Indexation des documents et Recherche d'Information

Ahmed Belkhir

Résumé

Dans ce rapport, on présente le travail qu'on a fait dans le projet de conception d'un système de recherche d'information. On a commencé par l'indexation d'une collection de documents avec Apache Solr en faisant les traitements nécessaires. Ensuite, on construit différents systèmes de recherche d'informations avec différents prétraitements et avec différents types de requêtes (requêtes courtes et longues). On a montré que BM-25 est le meilleur des 4 schémas de pondérations qu'on a testé. Dans une deuxième partie, on a essayé d'améliorer le système de recherche d'informations en utilisant les synonymes de WordNet et on a obtenu une légère amélioration du score MAP.

Mots clés: Recherche d'information, Traitement Automatique du langage Naturel, Apache Solr, Amélioration des requêtes

1 Introduction

Dans le cadre du cours Traitement Automatique du langage Naturel (INF7546) enseigné à l'UQAM par Prof. Fatiha Sadat, on a réalisé un projet de système de recherche d'information avec Apache Solr. La recherche d'information (RI) est une activité visant à obtenir les documents les plus pertinents par rapport à une requête donnée. Dans une première partie, on a commencé par faire l'indexation des documents et on a conçu différents systèmes avec différentes configurations de prétraitement afin de les comparer en se basant sur un document de jugement de pertinence. Dans une deuxième partie on a essayé d'améliorer le système de recherche d'information en faisant l'expansion de requêtes avec les synonymes de WordNet. On va présenter les résultats obtenus dans la dernière partie.

2 État de l'art

Dans le traitement automatique du langage naturel, et afin de faciliter la manipulation de l'information contenue dans les textes, le stemming (ou racinisation) (Lovins, 1968) est une technique souvent utilisée. C'est un procédé de transformation des flexions en leur radical ou racine. La racine d'un mot correspond à la partie du mot restante une fois que l'on a supprimé son (ses) préfixe(s) et suffixe(s). On a montré que cette technique améliore les résultats de recherche d'information (Orengo et al., 2006). La lemmatisation est une technique similaire. La différence est que le stemming consiste à supprimer les derniers caractères d'un mot, ce qui entraîne souvent des erreurs de sens et d'orthographe. La lemmatisation par contre tient compte du contexte et convertit le mot en sa forme de base significative et qui a un sens. On a comparé les deux techniques dans l'article (Balakrishnan and Lloyd-Yemoh, 2014). Une autre technique très utilisée est la tokenisation (Webster and Kit, 1992) qui consiste à l'identification de chaque unité "atomique" dans les mots, c'est la conversion d'une chaîne de caractères en une liste de symboles (appelés tokens).

Dans la recherche d'information, plusieurs schémas de pondérations peuvent être utilisés afin de faire le classement des documents selon leur pertinence. Par exemple, "Term frequency-inverse document frequency" (tf-idf) est l'un des schémas de pondération les plus couramment utilisés dans les systèmes actuels de recherche d'information. C'est le produit entre la fréquence du terme de la requête (Luhn, 1957) et la fréquence inverse du document (Jones, 1972). Il en existe d'autres comme Okapi-BM25 (Robertson et al., 1995).

Pour améliorer les performances des systèmes de recherche d'information, on fait souvent des techniques d'expansion de requêtes à travers plusieures techniques telles que la l'utilisation des plongements de mots (Ganguly et al., 2015) ou encore l'utilisation des ontologies ou bases lexicales comme WordNet (Pal et al., 2014).

3 Données et ressources utilisées

3.1 Collection de données

Dans ce projet, on a utilisé une collection de données conçue à partir d'un ensemble d'articles de nouvelles publiées par Associated Press entre 1988 et 1990. Cette collection comporte ce qui suit :

- Des documents en format XML
- Un ensemble d'exemples de requêtes en Anglais
- Un ensemble de jugements de pertinence pour ces mêmes requêtes

La base de donnée contient 1050 fichiers contenant au total plus que 200000 documents. Les requêtes sont du nombre de 8618 avec les jugements de pertinence correspondants.

3.2 Plateforme du système RI

Pour concevoir le système de recherche d'informations, on a utilisé Apache Solr qui est une plateforme de recherche d'informations basée sur basé sur Apache Lucene, écrite en Java, libre d'utilisation et conçue pour fournir une faible latence. En effet, Solr s'est imposé comme la plate-forme de facto pour la création d'applications de production. À l'exception de quelques moteurs de recherche Web commerciaux qui déploient une infrastructure personnalisée pour atteindre l'échelle nécessaire, la plupart des organisations tirent aujourd'hui parti de Solr, notamment Best Buy, Bloomberg, Comcast, Disney, eHarmony, Netflix, Reddit et Wikipedia. Bien que Solr soit conçu pour être évolutif via une architecture distribuée et partitionnée, on a utlisé notre machine locale pour ce projet. la plateforme est principalement .

4 Méthodologie

Dans cette section, on va expliquer la méthode qu'on a utilisée pour concevoir le système de recherche d'information. Il y en a deux parties: d'abord on commence par l'indexation des document avec différents prétraitements, et ensuite on essaye d'améliorer le meilleur système avec l'expansion des requêtes.

4.1 Partie 1: Indexation et différents prétraitements

4.1.1 Préparation des documents

Pour indexer les documents mentionnés précédemment, on a d'abord créé un noyeau (core) Solr qui va contenir l'index des documents. Avant cela, on a fait des modifications sur les documents .xml afin que l'indexation puisse s'effectuer sans erreurs. En bref, on a écrit un script python qui traite chacun des 1050 documents en introduisant une balise racine, en modifiant chaque balise par une synthaxe personnalisée. Par exemple, on a remplacé <DOCNO> par <field name="DOCNO"> et terminé chaque balise par </field>. Ces modifications entre autres ont été nécessaires car, sans elles, on aurait plein d'erreurs et le système de recherche d'informations n'aurait pas fonctionné avec la base de données.

4.1.2 Prétraitements

Avant d'indexer les documents avec Solr, on a configuré les prétraitements suivants dans l'ordre:

- Tokenisation avec StandardTokenizer (ST)
- Suppression des stop words avec StopFilter (SF)
- Remise du tout en miniscule avec LowerCaseFilter (LCF)
- Suppression des termes possessifs avec EnglishPosessFilter (EPF)
- Marquage des mots clés avec SetKeywordMarkerFilter (SKMF)
- Stemming de Porter avec PorterStemFilter (PSF)

Tous ces traitements là ont été faits l'un après l'autre grâce aux classes de Lucene mis en disposition à travers le fichier schema.xml qui a dû être personnalisé pour faire les modifications manuellement et non avec l'interface web Solr Admin UI et managed-schema qui n'offraient pas ce niveau de personnalisation. La figure 1 illustre les prétraitements sur la phrase "Apple's success is because Apples' coders ate apples.".

Ensuite, l'indexation a été faite par la commande linux suivante:

```
bin/post -c nom_core chemin_données
be
```

ST	Apple's	success	is	because	Apples	coders	ate	apples
SF	Apple's	success		because	Apples	coders	ate	apples
LCF	apple's	success		because	apples	coders	ate	apples
EPF	apple	success		because	apples	coders	ate	apples
SKMF	apple	success		because	apples	coders	ate	apples
PSF	appl	success		becaus	appl	coder	at	appl

Figure 1: Le pipeline de prétraitement

L'indexation a été faite en total 8 fois avec 4 schémas de pondérations différents, chacun avec des prétraitements et sans prétraitements. Les schémas de pondérations utilisés sont:

- TF-IDF

C'est le produit entre la fréquence du terme de la requête et la fréquence inverse du document. C'est souvent utilisé dans les systèmes RI.

- Okapi-BM25

C'est un modèle probabiliste développé dans les années 1970 et 1980 par Stephen E. Robertson, Karen Spärck Jones et d'autres spécialistes de l'informatique. (Robertson et al., 1995).

- LMDirichlet

C'est un modèle dont la formule attribue un score négatif aux documents qui contiennent le terme, mais dont le nombre d'occurrences est inférieur à celui prédit par le modèle de langage de la collection (Zhai and Lafferty, 2004).

- SweetSpot

C'est un modèle qui permet de définir un "point idéal" pour la longueur d'un champ en fonction d'une plage définie par min et max. Les champs dont la longueur se situe dans cette fourchette obtiendront le même score, tandis que les champs dont la longueur se situe en dehors de cette fourchette obtiendront un score inférieur,

4.1.3 Automatisation des requêtes

Chaque fois qu'on a fait l'indexation, on fait deux types de requêtes à partir du fichier des requêtes mentionné ci-dessus: des requêtes courtes avec le champ "title" seulement, et des requêtes longues avec le champ "title" et le champ "description". Le script Python développé lit le fichier des requêtes, détecte les champs concernés dans chaque requêtes, supprime les mots inutiles dans le champ descroption (par example "This document discusses"), fait une requête HTTP à travers curl pour communiquer avec Solr et écrit la réponse dans un fichier texte avec le formatage adéquat pour être évalué par la suite par trec_eval.

4.2 Partie 2: Amélioration du système RI

Après avoir fait les requêtes par les différentes méthodes, on a tenté d'améliorer le système de recherche d'information en faisant l'expansion de requêtes à travers les synonymes de la base lexicale WordNet. On a utilisé la librairie NLTK de Python pour étendre les requêtes avec des synonymes de chaque mot de la requête.

4.2.1 Pipeline du traitement des requêtes

Chaque requête, subit des traitements similaires aux traitements faits lors de l'indexation des documents. Cette fois-ci, on a fait les traitements manuellement avec le langage python et les librairies NLTK et re. En bref, le pipeline est le suivant: Requête –> Tokenisation –> Part Of Speech tagging (est-ce un verbe/nom..?) –> Supprimer les stopwords –> Wordnet Synset –> Différents traitements.

```
query:  This is a sample query for the IR system.
tokens:  ['This', 'is', 'a', 'sample', 'query', 'for', 'the', 'IR', 's
ystem', '.']
tokens:  [('This', 'DT'), ('is', 'VBZ'), ('a', 'DT'), ('sample', 'J
J'), ('query', 'NN'), ('for', 'IN'), ('the', 'DT'), ('IR', 'NNP'), ('s
ystem', 'NN'), ('.', '.')]
tokens:  [('sample', 'JJ'), ('query', 'NN'), ('ir', 'NNP'), ('system',
'NN'), ('.', '.')]
synsets:  [[], [Synset('question.n.01')], [Synset('iridium.n.01'), Syn
set('inland_revenue.n.01')], [Synset('system.n.01'), Synset('system.n.
02'), Synset('system.n.03'), Synset('system.n.04'), Synset('arrangemen
t.n.03'), Synset('system.n.06'), Synset('system.n.07'), Synset('syste
m.n.08'), Synset('organization.n.05')]]
synonyms:  {'question': 1, 'iridium': 1, 'inland_revenue': 1, 'syste
m': 1, 'arrangement': 1, 'organization': 1}
synonyms:  {'question': 1, 'iridium': 1, 'inland revenue': 1, 'syste
m': 1, 'arrangement': 1, 'organization': 1}
expanded query:  question iridium inland revenue system arrangement or
ganization This is a sample query for the IR system.
```

Figure 2: Exemple d'expansion de requêtes avec les différents prétraitements.

4.2.2 Différents traitements

Ce qu'on entend par différents traitement dans la dernière étape du pipeline de prétraitements, c'est qu'on a extrait, pour chaque mot de chaque requêts:

- Ses synsets (les mots ayant des sens proches)
- Ses synonymes (mots ayant le même sens)
- Ses hypernymes (mots ayant un sens plus large)
- Des combinaisons de ceux-ci

On a fait l'expansion de requêtes sur les mots d'une requête courte (constituée du champ "title"),

sur les mots des requêtes longues (champs "title" et "description") et sur les mots du champ "title" en ajoutant les mots du champ description sans expansion. On a choisi de faire ceci puisque si on fait l'expansion sur les termes du champ "description", on aurait une requête trop longue et probablement va diriger le système dans le mauvais sens.

On a essayé aussi de développer un peu l'expansion de requêtes avec les synonymes (approche Synonymes améliorées dans les tableaux 3 et 4) en choisissant de n'ajouter que les synonymes qui ont le même rôle dans la phrase (part of speech tag). Par exemple, il est plus logique d'ajouter les verbes synonymes de "fly" si le mot a été utilisé comme verbe dans la requête originale plutôt que d'étendre la requête avec le mot "fly" qui signifie mouche et qui va dérailler le système.

5 Évaluations et résultats

Pour évaluer notre système de recherche d'information, on s'est basé sur l'une des mesures les plus utilisées: la moyenne des précisions moyennes (Mean Average Precision MAP) qu'on obtient en utilisant la librairie trec_eval (https://github.com/usnistgov/trec_eval).

```
trec_eval qrels.txt results.txt
```

5.1 Résultats de la première partie

On rappelle que dans la première partie, on a fait l'indexation de 8 manières différentes, et on a fait des requêtes courtes et d'autres longues pour chacune d'elles. On présente les résultats obtenus dans le tableau 1 pour les requêtes courtes (champ "tilte" uniquement) et dans le tableau 2 pour les requêtes longues (champs "title" et "description" dans le fichier de requêtes).

On peut voir que les prétraitements sont efficaces dans tous les différentes configurations. En effet, en moyenne, les prétraitements sont 2.75% meilleurs pour les requêtes courtes et 2.5% meilleurs pour les requêtes longues.

En outre, comme on pouvait le prédire, les requêtes longues sont 20% plus efficaces que les requêtes courtes. C'est parceque les requêtes courtes sont plus ambigûs et trop générales pour que le système de recherche d'informations puisse trouver les bons résultats correspondants.

En comparant les différents schémas de pondérations, on peut voir que Okapi-BM25 est le meilleur entre les 4 modèles testés, que ce soit avec les requêtes courtes ou avec les requêtes longues.

Concernant les courbes rappel-précision, on peut voir dans les figures 3 à 6 que les différents systèmes développés ont donné des résultats similaires, toujours avec BM-25 comme meilleur schéma de pondération puisqu'il présente la plus grande aire sous la courbe.

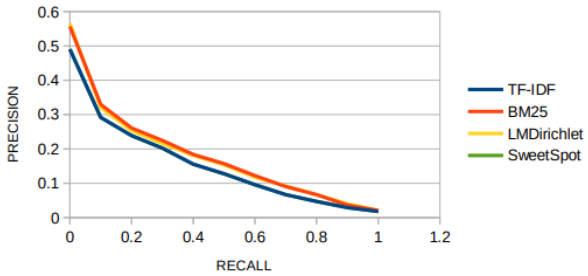

Figure 3: Requêtes courtes, sans prétraitements

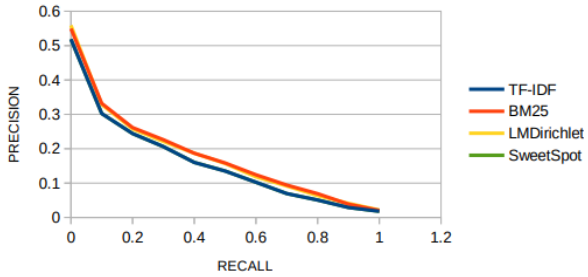

Figure 4: Requêtes courtes, avec prétraitements

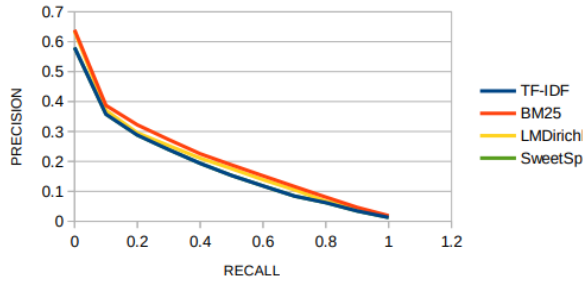

Figure 5: Requêtes longues, sans prétraitements

Pour conclure la première partie, le meilleur système en global était celui construit avec l'indexation avec les prétraitements mentionnés dans le paragraphe (4.12), en utilisant le schéma de pondération Okapi BM-25 avec les requêtes longues.

5.2 Résultats de la deuxième partie

Dans la deuxième partie, on a essayé d'améliorer les requêtes avec la base lexicale WordNet comme mentinoné précedemment. On peut voir dans le tableau 3 que l'expansion de requêtes n'est pas toujours efficace si on la fait sur les mots des requêtes

Schéma de pondération	MAP baseline	MAP baseline avec prétraitement	Amélioration
TF-IDF	14.09 %	14,65 %	4 %
Okapi BM25	16.58 %	16.70 %	1 %
LMDirichlet	16.29 %	16.49 %	2 %
SweetSpot	17.24 %	17.74 %	3 %

Table 1: Comapraison des MAPs avec et sans prétraitements pour les requêtes courtes.

Schéma de pondération	MAP baseline	MAP baseline avec prétraitement	Amélioration
TF-IDF	17.24 %	17.74 %	3 %
Okapi BM25	20.08 %	20.29 %	1 %
LMDirichlet	18.70 %	19.24 %	3 %
SweetSpot	17.24 %	17.74 %	3 %

Table 2: Comapraison des MAPs avec et sans prétraitements pour les requêtes longues.

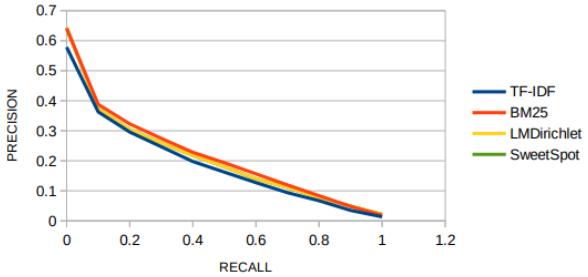

Figure 6: Requêtes longues, avec prétraitements

Termes ajoutés	MAP
Sans expansion de requête	*16.70 %*
Synonymes	17.28 %
Hypernymes	10.36 %
Synsets	16.36 %
Synsets et synonymes	16.16 %
Synsets et hypernymes	14.58 %
Synsets, hypernymes et synonymes	15.02 %
Synonymes améliorés	17.59%

Table 3: Résultats d'expansion des mots du champ "title" des requêtes courtes.

courtes. En effet, seul l'ajout de synonymes par les deux méthodes a prouvé d'être efficace pour améliorer la moyenne des précisions moyennes. Comparé aux requêtes sans expansion avec un MAP de 16.70%, on a obtenu un MAP de 17.28% avec les synonymes normaux et 17.59% avec les synonymes améliorés. On rappelle que dans cette méthode, on a choisi de ne pas ajouter n'importe quels synonymes, mais plutôt ceux qui portent le même part of speech tag. On a aussi limité le nombre de synonymes à 1 seul synonyme par mot de requête pour éviter que le système ne diverge trop du sujet de la requête originale.

Les deux tableaux suivants (4 et 5) nous mènent à des conclusions similaires: les différentes méthodes, à l'exception de l'ajout de synonymes, ont rapporté des scores MAP plus faibles que le système sans expansion de requêtes et seules les expansions de requêtes avec les synonymes sont efficaces, que ce soit en rajoutant tous les synonymes possibles ou en choisissant ceux ayant le même part of speech tag et en les limitant à un seul synonyme par mot (synonymes améliorés).

On peut expliquer le fait que le rajout d'hypernymes n'a pas apporté d'améliorations par le fait que les requêtes sont déjà un peu générales et pour améliorer le système RI, on veut plutôt avoir des requêtes plus spécifiques pour ne pas dérailler le système.

Les synsets n'ont pas rapporté d'améliorations non plus car ils peuvent contenir des mots ayant des sens un peu différents des mots de la requête originale.

Enfin, le meilleur système de recherche d'information obtenu est celui avec le rajout de synonymes "améliorés" apportant une amélioration relative jusqu'à 2% et avec lequel on peut atteindre une 20.56% de moyenne de précisions moyennes dans le cas d'expansion de requêtes longues en rajoutant un synonyme par mot du champ "title" seulement et en utilisant les prétraitements lors de l'indexation des documents avec le schéma de pondération BM-25.

Termes ajoutés	MAP
Sans expansion de requête	*20.15 %*
Synonymes	20.56 %
Hypernymes	17.96 %
Synsets	19.75 %
Synsets et synonymes	19.39 %
Synsets et hypernymes	18.51%
Synsets, hypernymes et synonymes	18.78 %
Synonymes améliorés	20.44%

Table 4: Résultats d'expansion des mots du champ "title" seulement des requêtes longues.

Termes ajoutés	MAP
Sans expansion de requête	*20.15 %*
Synonymes	20.38 %
Hypernymes	15.04 %
Synsets	19.07 %
Synsets et synonymes	17.98 %
Synsets et hypernymes	16.75 %
Synsets, hypernymes et synonymes	16.32 %
Synonymes améliorés	20.23 %

Table 5: Résultats d'expansion des mots du champ "title" et "description" des requêtes longues.

Conclusion et perspectives

Dans ce projet, on a beaucoup appris. On s'est familiarisé avec Solr qui est un système de recherche d'information utilisé dans beaucoup d'entreprises aujourd'hui. Après avoir testé une multitude de configurations, on peut conclure que les prétraitements (tokenisation, suppression des stopwords..) sont toujours efficaces pour la recherche d'information. On a trouvé aussi que BM-25 est le meilleur schéma de pondération parmi les 4 testés. Enfin, on a fait une légère amélioration du système RI avec l'ajout des synonymes de deux méthodes. Concernant les perspectives, notre système peut sûrement être amélioré et on peut proposer de travailler plus sur la méthode avec laquelle on fait le choix des synonymes. On peut choisir le synonyme qui occure le plus fréquemment avec les autres mots de la requête. On peut aussi essayer les plongements de mots ou les retours de pertinence et voir ce que cela pourrait donner comme résultat en termes de MAP.

References

Vimala Balakrishnan and Ethel Lloyd-Yemoh. 2014. Stemming and lemmatization: a comparison of retrieval performances.

Debasis Ganguly, Dwaipayan Roy, Mandar Mitra, and Gareth JF Jones. 2015. Word embedding based generalized language model for information retrieval. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, pages 795–798.

Karen Sparck Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*.

Julie Beth Lovins. 1968. Development of a stemming algorithm. *Mech. Transl. Comput. Linguistics*, 11(1-2):22–31.

Hans Peter Luhn. 1957. A statistical approach to mechanized encoding and searching of literary information. *IBM Journal of research and development*, 1(4):309–317.

Viviane Moreira Orengo, Luciana S Buriol, and Alexandre Ramos Coelho. 2006. A study on the use of stemming for monolingual ad-hoc portuguese information retrieval. In *Workshop of the Cross-Language Evaluation Forum for European Languages*, pages 91–98. Springer.

Dipasree Pal, Mandar Mitra, and Kalyankumar Datta. 2014. Improving query expansion using wordnet. *Journal of the Association for Information Science and Technology*, 65(12):2469–2478.

Stephen E Robertson, Steve Walker, Susan Jones, Micheline M Hancock-Beaulieu, Mike Gatford, et al. 1995. Okapi at trec-3. *Nist Special Publication Sp*, 109:109.

Jonathan J Webster and Chunyu Kit. 1992. Tokenization as the initial phase in nlp. In *COLING 1992 Volume 4: The 14th International Conference on Computational Linguistics*.

Chengxiang Zhai and John Lafferty. 2004. A study of smoothing methods for language models applied to information retrieval. *ACM Transactions on Information Systems (TOIS)*, 22(2):179–214.