


Recommandation de documents pour moteur de recherche Coveo

Philippe Blais
Philippe Blouin-Leclerc
William Bourget
Stéphane Caron
Samuel Lévesque


21 décembre 2018

Résumé

Faire ressortir les points saillants de l'article en un seul paragraphe question de titiller le lecteur.

L'ensemble du code et des documents qui ont servi à la résolution de cette problématique et à l'écriture de l'article se trouvent dans le répertoire  du projet.

1 Présentation du problème et état de l'art

Lors de ce projet proposé par Coveo , nous devions utiliser un historique de requêtes faites par des utilisateurs afin de développer un modèle de recommandation de document. Le but du modèle est de proposer une série de 5 documents d'intérêt en fonction de la recherche qui est faite par l'utilisateur et de certaines autres caractéristiques.

Toutefois, dans la plupart des approches les plus populaires, le modèle commence par extraire de l'information des documents cible et peut par la suite se définir une mesure de distance entre une requête et chacun des documents pour déterminer quel serait la meilleure correspondance requête-document. Malheureusement, pour ce projet, nous n'avons pas accès au contenu des documents que l'on souhaite prédire, mais bien à un jeu restreint de caractéristiques telles la source du document, son auteur et son titre.

Nous avons donc le choix entre attaquer cette problématique comme un problème de régression pour attribuer un score à chacun des documents pour recommander ceux dont le score est le plus élevé ou comme un problème de classification. Puisque l'approche par régression demandait d'attribuer un score aux documents pour les données de test sur lesquelles nous basons notre modèle, nous avons décidé d'approcher la situation comme un problème de classification

en utilisant les probabilités a posteriori de notre modèle pour définir les 5 documents les plus pertinents.

2 Approche proposée

Ici, on présente les grandes lignes conceptuelles qui ont basé notre travail. On souhaite entre autres présenter et référencer les modèles les plus importants qui ont été utilisés dans notre modèle.

On discute aussi des concepts du document recommendation et pourquoi on s'intéresse surtout à certaines variables (pourquoi beaucoup de travail sur les queries, utilisation de techniques du traitement de la langue naturelle, etc.)

– Début du vrai texte

Selon le livre *Introduction to Information Retrieval* de (Schütze, Manning, & Raghavan,), une approche standard en recherche d'information est de se servir du contenu des documents pour créer un jeu d'attributs pour chaque document disponible. On fait ensuite la même chose avec les recherches qui ont mené à ces documents et on peut par la suite se définir une mesure de similarité entre une recherche et un document de telle sorte que la similarité soit la plus grande dans les cas où le document était pertinent pour l'utilisateur.

Si nos attributs sont bien construits et qu'on définit bien notre mesure de similarité, on peut ainsi facilement recommander une liste des documents les plus pertinents pour une nouvelle requête en appliquant nos traitements sur la requête et en calculant la similarité avec chacun des documents.

Malheureusement, n'ayant pas accès au contenu des documents à recommander, nous avons décidé d'attaquer le problème comme une situation d'apprentissage supervisé où les classes sont l'ensemble des documents possibles et en construisant des attributs autour de nos requêtes.

Puisque nous pensons que la majorité de l'information utile à nos prédictions se trouve dans la requête textuelle, notre approche consiste à tester plusieurs techniques de vectorisation de texte pour transformer nos requêtes textuelles en information numérique utilisable pour entraîner des modèles d'apprentissage automatique.

Également, afin de bien capter les requêtes utilisant des mots de sens commun, nous souhaitons utiliser les plongements de mots, décrits dans le chapitre 6.8 du livre *Speech and Language Processing* de (?, ?).

On souhaite par la suite utiliser ces représentations numériques de nos requêtes pour comparer différents modèles d'apprentissage automatique et optimiser leurs hyperparamètres pour augmenter le pouvoir prédictif de notre modèle.

Finalement, puisqu'on s'attaque ici à un problème de classification à un très grand nombre de classes, nous souhaitons faire de l'apprentissage non-supervisé sur les attributs de nos documents pour regrouper certains d'eux et réduire le nombre de classes possibles. On utiliserait par la suite ces classes agrégées pour entraîner un modèle de classification qui retournerait plutôt un groupe de documents duquel on choisirait les 5 plus pertinents.

3 Méthodologie expérimentale

Ici, on parle de la manière dont on applique les grandes lignes décrites auparavant.

Points à traiter : - Méthodologie de validation (Séparation du jeu de données) - Mesure de score utilisée (Précision sur recommandation de 5 documents car métrique d'évaluation de Coveo) - Utilisation des données sans clicks et à plusieurs clicks - Pipeline et recherche en grille - Paramètres testés

- Début du vrai texte

Avant de débiter l'optimisation de notre modèle, nous avons défini quelle mesure de score allait être utilisée pour l'évaluation de notre modèle afin de baser nos développements sur celle-ci. Puisque l'évaluation de notre modèle sera faire en regardant si le document pertinent se trouve dans la liste des 5 documents les plus pertinents fournis selon notre modèle, nous nous sommes bâti une fonction de score qui retourne le pourcentage de réussite selon ce critère particulier. Également, afin d'évaluer notre modèle, nous avons décidé d'utiliser le partitionnement des données déjà fait par Coveo. Nous avons vérifié que cette séparation des données avait été faite de façon aléatoire en confirmant que les dates de recherche n'étaient pas ordonnées.

Puisque nos données brutes contiennent des informa-

tions de plusieurs types, dont des données textuelles qu'on ne peut pas directement utiliser dans les algorithmes d'apprentissage automatique, nous devons faire non seulement beaucoup de travail sur l'optimisation des hyper-paramètres de nos modèles, mais aussi sur le choix des pré-traitements à faire sur nos données.

Pour attaquer de façon claire le problème de l'optimisation des pré-traitements, nous avons utilisé le module *pipeline* de la librairie Python *sklearn*. Ce dernier nous permet de définir nos différentes étapes de pré-traitement par des classes dont les paramètres sont modifiables. On procédant ainsi, on peut simplement faire une recherche en grille comme on le ferait avec n'importe quel modèle, mais en testant plutôt différentes combinaisons de pré-traitements.

Également, puisque le nombre de classes possibles à prédire est très grand, nous avons analysé la possibilité de faire du clustering sur nos variables réponses pour créer des clusters de documents desquels on prédirait les 5 documents les plus fréquents. Ceci permettrait à notre modèle de travailler avec un nombre plus restreint de classes et ainsi de mieux capter le signal pour chacune d'elles. Les clusters ainsi créés représentent donc des documents semblables. Puisque le clustering des documents se fait sur leur titre, les clusters regroupent donc des documents traitant de sujets similaires.

Les différentes étapes de notre *pipeline* sont donc les suivantes :

- Fusion des données de recherche avec l'information des documents associés
- Filtre des champs conservés
- Normalisation des requêtes (stemming)
- Vectorisation des requêtes (tokenization, vecteur de fréquence, TF-IDF, Word2Vec)
- Transformation des variables catégoriques en variables indicatrices
- Imputation des données manquantes

4 Résultats expérimentaux

Présentation des résultats avec tableaux, figures et tests statistiques. On n'analyse rien ici, on ne fait que montrer ce que nous avons obtenue avec l'approche décrite plus haut.

5 Analyse des résultats

Faire du gros blabla sale sur les résultats. Pourquoi notre score n'est pas si élevé que ça, comment on au-

rait pu améliorer l'efficacité des embeddings. Techniques qui fonctionnent le mieux et avantages/inconvénients des différentes techniques en production (temps d'entraînement, mémoire, etc.)

6 Conclusion

Ouverture philosophique, constats du projet et apprentissages

Références

Ai, Q., Bi, K., Guo, J., Croft, W. B. (2018). Learning a deep listwise context model for ranking refinement.

doi : 10.1145/3209978.3209985
Alpaydin, E. (2010). *Introduction to machine learning* (2nd éd.). The MIT Press.
Goodfellow, I., Bengio, Y., Courville, A. (2016). *Deep learning*. MIT Press. (<http://www.deeplearningbook.org>)
Hastie, T., Tibshirani, R., Friedman, J. (2001). *The elements of statistical learning*. New York, NY, USA : Springer New York Inc.
Pang, L., Lan, Y., Guo, J., Xu, J., Xu, J., Cheng, X. (2017). DeepRank : A new deep architecture for relevance ranking in information retrieval.
doi : 10.1145/3132847.3132914
Schütze, H., Manning, C. D., Raghavan, P. (2008). *Introduction to information retrieval* (Vol. 39). Cambridge University Press.