



POLYTECH PARIS-SACLAY
ET5 INFO

RAPPORT PROJET NLP

SEARCH ENGINE

FORT Adrien – RAHOUAL Wassim |
<https://github.com/Adrien2209/NLP-Project-TFIDF>



SOMMAIRE

DESCRIPTION DU PROJET - 3

En introduction, nous expliquons brièvement le projet.

TF-IDF - 4

Dans cette partie nous allons expliquer comment fonctionne l'approche TF-IDF dans le cadre d'un moteur de recherche

NOTRE IMPLEMENTATION ET SES LIMITES - 5

Nous expliquons ici notre implémentation, tous les choix de traitement de données ainsi que le calcul du score et nous discutons des limites de notre implémentation ainsi que ses améliorations possibles

ANALYSE DES RESULTATS - 7

Enfin, nous analysons les résultats obtenus en accords avec notre implémentation et ses limites

CONCLUSION ET DIFFICULTES - 8

BIBLIOGRAPHIE - 9

$$TF_{w,d} = \frac{\# \text{ occurrences of } w \text{ in doc } d}{\# \text{ of words in doc } d}$$

$$idf_w = \log\left(\frac{\# \text{ of doc in the collection}}{\# \text{ of docs in which } w \text{ occurred}}\right)$$


$$TF.idf_{w,d} = TF_{w,d} \cdot idf_w$$



DESCRIPTION DU PROJET

Pour ce projet, nous devions créer un moteur de recherche en utilisant le jeu de données CISI. Nous devions implémenter une approche statistique basé sur TF-IDF, et analyser les résultats obtenus par rapport à une base de résultats.

Afin de parvenir à cet objectif, il y avait plusieurs étapes cruciales. Dans un premier temps, nous devions parser les différents jeu de données mis à notre disposition pour pouvoir les exploiter proprement. Nous avons fait le choix d'utiliser les dictionnaires afin de rapidement accéder aux différents éléments à l'aide de clés. Nous devions ensuite faire la relation entre les requêtes, les documents du jeu de données et la base de résultats afin d'analyser les performances de notre modèle. Enfin, l'amélioration du modèle passait avant tout par un traitement des données en entrées, que nous détaillerons dans la partie dédiée à notre implémentation. Nous avons aussi essayé plusieurs implémentations différentes dans la recherche d'une précision accrue et nous détaillons les raisons des changements de performances.



TF-IDF

Avantages TF-IDF

- Implémentation simple
- Facile à calculer et peu gourmand en performances
- Bon point de départ pour d'autres calculs (vectorization et similarité cosin)

Term frequency-inverse document frequency en anglais ou TF-IDF est une approche statistique de l'importance d'un mot dans un document donné d'une bibliothèque de document. Un des plus grands domaines d'application à ce jour est l'analyse de texte pour extraire automatiquement des informations d'un document à l'aide des mots qui le constitue, par exemple si le mot qualité apparaît de nombreuses fois dans des demandes de service après-vente d'un produit, cela suggère que les clients sont mécontents de la qualité des produits. Un autre domaine où cette technique est très utilisée est celui qui nous intéresse, le Natural Language Processing ou NLP.

TF-IDF a été inventé pour retrouver des informations dans un corpus de documents à partir d'une requête.

Inconvénients TF-IDF

- Aucune analyse du contexte
- TF-IDF ignore l'ordre de mots et donc « le président français » par exemple ne sera pas forcément rattaché au président de la France.

$$TF_{w,d} = \frac{\# \text{ occurrences of } w \text{ in doc } d}{\# \text{ of words in doc } d}$$

$$idf_w = \log\left(\frac{\# \text{ of doc in the collection}}{\# \text{ of docs in which } w \text{ occurred}}\right)$$

$$TF \cdot idf_{w,d} = TF_{w,d} \cdot idf_w$$

TF va calculer le nombre d'occurrence d'un mot donné dans un document, divisé par le nombre de mots dans le document. C'est donc la fréquence du mot dans un document.

idf calcul la rareté d'un mot dans le corpus de documents, en divisant le nombre de documents dans la collection par le nombre de documents où le mot donné apparaît. De ce fait, les mots commun ont un score proche de 0.

On multiplie ensuite ces deux valeurs et plus le score est haut, plus le mot est important dans un document donné.



NOTRE IMPLEMENTATION ET SES LIMITES

Notre implémentation repose sur une volonté de percevoir le changement en fonction des options de traitement des données, c'est-à-dire que nous avons essayé au possible de faire quelque chose de modulable afin de comparer l'utilité ou non d'une fonctionnalité. Nous allons dans un premier temps voir le parsing des documents et pourquoi l'utilisation assez massive de dictionnaire nous est apparu pertinente pour stocker un maximum d'informations sur les documents et les requêtes. Ensuite nous verrons les options de pré-traitement des données que nous avons implémenté, les justifier et mentionner les limites de celles-ci. Enfin, nous verrons les limites de notre implémentation, les améliorations que nous aurions aimé effectuer.

La première étape après avoir parsé les différents jeux de données était le traitement des différents mots. Notre parser extrait dans un premier temps le texte complet et nous effectuons ensuite une tokenisation, c'est-à-dire que chaque mot va devenir une clé dans un dictionnaire lié à un document, et son nombre d'apparition la valeur liée à cette clé. De cette manière il est facile d'itérer seulement sur les mots et si besoin, de récupérer leurs fréquence d'apparition afin de leur attribuer un poids différent par exemple. Enfin, nous récupérerons un maximum d'information sur le document comme son auteur, sa taille, son ID.

Une fois les mots du documents sous forme de token, nous allons supprimer toutes les majuscules et tous les signes de ponctuations à l'exception des apostrophes, qui seront traitées par le stemmer. Nous allons ensuite traiter ces données en supprimant les « stopwords » dans un premier temps. Ce sont des mots qui n'apportent aucune information comme « the », « is » ou « what » par exemple. Même si ces mots, via l'approche TF-IDF auront eu un poids très faible, nous avons remarqué une amélioration après les avoir retiré. Il y a néanmoins une limite, si une question devait ressembler à « What is the definition of other », le seul mot restant serait « definition », mais étant donné que l'approche TF-IDF ne tiens pas compte de l'ordre des mots, même sans ces stopwords nous aurions eu des résultats peu satisfaisant pour une telle requête.

Enfin, nous utilisons du stemming, une technique de normalisation de texte qui va transformer les mots en leur radical en supprimant les préfix et suffix. Une approche plus complète aurait été d'utiliser une autre technique dite de lemmatization en amont qui réduit un mot à un mot racine synonyme. Cependant, la lemmatization n'est pas très efficace mot par mot et l'est beaucoup plus phrase par phrase avec une analyse du contexte. C'est pourquoi nous ne l'avons pas implémenté pour nous contenter du stemming. Il y a d'autres fonctions qui serait utiles comme la conversion des nombres car notre implémentation identifierai « 10 » et « dix » comme deux token différents. Enfin, l'ajout de Named Entity Recognition permettrait d'allouer un poids différent à un nom propre par exemple.

Enfin, nous calculons le score TF-IDF de chaque document pour la requête et selon les paramètres nous sortons les N documents avec le plus haut score. Comme mentionné plus haut, nous avons plusieurs options possibles pour comparer les performances. La première sans répétition de mots, c'est-à-dire que si la requête contient 3 fois le mot « system », nous n'allons le compter qu'une fois, à l'inverse, avec répétition nous allons le compter le nombre de fois qu'il apparait pour lui attribuer plus de poids. Nous pouvons aussi choisir avec ou sans stemming. Enfin, pour avoir des résultats de précision cohérents avec la base de résultats, nous pouvons choisir les N premiers TF-IDF les plus grands, et aligner N avec le nombre de documents attendus par le fichier CISI.REL. De ce fait, nous n'implémentons comme métrique uniquement la précision étant donnée que le rappel est toujours égal à la précision dans notre cas. En effet, nous classons les documents par rapport à leur score TF-IDF mais de ce fait, aucun document n'est positif ou négatif, ils sont justes plus ou moins pertinents. Nous justifions cette approche en prenant comme exemple un moteur de recherche, les premiers onglets contiennent les meilleurs pages et les plus lointains celles qui sont moins susceptible de nous intéresser.

Une des limites de notre implémentation s'explique clairement pour la requête n°6. D'après la base de résultats, cette requête n'attend qu'un seul document, le document 400 qui est assez long. Notre modèle nous retourne en premier le document 1357 avec un score tout de même assez faible. Celui-ci beaucoup moins long contient énormément de fois le mot « communication », présent dans la requête, sur un document plus court. De ce fait, il apparaît aux yeux de notre implémentation comme le document le plus pertinent. Ce résultat traduit une limite dans le calcul du TF-IDF qui va se retrouver biaisé sur des documents très courts par rapport à des documents très longs. Il traduit aussi une autre limite qui ne tient pas compte du contexte. Dans la requête n°6 à savoir : « What possibilities are there for verbal communication between computers and humans, that is, communication via the spoken word? » Les mots primordiaux sont "communication", "computers", "humans" et "spoken". Cependant, comme nous n'effectuons pas d'analyse contextuelle, il nous est impossible d'affecter des poids différents aux mots primordiaux.

Cette limite aurait pu être résolue en partie par un calcul de similarité cosinus, qui s'attarderait plus sur la similarité entre les documents, au-delà d'un simple score. Une autre façon de lever cette limite aurait été d'expérimenter un modèle d'apprentissage supervisé avec un jeu de requête et de réponse attendu plus conséquent. Ce sont des aspects que nous avons en partie explorés mais que nous n'avons pas su implémenter.

ANALYSE DES RESULTATS

Ci-dessous les résultats de notre implémentation.

Pour calculer la précision de l'ensemble de l'implémentation, nous faisons une moyenne des précisions pour chaque requête.

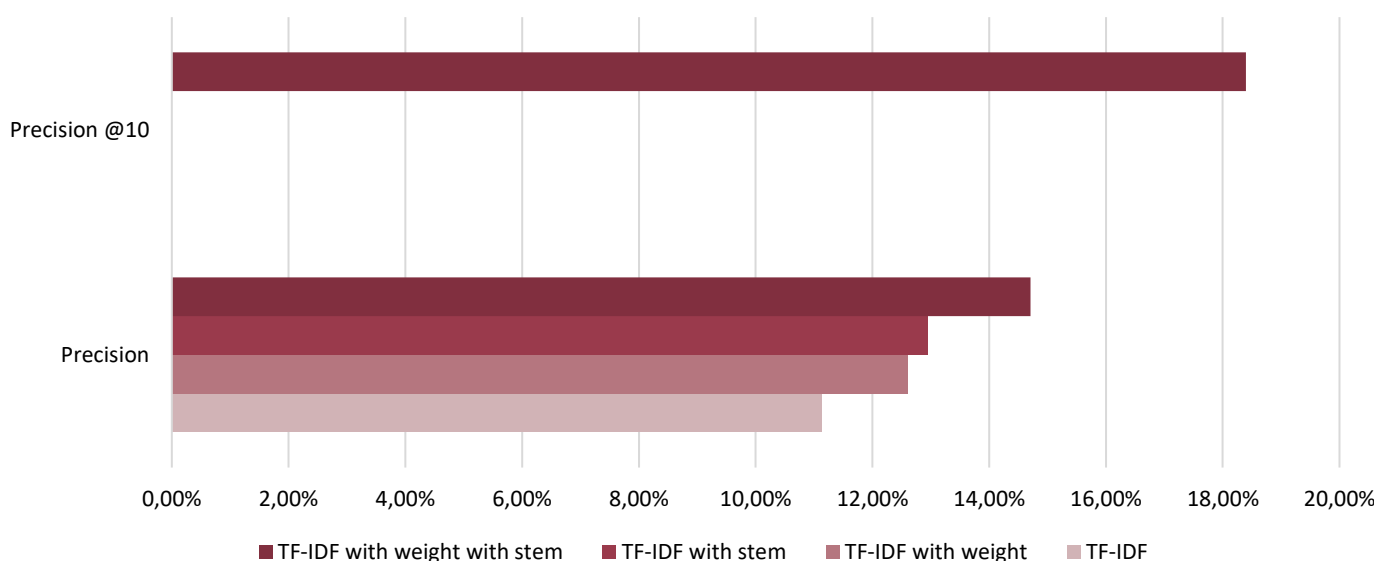
Quand nous calculons la précision pour une requête, nous prenons les N premiers résultats où N correspond au nombre de résultats attendus par la requête selon le document CISI.REL. Par exemple pour la requête 1, 46 documents sont attendus alors nous prenons les 46 premiers résultats.

Dans le document CISI.REL, certaines requêtes n'attendent pas de résultats comme la requête 36, absente du fichier. Nous ne prenons dans ce cas pas en compte la précision pour ce cas de figure car il nous est impossible de la calculer.


Enfin, pour la Precision@10, nous prenons les 10 premiers documents et nous les comparons seulement aux documents des requêtes qui attendent au moins 10 documents. Autrement, les requêtes comme la n°6 n'attendant qu'un seul document viennent diminuer la moyenne.

	TF-IDF	Weight	Stem	Weight & Stem
Precision	11,14%	12,60%	12,94%	14,71%
Precision@10				18,40%

CROISSANCE PAR SECTEUR



Ces résultats nous montrent l'importance du pré-traitement des données, qui pour chaque ajout va augmenter la précision. Cependant, nous imaginons un soft cap à 30% de précision après ajout de lemmatisation et utilisation de similarité cosinus, ou le modèle TF-IDF seul ne suffirait pas, et où l'analyse contextuelle viendrait faire toute la différence.



CONCLUSION ET DIFFICULTES

Ce projet a su piquer notre curiosité et nous comptons le poursuivre en récupérant un jeu de données plus conséquent et en explorant la voie de l'apprentissage supervisé via la librairie scikit-learn ou encore du deep learning via keras et tensorflow pour façonner un modèle capable d'approcher les 100% de précision sur les données fournies avec ce projet. Nous sommes satisfait de nos résultats car même si 18% de précision semble peu, nous avons vu ce pourcentage évoluer positivement au fil des implémentations ce qui nous a permis de comprendre les enjeux liées aux moteurs de recherche. La plus grosse difficulté à été de garder un cadre car au début nous sommes partis un peu dans tous les sens, explorant les différentes possibilités qui s'offraient à nous, essayant d'implémenter le plus de traitement possible, sans remplir la condition principale à savoir un modèle TF-IDF fonctionnel et des mesures. Nous voulions dès la première semaine partir sur un modèle d'apprentissage sans connaître les différentes librairies ni le fonctionnement d'une telle approche, assez loin du modèle TF-IDF de la consigne. Nous avons donc du recadrer notre travail pour fournir ce travail et nous sommes persuadés que si nous avions fonctionné étapes par étapes, nous aurions pu fournir un projet plus complet, en implémentant aussi le modèle BM-25 afin de comparer les résultats, et d'autres modèles encore.



BIBLIOGRAPHIE

<https://www.capitalone.com/tech/machine-learning/understanding-tf-idf/>

<https://medium.com/dair-ai/deep-learning-for-nlp-an-overview-of-recent-trends-d0d8f40a776d>

<https://www.datasciencelearner.com/sklearn-cosine-similarity-implementation/>

<https://m16marketing.com/digital-marketing-blog/what-is-tf-idf-and-how-does-it-apply-to-search-engine-optimization-seo/>