


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

Dans le cadre de ce projet proposé par la compagnie Coveo , nous devons utiliser un historique de requêtes faites par des utilisateurs afin de développer un modèle de recommandation de documents. L'objectif 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.

Il existe plusieurs méthodes pour bâtir des systèmes de recommandation. Parmi ces méthodes, il en existe deux qui sont fréquemment utilisées en pratique. Ces méthodes sont le filtrage collaboratif et les systèmes basés sur le contenu. La première consiste à calculer des associations entre des items (documents) ou des utilisateurs pour ainsi utiliser cette information pour faire une recommandation. Cette méthode a l'avantage d'être relativement simple à implémenter. La deuxième méthode consiste à utiliser les attributs des items ou des utilisateurs pour prédire les documents d'intérêt. Cette méthode a l'avantage d'apprendre des liens entre certains attributs pour raffiner la qualité des recommandations.

Dans ces différentes approches décrites, 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. Dans notre cas, nous n'avons pas accès au contenu détaillé des documents que l'on souhaite prédire, mais seulement à un jeu restreint de caractéristiques comme la source du document, son auteur et son titre.

Nous avons également le choix d'attaquer la problématique comme un problème de régression ou comme un problème de classification. Dans le premier cas, il faut attribuer un score à chacun des documents et ainsi recommander ceux dont le score est le plus élevé. Dans le deuxième cas, il faut tenter de prédire directement une classe, qui correspond à un document en particulier.

Nous avons décidé d'utiliser un système basé sur le contenu dans un contexte de classification. Étant donné que nous avons beaucoup d'utilisateurs différents (environ 600) et beaucoup de documents différents (environ 6000), la méthode basée sur le filtrage collaboratif nous apparaissait moins efficace. De plus, certains utilisateurs sont inconnus par le système, ce qui rend plus complexe la tâche d'utiliser cette information.

Le reste du document est structuré de cette façon : la section 2 décrit notre approche de manière plus détaillée, la section 3 présente notre méthodologie expérimentale, la section 4 présente les résultats expérimentaux, la section 5 fait l'analyse de ces résultats. Finalement, la section 6 présente les différents constats et leçons que nous avons tirés de ce projet.

2 Approche proposée

Selon le livre *Introduction to Information Retrieval* de (Schutze, 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. Ces attributs sont ensuite utilisés pour faire l'apprentissage d'un algorithme prédictif. Cela correspond à l'approche basée sur le contenu que nous avons choisi d'emprunter. On peut également utiliser cette approche pour créer des attributs pour chaque de nos requêtes. Une fois que nous avons deux ensembles d'attributs (requêtes et documents), nous pouvons définir une mesure de similarité qui permettra d'associer une requête à un document. En ayant de bons attributs, il est possible de penser que la mesure de similarité permettra de trouver des associations vers des documents pertinents pour l'utilisateur.

Dans notre cas, nous n'avons pas accès au contenu des documents. Ainsi, au lieu de créer deux ensembles d'attributs, nous avons créé des attributs pour les requêtes seulement. Ensuite, nous allons utiliser ces attributs pour entraîner un modèle supervisé où les classes à prédire sont les différents documents possibles.

Pour bâtir ce modèle, nous pouvons séparer le travail en 2 grandes étapes : prétraitement des données et modélisation. La première étape consiste essentiellement à créer les attributs, alors que la deuxième consiste à entraîner des modèles en utilisant ces attributs.

Pour ce qui est de la création d'attributs, nous pensons qu'une grande partie de l'information prédictive réside dans le contenu de la requête directement. Pour utiliser cette information, nous allons tester plusieurs méthodes de vectorisation de texte pour transformer ces requêtes textuelles en information numérique. De plus, afin de mieux capter le contexte autour des mots dans les requêtes, nous allons utiliser les plongements de mots, décrits dans le chapitre 6.8 du livre *Speech and Language Processing* de (Jurafsky & Martin,). Pour faciliter l'apprentissage, nous allons également normaliser nos requêtes. Plus de détails sont donnés sur ces éléments dans la prochaine section.

Pour ce qui est de la partie modélisation, on souhaite utiliser les 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 ayant un très grand nombre de classes, nous allons également tenter de faire un modèle basé sur de l'apprentissage non-supervisé. Pour ce faire, nous allons créer des attributs de nos documents pour ainsi regrouper certains d'eux et réduire le nombre de classes possibles. On utiliserait par la suite ces classes

aggrégées pour entraîner un modèle de classification (qui serait cette fois-ci supervisé) retournant plutôt le groupe de documents duquel on choisirait les 5 plus pertinents (fréquents).

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 clics et à plusieurs clics - 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 faite 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 informations 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 per-

mettrait 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 aurait pu améliorer l'efficacité des embeddings. Tech-

niques 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.
- Jurafsky, D., Martin, J. H. (2014). *Speech and language processing* (Vol. 3). Pearson London.
- 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
- Schutze, H., Manning, C. D., Raghavan, P. (2008). *Introduction to information retrieval* (Vol. 39). Cambridge University Press.