique pas à l'intégration interpolée. Prenons le cas extrême, où λ_d=0. Dans cette situation, un mot d'un bigramme de la requête, pour lequel le bigramme se trouve aussi dans le document, aura une contribution nulle au score, car les probabilités des mots simples ne contribuent au score que si le contexte est absent du document. Dans ce cas, le score attribué au document par le modèle backoff est nul, tandis que le score calculé par le modèle interpolé est équivalent au score du modèle 1-gramme. Le problème de sous-représentation de données est plus sévère pour les bigrammes que pour les mots simples, surtout avec un document de seulement 100 à 200 mots. Ceci implique qu'il est préférable de toujours consulter un modèle 1-gramme, plutôt que de faire confiance uniquement au modèle 2-gramme, lorsqu'un bigramme de la requête est présent dans le document. Ceci est conforme aux observations de Chen et Goodman [2] qui confirment que les approches backoff sont plus adaptées lorsque les comptes des bigrammes sont élevés.

3.4.3 Amélioration des bigrammes versus du lissage des mots simples

On peut considérer l'approche 1-gramme avec un lissage fixe $p(unk)$ (équation 3.7) comme une approche de base que l'on cherche à améliorer. Les approches que nous avons investiguées pour une amélioration par rapport à l'équation 3.7 sont les modèles 1-gramme intégrés avec

	BOFIXE $p(unk) = 10E-3$		BO2 $\lambda_d = 0.6$	BO2 $\lambda_d = 0.4$
λ_2	titre	titre&desc	titre	titre&desc
modèle 1-gramme	0.3536	0.3464	0.4099	0.4245
0.001	0.2187	0.2439	0.3472	0.3630
0.01	0.3022	0.3061	0.3876	0.3976
0.1	0.3339	0.3335	0.3913	0.4048
0.2	0.3280	0.3274	0.3906	0.4071
0.4	0.3196	0.3185	0.3820	0.4085
0.6	0.3136	0.2993	0.3763	0.4098
0.8	0.2968	0.2734	0.3726	0.4017

TAB. 3.6 – Précision moyenne avec le modèle bigramme, approche backoff

le modèle de corpus (équations 3.10 et 3.15) et l'intégration avec un modèle 2-gramme (INTERPFIXE et BOFIXE). Le modèle 1-gramme qui intègre le modèle du corpus permet une amélioration significative (une hausse en précision de 5.4% (de 35.36% à 40.71%) pour les requêtes titres et de 8.1% (de 34.64% à 42.75%) pour les requêtes titres et descriptions), tandis que les gains respectifs obtenus par le modèle 2-gramme sont au mieux de 0.53% et 0.80%. Cela semble indiquer que lorsque les modèles n-gramme sont utilisés pour la recherche d'information où les documents sont petits, il est important d'optimiser le modèle 1-gramme avant d'intégrer des modèles d'ordre supérieur.

Chapitre 4

Affinités lexicales

4.1 Introduction

L'hypothèse d'indépendance entre mots du modèle 1-gramme (et d'une grande partie des approches à la recherche d'information) n'est pas toujours justifiée. Par exemple, supposons qu'un usager précise la requête "information retrieval," et qu'il existe deux documents dans la collection qui contiennent ces deux mots, avec des probabilités égales. Le premier document parle de la recherche d'information dans le cadre des moteurs de recherche, et contient le terme composé "information retrieval" une ou plusieurs fois. L'autre document ne traite pas des moteurs de recherche et les mots "information" et "retrieval" ne se trouvent pas ensembles. Le modèle 1-gramme donnerait le même score aux deux documents, malgré le fait que le premier soit probablement plus pertinent.

Une approche possible pour palier ce problème consiste à combiner le modèle unigramme avec un modèle bigramme voire trigramme. Ces modèles sont utiles dans des applications de traitement des langues naturelles, telles que la reconnaissance de la parole et la traduction automatique, où un modèle est utilisé pour prédire la probabilité qu'un mot suive les deux ou trois mots qui précédent. Cependant, pour la recherche d'information, l'ordre des mots dans une requête n'est pas toujours pertinent. Souvent, l'usager précise juste deux ou trois mots-clés qui ne forment pas une phrase, et qu'il/elle s'attend à voir proches, dans un ordre arbitraire, dans les documents pertinents. Quelques exemples sont : "louer appartement" où "bus Montréal Ottawa". Dans le premier exemple, l'usager s'attend à trouver des documents avec des phrases comme "louer un appartement" ou "appartements à louer". Dans le deuxième, si l'usager veut faire un voyage aller retour, les documents avec les phrases "au-

tobus de Montréal à Ottawa" et "autobus entre Ottawa et Montréal" serait tous les deux pertinents. En plus, comme nous l'avons décrit dans le chapitre 3, nous n'avons pas mesuré de performance intéressante avec l'utilisation des modèles bigrammes.

Un autre type d'unité qui tient compte des relations entre mots, sans contrainte d'ordre, est *l'affinité lexicale*. Maarek [11] définit une affinité lexicale comme une représentation d'une corrélation de co-occurrences entre deux mots d'une langue. Ici le terme affinité lexicale sera utilisé pour signifier toute paire de mots se trouvant dans un voisinage proche (une notion que nous préciserons). D'autres approches qui traitent des termes composés limitent parfois les paires aux mots dans des groupes nominaux (ex : "appareil photo") ou des mots dans une relation particulière comme modificateur-modifié (ex : "premier ministre") [11]. Ne faisant pas usage d'un tagger syntaxique dans ce travail, nous considérons toutes les paires possibles dans une fenêtre de taille fixe (de cinq mots par exemple).

Maarek introduit le concept de pouvoir de résolution d'une paire de mots. Comme le modèle d'indexation avec *idf*, utilisé par le modèle vectoriel, ou la combinaison du modèle de corpus avec le modèle 1-gramme, l'idée principale derrière le pouvoir de résolution est que la pertinence d'un terme pour un document est liée à la fréquence de ce terme dans le document ainsi que sa fréquence dans le corpus global. Conformément à la pratique en théorie de l'information, la quantité d'information d'un mot w dans un corpus est définie comme

$$\text{INFO}(w) = -\log_2(p_{corpus}(w)) \tag{4.1}$$

La quantité d'information est le pouvoir de discrimination du terme parmi les documents de la collection. Plus le mot est fréquent dans le corpus, moins le mot peut être utilisé pour distinguer les différents documents. Il suit que la quantité d'information d'une paire de mots $< u, v >$ est :

$$\text{INFO}(< u, v >) = -\log_2(p_{corpus}(< u, v >)) \tag{4.2}$$

Pour éviter le problème de sous-représentation de données, ainsi que le calcul prohibitif des probabilités pour toutes les paires possibles dans le corpus, une hypothèse est faite sur l'indépendance entre les mots dans le contexte de la langue globale (représentée par le corpus), et la quantité d'information d'une paire devient :

$$\text{INFO}_{indep}(< u, v >) = -\log_2(p_{corpus}(u) \times p_{corpus}(v)) \tag{4.3}$$

Le pouvoir de résolution ρ de la paire $< u, v >$ pour un document d combine alors la quantité d'information et la fréquence de la paire dans le document :

$$\rho_d(< u, v >) = c_d(< u, v >) \times \text{INFO}_{indep}(< u, v >) \tag{4.4}$$

où $c_d(< u, v >)$ est le nombre de fois où la paire $< u, v >$ a été vue dans le document d.

L'application pour laquelle les affinités lexicales ont été originalement proposées était la documentation pour une bibliothèque de logiciel, avec le but de permettre à un ingénieur logiciel de trouver des composants réutilisables [11]. Les deux tâches de l'application comprenaient du clustering ainsi que la recherche de composants. Chaque document d était représenté par un profil X, qui était simplement l'ensemble des affinités lexicales du document, avec leur pouvoir de résolution :

$$X = \{(< u, v >), \rho(< u, v >))\} \tag{4.5}$$

Pour la tâche de clustering, les documents étaient regroupés selon une mesure de distance entre leurs profils (décrite dans la section 4.2.2). Pour la recherche, un profil était construit pour la requête, et les documents étaient ordonnés selon la distance entre le profil du document et celui de la requête. Le but était d'augmenter la précision des premiers documents retrouvés. L'usager choisirait alors un des premiers documents retrouvés et, si nécessaire, à partir de ce document, entrerait et parcourrait la hiérarchie des clusters. L'utilisation des mots simples, était notée comme une possibilité à considérer pour augmenter le rappel.

Cependant, dans notre travail, nous ne considérons que la tâche de recherche. Pour éviter qu'un document pertinent, qui contient les termes d'une requête, mais pas dans la fenêtre de mots précisée, ait un score nul, le score de pertinence devrait tenir compte des mots simples ainsi que des paires. Évidement, il serait nécessaire de compter les mots simples dans le score de pertinence pour les requêtes d'un seul mot. Avant d'explorer comment incorporer les mots simples avec les paires, nous décrivons dans la section 4.2 nos tentatives pour trouver la meilleure façon d'employer exclusivement les affinités lexicales. Dans la section 4.3, nous décrivons un premier modèle probabiliste (AL-1) qui intègre les mots simples dans le lissage, et dans la section, 4.4.1 un deuxième modèle qui combine les mots simples et paires dans un modèle unigramme.

Compter les affinités lexicales d'un document

L'algorithme suivant traite un document d et stocke les affinités lexicales rencontrées dans une fenêtre de win mots :

```
COUNTLAS(document d, fenêtre win)
 1  //stocker les paires avec leur fréquence
 2  for each phrase s = w_1 ... w_n in d
 3     do for i ← 1 to n
 4           do for j ← i + 1 to min(n, i + win)
 5                 do //stocker les comptes pour les mots dans l'ordre lexicographique
 6                    if w_i < w_j
 7                       then c_d((w_i, w_j)) + +
 8                       else if w_i > w_j
 9                                then c_d((w_j, w_i)) + +
10  //calculer le pouvoir de résolution pour chaque paire
11  for each paire < u, v >
12     do info ← − log(p_corpus(u) × p_corpus(v))
13        ρ(< u, v >) = info × c_d(< u, v >)
```

Les paires $< u, v >$ sont toujours stockées dans l'ordre lexicographique, où u est inférieur à v. Le but est double : cela permet d'une part de ne pas imposer de contrainte sur l'ordre des mots, et d'autre part une implémentation plus efficace (plus rapide et plus compacte en mémoire). Il est à noter que les paires $< w, w >$ ne sont pas considérées par cet algorithme, car elles ne véhiculent pas d'information utile pour la recherche d'information.

Le tableau 4.1 montre plusieurs affinités lexicales dans un document qui parle du 200$^{\text{ième}}$ anniversaire de la cour suprême des États-Unis. Pour chaque paire, la fréquence, le pouvoir de résolution, la probabilité de la paire (décrite dans la section 4.3), et le log des probabilités globales des mots individuels sont donnés. Les paires sont ordonnées par fréquence, et puis par pouvoir de résolution. On peut remarquer que, parmi les paires avec la même fréquence dans le document, celles qui contiennent des mots plus rares ont un pouvoir de discrimination plus élevé. Ceci est plus marqué pour les paires de fréquence 1, si l'on compare la paire "fourth-grader francavilla" (pouvoir de résolution de 13) dont les deux mots ont une fréquence globale plutôt basse, avec la paire "call state" (pouvoir de résolution de 5) dont les deux mots sont

affinité lexicale ($< u, v >$)	fréquence	ρ	probabilité	$p_{corpus}(u)$	$p_{corpus}(v)$
court supreme	7	43,5674	0,00574	-2.79	-3.43
court property	6	38,3014	0,00505	-2.79	-3.59
court night	5	29,6802	0,00391	-2.79	3.14
public sidewalk	4	30,4571	0,00401	-3.02	-4.59
court justice	4	24,5999	0,00324	-2.79	3.36
court write	4	24,0759	0,00317	-2.79	3.23
court member	4	22,5521	0,00297	-2.79	-2.85
property sidewalk	3	24,5551	0,00324	-3.59	-4.59
court neat	3	23,5860	0,00311	-2.79	-5.10
demonstrator property	3	22,2532	0,00293	-3.82	-3.59
		...			
fourth-grader francavilla	1	13,1935	0,00174	-6.21	-6.99
brandi fourth-grader	1	13,0174	0,00172	-6.81	-6.21
francavilla grafton	1	12,8103	0,00169	-6.99	-5.82
brandi grafton	1	12,6342	0,00167	-6.81	-5.82
hand-printed multicolored	1	12,4154	0,00164	-6.81	-5.61
		...			
court people	1	5,26036	0,00069	-2.79	-2.47
court state	1	5,14814	0,00068	-2.79	-2.36
make time	1	5,12372	0,00068	-2.52	-2.60
state united	1	5,07122	0,00067	-2.36	-2.72
call state	1	5,06964	0,00067	-2.71	-2.36

TAB. 4.1 – Exemples d'affinités lexicales pour un document (AP900302-10) dédié au 200ème anniversaire de la cour suprême des États-Unis. La probabilité d'une affinité lexicale est décrite dans la section 4.3

beaucoup plus fréquents.

On observe aussi que le mot "court" domine les paires, apparaissant dans la moitié des paires montrées.

4.2 Pertinence basée sur le pouvoir de résolution

Dans cette section nous décrivons différents scores qui permettent d'utiliser le pouvoir de résolution dans le score de pertinence.

4.2.1 Somme des pouvoirs de résolution

Le score de pertinence le plus simple pour un document d et une requête q consiste à additionner les pouvoirs de résolution (tels que calculés par le document d) de toutes les

affinités lexicales de la requête. Formellement :

$$score_{sum\_\rho}(d, q) = \sum_{a \in q} \rho_d(a) \tag{4.6}$$

où $a =< u, v >$ est une affinité lexicale dans la requête, et $\rho_d(a)$ est son pouvoir de résolution dans le document d. Cette approche favorise les documents longs, qui ont à priori plus de chance de contenir davantage d'affinités lexicales, qu'un document plus court. L'exemple extrême serait un dictionnaire (si la collection contenait un dictionnaire comme document), qui aurait tous les termes de la requête, alors que le dictionnaire n'est pas du tout un document spécifiquement approprié pour une requête.

4.2.2 Produit interne

Le produit interne est l'approche utilisée par Maarek, comme mesure de distance entre le profil d'un document et celui de la requête :

$$score_{prod\_int}(d, q) = \sum_{a \in q \cap d} (\rho_d(a) \times \rho_q(a)) \tag{4.7}$$

Le biais vers les documents plus longs demeure un problème avec le produit interne. L'avantage du produit interne, par rapport à la sommation simple des pouvoirs de résolution, est qu'on considère aussi le pouvoir de discrimination des paires dans la requête. Le pouvoir $\rho_q(a)$ nous permet de différencier une paire spécifique pour décrire le besoin d'information d'une paire commune sans signification particulière. Par exemple, supposons que l'usager formule la requête q_1="java programming language" et qu'une collection d'informatique et deux documents en particulier dans cette collection contiennent les fréquences et probabilités reportées dans la figure 4.1. Dans cette collection, les mots "programming" et "language" sont plus fréquents que "java", et donc la paire "java programming" est plus discriminante entre les documents que la paire "language,programming". Spécifiquement, le pouvoir de résolution pour la paire "language,programming" vue une fois est $-(-3-3) \times 1 = 6$ et le pouvoir de résolution pour "java,programming" est $-(-3-4) \times 1 = 7$. Les documents d_1 et d_2 contiennent chacun trois occurrences des paires dans la requête q_1, mais d_1 contient la paire plus discriminante, "java,programming" deux fois, tandis que le document d_2 la contient une fois. Intuitivement, d_1 est probablement plus pertinent que d_2 pour la requête donnée. On voit que la sommation simple et le produit interne donnent un score plus élevé au document d_2, mais l'écart entre les scores des deux documents est plus prononcé avec le produit interne.

corpus	
$\log p(java)$	-4
$\log p(programming)$	-3
$\log p(language)$	-3

paire	q_1	d_1	d_2
c(java programming)	1	2	1
c(language programming)	1	1	2
ρ(java programming)	7	14	7
ρ(language programming)	6	6	12
$\sum \rho$		14+6=20	7+12=19
produit interne		$(7 \times 14) + (6 \times 6) = 134$	$(7 \times 7) + (6 \times 12) = 121$

FIG. 4.1 – Sommation des pouvoirs de résolution vs produit interne

4.2.3 Normalisations

Afin de contourner le problème du biais vers les documents longs, nous avons essayé plusieurs approches de normalisation qui sont maintenant décrites. Pour plus d'information sur l'effet des techniques de normalisation sur la performance d'un système, nous invitons le lecteur à lire [16], où Singhal et. al. décrivent une approche à optimiser une normalisation donnée.

Normalisation par la longueur du document

Il est par exemple possible de normaliser le score par le nombre total d'affinités lexicales dans le document, soit :

$$score_{norm\_|d|}(d, q) = \frac{score(d, q)}{|d|} \tag{4.8}$$

où $|d|$ est le nombre de paires dans d et $score(d, q)$ est l'un des scores précédemment décrits (équation 4.6 ou 4.7).

Normalisation par la somme des pouvoirs de résolution du document

Il est également possible de normaliser par la somme des pouvoirs de résolution de toutes les paires dans le profil du document :

$$score_{norm\_sum\_\rho}(d, q) = \frac{score(d, q)}{\sum_{a \in d} \rho_d(a)} \tag{4.9}$$

	d_3	d_4
# paires	20	50
c(java programming)		1
c(language programming)	1	1
ρ(java programming)		7
ρ(language programming)	6	6
$\sum \rho$	6	13
$\sum \rho / \lvert d \rvert$	$\frac{6}{20} = 0.30$	$\frac{13}{50} = 0.26$
$\sum \rho / \log \lvert d \rvert$	$\frac{6}{1.30} = 4.61$	$\frac{13}{1.70} = 7.65$

TAB. 4.2 – Diviser la somme des pouvoirs de résolution par la longueur du document

Ce calcul revient à sommer les pouvoirs de résolution relatifs dans d pour les paires dans q.

Ces deux normalisations sont peut-être trop sévères. Prenons par exemple deux autres documents d_3 et d_4 pour la même requête q_1="java programming language" avec les fréquences et pouvoirs de résolution reportés dans le tableau 4.2. Le document d_3 est un document court qui parle des langages de programmation, mais pas nécessairement de Java. Le document d_4 est plus long mais traite clairement du langage Java. Intuitivement, le document d_4 est plus pertinent que d_3, non seulement parce qu'il contient deux des paires de la requête, mais aussi parce qu'il contient une paire qui est plus discriminante que la paire dans d_3. Cependant, le score pour d_3, normalisé par la longueur du document, est plus élevé. Ce phénomène est plus prononcé pour les requêtes plus longues, car la possibilité de trouver un document relativement court contenant juste une des paires moins discriminantes augmente avec la longueur de la requête.

Diviser par le log de la longueur (ou le log de la somme de pouvoirs de résolution pour toutes paires dans le document) permet de prendre en compte la taille du document d'une façon moins marquée, soit :

$$score_{norm\_|d|}(d,q) = \frac{score(d,q)}{\log |d|} \tag{4.10}$$

$$score_{norm\_sum\_\rho}(d,q) = \frac{score(d,q)}{\log\left(\sum_{a \in d} \rho_d(a)\right)} \tag{4.11}$$

Nous voyons sur notre exemple (voir le tableau 4.2), que l'ordre de documents ne change pas si la normalisation utilise le log de la longueur du document, plutôt que la longueur directement.

Dice et Jaccard

Les normalisations Dice et Jaccard du produit interne sont souvent utilisées par le modèle vectoriel, et peuvent être appliquées aux pouvoirs de résolution des profils du document et de la requête. La normalisation Dice dans notre contexte prend la forme :

$$score_{Dice}(d,q) = \frac{2 \times \sum_{a \in q \cap d} \rho_d(a) \times \rho_q(a)}{\sum_{a \in d} \rho_d(a) + \sum_{a \in q} \rho_q(a)} \tag{4.12}$$

Avec les requêtes courtes (ou même avec une requête relativement longue mais assez courte en comparaison avec le document), cette normalisation approche celle de la division par $\sum_{a \in d} \rho_d(a)$, proposée dans l'équation 4.9.

La normalisation Jaccard est conceptuellement l'intersection du document et de la requête (le produit interne) divisé par l'union, dans notre contexte :

$$score_{Jaccard}(d,q) = \frac{\sum_{a \in q \cap d} \rho_d(a) \times \rho_q(a)}{\sum_{a \in d} \rho_d(a) + \sum_{a \in q} \rho_q(a) - \sum_{a \in q \cap d} \rho_d(a) \times \rho_q(a)} \tag{4.13}$$

Cette normalisation devrait aussi approcher celle de l'équation 4.9, si le produit interne est relativement petit en comparaison avec la taille du document.

Cosinus

On peut également utiliser le profil de la requête et trier les documents selon la mesure de similarité du cosinus. Le cosinus, utilisé souvent dans le modèle vectoriel, peut être adapté pour les profils d'affinités lexicales de la manière suivante :

$$score_{cos}(d,q) = \frac{\sum_{a \in d \cap q} \rho_d(a)\rho_q(a)}{\sqrt{\sum_{a \in d} \rho_d(a)^2} \sqrt{\sum_{a \in q} \rho_q(a)^2}} \tag{4.14}$$

Le cosinus prend en compte la longueur du document, implicitement, dans le premier terme du dénominateur. La longueur de la requête est aussi comprise dans le calcul, mais comme cette valeur est égale pour tous les documents, elle ne change pas l'ordre des documents retrouvés. On peut en fait l'enlever du score pour des raisons d'efficacité. D'un point de vue géometrique, le cosinus mesure l'angle entre deux vecteurs, sans égard à leurs longueurs.

Normalisation par le nombre de paires partagées entre le document et la requête

On peut aussi diviser le score brut par le nombre d'affinités lexicales que la requête et le document ont en commun :

$$score_{norm\_intersect}(d, q) = \frac{score(d, q)}{|q \cap d|} \tag{4.15}$$

où $|q \cap d|$ est le nombre de paires qui se trouvent et dans le document et dans la requête. Ce score est essentiellement la moyenne des pouvoirs de résolution des paires pertinentes dans le document. Il convient de noter qu'un document qui contient plusieurs paires de la requête obtiendrait le même score qu'un autre document qui n'en contiendrait qu'une, dans le cas où les pouvoirs de résolution sont égaux pour chaque paire.

4.2.4 L1 norm

Une autre mesure possible de distance entre deux vecteurs est la différence absolue (aussi connue sous les noms *city block*, *Manhattan distance*, *L1 norm*, ou encore *Hamming distance*) :

$$score_{L1}(d, q) = -\sum_{a \in q} |\rho_d(a) - \rho_q(a)| \tag{4.16}$$

Comme la valeur absolue de la différence évalue la distance plutôt que la similarité entre un document et une requête, les documents sont ordonnés dans l'ordre inverse de la différence. Le fait que, normalement, une requête ne contienne une paire donnée qu'une fois, et qu'il est possible qu'un document pertinent la contienne plusieurs fois, présente un problème. Pour une paire donnée, un document qui ne contient pas la paire aura la même distance à la requête qu'un document pertinent qui la contient deux fois. En outre, un document contenant la paire plus que deux fois aura une distance plus élevée (ce qui implique un score plus bas) qu'un document qui manque la paire.

paire	q_2	d_5	d_6
c(java language)	1	2	1
ρ(java language)	7	14	7
score=-$\lvert\rho_d - \rho_q\rvert$		$-\lvert 14-7\rvert = -7$	$-\lvert 7-7\rvert = 0$

TAB. 4.3 – Exemple de la distance entre un document et une requête

Par exemple, considérons une requête q_2 = "java language" et deux documents d_5 et d_6 avec les fréquences et pouvoirs de résolution rapportés dans le tableau 4.3. Le document d_5 est intuitivement plus pertinent que le document d_6, contenant la paire de la requête deux fois, tandis que la paire a une fréquence de 1 dans le document d_6. Cependant, comme la fréquence de cette paire est 1 dans la requête, la distance entre d_6 et la requête est plus petite (et implique donc un score plus élevé) qu'entre d_5 et la requête. Cette mesure serait plus adaptée pour une tâche de catégorisation où les pouvoirs de résolution entre documents sont plus comparables qu'entre un document et une requête. Un autre moyen de l'employer serait avec des pouvoirs de résolution normalisés. Cette approche sera étudiée dans la section 4.3.2.

4.2.5 Comparaison des mesures

Le tableau 4.4 montre la précision moyenne réalisée avec les scores de pertinence décrits jusqu'ici pour les affinités lexicales. On peut tirer plusieurs conclusions à partir de ces résultats :

- La mesure qui donne les moins bons résultats est la distance absolue $L1$; ce qui n'est pas étonnant étant donné le problème que nous avons décrit.
- Le produit interne (eq 4.7) est plus performant qu'une sommation simple sur les pouvoirs de résolution du document (eq 4.6). Comme nous l'avons indiqué, ceci peut s'expliquer par le fait que le produit interne incorpore de l'information de la requête, augmentant le poids relatif des paires plus discriminantes dans le score.
- La normalisation par le nombre de paires dans le document (eq 4.8) est comparable à la division par la somme de pouvoirs de résolution des paires du document (eq 4.9). Dans les deux cas, la normalisation moins sévère, par le log (eqs 4.10 et 4.11) plutôt que par la valeur brute, donne de meilleurs résultats. Pour le produit interne ainsi que pour la sommation simple, la normalisation directe par la longueur du document dégrade les performances, tandis que la normalisation avec le log permet une augmentation en précision. Ces résultats confirment les raisonnements faits dans la section 4.2.3.
- Les normalisations Dice et Jaccard sont comparables à celle du produit interne divisé

Mesure de score	équation(s)	titres	titres et desc.
cosinus	4.14	0.2272	0.2436
produit interne/log $\|d\|$	4.7,4.10	0.2004	0.2469
produit interne/log $\sum \rho_d$	4.7,4.11	0.2012	0.2464
produit interne	4.7	0.1941	0.2437
produit interne/$\|d\|$	4.7,4.8	0.1905	0.2198
produit interne/$\sum \rho_d$	4.7,4.9	0.1893	0.2207
Dice	4.12	0.1895	0.2235
Jaccard	4.13	0.1895	0.2235
sum_ρ/log $\|d\|$	4.6,4.10	0.1997	0.2080
sum_ρ/log $\sum \rho_d$	4.6,4.11	0.1998	0.2076
sum_ρ	4.6	0.1939	0.2083
sum_ρ/$\|d\|$	4.6,4.8	0.1887	0.1796
sum_ρ/$\sum \rho_d$	4.6,4.9	0.1873	0.1781
sum_ρ/$\|d \cap q\|$	4.15	0.1785	0.1390
L1	4.16	0.1093	0.1653

TAB. 4.4 – Précision moyenne avec les affinités lexicales

par $\sum_{a \in d} \rho_d(a)$ (eq 4.9), pour les raisons que nous avons déjà discutées.

- Parmi ces mesures de score qui utilisent les pouvoirs de résolution directement, le cosinus donne la précision la plus élevée. Le cosinus prend en compte implicitement la longueur du document, ce qui a pour effet d'atténuer les problèmes de dépendance à la longueur du document. Comme dans le cas du produit interne, l'inclusion du profil de la requête, dans le numérateur, augmente les poids relatifs des paires plus discriminantes dans le score.

Aucune de nos expériences qui se basent uniquement sur les affinités lexicales ne dépasse le baseline. Comme nous l'avons discuté dans la première section de ce chapitre, des documents pertinents qui contiennent les mots de la requête, mais pas dans la fenêtre précisée, auront un score nul. Afin de permettre une augmentation en rappel ainsi qu'en précision, il est nécessaire d'intégrer les mots simples dans le score de pertinence. Nous examinons maintenant deux techniques de combinaison des mots simples et des affinités lexicales.

4.3 AL-1 : Un modèle probabiliste

Au lieu d'utiliser les pouvoirs de résolution bruts, on peut construire un modèle probabiliste basé sur les affinités lexicales. Ce modèle incorpore implicitement la longueur du document, et peut être combiné naturellement avec un modèle de langue unigramme.

Si une paire de mots $< u, v >$ est considérée comme une unité lexicale, la probabilité de voir la paire dans un document d est simplement sa fréquence relative, calculée de la même façon que pour le modèle unigramme (eq 1.14) :

$$p_{d_{MLE,freq}}(< u, v >) = \frac{c_d(< u, v >)}{\sum_{u,v} c_d(< u, v >)} \tag{4.17}$$

Afin d'incorporer le pouvoir de discrimination des mots dans le modèle probabiliste, on peut utiliser le pouvoir de résolution comme une fréquence modifiée :

$$p_{d_{MLE}}(< u, v >) = \frac{\rho_d(< u, v >)}{\sum_{u,v} \rho_d(< u, v >)} \tag{4.18}$$

Cette probabilité incorpore implicitement la fréquence relative de la paire dans le document ainsi que que son pouvoir de discrimination.

4.3.1 Lissage

Le problème de sous-représentation de données a un impact plus profond sur un modèle probabiliste d'affinités lexicales que sur un modèle 1-gramme. Il existe deux classes d'événements inconnus : une paire $< u, v >$, où u et v ont été vus dans le document d, mais jamais dans la fenêtre précisée, et une paire $< unk, w >$, où l'un (ou possiblement deux) des mots de la paire n'apparaît pas dans le document. Intuitivement, le premier événement devrait recevoir une probabilité plus élevée que le second.

Une façon d'accorder une partie de la masse de probabilités aux paires non vues consiste à l'instar de ce que nous avons proposé pour les modèles 1-gramme, d'attribuer une probabilité fixe p_{unk} à tout évènement inconnu, et à normaliser le modèle en intégrant ce paramètre :

$$p_d(< u, v >) = p_{d_{MLE}}(< u, v >) \times (1 - p_{unk}) \tag{4.19}$$

Lissage backoff

Cependant, une meilleure approche consiste à intégrer les probabilités des mots individuels dans le lissage, permettant ainsi qu'une paire $< u, v >$ dont les deux mots apparaissent dans le document reçoive une probabilité plus élevée qu'une paire dont un ou deux mots sont inconnus du document. Par exemple, on peut appliquer l'idée du modèle de backoff et

consulter le modèle unigramme quand une paire est inconnue du document :

$$p_d(< u, v >) = \lambda_2 p_{d_{MLE}}(< u, v >) + \theta_{u,v} \left[\frac{\alpha p(u)}{|V|} + \frac{\beta p(v)}{|V|} \right]$$

où

$$\theta_{u,v} = \begin{cases} 1 & \text{lorsque } p_{d_{MLE}}(< u, v >) = 0 \\ 0 & \text{sinon} \end{cases} \tag{4.20}$$

et $p(w)$ est calculé selon le modèle unigramme comme décrit dans la section 3.2.1 ou 3.2.3. On remarque que $\sum_{u,v} \frac{p(u)}{|V|} = 1$. Comme il n'y a pas de raison à priori pour donner plus de poids à la probabilité du mot simple u qu'à la probabilité du mot v, on peut poser $\alpha = \beta$ et $\gamma = \frac{\alpha}{|V|} = \frac{\beta}{|V|}$ puis simplifier l'équation comme suit :

$$p_d(< u, v >) = \lambda_2 p_{d_{MLE}}(< u, v >) + \theta_{u,v} \gamma \left(p(u) + p(v) \right) \tag{4.21}$$

γ doit être calculé de manière à respecter la contrainte stochastique du modèle :

$$\begin{aligned} \sum_{u,v} p(< u, v >) = 1 &= \sum_{u,v} \lambda_2 p_{d_{MLE}}(< u, v >) + \sum_{u,v} \theta_{u,v} \gamma \left(p(u) + p(v) \right) \\ 1 &= \lambda_2 + \gamma \sum_{u,v} \theta_{u,v} \left(p(u) + p(v) \right) \end{aligned}$$

$$1 - \lambda_2 = \gamma \sum_{u,v} \theta_{u,v} \left(p(u) + p(v) \right)$$

$$\gamma = \frac{1 - \lambda_2}{\sum_{u,v} \theta_{u,v} \left(p(u) + p(v) \right)}$$

Afin d'illustrer une valeur typique de γ, considérons un document de 250 mots (avec 150 mots uniques) qui contient 200 paires uniques. La probabilité moyenne d'un mot simple serait $1/150 = 0.0067$. Le nombre maximum de paires différentes est $150 \times 150 = 22500$. Cependant, en ne gardant que les paires $< u, v >$ où u est inférieur en ordre lexicographique à v, le nombre total de paires considérées est en fait $(22500 - 150)/2 = 11175$, alors que le nombre de paires qui n'existent pas dans le document est $11175 - 200 = 10975$. La valeur de γ dans ce cas est :

$$\gamma \approx \frac{1 - \lambda_2}{10975 \times (0.0067 + 0.0067)} = \frac{1 - \lambda_2}{146}$$

Nous pouvons voir que la contribution des mots simples dans l'équation 4.21 est relativement faible par rapport à celle de les paires observées, à cause du grand nombre possible des paires

possibles.

Lissage interpolé

Un lissage interpolé, où les modèles unigrammes sont toujours consultés, est aussi possible :

$$p_d(< u, v >) = \lambda_2 p_{d_{MLE}}(< u, v >) + \alpha \frac{p(u)}{|V|} + \beta \frac{p(u)}{|V|} \tag{4.22}$$

Comme pour le lissage backoff, on peut poser $\alpha = \beta$ et $\gamma = \frac{\alpha}{|V|} = \frac{\beta}{|V|}$. L'équation simplifiée devient alors :

$$p_d(< u, v >) = \lambda_2 p_{d_{MLE}}(< u, v >) + \gamma \left(p(u) + p(v)\right) \tag{4.23}$$

Et on résout γ en fonction de λ_2 par :

$$\gamma = \frac{1 - \lambda_2}{\sum_{u,v} (p(u) + p(v))} = \frac{1 - \lambda_2}{2|V|} \tag{4.24}$$

Ici la valeur de γ pour le même document de 150 mots uniques serait $\frac{1-\lambda_2}{2\times 150}$. Cependant, en ne gardant que les paires $< u, v >$ où $u < v$, γ devient $\frac{1-\lambda_2}{150}$, une valeur légèrement plus basse que dans l'approche backoff.

4.3.2 Scores de pertinence basés sur le modèle AL-1

Pertinence basée sur la probabilité de générer la requête

Comme le modèle AL-1 traite une paire comme une unité lexicale complète, il est comparable à un modèle unigramme, et la score de pertinence d'un document d pour une requête q est le produit des probabilités de chaque paire selon le modèle du document :

$$score_{gen}(d, q) = p(q|d) = \prod_{a \in q} p_d(a)^{c_q(a)} \tag{4.25}$$

où $a =< u, v >$ est une paire dans la requête q, $p_d(a)$ est sa probabilité telle que calculée par l'équation 4.20 ou l'équation 4.22 ; et $c_q(a)$ est le nombre de fois que la paire est vue dans q. Comme pour le modèle 1-gramme, une hypothèse d'indépendance entre les unités lexicales est faite. Il est possible que l'hypothèse d'indépendance entre affinités lexicales soit plus forte que l'indépendance entre mots, si l'on considère, par exemple, la dépendance entre les mots "programming" et "java" en comparaison avec la dépendance entre les paires "programming language" et "language java".

KL-divergence

D'autres mesures de score de pertinence, adaptées pour des distributions probabilistes, sont aussi possibles. La *Kullback-Leibler divergence* $D(P||Q)$, aussi connue comme entropie relative, est une mesure de pseudo-distance entre deux distributions probabilistes P et Q prenant leurs valeurs sur le même ensemble d'événements. On parle ici de pseudo-distance car la divergence n'est pas symétrique (c'est à dire, $D(P||Q) \neq P(Q||P)$). L'entropie relative est aussi une mesure de l'information perdue si l'on utilise la distribution Q comme approximation d'une distribution observée P. Pour la recherche d'information, la distribution observée est le modèle de langue pour la requête, et le but est d'ordonner les documents en ordre décroissant de la divergence entre le modèle de la requête et celui du document d.

$$D(q||d) = \sum_{a \in q} p_q(a) log \left(\frac{p_q(a)}{p_d(a)} \right) \tag{4.26}$$

Le score de pertinence d'un document d pour la requête q est donc :

$$\begin{aligned} score_{KL}(d,q) &= -|D(q\|d)| \\ &= -\left| \sum_{a \in q} p_q(a) \left(\log p_q(a) - \log p_d(a) \right) \right| \end{aligned} \tag{4.27}$$

Plus la distribution d'un document est proche de celle de la requête, plus la divergence approche zéro. Il existe une variante symétrique de la KL-divergence : $D_s(P||Q) = D(P||\frac{P+Q}{2}) + D(Q||\frac{P+Q}{2})$. En plus d'être une mesure de distance symétrique, la KL-divergence symétrique est toujours non-negative, comme $D(P||Q)$. Avec cette version, le score de pertinence devient :

$$
\begin{aligned}
score_{KLsym}(d,q) &= -D_s(q\|d) \\
&= -\Bigg[\sum_{a\in q} p_q(a)\log p_q(a) - \sum_{a\in q} p_q(a)\log\left(\frac{p_q(a)+p_d(a)}{2}\right) \\
&\quad + \sum_{a\in d} p_d(a)\log p_d(a) - \sum_{a\in d} p_d(a)\log\left(\frac{p_d(a)+p_q(a)}{2}\right)\Bigg] \\
&= -\Bigg[\sum_{a\in q} H(p_q(a)) - \sum_{a\in q} p_q(a)\log(p_q(a)+p_d(a)) + \sum_{a\in q} p_q(a)\log 2 \\
&\quad + \sum_{a\in d} H(p_d(a)) - \sum_{a\in d} p_d(a)\log(p_q(a)+p_d(a)) + \sum_{a\in d} p_d(a)\log 2\Bigg] \\
&= -\Bigg[2\log 2 + \sum_{a\in q} H(p_q(a)) - \sum_{a\in q} p_q(a)\log(p_q(a)+p_d(a)) \\
&\quad + \sum_{a\in d} H(p_d(a)) - \sum p_d(a)\log(p_q(a)+p_d(a))\Bigg] \qquad (4.28)
\end{aligned}
$$

où $H(x) = p(x)\log p(x)$.

Le calcul peut être simplifié en prenant les sommations sur l'intersection des termes dans q et d. Si une paire est manquante dans le document d le deuxième terme et le troisième terme de l'équation 4.28 s'annulent :

$$
\begin{aligned}
& H(p_q(a)) - p_q(a)\log\left(p_q(a)+p_d(a)\right) \\
= \;& p_q(a)\log p_q(a) - p_q(a)\log(p_q(a)+0) = 0
\end{aligned}
$$

Similairement, pour les paires absentes dans la requête q, le quatrième terme et le cinquième terme s'annulent :

$$
\begin{aligned}
& H(p_d(a)) - p_d(a) log\left(p_q(a)+p_d(a)\right) \\
= \;& p_d(a)\log p_d(a) - p_d(a)\log(0+p_d(a)) = 0.
\end{aligned}
$$

Alors,

$$\begin{aligned} score_{KLsym}(d,q) &= -\Bigg[2\log 2 + \sum_{a\in q\cap d}\Bigg[H(p_q(a)) - p_q(a)\log(p_q(a)+p_d(a)) \\ &\quad + H(p_d(a)) - p_d(a)\log(p_q(a)+p_d(a))\Bigg]\Bigg] \\ &= -2\log 2 - \sum_{a\in q\cap d}\Big[H(p_q(a)) + H(p_d(a)) - (p_d(a)+p_q(a))\log\left(p_q(a)+p_d(a)\right)\Big] \\ &= -2\log 2 - \sum_{a\in q\cap d}\Big[H(p_q(a)) + H(p_d(a)) - H(p_q(a)+p_d(a))\Big] \end{aligned} \tag{4.29}$$

De plus, les termes qui sont constants pour tous les documents peuvent être enlevés du score et finalement :

$$score_{KLsym}(d,q) = -\sum_{a\in q\cap d}\left[H(p_d(a)) - H(p_q(a)+p_d(a))\right] \tag{4.30}$$

L1 norm

Une dernière mesure que nous avons essayée pour intégrer les modèles probabilistes des affinités lexicales est la distance absolue :

$$score_{L1_{prb}}(d,q) = -\sum_{a\in q}|p_d(a) - p_q(a)| \tag{4.31}$$

Nous attendons à une meilleure performance avec la mesure L1 basée sur les probabilités, en comparaison avec la mesure L1 basée sur les pouvoirs de résolution bruts (voir la section 4.2.4). Comme la probabilité d'une paire dans la requête est généralement supérieure à celle dans le document, la distance entre le document et la requête pour une paire donnée est plus petite si la probabilité dans le document est plus grande, en contraste avec la distance des pouvoirs de résolution bruts.

4.3.3 Optimisation des paramètres

Dans cette section nous présentons les expériences que nous avons effectuées pour trouver les paramètres optimaux pour les approches que nous avons décrites, notamment la contribution λ des paires vues dans un document d, dans $p_d(< u, v >)$, la valeur de $p(unk)$ pour le lissage fixe, et la contribution du modèle de corpus dans le modèle unigramme.

λ_2	titres	titres & descr
0	0.2222	0.2847
0.1	0.2587	0.2903
0.5	0.2575	0.2899
0.9	0.2568	0.2860
1	0.1072	0.0695

TAB. 4.5 – Précision moyenne avec le modèle AL-1 (approche backoff, eq 4.20), avec plusieurs valeurs de λ_2, la contribution de la probabilité des paires vues.

p_{unk}	titres	titres & descr
10E-4	0.1936	0.2354
10E-5	0.1975	0.2312
10E-6	0.1975	0.2216
10E-3	0.1600	0.1795
10E-2	0.0158	0.0107

TAB. 4.6 – Précision moyenne avec le modèle AL-1 (approche fixe, eq 4.19), avec plusieurs valeurs de p_{unk}, la probabilité allouée aux paires non vues.

Les contributions relatives des modèles de paires et mots simples (λ_2), approche backoff

Le tableau 4.5 montre l'effet du paramètre λ_2, la contribution de la probabilité des paires vues, dans l'approche backoff (équation 4.20). Le score de pertinence pour ces expériences est la probabilité de générer la requête, selon l'équation 4.25. Sans aucun lissage, si seulement les probabilités des paires vues sont utilisées ($\lambda_2 = 1$), la sous-représentation de données devient problématique et la précision atteinte ne dépasse pas 10%. À l'autre extrémité, si seulement les probabilités des mots simples sont utilisées, la performance est inférieure à celles mesurées lorsque les paires et les mots simples sont intégrés. On remarque que la performance n'est pas très sensible aux valeurs de λ_2 entre 0.1 et 0.9. Pour ces expériences, la probabilité des mots simples utilisait un lissage avec le modèle du corpus (eq 3.10), avec λ_d de l'équation 3.10 = 0.9.

La masse de probabilité allouée aux paires inconnues, $p(unk)$, lissage fixe

Le tableau 4.6 montre l'effet de la masse allouée aux paires non vues dans l'approche de lissage fixe (équation 4.19). Comme nous l'avons vu pour le modèle unigramme, dans la section 3.3.2, si la probabilité allouée aux paires non vues est trop grande (supérieure aux probabilités des paires vues), la précision tombe, car une paire non vue dans un document

aura une probabilité plus élevée qu'une paire vue dans un autre document. Si le nombre moyen de paires dans un document est environ 880 (voir la section 2.1.1), la probabilité la plus faible d'une paire est environ 0.001. On peut voir que la précision moyenne tombe considérablement (à 1%) quand la probabilité accordée aux paires non vues dépasse 0.001.

La contribution du modèle du corpus dans le modèle unigramme intégré (λ_d)

λ_d	titres	titres & descr
0.9	0.2587	0.2903
0.5	0.2579	0.2956
0.1	0.2457	0.2933
0	0.1972	0.2381
1	0.1900	0.1049

TAB. 4.7 – Précision moyenne avec le modèle AL-1 (approche backoff, eq 4.20), avec plusieurs valeurs de λ_d (équation 3.10), la contribution des mots vus dans le modèle unigramme, dans l'intégration avec le modèle du corpus

Le tableau 4.7 montre l'effet du paramètre, λ_d, du modèle unigramme (équation 3.10) dans l'approche backoff (équation 4.20). On peut voir que le poids de la contribution du modèle de corpus n'a pas un impact significatif sur les résultats, sauf que la précision moyenne tombe pour les valeurs extrêmes de 0 (seulement le modèle du corpus est utilisé) ou 1 (aucun lissage n'est fait avec le modèle du corpus).

4.3.4 Comparaison de mesures de score de pertinence

Dans cette section, nous comparons les performances des approches discutées et une analyse de chaque approche est présentée.

Le tableau 4.8 montre la précision mesurée avec le modèle AL-1 et les différentes mesures de score. Pour ces expériences, les valeurs optimales des paramètres ont été utilisées pour

		backoff (eq4.20)		interpolé (eq4.22)		lissage fixe (eq 4.19)	
Mesure de score	équation	titre	titre&dsc	titre	titre&dsc	titre	titre&dsc
KL-div	4.27	0.2512	0.3034	0.2511	0.3034	0.1961	0.2470
générer la requête	4.25	0.2587	0.2903	0.2586	0.2904	0.1936	0.2354
KL-div symétrique	4.30	0.2132	0.2017	0.2132	0.2017	0.1934	0.1980
L1 probabiliste	4.16	0.2071	0.1842	0.2071	0.1842	0.1909	0.1773

TAB. 4.8 – Précision moyenne avec le modèle probabiliste des affinités lexicales

chaque approche. Pour les approches backoff et interpolation, la valeur de λ_2, la contribution de $p_{d_{MLE}}(< u, v >)$ dans les équations 4.20 et 4.22 était 0.1 (valeur optimale du tableau 4.5), et la valeur de λ_d pour le modèle unigramme était 0.9 (valeur optimale du tableau 4.7). Pour le lissage fixe (équation 4.19), la probabilité allouée aux paires non vues était 10E-4 (valeur optimale du tableau 4.6). Le modèle probabiliste utilisait les pouvoirs de résolution (eq 4.18). L'observation la plus saillante est que la précision obtenue en prenant en compte les probabilités des mots simples (les équations 4.20 et 4.22) est meilleure que la précision obtenue par un lissage fixe (l'équation 4.19). Ceci est vrai indépendamment de la méthode de lissage utilisée : backoff ou interpolé. On peut noter aussi que les approches de backoff et interpolation donnent des résultats quasiment équivalents.

KL-divergence asymétrique et la probabilité de générer la requête

Pour les requêtes titres, la KL-divergence et la probabilité de générer la requête donnent des résultats presque égaux ; tous les deux avec une précision moyenne d'environ 23%. Pour les requêtes titres et descriptions, la précision réalisée par la KL-divergence est légèrement plus élevée qu'avec la probabilité de générer la requête. Ces résultats sont raisonnables si l'on observe que dans la plupart des cas, le classement par KL-divergence est équivalent à celui obtenu avec la probabilité de générer la requête, mais contenant plus d'information sur le pouvoir de discrimination des mots dans la requête. En effet, dans la plupart des cas, la probabilité d'une paire donnée est plus élevée dans une requête que dans un document. Dans ce cas, la divergence est positive :

$$\begin{aligned} score_{KL}(d,q) &= -\left[\sum_{a\in q} p_q(a)\left(\log p_q(a) - \log p_d(a)\right)\right] \\ &= -\sum_{a\in q} p_q(a)\log p_q(a) + \sum_{a\in q} p_q(a)\log p_d(a) \end{aligned} \tag{4.32}$$

Le premier terme étant identique pour tous les documents, on peut simplifier l'équation 4.32 en :

$$\begin{aligned} score_{KL}(d,q) &= \sum_{a\in q} p_q(a)\log p_d(a) \\ &= \log\left(\prod_{a\in q} p_d(a)^{p_q(a)}\right) \end{aligned} \tag{4.33}$$

La seule différence entre cette formule et celle de la probabilité de générer la requête est qu'ici, une paire est comprise dans le produit $p_q(a)$ fois plutôt que $c_q(a)$ fois.

La distance absolue Comme nous avons anticipé, la distance absolue se comporte mieux quand la distance se base sur les probabilités, plutôt que sur les pouvoirs de résolution bruts. Cependant, la performance que donne la distance absolue est encore considérablement inférieure à ce que KL-divergence offre, en particulier avec les requêtes plus longues. La raison principale est l'écart entre les probabilités des paires dans une requête de 3-10 mots et dans un document d'environ 200 mots, ce qui rend une distance linéaire entre les deux distributions peu pertinente. Prenons l'exemple suivant d'une requête de trois mots w_1, w_2, et w_3, et deux documents d_1 et d_2 :

paire	q	d_1	d_2
$p(< w_1, w_2 >)$	0.33	0.001	0.002
$p(< w_1, w_3 >)$	0.33	0	0.001
$p(< w_2, w_3 >)$	0.33	0	0.002
$p(unk\_paire)$		10E-5	
$-\sum \lvert p_q - p_d \rvert$		-0.9888	-0.985
KL-div		2.3E-4	1.7E-3

Le document d_1 ne contient qu'une paire de la requête, et avec une probabilité faible en comparaison des probabilités des paires dans le document d_2. À cause de l'écart entre les probabilités de la requête (toutes sont de 0.33) et des documents (de l'ordre de 0.001) la distance absolue accorde des scores presque égaux aux deux documents. Le KL-divergence accorde quant à elle un score presque 10 fois supérieur au document d_2.

KL-divergence asymétrique La version symétrique de la KL-divergence est moins performante que la version asymétrique. L'écart entre les probabilités de la requête et du document est encore l'explication. Si on considère que la probabilité $p_d(< u, v >) \ll p_q(< u, v >)$, on a :

$$\begin{aligned} score_{KLsym}(d,q) &= -\sum_{a \in q \cap d} [H(p_d(a)) - H(p_q(a) + p_d(a))] \\ &\approx -\sum_{a \in q \cap d} [H(p_d(a)) - H(p_q(a))] \end{aligned} \tag{4.34}$$

et comme le dernier terme est égal pour tous documents :

$$\begin{aligned} score_{KLsym}(d,q) &\approx -\sum_{a\in q\cap d} H(p_d(a)) \\ &\approx -\sum_{a\in q\cap d} [p_d(a)\log p_d(a)] \end{aligned} \tag{4.35}$$

Ce score ne tient pas compte des probabilités dans la requête. Le fait que les requêtes avec les titres et les descriptions contiennent souvent une ou deux paires plus d'une fois, et donc ont plus de variation sur les probabilités des paires, pourrait expliquer pourquoi il y a une plus grande différence entre la performance des deux versions de KL-divergence avec les requêtes longues qu'avec les requêtes titres seulement.

4.3.5 Vérification des hypothèses

Dans cette section nous présentons comment nous avons testé les hypothèses que nous avons faites pour les expériences effectuées, notamment l'importance de l'information d'une paire INFO($< u, v >$), l'hypothèse d'indépendance entre mots dans le contexte global, l'hypothèse que seulement les paires prometteuses (avec $\rho > \bar{\rho} + \sigma$) doivent être gardées, et l'hypothèse que les paires intéressantes se trouvent dans une fenêtre de cinq mots.

Utiliser la fréquence des paires pour le pouvoir de résolution

L'importance de l'incorporation du pouvoir de discrimination des mots, INFO($< u, v >$), dans le score de pertinence d'un document pour une requête est vérifiée par des expériences où l'information d'une paire ne se figure pas dans le pouvoir de résolution. Spécifiquement :

$$\rho_d(< u, v >) = c_d(< u, v >) \tag{4.36}$$

Ici, seulement la fréquence de la paire dans le document détermine son pouvoir de résolution, et les probabilités globales des mots individuels ne sont pas comprises dans le pouvoir de résolution. Le tableau 4.9 montre les résultats que nous avons obtenus. On peut remarquer que le score donné par la KL-divergence est équivalent à la probabilité de générer la requête, si le modèle probabiliste se base sur les fréquences des paires (la colonne "fréq") plutôt que prendre en compte l'information de la paire (la colonne "$\rho_{eq4.4}$").

On remarque aussi que pour tous les tests, sauf pour la KL-divergence pour les requêtes titres, la précision moyenne obtenue avec la fréquence comme pouvoir de résolution est

	titres			titres et desc.		
Mesure de score Équation	$\rho_{eq4.4}$ 4.4	fréq 4.36	dép 4.37	$\rho_{eq4.4}$ 4.4	fréq 4.36	dép 4.37
KL-div	0.2512	0.2577	0.2503	0.3034	0.2884	0.3030
générer la requête	0.2587	0.2577	0.2582	0.2903	0.2884	0.2899
cosinus	0.2272	0.2224	0.2269	0.2436	0.2216	0.2393
sum_ρ	0.1939	0.1890	0.1913	0.2083	0.1776	0.1933

TAB. 4.9 – Précision moyenne avec les affinités lexicales, et l'utilisation de la fréquence brute ("fréq") vs le pouvoir de résolution (ρ) dans le score de pertinence. Les résultats avec une dépendance entre les mots d'une paire, dans le contexte global (la colonne "dép"). Pour ces expériences, le score de pertinence est la probabilité de générer la requête (équation 4.25) et les probabilités des paires sont lissées avec les mots simples avec un lissage backoff (équation 4.20).

inférieure à celle obtenue si l'information des paires est prise en compte (l'équation 4.4).

L'hypothèse d'indépendance

Nous avons testé l'hypothèse d'indépendance entre les mots d'une affinité lexicale dans le contexte global, de l'équation 4.3. Nous avons utilisé la probabilité d'une paire (plutôt que les probabilités des mots individuels) pour le calcul de l'information d'une paire, INFO($< u, v >$), avec l'équation 4.2. Plus précisément, le pouvoir de résolution d'une paire $< u, v >$ devient :

$$\begin{aligned} \rho_d(< u, v >) &= c_d(< u, v >) \times \text{INFO}(< u, v >) \\ &= c_d(< u, v >) \times (-\log_2 (p_{corpus}(< u, v >))) \end{aligned} \tag{4.37}$$

Les résultats sont montrés dans le tableau 4.9, dans les colonnes intitulées *dép*. La précision obtenue sans hypothèse d'indépendance est entre celle obtenue avec le pouvoir de résolution basé sur les probabilités des mots indépendants (équation 4.4), et celle où l'information de la paire n'est pas comprise dans le pouvoir de résolution (équation 4.36). Cela est raisonnable si l'on considère le problème de sous-représentation des données. Une grande partie des paires ne se trouvent dans le corpus qu'une seule fois (précisément, 10 millions des 69 millions de paires, soit 14%, ont une fréquence de 1, comme nous l'avons décrit dans le chapitre 2). De plus, la fréquence moyenne des paires, 4, est plutôt basse. Dans le cas extrême où chaque paire aurait une fréquence globale de 1, enlever l'hypothèse d'indépendance serait équivalent à ne pas prendre en compte l'information des paires, car l'information serait égale pour toute

paire.

Filtrer les paires

Pour réduire le bruit, ou le nombre de paires inintéressantes, Maarek et. al. suggèrent de garder d'un document d seulement les affinités lexicales avec les pouvoirs de résolution les plus élevés. Plus précisément, les paires qui ont un pouvoir de résolution ρ plus grand que la moyenne plus l'écart type ($\bar{\rho} + \sigma$) sont gardées. On s'attend à une augmentation de précision, au moins pour les premiers documents retrouvés, si des paires "vides" (e.g., "has a") ne sont pas comptées dans le calcul de pertinence. Cependant, l'efficacité de ce filtrage dépend des pré-traîtements effectués sur le corpus avant l'entraînement des modèles. Dans leurs expériences, Maarek et al. lemmatisaient son corpus, mais aucune référence n'est faite à l'utilisation d'une stopliste. Dans nos expériences, une stopliste a été employée pour enlever des mots vides du corpus. Un deuxième filtrage sur les pouvoirs de résolution des paires cause une perte de précision, enlevant des paires qui ne sont pas nécessairement non-informatives. Le tableau 4.10 nous permet de comparer la précision obtenue en gardant toutes les paires et en ne retenant que les plus prometteuses, qui ont un pouvoir de résolution $\rho \geq \bar{\rho}$ où $\rho \geq \bar{\rho} + \sigma$.

On observe dans le tableau 4.10 que la précision diminue dans toutes les expériences avec le filtrage plus sévère, où on ne garde que les paires avec un pouvoir de résolution $\rho \geq \bar{\rho} + \sigma$. La seule amélioration significative avec le filtrage de $\rho \geq \bar{\rho}$ est avec la KL-divergence pour les requêtes titres et descriptions (une hausse de 1.3%, de 30.3% à 31.6%). Avec les autres mesures, ce filtrage n'amène pas une augmentation significative de précision, ou cause une perte en performance, notamment avec les mesures qui n'ont aucun lissage avec les mots simples (le cosinus et sum_ρ).

Afin de démontrer comment un filtrage sur les paires peut éliminer des documents pertinents lors d'une recherche, considérons la requête 10 de notre collection : "solar car". Le document AP901116-46 a été jugé pertinent, et contient les mots de la requête dans les contextes suivants (ici nous présentons la version sans pré-traitement, pour faciliter la lecture) :

. . . `A Swiss vehicle using newly developed solar cells` . . .

. . . `The car was powered by revolutionary silicon solar cells` . . .

La paire "car,solar" apparaît une fois dans ce document, et a un pouvoir de résolution de $\rho = 7.7$. La moyenne des pouvoirs de résolution est $\bar{\rho} = 8.0$ et l'écart type est $\sigma = 2.0$.

	titres			titres et desc.		
Mesure de score	toutes	$\rho \geq \bar{\rho}$	$\rho \geq \bar{\rho} + \sigma$	toutes	$\rho \geq \bar{\rho}$	$\rho \geq \bar{\rho} + \sigma$
KL-div	0.2512	0.2557	0.2115	0.3034	0.3160	0.2773
générer la requête	0.2587	0.2533	0.2095	0.2903	0.2974	0.2587
KL-div symétrique	0.2132	0.2049	0.1732	0.2017	0.2004	0.1758
cosinus	0.2272	0.2031	0.1635	0.2436	0.2045	0.1720
sum_ρ	0.1939	0.1718	0.1336	0.2083	0.1820	0.1574

TAB. 4.10 – Précision moyenne avec les affinités lexicales, l'effet d'éliminer les paires avec un pouvoir de résolution relativement faible

Les deux filtrages éliminent la paire du document, et donc, si les mots simples ne sont pas utilisés dans un lissage, éliminent le document des documents retrouvés par le système. Le problème est plus grave pour les requêtes courtes. Il est plus probable qu'un document sera complètement éliminé si le filtrage enlève la seule paire de la requête (comme nous le voyons dans cet exemple), que si le document contient encore au moins une paire d'une requête longue, après filtrage.

La distance entre les mots qui forment une paire

L'idée que les mots qui doivent former une affinité lexicale se trouvent dans une fenêtre de 5 mots, originellement proposé par Martin et. al. [12] et implanté par Maarek et. al. [11] ne prend pas en compte l'utilisation d'une stopliste. Maarek note cependant, que seulement les paires de mots dans un groupe nominal ou une relation modificateur-modifié sont gardées. Dans notre travail, une stopliste est appliqué et environ 47% des mots sont enlevés du texte (voir la section 2.1). Comme nous utilisons encore une fenêtre de cinq mots, et que toute paire de mots dans la fenêtre est comptée, il est possible que le nombre de paires comptées soit notablement supérieur au nombre de paires comptées par une implantation sans l'emploi d'une stopliste. Cependant, les améliorations (ou désavantages) possibles en réduisant la taille de la fenêtre ne sont pas nécessairement intuitives.

En réduisant la taille de la fenêtre, on élimine des paires qui ne sont pas significatives, et des paires de mots qui ont une relation importante. À titre d'exemple, la figure 4.2 contient les paires de mots d'un document de la collection (AP900302-33) qui concernent le mot "senate" lorsqu'une fenêtre de 5 mots est considérée (colonne de gauche) puis dans le cas d'une fenêtre de 1 mot (colonne de droite). On peut voir que si certaines paires sont légitimement éliminées

fenêtre de 5 mots	fenêtre de 1 mot
125 senate	78-19 senate
78-19 senate	approve senate
amendment senate	balky senate
approve senate	day senate
attempt senate	fight senate
balky senate	lawmaker senate
bicker senate	senate thursday
day senate	senate vote
fight senate	
floor senate	
house senate	
intense senate	
lawmaker senate	
manager senate	
maneuvering senate	
million senate	
pile senate	
plan senate	
sen senate	
senate shoehorn	
senate sponsor	
senate spur	
senate thursday	
senate unrelated	
senate vote	

FIG. 4.2 – Affinités lexicales du document AP900302-33, gardées avec une fenêtre de 5 mots et de 1 mot, contenant le mot "senate."

(ex : "intense senate"), d'autres ne devraient peut-être pas l'être (ex : "house senate").

Nous rapportons les résultats des expériences de plusieurs tailles de fenêtre *win* dans le tableau 4.11. L'approche de lissage utilisée pour ces tests est le lissage backoff (équation 4.20), avec λ_d=0.1. Des fenêtres de 5, 2, et 1 mots ont été évaluées pour les documents ainsi que les requêtes. Pour les requêtes, la performance est meilleure avec la fenêtre la plus grande, de 5 mots. Une requête est normalement plutôt concise, ne contenant que les mots essentiels pour exprimer le besoin d'information. Il est donc fortement possible que des mots reliés se trouvent dans une fenêtre relativement grande.

La taille optimale de fenêtre employée pour les documents dépend de la longueur des requêtes. Les meilleurs résultats pour les requêtes titres (qui ont en moyenne 2.2 affinités lexicales, voir le tableau 5.3) sont obtenus quand la fenêtre de mots des documents est 5, tandis

		générer la requête équation 4.25		KL-div équation 4.27	
win_d	win_q	titre	titre&dsc	titre	titre&dsc
5	5	0.2587	0.2903	0.2512	0.3034
2	5	0.2541	0.2929	0.2469	0.3047
2	2	0.2550	0.2848	0.2470	0.3030
1	5	0.2480	0.3002	0.2357	0.3141
1	1	0.2105	0.2740	0.2111	0.2887

TAB. 4.11 – La distance entre mots d'une paire

qu'une fenêtre de 1 dans les documents est optimale pour les requêtes titres et descriptions (qui ont 18.2 paires en moyenne). Si une requête a un grand nombre d'affinités lexicales, une fenêtre petite dans les documents limite le nombre de documents non-pertinents retrouvés qui contiennent juste des paires moins importantes de la requête. En revanche, si une requête ne contient qu'une ou deux paires et si on ne cherche ces paires que dans une fenêtre d'un mot dans les documents, il est probable qu'aucune paire de la requête ne se figure dans le score de calcul d'une grande partie des documents pertinents, qui contiennent les paires dans une fenêtre de 2 ou 3 mots.

4.3.6 Résumé du modèle AL-1

Nous pouvons observer que le modèle AL-1 permet une précision plus élevée que les approches de la section 4.2 qui ne comprennent pas les mots simples dans le score de pertinence. Cependant, le modèle AL-1 que nous avons développé ne dépasse pas non plus le baseline. Cela est probablement à cause des contributions relatives des paires et des mots simples dans ce modèle. Quelle que soit la valeur du coefficient des paires observées, λ_2, la contribution des mots simples dans le score de pertinence est beaucoup plus basse que la contribution des paires. Dans la section 4.3.1, nous avons vu un exemple d'un document typique où $\gamma = \frac{1-\lambda_2}{146}$. Nous pouvons voir que la contribution des paires dans l'équation 4.22 est disproportionellement grande par rapport aux mots simples. Nous décrivons maintenant un deuxième modèle qui résout ce problème.

4.4 Modèle AL-2 : Traiter les affinités lexicales comme des unités atomiques

Une dernière approche que nous avons testée qui emploie les affinités lexicales consiste à considérer une affinité lexicale comme une unité lexicale complète et ne pas la décomposer en mots indépendants lors d'un lissage. Par exemple, la paire qui contient les deux mots "famine" et "sudan" est considérée comme un mot "famine,sudan". Ces unités lexicales sont combinées avec les mots simples dans un seul modèle. Dans cette section, nous présentons le modèle AL-2 (section 4.4.1), nos expériences (section 4.4.2), et une discussion de l'utilité de ce modèle (section 4.4.3).

4.4.1 Un nouveau modèle unigramme

Pour l'entraînement d'un modèle unigramme qui contient les paires ainsi que les mots simples, nous modifions les fréquences des paires, et cherchons à optimiser les contributions relatives des paires. Formellement, le compte utilisé d'un mot ou d'une paire est comme suit :

$$c_d^*(w') = \begin{cases} c_d(w) & \text{si } w' \text{ est un mot simple } w \\ \beta_d c_d(< u, v >) & \text{sinon} \end{cases} \tag{4.38}$$

où w' est soit un mot simple w, soit une paire $< u, v >$ représentée comme un mot, et $c_d(w')$ est le compte observé. Les comptes observés des paires sont multipliés par le paramètre β_d, afin de configurer la contribution relative des paires par rapport aux mots simples, aux modèles des documents. Dans ce travail, β_d est constant à travers la collection, et ne dépend pas du document.

La fréquence modifiée d'une paire peut aussi se baser sur le pouvoir de résolution de la paire plutôt que sur sa fréquence observée :

$$c_d^*(w') = \begin{cases} c_d(w) & \text{si } w' \text{ est un mot simple } w \\ \beta_d \rho_d(< u, v >) & \text{sinon} \end{cases} \tag{4.39}$$

Les probabilités des mots w et des paires $< u, v >$ se basent sur leurs fréquences relatives

et sont alors :

$$p_{d_{MLE}}(w) = \frac{c_d^*(w)}{\sum_{w'} c_d^*(w')}$$
$$p_{d_{MLE}}(< u, v >) = \frac{c_d^*(< u, v >)}{\sum_{w'} c_d^*(w')} \tag{4.40}$$

Pour pallier au problème de sous-représentation des données, nous pouvons appliquer les approches décrites dans le chapitre 3 pour les modèles unigramme. Spécifiquement, une probabilité fixe $p(unk)$ peut être allouée aux mots et paires non-vus dans un document d :

$$p_{d_{fixe\_collection}}(w') = p_{d_{MLE}}(w') \times (1 - p(unk)) \tag{4.41}$$

Nous pouvons aussi entraîner un modèle du corpus $p_{corpus_{MLE}}(w')$ de la même manière que pour les modèles de document $p_{d_{MLE}}(w')$ avec un paramètre β_{corpus}, et interpoler ce modèle avec le modèle du document :

$$p_{d_{corpus\_interp}}(w') = \lambda_d p_{d_{MLE}}(w') + (1 - \lambda_d) p_{corpus\_fixe}(w')) \tag{4.42}$$

où le modèle de corpus alloue une probabilité fixe $p_{corpus}(unk)$ aux mots et paires non-vus dans la collection :

$$p_{corpus\_fixe}(w') = p_{corpus_{MLE}}(w') \times (1 - p_{corpus}(unk)) \tag{4.43}$$

Le score de pertinence d'un document d pour une requête $q = \{w'_1, \ldots, w'_N\}$ est la probabilité de générer la requête selon le modèle du document :

$$\begin{aligned} score(d, q) = p_d(q) &= \prod_{i=1}^{N} p_d(w'_i) \\ &= \prod_{w' \in q} p_d(w')^{c_q(w')} \end{aligned} \tag{4.44}$$

Nous pouvons aussi optimiser les contributions relatives des mots simples et des paires dans la requête, et modifier leur fréquence avec les facteur β_q comme pour les documents. Le score est alors :

$$score(d, q) = \prod_{w' \in q} p_d(w')^{c_q^*(w')} \tag{4.45}$$

	titres				titres et descr			
β_q / β_d	0.1	0.5	1	2	0.1	0.3	0.5	1
0.01	0.3575	0.3581	0.3577	0.3582	0.3442	0.3446	0.3437	0.3448
0.1	0.3590†	0.3660*†	0.3662*†	0.3606†	0.3477*	0.3491*	0.3537*	0.3458
0.5	0.3483	0.3633†	0.3588	0.3510	0.3526*	0.3572*†	0.3510*	0.3396
1	0.3429	0.3585	0.3529	0.3456	0.3559*†	0.3588*†	0.3493*	0.3367
baseline	0.3349				0.3498			
unigramme sans paires	0.3639				0.3464			
bigramme	0.3589				0.3544			

TAB. 4.12 – Précision obtenue avec un modèle unigramme qui contient les mots simples et les paires représentées comme mots simples, avec plusieurs valeurs de β_q et β_d. La fréquence modifiée des paires $c^*(< u, v >)$ est calculée selon l'équation 4.38. Les résultats qui dépassent la précision du baseline et du modèle unigramme sans affinités lexicales sont indiqués avec une étoile (*), et ceux qui dépassent le baseline et le modèle bigramme sont indiqués avec une croix (†).

4.4.2 Expériences

Dans cette section nous présentons les résultats de plusieurs expériences avec le modèle AL-2. Premièrement, nous présentons les performances avec un lissage fixe (équation 4.41), lorsque les comptes bruts des paires (équation 4.38) ou les pouvoirs de résolution (équation 4.39) sont employés. Ensuite nous étudions l'effet d'un filtrage sur les paires dans les requêtes. Finalement, les précisions mesurées avec lissage avec un modèle de corpus (équation 4.42) sont presentées.

Utiliser la fréquence brute d'une paire

Dans le tableau 4.12, nous rapportons les précisions obtenues avec le modèle AL-2 lissé avec une probabilité fixe $p(unk)$ pour les mots inconnus (l'équation 4.41), avec plusieurs combinaisons de β_q et β_d, les facteurs des contributions des paires dans les requêtes et documents. Dans ces expériences, nous avons utilisé la fréquence observée d'une paire (l'équation 4.38) plutôt que son pouvoir de résolution. Pour les documents, seulement les paires avec un pouvoir de résolution $\rho_d(< u, v >)$ supérieure à la moyenne + l'écart type $(\bar{\rho}_d(< u, v >) + \sigma)$ des paires sont comptées, tandis que toutes les paires des requêtes sont gardées. Les résultats du baseline ainsi que les meilleures précisions obtenues par le modèle unigramme qui emploie un lissage fixe $p(unk)$ pour les mots inconnus, décrit dans le chapitre 3, sont présentés.

	titres				titres et descr			
β_q / β_d	0.01	0.05	0.1	0.5	0.01	0.05	0.1	0.3
0.01	0.3585	0.3612†	0.3657*†	0.3531	0.3457	0.3497*	0.3571*†	0.3409
0.04	0.3488	0.3640*†	0.3637†	0.3494	0.3485*	0.3560*†	0.3539*	0.3163
0.1	0.3477	0.3642*†	0.3600†	0.3496	0.3506*	0.3585*†	0.3478	0.3026
0.5	0.3466	0.3462	0.3510	0.3282	0.3545*†	0.3501*	0.3339	0.2884
baseline	0.3349				0.3498			
unigramme sans paires	0.3639				0.3464			
bigramme	0.3589				0.3544			

TAB. 4.13 – Précision obtenue avec un modèle unigramme qui contient les mots simples et les paires représentées comme mots simples, avec plusieurs valeurs de β_q et β_d. La fréquence modifiée des paires $c^*(< u, v >)$ est calculée selon l'équation 4.39. Les résultats qui dépassent la précision du baseline et du modèle unigramme sans affinités lexicales sont indiqués avec une étoile (*), et ceux qui dépassent le modèle bigramme sont indiqués avec une croix (†).

Les résultats marqués d'une étoile (*) dépassent la précision du modèle unigramme du chapitre 3 qui ne contient pas les paires et qui emploie un lissage fixe $p(unk)$ (l'équation 3.7), et les résultats indiqués avec une croix (†) dépassent le modèle bigramme. On observe que, pour les requêtes titres, la précision n'est pas très sensible à la valeur des paramètres β_d et β_q, avec une plage de variation de 2.33% (de 34.29% à 36.62%) pour toutes les combinaisons essayées. De même, la plage de variation de précision pour les requêtes titre et description est de 2.2% (de 33.67% à 35.88%). Le modèle AL-2 dépasse le modèle unigramme sans paires, et le modèle bigramme, pour plusieurs combinaisons de β_d et β_q, avec plus de gains notés pour les requêtes titres et descriptions que pour les requêtes titres.

Utiliser le pouvoir de résolution des paires

Le tableau 4.13 montre les précisions obtenues lorsque le pouvoir de résolution est utilisé dans la fréquence modifiée des paires (l'équation 4.39). Comme le pouvoir de résolution d'une paire peut être entre 5 - 10 fois sa fréquence (voir le tableau 4.1 pour des exemples des fréquences et pouvoirs de résolution dans un document typique), les valeurs de β sont plutôt petites en comparaison avec les expériences dans le tableau 4.12 où la fréquence brute est employée (l'équation 4.38). Pour un document donné, seulement les paires avec un pouvoir de résolution ρ supérieur au pouvoir de résolution moyen plus l'écart type $\bar{\rho} + \sigma$ des paires dans le document sont comptées. Les combinaisons de β_d et β_q qui donnent une précision supérieure à celle du meilleur modèle unigramme sans paires (l'équation 3.7) sont indiquées

avec une étoile (*) et les résultats qui dépassent le modèle bigramme indiqués avec une croix (†). Là encore, cette approche est plus adaptée pour les requêtes longues.

On observe que la plage de variation des précisions obtenues est plus grande lorsque le pouvoir de résolution est utilisé plutôt que la fréquence brute des paires, soit de 3.75% pour les requêtes titres (de 32.82% à 36.57%) et de 7.01% pour les requêtes titres et descriptions (de 28.84% à 35.85%). De plus, une perte de précision est observée lorsque la contribution des paires dans la requête est relativement grande. Pour les requêtes titres, qui contiennent en moyenne 2.2 paires (voir le tableau 5.6 pour des statistiques du nombre de mots et paires dans les requêtes), la performance tombe pour $\beta_d \geq 0.5$ et $\beta_q \geq 0.5$, la précision la plus basse étant 32.82% pour $\beta_d = 0.5$ et $\beta_q = 0.5$. Les requêtes titres et descriptions, cependant, qui contiennent environ 8 fois le nombre de paires des requêtes courtes (18.2 paires en moyenne), sont plus sensibles à la contribution des paires dans la requête, β_q. La perte de performance est généralement notable pour $\beta_d \geq 0.5$ et $\beta_q \geq 0.3$, et tombe jusqu'à 28.8% pour $\beta_d = 0.5$ et $\beta_q = 0.3$.

Filtrer les paires dans les requêtes

Les deux expériences précédentes filtrent les paires dans les documents. Il est possible qu'un filtrage sur les paires dans une requête soit particulièrement utile pour les requêtes longues. Prenons les requêtes 37 et 47 dans notre collection (après pré-traitement) :

- requête 37 :

  `france west germany unity`

  `identify document discuss joint effort unity france west germany`

- requête 47 :

  `hole ozone layer`

  `concern negative effect hole ozone layer public health justify`

Le tableau 4.14 montre toutes les affinités lexicales de ces requêtes. Les affinités lexicales avec un pouvoir de résolution supérieur à la moyenne ($\rho \geq \bar{\rho}$) sont indiquées avec une étoile (*). Nous pouvons voir que les paires qui n'ont aucune importance comme "discuss,identify" et "discuss,document" de la requête 37, et "concern,effect"de la requête 47, sont enlevés, et que les paires qui sont probablement les paires les plus intéressantes de la requête, telles que "germany,west" et "france,germany" de la requête 37 et "layer ozone"de la requête 47, sont gardées.

Requête 37		
Paire	fréquence	ρ
*france,unity	2	15.1
*germany,unity	2	14.1
*unity,west	2	13.8
*france,germany	2	13.3
*france,west	2	12.9
*germany,west	2	12.0
joint,unity	1	7.7
identify,unity	1	7.5
identify,joint	1	7.2
france,joint	1	7.2
effort,unity	1	7.2
document,unity	1	7.4
document,joint	1	7.2
document,identify	1	7.1
document,france	1	7.1
discuss,unity	1	7.4
discuss,joint	1	7.1
discuss,identify	1	7.0
discuss,france	1	7.0
discuss,document	1	7.0
joint,west	1	6.6
germany,joint	1	6.7
effort,west	1	6.1
effort,joint	1	6.8
effort,identify	1	6.7
effort,germany	1	6.2
effort,france	1	6.7
document,effort	1	6.8
discuss,west	1	6.4
discuss,effort	1	6.6
$\bar{\rho}$	8.3	

Requête 47		
Paire	fréquence	ρ
*layer,ozone	2	18.6
*hole,ozone	2	17.6
*hole,layer	2	17.5
*justify,ozone	1	8.9
*justify,layer	1	8.9
*negative,ozone	1	8.7
*layer,negative	1	8.7
hole,justify	1	8.4
hole,negative	1	8.2
effect,ozone	1	8.1
effect,layer	1	8.0
health,ozone	1	7.9
concern,ozone	1	7.9
health,layer	1	7.8
concern,layer	1	7.8
ozone,public	1	7.7
layer,public	1	7.6
effect,hole	1	7.6
health,justify	1	7.5
effect,negative	1	7.5
health,hole	1	7.4
concern,hole	1	7.4
concern,negative	1	7.3
justify,public	1	7.3
hole,public	1	7.1
negative,public	1	7.1
effect,health	1	6.7
concern,effect	1	6.7
effect,public	1	6.4
health,public	1	6.2
$\bar{\rho}$	8.7	

TAB. 4.14 – Les fréquences et pouvoirs de résolution pour les requêtes 37 et 47. Les paires indiquées avec une étoile (*) sont celles avec un pouvoir de résolution $\rho \geq \bar{\rho}$.

	titres				titres et descr			
β_q / β_d	0.01	0.05	0.1	0.5	0.01	0.05	0.1	0.3
0.01	0.3479	0.3481	0.3510	0.3429	0.3449	0.3506*	0.3544	0.3509*
0.04	0.3487	0.3500	0.3498	0.3396	0.3482	0.3547	0.3541*	0.3419*
0.1	0.3472	0.3500	0.3474	0.3400	0.3502	0.3567	0.3557*	0.3402*
0.5	0.3423	0.3409	0.3359	0.3310*	0.3516	0.3534*	0.3474*	0.3345*
baseline	0.3349				0.3498			
unigramme sans paires	0.3639				0.3464			
bigramme	0.3589				0.3544			

TAB. 4.15 – Filtrer les paires dans les requêtes. Pour ces expériences, seulement les paires prometteuses des requêtes sont gardées, où $\rho \geq \bar{\rho}$. Les résultats indiqués avec une étoile (*) sont ceux qui donne une amélioration par rapport au tableau 4.13, où toutes les paires des requêtes sont gardées.

Cependant, l'utilité du filtrage dépend de la nature de la requête. Intuitivement, trois des 13 mots de la requête 37 ne sont pas importants pour la requête ("identify", "document", et "discuss"), tandis le mot "concern" est probablement le seul mot de la requête 47 qui n'est pas significatif. Par consequence, des paires probablement intéressantes de la requête 47 sont enlevées, comme "health,public." Il est à noter aussi qu'un tel filtrage ne devrait pas augmenter la performance du système pour les requêtes courtes qui ne contiennent qu'une ou deux affinités lexicales.

Nous rapportons dans le tableau 4.15 les performances qu'un filtrage sur les paires dans les requêtes amène. Pour ces expériences, le pouvoir de résolution est utilisé pour les documents (l'équation 4.39) et les requêtes, et seulement les paires prometteuses (avec $\rho \geq \bar{\rho} + \sigma$) sont gardées, dans les documents. Un filtrage moins sévère est appliqué aux requêtes, où les paires avec $\rho \geq \bar{\rho}$ sont gardées.

On voit qu'un filtrage sur les paires de la requête améliore la performance lorsque la contribution des paires (β_d ou β_q) est grande. Cependant, les meilleures précisions moyennes que nous avons obtenues avec ce filtrage, de 35.10% pour les requêtes titres, et de 35.67% pour les requêtes titres et descriptions, sont inférieures aux meilleures précisions moyennes obtenues quand toutes les paires de la requête sont gardées, soit de 36.57% et de 35.85%.

Une explication possible est que les requêtes, surtout les requêtes courtes, sont normalement plus concises en comparaison avec les documents, et un filtrage sur les requêtes peut causer

	titres				titres et descr			
β_q / β_d	0.005	0.01	0.05	0.1	0.005	0.01	0.05	0.1
$\beta_{corpus} = 0.0001$								
0.005	0.4140*	0.4154*	0.3948	0.3877	0.4219	0.4213	0.3961	0.3638
0.01	0.4136*	0.4154*	0.3932	0.3871	0.4221	0.4218	0.3965	0.3585
0.04	0.4064	0.4081*	0.3849	0.3793	0.4239	0.4239	0.3962	0.3519
0.1	0.4066	0.4087*	0.3849	0.3802	0.4264	0.4257	0.3904	0.3474
0.5	0.3798	0.3795	0.3773	0.3730	0.4192	0.4196	0.3813	0.3384
$\beta_{corpus} = 0.01$								
0.005	0.4126*	0.4145*	0.3977	0.3909	0.4221	0.4208	0.4049	0.3900
0.01	0.4125*	0.4143*	0.3958	0.3896	0.4229	0.4223	0.4027	0.3850
0.04	0.4059	0.4071	0.3880	0.3814	0.4238	0.4243	0.4004	0.3791
0.1	0.3955	0.4065	0.3855	0.3800	0.4225	0.4248	0.3984	0.3727
0.5	0.3795	0.3798	0.3769	0.3743	0.4196	0.4192	0.3924	0.3600
baseline	0.3349				0.3498			
unigramme sans paires	0.4071				0.4275			
bigramme	0.4072				0.4277			

TAB. 4.16 – Précision obtenue avec une interpolation avec un modèle de corpus (l'équation 4.42), avec plusieurs valeurs de β_d, β_q, et β_{corpus}. Dans toutes ces expériences, $\lambda_d = 0.5$. Les précisions indiquées avec une étoile (*) sont supérieures à celles obtenues par le modèle unigramme qui ne contient pas les paires ainsi que le modèle bigramme.

que des paires importantes soient enlevées, comme nous l'avons vu avec l'exemple de la requête 47.

Interpolation avec un modèle de corpus

Dans les tableaux 4.16 et 4.17, nous présentons la précision obtenue par le modèle AL-2, lorsque le modèle est lissé avec un modèle de corpus (l'équation 4.42).

Le tableau 4.16 montre les résultats pour plusieurs combinaisons de β_d, β_q, et β_{corpus} avec λ_d=0.5 pour toutes les expériences. Les combinaisons de β_d, β_q et β_{corpus} qui dépassent le meilleur modèle unigramme sans paires, et le meilleur modèle bigramme, sont indiquées avec une étoile (*). Avec les paramètres testés, la performance pour les requêtes titres dépasse celle des meilleurs modèles unigramme (sans paires) et bigramme, mais la performance pour les requêtes titres et descriptions est inférieure aux modèles unigramme et bigramme.

Dans le tableau 4.17, nous cherchons à déterminer la contribution optimale du modèle du corpus, et présentons les résultats pour plusieurs valeurs de λ_d. Ici le modèle AL-2 donne des

	titres		titres et descr	
$(\beta_d,\beta_q,\beta_{corpus})$ λ_d	(0.01,0.01,0.01)	(0.01,0.01,0.0001)	(0.01,0.005,0.01)	(0.04,0.005,0.0001)
0.1	0.3920	0.3939	0.4118	0.4095
0.2	0.4077*	0.4092*	0.4320*†	0.4314*†
0.4	0.4087*	0.4095*	0.4266	0.4274
0.5	0.4143*	0.4154*	0.4229	0.4239
0.6	0.4113*	0.4139*	0.4196	0.4200
baseline	0.3349		0.3498	
unigramme sans paires	0.4071		0.4275	
bigramme	0.4072		0.4277	

TAB. 4.17 – Précision obtenue avec une interpolation avec un modèle de corpus (l'équation 4.42), avec plusieurs valeurs de λ_d, β_d, β_q, et β_{corpus}. Les précisions indiquées avec une étoile (*) sont supérieures à celles obtenues par le modèle unigramme qui ne contient pas les paires. Une croix (†) indique une amélioration significative par rapport au modèle bigramme, selon le test de Wilcoxon des rangs signés avec un intervalle de confiance de 95%.

résultats supérieurs aux modèles unigramme et bigramme pour les deux types de requête. Pour les requêtes titres, le gain en performance le plus grand est de 40.72% à 41.54%, et pour les requêtes titre et description, de 42.77% à 43.20%.

4.4.3 Analyse du modèle

Dans nos expériences, nous avons montré que le modèle AL-2 peut dépasser un modèle unigramme sans paires, et un modèle bigramme, pour certaines combinaisons de paramètres (la contribution relative des paires β_d, β_q et β_{corpus}, et le poids du modèle de corpus (1-λ_d)). Le défi est de trouver les valeurs optimales de ces paramètres. Dans toutes les expériences, la précision diminue lorsque la contribution des paires, soit dans les documents, soit dans la requête, est trop grande.

Nous présentons maintenant deux exemples qui illustrent comment ce modèle peut amener un gain en précision (le tableau 4.18), ou une perte (le tableau 4.19), en comparaison avec un modèle unigramme sans paire décrit dans le chapitre 3. Afin de simplifier l'illustration, un lissage fixe $p(unk) = 0.0001$ est utilisé, et la contribution des paires dans la requête, β_q=1.

Illustration d'un gain (tableau 4.18)

Le document AP900301-222 a été jugé pertinent pour la requête titre #38. Ce document contient deux des trois mots de la requête ("debt" et "poland"), ainsi qu'une paire

Requête #38 :

`titre: conversion debt poland`

	1-gramme sans paires		AL-2	
	AP900301-222*	AP900622-142	AP900301-222*	AP900622-142
conversion	-	-2.28	-	-2.44
debt	-1.41	-2.28	-1.52	-2.44
poland	-1.11	-1.68	-1.22	-1.84
debt,poland	-	-	-1.54	-
conversion,debt	-	-	-	-
conversion,poland	-	-	-	-
score	-6.52	-5.24†	-16.28†	-18.72

TAB. 4.18 – Scores pour un document pertinent et non-pertinent, pour la requête #38. Une étoile(*) indique un document pertinent selon le jugement de pertinence, et une croix (†) indique le document auquel le modèle accorde un score plus élevé

("debt,poland"). Le document AP900622-142 a été jugé non-pertinent. Ce document contient les trois mots de la requête, mais avec des fréquences plus faibles que le document pertinent, et le document AP900622-142 ne contient aucune paire de la requête. Le modèle unigramme sans paire accorde un score plus élevé au document non-pertinent, tandis que le modèle qui comprend les paires favorise le document pertinent, AP900301-222.

Illustration d'une perte d'information (tableau 4.19)

Deux documents sont évalués pour la requête titre et description, #5. Le document AP900308-239, qui contient deux des trois mots de la requête ("acupuncture" et "study") a été jugé pertinent, tandis que le document AP901206-162, qui contient aussi deux mots de la requête ("case" et "study") ainsi qu'une paire ("case,study") a été jugé non-pertinent. Intuitivement, le document AP900308-239 devrait être plus pertinent, contenant le mot moins commun, "acupuncture". Bien que le document AP901206-162 contienne le même nombre de mots de la requête comme le document AP900308-239, ainsi qu'une paire de la requête, les mots "case" et "study" sont plutôt généraux, et ce document ne satisfait probablement pas le besoin d'information de l'usager. Dans cet exemple, le modèle unigramme sans paire accorde un score plus élevé au document pertinent. Cependant, la présence de la paire "case,study" augmente trop le score du document non-pertinent, AP901206-162, calculé par le modèle AL-2. Si les mots "case", "study", ainsi que les autres mots non-significatifs que nous avons vus dans les requêtes 37 et 47 (le tableau 4.14) comme "document" et "discuss" sont ajoutés à la stopliste, la possibilité que des paires non-informatives contribuent trop au score de pertience

Requête #5 :

```
titre: acupuncture
description: case study acupuncture
```

	1-gramme sans paires		AL-2	
	AP900308-239*	AP901206-162	AP900308-239*	AP901206-162
acupuncture	-2.42	-	-2.56	-
acupuncture	-2.42	-	-2.56	-
case	-	-1.32	-	-1.50
study	-1.72	-1.45	-1.86	-1.63
acupuncture,case	-	-	-	-
case,study	-	-	-	-1.93
acupuncture,study	-	-	-	-
score	-10.56†	-10.77	-22.98	-21.06†

TAB. 4.19 – Scores pour un document pertinent et non-pertinent, pour la requête #5. Une étoile(*) indique un document pertinent selon le jugement de pertinence, et une croix (†) indique le document auquel le modèle accorde un score plus élevé

sera minimisée.

Chapitre 5

La couverture des affinités lexicales, bigrammes, et mots simples

Dans ce chapitre, nous présentons des statistiques sur la couverture des unités lexicales (unigrammes, bigrammes, et affinités lexicales) des requêtes dans les documents jugés pertinents, ainsi que dans la collection au niveau global.

Les tableaux 5.1, 5.2, 5.3, 5.5, et 5.4 montrent la couverture des mots, bigrammes et paires d'une requête dans les documents jugés pertinents. Pour les deux types de requête (titres, titres et descriptions), le nombre moyen de mots simples, bigrammes, et paires, par requête, est rapporté. De même, le nombre moyen de mots (ou bigrammes, ou affinités lexicales) d'une requête qui sont également dans les documents pertinents est présenté. Les valeurs de précision et rappel pour l'ensemble des documents qui contiennent au moins un mot (ou bigramme, ou affinité lexicale) des requêtes sont montrées. Pour les affinités lexicales, la couverture des paires est présentée, selon la taille de fenêtre utilisée dans les documents (le tableau 5.3) et selon le filtrage des paires par pouvoir de résolution (le tableau 5.5).

Pour toutes les statistiques, seulement les mots (ou bigrammes ou affinités lexicales) uniques sont considérés. Par exemple, prenons la requête 47 de notre collection :

<title>hole ozone layer< /title>
<description>concern negative effect hole ozone layer public health justify< /description>

Et considérons un document qui contient les mots "ozone" et "layer" trois fois chacun. La requête contient 9 mots uniques, et le document contient deux mots de la requête. Dans ce cas, la couverture des mots de la requête dans le document est 2 / 9 = 22%.

Mots simples		
	titre	titre et desc
#mots par requête	2.5	6.9
#mots dans les documents pertinents	1.7	2.9
% des mots dans les docs pertinents	68%	42%
Précision	4%	2%
Rappel	95%	97%

TAB. 5.1 – Couverture des mots simples, par document : le nombre (et pourcentage) des mots d'une requête qui sont également dans un document pertinent. La moyenne, pour l'ensemble de documents pertinents d'une requête, et pour l'ensemble des requêtes, est montrée.

Bigrammes		
	titre	titre et desc
#bigrammes par requête	1.4	6.0
#bigrammes dans un document pertinent	0.44	0.53
% des bigrammes dans un doc pertinent	30%	9%
Précision	410/6353=6%	454/50645=1%
Rappel	410/1172=35%	454/1172=39%

TAB. 5.2 – Couverture des bigrammes, par document : le nombre (et pourcentage) des bigrammes d'une requête qui sont également dans un document pertinent. La moyenne, pour l'ensemble de documents pertinents d'une requête, et pour l'ensemble des requêtes, est montrée.

Pour la précision et le rappel, les calculs absolus sont montrés. Quant à la précision, le nombre de documents pertinents qui contiennent au moins un mot (ou bigramme ou paire) de la requête divisé par le nombre total de documents dans la collection qui contiennent au moins un mot est donné. En ce qui concerne le rappel, le nombre de documents pertinents qui contiennent au moins un mot divisé par le nombre de documents jugés pertinents est donné. Nous présentons maintenant quelques analyses qui permettent de comparer le niveau auquel une approche (par exemple, utiliser une fenêtre de deux mots dans les documents versus une fenêtre de cinq mots) augmente ou diminue le nombre de documents pertinents et non-pertinents qui contiennent un mot ou paire d'une requête.

Affinités lexicales						
	titre			titre et desc		
	$win_d = 5$	$win_d = 2$	$win_d = 1$	$win_d = 5$	$win_d = 2$	$win_d = 1$
#paires / req.	2.2			18.2		
#paires / doc. pert.	0.56	0.47	0.38	1.3	0.88	0.66
% des paires/ doc pert	25%	21%	17%	7%	5%	4%
Précision	$\frac{596}{8810} = 7\%$	$\frac{502}{7595} = 7\%$	$\frac{456}{6946} = 7\%$	$\frac{678}{77947} = 1\%$	$\frac{580}{65234} = 1\%$	$\frac{524}{58101} = 1\%$
Rappel	$\frac{596}{1172} = 51\%$	$\frac{502}{1172} = 43\%$	$\frac{456}{1172} = 39\%$	$\frac{678}{1172} = 58\%$	$\frac{580}{1172} = 49\%$	$\frac{524}{1172} = 45\%$

TAB. 5.3 – Couverture des affinités lexicales par document, selon la taille de fenêtre

5.1 L'effet de la taille de la fenêtre sur la couverture des affinités lexicales

Nous pouvons observer dans le tableau 5.3 qu'utiliser une fenêtre de 2 mots ou 1 mot dans les documents plutôt que 5 réduit la couverture. Plus précisément, le nombre de documents pertinents contenant au moins une paire de la requête, ainsi que le pourcentage des paires de la requête qui sont également dans un document pertinent, diminue. Le rappel de tous les documents de la collection avec au moins une paire de la requête diminue d'une manière significative si l'on emploie une fenêtre de 1 mot plutôt que 5 mots dans le document (par exemple, de 51% à 39% pour les requêtes titres), tandis que la précision de ces documents (7%) ne change pas. Autrement dit, en utilisant une fenêtre petite dans les documents, on perd des documents pertinents. Cependant, on perd même plus des documents non-pertinents, et parmi les documents qui restent, la précision ne décroît pas.

5.2 L'effet du filtrage des affinités lexicales sur la couverture de mots simples

Le tableau 5.4 montre la couverture des mots simples qui forment les affinités lexicales. Considérons encore la requête 47 et un document avec les paires "ozone layer" et "negative effect". Ici la couverture des mots simples est 4/9=44%, car le document contient 4 des mots uniques de la requête. Dans l'ensemble, si on n'applique aucun filtrage sur les affinités lexicales (toutes les paires sont gardées), la couverture des mots simples est égale à celle dans le tableau 5.1 (68% pour les requêtes titres, 42% pour les requêtes titres et descriptions). Si on garde juste les paires avec un pouvoir de résolution plus grand que la moyenne ($\rho \geq \bar{\rho}$), les paires gardées dans les documents pertinents contiennent moins des mots simples de la

Affinités lexicales						
	titre			titre et desc		
	toutes	$> \bar{\rho}$	$> \bar{\rho} + \sigma$	toutes	$> \bar{\rho}$	$> \bar{\rho} + \sigma$
#mots/ doc. pert.	1.7	1.6	1.33	2.9	2.5	1.9
% des paires/ doc pert	68%	64%	53%	42%	36%	28%
% de docs perts avec au moins 1 mot	95%	93%	84%	97%	96%	89%

TAB. 5.4 – Couverture des mots simples des affinités lexicales par document, selon le filtrage par pouvoir de résolution

Affinités lexicales						
	titre			titre et desc		
	toutes	$> \bar{\rho}$	$> \bar{\rho} + \sigma$	toutes	$> \bar{\rho}$	$> \bar{\rho} + \sigma$
#paires / req.	2.2			18.2		
#paires / doc. pert.	0.56	0.40	0.25	1.3	0.69	0.46
% des paires/ doc pert	25%	18%	10%	7%	4%	3%
Précision	$\frac{596}{8810} = 7\%$	$\frac{423}{3070} = 14\%$	$\frac{342}{2586} = 13\%$	$\frac{678}{77947} = 1\%$	$\frac{472}{21951} = 2\%$	$\frac{380}{16366} = 2\%$
Rappel	$\frac{596}{1172} = 51\%$	$\frac{423}{1172} = 36\%$	$\frac{342}{1172} = 29\%$	$\frac{678}{1172} = 58\%$	$\frac{472}{1172} = 40\%$	$\frac{380}{1172} = 32\%$

TAB. 5.5 – Couverture des affinités lexicales par document, selon le filtrage des paires par pouvoir de résolution

requête (64% pour les requêtes titres, 36% pour les requêtes titres et descriptions). Supposons dans notre exemple, que toutes les paires qui contiennent le mot "effect" ont un pouvoir de résolution faible ($\rho < \bar{\rho}$) mais que "negative" existe encore dans une autre paire avec un grand pouvoir de résolution. Dans ce cas, la couverture diminue à 3/9=33%, car le mot "effect" a été enlevé du document.

On observe dans le tableau 5.5 qu'en filtrant les paires avec un pouvoir de résolution faible, on perd des documents pertinents (le rappel diminue), mais la précision des documents restants augmente (de 7% à 14% pour les requêtes titres, et de 1% à 2% pour les requêtes titres et descriptions). Il est à noter que le filtrage plus sévère, où on ne garde que les paires avec un pouvoir de résolution $\rho > \hat{\rho} + \sigma$ diminue le rappel et n'amène pas de gain en précision (la précision décroît de 14% à 13% pour les requêtes titres, et demeure à 2% pour les requêtes longues).

	titres	titres et desc
# mots par requête	2.4	6.5
# mots de la requête présents au corpus	2.4	6.4
% couverture	100%	98%
# paires par requête	2.2	18.2
# paires de la requête présentes au corpus	1.9	5.6
% couverture	95%	31%

TAB. 5.6 – Couverture des mots et paires dans le corpus, où le corpus contient seulement les paires prometteuses, dont $\rho > \bar{\rho} + \sigma$

5.3 Couverture des unités lexicales dans la collection

Le tableau 5.6 montre la couverture des unigrammes et des affinités lexicales dans le corpus. Pour les requêtes titres dans notre collection, tous les mots simples sont présents dans le corpus, et presque tous (98%) les mots simples des requêtes longues sont aussi présents. Quant aux affinités lexicales, la couverture des paires des requêtes est inférieure à celle des mots simples, surtout pour les requêtes longues, dont seulement 31% des paires sont présentes dans la collection.

5.4 Résumé : Comparaison de couverture des mots simples, bigrammes, et affinités lexicales

On peut observer la magnitude du problème des mots inconnus, car un document pertinent ne contient habituellement pas un pourcentage significatif des mots d'une requête. Au fur et à mesure que la taille de la requête augmente, la couverture des mots de la requête dans un document pertinent diminue. Dans notre corpus, la couverture des mots simples décline de 68% pour les requêtes titres à 42% pour les requêtes avec les titres et descriptions. De plus, les documents de la collection ne contiennent que 30% des bigrammes des requêtes courtes et 9% des bigrammes des requêtes longues. La couverture des affinités lexicales (25%) est inférieure à celle des bigrammes (30%), pour les requêtes courtes, ainsi que pour les requêtes longues (9% couverture des bigrammes, 7% couverture des affinités lexicales). La précision et le rappel des documents contenant au moins une affinité lexicale de la requête approche la précision et le rappel des bigrammes, quand la fenêtre dans les documents est de 1 mot. Cela n'est pas étonnant si l'on considère un bigramme comme un cas spécial d'affinité lexicale,

avec une fenêtre d'un mot dans une direction. Il faudra rappeler, pourtant, qu'une fenêtre de 5 mots est toujours utilisée dans les requêtes. En conclusion, il est clair qu'on ne peut pas compter uniquement sur les affinités lexicales ou bigrammes pour retrouver les documents pertinents d'une requête.

Chapitre 6

Conclusion

Dans notre travail, nous avons étudié différentes techniques de lissage des modèles n-gramme pour la recherche d'information ainsi que l'utilisation des affinités lexicales, paires de mots sans contrainte sur l'ordre ou l'adjacence, dans des modèles de langue en recherche d'information.

Comme les approches antérieures aux modèles de langue en recherche d'information, nous avons observé des améliorations intéressantes avec un modèle unigramme par rapport au modèle vectoriel classique. Les améliorations légères que nous avons observées avec l'approche bigramme sont aussi cohérentes avec les approches décrites dans la littérature.

Des améliorations ont été observées quand les mots composés sont utilisés dans le modèle vectoriel [13]. Nous avons développé deux nouveaux modèles de langue qui incorporent des paires de mots, et avons vu des améliorations par rapport aux modèles de langue unigramme et bigramme, lorsque la contribution des paires de mots au score de pertinence est faible en comparaison avec les mots simples.

6.1 La fréquence globale des mots

6.1.1 Le modèle unigramme

Comme dans le modèle vectoriel classique, avec le facteur *idf*, nous avons observé dans nos expériences qu'il est important de prendre en compte la fréquence globale des mots dans un score de pertinence, en particulier avec les requêtes plus longues. Nous avons étudié deux classes de lissage pour le modèle unigramme : un lissage fixe, où une probabilité fixe $p(unk)$ (soit par document, soit constante à travers la collection) est allouée à tous les mots innconnus

	sans modèle de corpus				avec un modèle de corpus			
modèle	titres		titres et descr		titres		titres et descr	
baseline	33.49		34.98		33.49		34.98	
unigramme	36.39	(8.7)	34.64	(-1.0)	40.71	(21.6)	42.75	(22.2)
bigramme	35.89	(7.2)	35.44	(1.3)	40.72	(21.6)	42.77	(22.3)
AL-1	25.77	(-23.1)	28.84	(-17.6)	25.87	(-22.8)	30.34	(-13.3)
AL-2	36.62	(9.3)	35.88	(2.6)	41.54	(24.0)	43.20	(23.5)

TAB. 6.1 – Les meilleures précisions mesurées pour chaque modèle, lorsqu'un modèle de corpus est incorporé dans le score de pertinence ou pas. La précision et le gain relatif par rapport au modèle vectoriel sont montrés.

d'un document, et une integration avec un modèle de corpus, qui tient compte des fréquences globales des mots. Notre modèle unigramme dépasse en performance le modèle vectoriel avec le lissage fixe seulement pour les requêtes courtes, de 2-3 mots. Nous avons constaté une amélioration importante en performance lorsque le modèle de corpus est employé, le gain en performance étant plus marqué pour les requêtes longues. De plus, la contribution optimale du modèle de corpus au score de pertinence est plus grande pour les requêtes longues. L'utilisation du modèle de corpus permet au modèle unigramme de dépasser le modèle vectoriel de manière significative pour les requêtes longues et courtes.

6.1.2 Les modèles d'affinités lexicales

Pour les affinités lexicales, la fréquence globale des mots individuels figure dans l'information d'une paire $< u, v >$, INFO($< u, v >$). Dans nos modèles qui utilisent les affinités lexicales, AL-1 et AL-2, nous observons les meilleurs résultats lorsque l'information des paires est comprise dans le score de pertinence, plutot qu'utiliser uniquement les fréquences des paires dans les documents. Comme pour les modèles unigramme, cette information globale amène un gain en performance plus marqué pour les requêtes plus longues.

Comme pour le modèle unigramme, les deux classes de lissage, fixe et avec l'integration d'un modèle de corpus, ont été evaluées avec les modèles AL-1 et AL-2. Là encore, les performances sont supérieures lorsque le modèle de corpus est incorporé dans le score de pertinence.

Nous présentons dans le tableau 6.1 un sommaire des précisions obtenues avec les modèles unigramme, bigramme, AL-1 et AL-2. Nous comparons les meilleures performances où l'approche n'utilise pas un modèle de corpus et lorsqu'un modèle de corpus est incorporé dans le score de pertinence.

6.2 L'importance relative des mots simples et bigrammes ou paires

Comme nous l'avons presenté dans le chapitre 5, la couverture des bigrammes et les paires d'une requête dans les documents pertinents est plutôt faible en comparaison avec la couverture des mots simples. Bien que l'intégration de bigrammes ou paires peuvent améliorer la performance d'un système de recherche d'information, la manière dont les bigrammes ou paires sont compris dans le score de pertinence a un effet important sur leur utilité. Si trop de poids est accordé aux bigrammes ou paires, le problème de sous représentation aboutit à une perte en performance.

6.2.1 Bigrammes

Deux approches à l'integration des bigrammes au score de pertinence ont été evaluées : interpolation, où les mots simples sont toujours compris dans le score, et backoff, où les mots simples sont consultés seulement quand un bigramme de la requête n'est pas présent dans un document. Quand les mots simples sont toujours compris dans le score de pertinence (interpolation), incorporer les bigrammes permet une légère amélioration. Il convient de noter, cependant, que l'amélioration est possible seulement lorsque la contribution relative des bigrammes au score est plutôt faible. L'approche backoff n'amène pas d'amélioration.

6.2.2 Affinités lexicales

Avant d'explorer des modèles de langue qui incorporent les affinités lexicales et les mots simples, nous avons evalué les performances possibles avec un score basé uniquement sur les paires. Avec ces approches, les précisions obtenues sont toutes inférieures au modèle unigramme ainsi que le modèle vectoriel.

Le modèle AL-1 est basé principalement sur les affinités lexicales, et incorpore les mots simples dans le lissage des paires. Bien que la contribution des mots simples au score de pertinence soit configurable, elle est toujours plutôt faible par rapport à la contribution des paires. Les résultats sont meilleurs que les approches qui employent uniquement les paires mais sont toujours moins bons que les modèles unigramme et bigramme.

Dans le modèle AL-2, les mots simples et paires représentées comme mots simples, sont combinés dans un modèle unigramme, et les poids relatifs des paires sont plus configurables

qu'avec le modèle AL-1. Cette approche dépasse les modèles unigramme et bigramme. Comme pour les bigrammes, les améliorations sont possibles lorsque la contribution des paires au score est relativement basse.

6.3 Travail futur

Plusieurs études sont possibles pour tenter d'améliorer nos résultats. Dans les expériences décrites ici, chaque paramètre est constant à travers la collection, et ne dépend pas d'un document ou d'une requête. Une amélioration possible consiste à rendre le poids donné au modèle de corpus dépendant de chaque mot de la requête. Pour une requête donnée, les n premiers documents retrouvés par le système sont considérés pertinents, et un algorithme EM est utilisé pour optimiser les paramètres pour cet ensemble de documents. Cette approche a été étudiée par Hiemstra [7] pour le modèle unigramme, mais peut s'appliquer à tous les modèles developpés dans ce travail.

Nous avons vu dans le chapitre 4 que les requêtes longues contiennent souvent plusieurs paires qui ne sont probablement pas très pertinentes. Cependant, nos tentatives de filtrer les paires de la requête n'ont pas offert des améliorations importantes. Pour le modèle AL-1, nous avons essayé de limiter la fenêtre des mots où les paires se trouvent, à deux ou un mot (à droite ou à gauche d'un mot donné), et pour le modèle AL-2, le filtrage éliminait des paires avec un pouvoir de résolution relativement faible ($\rho < \bar{\rho}$). Avec ce dernier filtrage, seulement environ 20-25% des paires d'une requête étaient gardées. Un filtrage possible est d'éliminer les paires avec un pouvoir de résolution faible, mais avec un filtrage moins sévère. Par exemple, 10% des paires peuvent être eliminées, selon leur pouvoir de résolution.

Un autre filtrage possible sur les affinités lexicales, qui s'applique aux documents et requêtes, consiste à employer un *part-of-speech tagger* et de filtrer les mots selon les parties du discours, ou de ne garder que les paires dans une relation particulière, comme un groupe nominal ou une relation modificateur-modifié.

Un problème potentiel lorsque des mots composés sont incorporés dans la recherche d'information est la double représentation de mots. Si la paire "famine,sudan" apparaît dans un document ou une requête, le mot "famine" sera représenté deux fois dans notre modèle AL-2 : comme le mot simple "famine" et dans la paire "famine,sudan". Il est possible que la performance soit améliorée si le modèle prend en compte les relations entre les paires et

les mots simples. Dans l'approche backoff de notre modèle AL-1, cette double représentation n'est pas présente, car les mots simples d'une paire sont compris dans le score de pertinence seulement si la paire n'est pas présente dans le document. Cependant, les précisions moyennes mesurées avec ce modèle sont relativement basses, à cause de la contribution trop faible des mots simples au score de pertinence. Une approche possible qui peut résoudre ce problème consiste à utiliser le modèle AL-2, où les mots simples ont une contribution importante, et à représenter la paire "famine,sudan" soit par les deux mots simples "famine" et "sudan", soit par la paire "famine,sudan". Le choix entre ces deux représentations peut se baser sur un lexique de mots composés, ou sur une analyse statistique du corpus.

Bibliographie

[1] Berger, A., et Lafferty, J.D. Information retrieval as statistical translation. *Research and development in information retrieval*, pages 222-229, 1999.

[2] Chen, S.F, et Goodman, J. An empirical study of smoothing techniques for language modeling. *Tehnical report TR-10-98*, Harvard University, 1998.

[3] Fuhr, N. Probabilistic models in information retrieval. *The computer journal*, 35 (3), pages 243-255, 1992.

[4] Goodman, J. A bit of progress in language modeling. *Technical report, Microsoft Research 2000*, 2000.

[5] Hiemstra, D., et de Vries, A. Relating the new language models of information retrieval to the traditional retrieval models. *CTIT Technical report TR-CTIT-00-09, Centre for Telematics and Information Technology*, May 2000.

[6] Jin, R., Hauptmann, A.G., et Zhai, C.X. Title language model for information retrieval. *Proceedings on the 25th annual international ACM SIGIR conference*, pages 42-48, 2002.

[7] Hiemstra, D. Term-specific smoothing for the language modeling approach to information retrieval : the importance of a query term. *25th annual international ACM SIGIR conference on research and development in information retrieval*, Tampere, Finland, 2002.

[8] Lafferty, J.D., et Berger, A. The Weaver system for document retrieval. *Proceedings of the eighth Text REtrieval Conference (TREC-8)*, 1999.

[9] Lafferty, J., et Zhai, C. Document language models, query models and risk minimization for information retrieval. *Proceedings on the 24th annual international ACM SIGIR conference*, pages 111-119, 2001.

[10] Lavrenko, V. et Croft, W.B. Relevance-based language models. *Research and development in information retrieval*, pages 120-127, 2001.

[11] Maarek, Y.S., Berry, D.M., et Kaiser, G.E. An information retrieval approach for automatically constructing software libraries. *IEEE transactions on software engineering, 17(8)*, pages 800-813, 1991.

[12] Martin, W.J.R., Al, B.P.F., et van Sterkenburg, P.J.G. On the processing of a text corpus : From textual data to lexicographic information. *Lexicography : Principles and Practice*, R.R.K. Hartmann, editor. Applied Language Studies Series, Academic Press, London, 1983.

[13] Nie, J.-Y., et Dufort, J.F. Combining words and compound terms for monolingual and cross-language information retrieval, Information 2002, Beijing, July 2002.

[14] Ponte, J.M., et Croft, W.B. A language modeling approach to information retrieval. *Proceedings of SIGIR '98*, pages 275-281. Melbourne, Australia, 1998.

[15] Rosenfeld, R. Two decades of statistical language modeling : Where do we go from here ? *Proceedings of the IEEE*, 88(8), 2000.

[16] Signhal, A., Buckley, C., et Mitra, M. Pivoted document length normalization. *Research and development in information retrieval*, pages 21-29. 1996.

[17] Song, F., et Croft, W.B. A General Language Model for Information Retrieval. *Proceedings on the 22nd annual international ACM SIGIR conference*, pages 279-280, 1999.