Recherche d'information avec PyLucene

Francois Huppé-Marcoux

Mots clés : pyLucene, recherche d'information, stemming, BM25

Résumé

Ce document présente différentes méthodes de recherche d'information (RI). Le but de ce travail est d'explorer les différents pré-traitements et méthodes d'indexation sur un jeu de données. Les mesures de similarité présentées sont : TF-IDF, BM25 et LM Jelinek-Mercer. Il sera présenté des métriques d'évaluation avec une analyse des problèmes rencontrés lors de la recherche d'information et des explications des résultats obtenus.

1 Introduction

La recherche d'information existe sous plusieurs formes depuis que l'on classe des éléments dans une collection. Une recherche dans le dictionnaire est un exemple de classement de mots par ordre alphabétique pour retrouver plus rapidement l'information. Aujourd'hui, les systèmes de recherche d'information sont présents dans plusieurs domaines, que ce soit une recherche dans un texte ou une recherche web parmi des millions de documents disponibles.

2 Problèmatiques

Les problèmes que rencontrent les systèmes de recherche d'information sont les problèmes polysémiques, de synonymes et des écarts lexicaux entre la requête et les documents.

Un mot peut avoir plusieurs sens et s'écrire de la même façon. La sémantique du mot dépend du contexte de la phrase. Malheureusement, il n'existe pas d'algorithme qui soit capable de désambiguïser tous les mots d'un texte.

Pour faciliter le traitement d'un corpus comprenant plusieurs milliers de documents, on utilise la tokenisation des mots. La tokenisation est le processus de prendre un texte qui contient plusieurs phrases à une suite de mots formatés et prétraités. Une fois la liste de mots créée, on peut appliquer différents traitements pour simplifier les mots dans le but de restreindre le vocabulaire. Le but de l'opération est de plus facilement associer des mots de la requête avec les mots des documents. Le stemming est un algorithme qui enlève arbitrairement la fin d'un mot. L'algorithme de stemming, de par sa nature arbitraire, a tendance à couper un trop gros suffixe d'un mot ou à ne pas couper assez de mot. Dans le cas d'une coupe excessive, des mots qui sont différents seront mis sous la même base et cela causera des problèmes dans la recherche. Des mots qui ont des sens différents seront identiques. Dans le cas où le mot n'est pas assez coupé, le dictionnaire de mots sera trop grand et des mots similaires ne seront pas associés ensemble.

Le lemmatiseur WordNet utilise des types de base pour identifier des mots. Des exemples de types sont : Location, Physical object, Time, etc. Ces lemmes sont assignés aux mots et un mot peut avoir plusieurs lemmes. Par exemple, le mot 'Homard' est considéré à la fois comme un animal et comme de la nourriture. Ensuite, il utilise ces étiquettes pour traiter les mots. WordNet gère mieux les synonymes qu'un simple stemmer.

3 État de l'art

Pour la recherche d'information avec Lucene, il existe plusieurs schémas de pondération. Un schéma de pondération permet de mesurer la distance sémantique entre deux documents. Les méthodes classiques de pondération qui ont fait leurs preuves sont le TF-IDF, LM et BM25.

En plus des méthodes classiques, il existe d'autres façons de pondérer la recherche. Le Deep Contextual Term Weighting (DeepCT)

est un schéma de pondération qui utilise l'apprentissage profond par transformeur pour comprendre le contexte de l'apparition des termes dans les documents. Le DeepCT améliore la mesure classique du TF-IDF en ajoutant le contexte des mots à la mesure classique. Prendre le contexte en compte a l'avantage de diminuer les problèmes polysémiques et de termes ambigus. De plus, il donne de meilleurs résultats pour des mots rares. D'autre part, le schéma de pondération Global Analysis of Relevance (GAR) considère la distribution des mots et la pertinence à travers tous les documents du corpus. L'algorithme normalise la pondération pour que les longs documents n'aient pas un avantage sur les courts documents, contrairement à la mesure du TF-IDF. Il est possible d'ajouter une rétroaction des utilisateurs au schéma de pondération.

Des librairies telles que Lucene, OpenSearch et Solr utilisent l'indexation pour enregistrer des documents. La méthode d'indexation utilisée par Lucene est le Hierarchical Navigable Small-World Network (HNSW). Cette méthode se différencie des bases de vecteurs, car elle ne nécessite pas de base de données vectorielles. Le coût de calcul pour une base de données non vectorielle est moins important qu'avec Lucene, qui utilise la recherche de documents par sac de mots. Une recherche démontre que l'utilisation de l'intelligence artificielle pour l'indexation ne donne pas nécessairement de meilleurs résultats et que ses modèles à base d'API sont beaucoup plus coûteux en termes de calcul. En plus des coûts pour entraîner les poids des modèles.

L'algorithme HNSW est une structure incrémentale basée sur un graphe pour la recherche K-ANNS (K-Approximate Nearest Neighbors Search) qui offre une meilleure échelle de complexité logarithmique. Il a comme caractéristique d'offrir une méthode de sélection explicite pour le point d'entrée du graphe, la catégorisation des liens par différentes échelles et l'adoption d'une heuristique avancée pour la sélection des voisins. Cette méthode surpasse significativement les approches open-source état de l'art existantes, qui sont généralement limitées aux espaces vectoriels. Les techniques utilisées pour pallier aux problématiques de la RI sont les systèmes de réseau neuronal convolutionnel (CNN), les réseaux de neurones récurrents (RNN) et des techniques de transfert de connaissance. Ces techniques améliorent la recherche d'information en comprenant mieux la sémantique de la recherche. De plus, les transformeurs ont été utilisés, avec l'avantage de l'attention, pour augmenter les résultats d'une requête. L'utilisation des modèles BERT et GPT-2 a démontré une capacité accrue pour la RI. Certains modèles multi-modaux prennent plusieurs types de requêtes, comme des documents textuels ou des images, pour effectuer des recherches d'information, ce qui augmente les informations contenues dans une recherche. L'augmentation de la requête est un aspect important de la recherche, car les résultats de la recherche dépendent de la longueur et de la sémantique de la requête. Une recherche avec peu de mots donne des résultats médiocres en comparaison d'une requête avec beaucoup de mots. L'augmentation de la recherche par différents moyens donne de meilleurs résultats, car elle spécifie la recherche dans le cas d'une requête avec peu d'information.

Augmentation de la recherche et recherche courte.

Les systèmes de recherche d’information ont des difficultés avec des requêtes courtes. Des requêtes avec peu de mots donnent peu d’information sur le sujet de recherche. Avec une collection de 240 000 documents et un jugement de 300 documents pertinents, pour une recherche, les chances que les documents pertinents se trouvent dans le résultat de la recherche sont minces. Il y a beaucoup de documents qui ont des sujets

similaires. Alors, en donnant un sujet de recherche large, qui contient peu de mots, les chances que des mauvais documents soient proposés augmentent. Tandis qu'une longue requête spécifie mieux le sujet de la requête.

Pour pallier aux requêtes courtes, les systèmes de recherche d'information moderne utilisent l'expansion de requête. Il existe plusieurs stratégies d'expansion. Entre autres, la rétroaction pseudo-pertinence (Pseudo-relevance feedback) suppose que les premiers documents qui sont retournés sont bons et ils sont utilisés pour faire la recherche du reste des documents. Une autre approche est d'utiliser des vecteurs sémantiques. C'est une approche probabiliste des termes apparaissant dans les documents qui encode les similarités entre les documents. Les documents sont encodés dans un vecteur à plusieurs dimensions. Puis, on calcule la ressemblance entre les vecteurs pour trouver des documents semblables. La mesure de similarité cosinus est souvent utilisée pour la recherche vectorielle. Une simple expansion de recherche avec les meilleurs documents n'est pas prouvée d'augmenter constamment les résultats de la recherche. Tandis qu'une augmentation en allant chercher des mots importants des meilleurs documents donne de meilleurs résultats en augmentant systématiquement les résultats obtenus.

Plus récemment, l'expansion de la recherche utilise les grands modèles de langue (GML) pour améliorer la recherche. L'avantage d'utiliser un GML est qu'on ne se base pas sur le top K documents pour améliorer la recherche. Pour une requête courte, les GML n'ont pas de limitation quant aux top K retrouvés. Il existe deux types d'augmentation de recherche avec des GML, soit le zero-essai (zero-shot), quelque essai (few-shot) ou la chaîne de pensée (chain of thought).

La nature de l'expansion de recherche dépend des domaines du corpus. Un corpus comme Wikipédia aura besoin d'une expansion de recherche spécifique aux entités nommées. Les principales méthodes d'augmentation de la requête sont d'utiliser des modèles globaux et d'incorporer les concepts pertinents à la requête, de faire des associations sémantiques à travers les documents et de les relier à la requête, ou d'intégrer une rétroaction de l'utilisateur dans la requête (algorithme Rocchio de rétroaction de pertinence).

4 Données et ressources utilisées

Les données utilisées proviennent du corpus TREC AP 88-90. Il s'agit d'une collection de documents issue de la Conférence sur la Recherche d'Informations Textuelles (Text Retrieval Conference), comprenant des articles de l'Associated Press couvrant la période de 1988 à 1990. Ce jeu de données fournit un ensemble riche et diversifié de textes en langage naturel non structurés, ce qui en fait un outil idéal pour le développement, le test et l'évaluation comparative des systèmes de RI. Le corpus sert de standard de référence pour évaluer l'efficacité des différentes approches dans le domaine de la RI.

Le corpus est composé de 242 918 documents, chacun ayant des informations sur le numéro du document, le titre, la provenance et le texte.

Nombre total de mots	147'214'366
Nombre de mots unique	1'218'173
Mots par documents	606
Mots fréquents	'the' : 5'949'718, of: 3'018'995, 'to': 2'832'219, 'a': 2'480'928, 'and' : 2'450'984

Les sujets de requêtes sont inclus avec les corpus. Il y a en tout 150 sujets avec lesquels nous pouvons formuler des requêtes. Finalement, on a 150 jugements associés aux sujets de requête. Ces deux ensembles servent lors de l'évaluation.

5 Méthodologie

Pour effectuer les tests sur l'ensemble des données, le logiciel PyLucene version 9.7.0 est utilisé. PyLucene est une interface permettant d'utiliser Lucene en Python. Pour ce faire, on démarre une machine virtuelle en Java, puis on lance des exécutions de Lucene dans la machine virtuelle Java. Pour utiliser PyLucene, il faut suivre les instructions disponibles sur le site officiel en prenant le temps d'installer Java et son compilateur. Cette version de PyLucene utilise l'algorithme HNSW pour indexer les fichiers. PyLucene est également utilisé pour la tokenisation avec la bibliothèque EnglishAnalyzer. Cette bibliothèque tokenize les mots selon la méthode de stemming avec l'algorithme de Porter. Dans le test de base, on utilise une tokenisation de base en enlevant seulement les stopwords. Dans le deuxiemes tests, on utilise la lemmatisation pour tokeniser. Ensuite, on indexe les documents selon l'algorithme HNSW. Une fois que l'indexation est terminée, on lance des recherches dans l'index. L'engin de lucene se charge de trouver les documents pertinents.

On teste la recherche avec les sujets fournis dans les données. Au total, 150 requêtes sont lancées au moteur de recherche, celui-ci trouvant un maximum de 1000 documents. On crée un fichier 'trec_run' contenant les résultats des 1000 résultats pour les 150 recherches.

On observer la différence entre les requêtes courtes et longues. Pour formuler les requêtes courtes, on prend seulement les titres, tandis que pour les requêtes longues, on prend le titre et la description des sujets fournis dans les données.

Enfin, on teste les trois schémas de pondération : TF-IDF, BM25 et LMJelinekMercer. Ces schémas sont responsables du calcul de la similarité entre la requête et l'indexation du corpus.

6 Évaluations et résultats

Pour les tests, nous allons observer la métrique du Mean Average Precision (MAP). Cette métrique est couramment utilisée pour l'évaluation de systèmes de RI. Cette métrique révèle la pertinence des documents trouvés et leur rang. Une recherche qui trouve tous les documents pertinents, mais qui les classe loin dans les rangs n'aura pas un bon résultat de MAP. Alors qu'une recherche avec quelques bons documents, mais qui apparaissent dans les premiers résultats, aura un résultat semblable.

Voiciles résultat obtenu lors des tests.

Types de Requêtes	Schéma de pondération	MAP
Courtes	Tf*idf	0.1130
Courtes	BM25	**0.1569**
Courtes	LM	0.1389
Longues	Tf*idf	0.1273
Longues	BM25	**0.1668**
Longues	LM	0.1498

Résultats de la MAP@1000 pour 150 requêtes avec un tokenisation de base.

Types de Requêtes	Schéma de pondération	MAP
Courtes	Tf*idf	0.1353
Courtes	BM25	**0.1813**
Courtes	LM	0.1645
Longues	Tf*idf	0.1490
Longues	BM25	**0.1906**
Longues	LM	0.1750

Résultats de la MAP@1000 pour 150 requêtes avec un tokenisation stemme.

Selon les tests exécutés, les requêtes longues sont en moyenne 1,14 % meilleures avec la mesure du MAP. Certains tests avec des requêtes courtes donnent de meilleurs résultats à cause du schéma de pondération. Le meilleur schéma de pondération est le BM25 comparativement aux autres schémas avec le même prétraitement. En moyenne, le prétraitement augmente la mesure du MAP de 2,38 %. Le meilleur résultat est avec un prétraitement stemmé et le schéma de pondération BM25.

Si l'on examine plus en détail le test long BM25, le nombre de documents retournés pour les 150 requêtes est de 150 000. À chaque test, Lucene retourne 1000 résultats. Le nombre de documents pertinents selon les jugements est de 25 549. Ensuite, le nombre de documents que Lucene a réussi à trouver est de 12 365. La précision des 5 premiers documents est de 39 %, à 10 documents elle est de 38 % et elle continue à descendre tranquillement pour atteindre une précision de 8 % à 1000 documents.

Une partie de ces résultats s'explique car il y a beaucoup de documents et qu'on utilise un algorithme de stemming. Le nombre de mots uniques est d'un peu plus de 1,2 million. Lucene utilise l'algorithme de Porter pour faire son stemming. Cet algorithme fait un stemming doux, donc plusieurs mots qui ont la même sémantique gardent leur forme d'origine.

En regardant les différents types de tokenisation.

Base	Stemme
and	Ø
Banking ou banks	bank
agreed	agre
milwaukee	milwauke
executive	execut

Pour certains mots, la tokenisation par stemming fonctionne bien. Elle enlève les mots comme "and" qui donnent peu d'information et qui sont très courants dans les documents. Elle simplifie les mots qui peuvent être des verbes ou au pluriel à une forme plus simple. De plus, elle simplifie les temps des verbes. Par contre, l'algorithme ici a enlevé un 'e' au nom de la ville 'Milwaukee'. Ce qui ne change pas grand-chose puisque la requête est tokenisée de la même façon. Pour le dernier mot 'executive', qui est réduit à 'execut', cela peut causer des problèmes. Dans un cas, on pense à un titre d'employé dans une compagnie. Dans un autre cas, si on prend le mot 'execut' qui sera tokenisé de la même manière, on pense au verbe s'exécuter. Cela peut causer des collisions dans la recherche.

En observant la longueur des textes trouvés par l'engin de recherche avec la mesure de similarité BM25, on observe que l'engin n'a pas tendance à favoriser des longueurs de contenu en particulier.

En analysant les erreurs, pour la première requête, le sujet et la description donnent ce

texte : "Antitrust Cases Pending Document discusses a pending antitrust case". La reqête est tokenisé et la mesure du BM25 regardes les termes suivants : "antitrust pend case". Selon le calcul du BM25, le mot "antitrust" pèse plus dans la balance, car c'est un mot rare qui apparaît 5 fois dans le document trouvé. En analysant les documents qui sont pertinents, on remarque que le mot "discuss" est important à la recherche, car les documents pertinents sont des documents qui expliquent des cas. Pour faire une meilleur recherche dans les dossier, il faudrait que la sémantique du mot ‘discuss’ soit gardé dans la requête, alors que le mot n’est pas pris en compte dans la mesure du BM25. En somme, les documents qui sont classés en premier contiennent des mots de la requête. Ils sont évalués bons, mais ils n'ont pas la subtilité qu'ils doivent avoir pour discuter d'un cas. Le problème ici est que le mot "discuss" n'a pas été jugé important dans la mesure du BM25.

7 Conclusion

Avec une tokenisation de stemming et aucune expansion de requête, nous arrivons tout de même à trouver des documents pertinents dans la collection de données. La recherche des requêtes se fait rapidement grâce à l'indexation de Lucene.

Le tokeniseur WordNet et sa capacité à lemmatiser les mots donnent de meilleurs résultats, car contrairement au stemmer, les mots sont désambiguïsés en gardant la sémantique du mot.

Pour augmenter les résultats de la recherche, une étude a démontré qu'une combinaison du Pseudo-Relevance Feedback (PRF) et d'une chaîne de pensée obtient de meilleurs résultats.

Les résultats du MAP sont affectés par l'ordre dans lequel les documents sont affichés. Pour une future amélioration, il serait bon de créer un module particulier pour reclasser les documents trouvés.

Références

ALMasri, Mohannad, Catherine Berrut, and Jean-Pierre Chevallet. "A comparison of deep learning based query expansion with pseudo-relevance feedback and mutual information." Advances in Information Retrieval: 38th European Conference on IR Research, ECIR 2016, Padua, Italy, March 20–23, 2016. Proceedings 38. Springer International Publishing, 2016.

Apache Software Foundation. (2023, juillet). Welcome to PyLucene. lucene.apache. https://lucene.apache.org/pylucene/

Catherine McCabe. (1999, 12). TRECEVAL is a program to evaluate TREC results using the standard, NIST evaluation procedures. faculty.washington. https://faculty.washington.edu/levow/courses/ling573_SPR2011/hw/trec_eval_desc.htm

Hambarde, Kailash A., and Hugo Proenca. "Information Retrieval: Recent Advances and Beyond." arXiv preprint arXiv:2301.08801 (2023).

Introduction to Information Retrieval. Christopher D. Manning, Prabhakar Raghavan and Hinrich Schutze, Cambridge University Press, 2008 (online version available)

Jagerman, Rolf, et al. "Query Expansion by Prompting Large Language Models." arXiv preprint arXiv:2305.03653 (2023).

Lin, Jimmy, et al. "Vector search with OpenAI embeddings: Lucene is all you need." arXiv preprint arXiv:2308.14963 (2023).

Malkov, Yu A., and Dmitry A. Yashunin. "Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs." IEEE transactions on pattern analysis and machine intelligence 42.4 (2018): 824-836.

Moral, Cristian, et al. "A survey of stemming algorithms in information retrieval." Information Research: An International Electronic Journal 19.1 (2014): n1.

Utt, Jason, and Sebastian Padó. "Ontology-based distinction between polysemy and homonymy." Proceedings of the Ninth International Conference on Computational Semantics (IWCS 2011). 2011.