Rapport de projet

Fouilles de données

PARIS DIDEROT

Enseignant : Anne-Claire Haury

Université Paris Diderot

Master Informatique 2ème année

Quentin BOUILLAGET - Stéphane SCHMIDELY

Sommaire

Préface

Dans le cadre de l'UE « Fouilles de données », nous devions concevoir une application mettant en pratique les différentes notions et algorithmes utilisés en datamining.

Pour notre projet, nous sommes parti sur un logiciel capable de recommander des produits en fonction des tweets d'un utilisateur. Ce document a pour but d'expliquer le problème soulevé, nos motivations ainsi que les différents moyens mis en place pour le résoudre.

Introduction

L'application que nous avons mis en place souhaite répondre à la problématique suivante : « Pouvoir proposer à un utilisateur des produits qui lui correspondent ». Nous voulions pouvoir permettre à un utilisateur de lui recommander différents produits basés sur le contenu qu'il publie, qu'il aime ...

Cette idée nous est venue alors que nous comptions choisir le sujet « Recommandation de produits ». Qu'est ce que la « recommandation de produit ? »Recommander aux utilisateurs des produits semblables à la façon d'Amazon ou de n'importe quel autre cyber-commercant ou recommander à utilisateur un produit avant même qu'il ne fasse le premier pas ?

Sur la base du second postulat, nous avons eu l'idée de mêler de croiser deux bases de données : celui d'un réseau social et d'un site marchant. Twitter, étant un réseau social beaucoup plus « ouvert »que Facebook (il n'y a que très peu de profils privés), notre choix s'est orienté vers celui-ci. De même pour Amazon, qui de par les possibilités offertes via son API (accès à une base de données gigantesque, monétisation...) a été choisi en lieu et place de comparateurs de prix ou de sites plus « locaux ».

Les résultats obtenus avec notre application ont été quasiment à la hauteur de nos attentes. Nous pouvons avancer plusieurs raisons que nous aborderons dans la section « Résultats ».

Méthodes

Base de données

Malgré le fait que l'application fonctionne en temps-réel, nous avons eu besoin de créer une base de données. Nous nous en servons lorsque nous devons comparé la liste des tweets d'un utilisateur à celle des autres utilisateurs. Pour cela, des informations ont été collectées et sauvegardées.

Nous stockons dans notre base de données trois tables : La première « IDF »stocke le nombre d'occurrences (nbDoc) où un mot donné (token) apparaît au moins une fois dans une timeline (ensemble des tweets d'une même personne). Cela nous permettra de calculer le TF-IDF pour l'ensemble des mots présent dans la timeline d'une personne dont on souhaite recommander un produit.

La seconde « Meta »fonctionne sur un principe de clé/valeur, pour stocker les valuers importantes. Il ne sert actuellement qu'à stocker le nombre de documents (timelines) total auquel on a appliqué un TF-IDF.

La troisième « Twittos »stocke l'ensemble des identifiants utilisateurs (username) dont on a déjà analysé la timeline ainsi que l'identifiant du dernier tweet analysé. Cette table permet de ne pas relancer le calcul du TF-IDF et le stockage dans une base de données sur la timeline d'un utilisateur qui n'aurait pas évoluée.

L'usage de l'ensemble des données récoltées par le biais de l'API Twitter est limité dans le cadre du « Twitter Developer Agreement ». (https://dev.twitter.com/overview/terms/agreement-and-policy)

Ces données seront utilisées lors des traitements que nous sommes amenés à faire dans notre application qui résulte du choix de l'utilisateur à un instant donné. En effet, le contenu dépendra de l'utilisateur selectionné mais aussi du contenu publié par ce dernier au moment de la requête et des tweets des autres utilisateurs présents dans la base de données.

Langage et librairies

Nous avons choisi de développer l'application en Python notamment dû au fait qu'il dispose d'un certains nombre de librairies dédié au datamining. Par ailleurs, ce projet était aussi une occasion pour nous d'apprendre ou de pratiquer de nouveau ce langage. Enfin, il nous a permis de découvrir un nouveau framework Web : Django. Au niveau de la base de données, nous avons pris SQLite dû fait de sa simplicité d'administration mais aussi au vue des données que nous avions prévu de sauvegarder.

Lors du développement de notre application, nous avons eu recours à plusieurs librairies externes :

Amazon Product : https://pypi.python.org/pypi/python-amazon-product-api/

Hunspell : https://pypi.python.org/pypi/hunspell

Tweepy : https://github.com/tweepy/tweepy

Amazon Product

« python-amazon-product-api »est une des seules bibliothèques Python qui permet d'interfacer avec la « Product Advertisement API »d'Amazon et qui l'implémente correctement (les spécifications de celle-ci datant de 2011). Cette API permet d'accéder aux différents produits disponibles sur Amazon via les fonctions de recherches, les informations autour de ces produits ou encore les produits similaires. De plus, en étant affilié au programme Amazon, l'utilisation de cette API pour générer des ventes peut conduire à gagner un petit pourcentage sur cette dernière.

Hunspell

Nous avons décidé d'avoir recours à cette librairie car elle permet entre autres de vérifier si des mots existent dans les dictionnaires français et anglais, de proposer des synonymes et de donner la racine d'un mot dans le but d'uniformiser et de corriger les données que nous allions recevoir en entrée.

Par ailleurs, Hunspell est une librairie utilisée dans de nombreux programmes comme LibreOffice, Firefox, Thunderbird, Google Chrome ce qui nous a conforté dans notre choix.

Tweepy

Tweepy est une des nombreuses librairies disponible en Python pour interfacer avec l'API de Twitter. Nous avons décidé de la choisir en raison de sa popularité mais aussi grâce à sa facilité d'utilisation. Elle nous permet de récuperer les différents tweets d'un utilisateur donné mais aussi lors de la saisie du nom du compte Twitter afin de lui suggérer différents profils associés à sa saisie.

Traitements

La seule action requise par l'utilisateur est de saisir le nom du compte Twitter dont il souhaite connaître les différents produits associés à ses contenus. Une fois la validation effectuée, le programme se charge de :

- récupérer les 100 derniers tweets du profil choisi
- nettoyer chacun des tweets
  - supprimer les URL
  - supprimer les smileys
  - supprimer la ponctuation
  - supprimer le # des hashtags
  - supprimer les références au profil (@compte)
  - supprimer les prépositions/stop-words (le, la, du ...)
  - convertir le tout en minuscule
  - corriger les mots et remplacer par la racine de chaque mot

    - tester si le mot appartient au dictionnaire français ou anglais
    - s'il appartient à un des deux dictionnaires, alors on cherche la racine du mot dans la langue correspondante
    - s'il existe dans les deux dictionnaires, on se base sur la langue du tweet et on cherche sa racine
    - s'il n'existe dans aucun des deux, on regarde s'il existe un mot se rapprochant le plus du mot en recherchant dans le dictionnaire correspondant à la langue du tweet sinon on l'ajoute dans les mots inconnus (ex de cas particulier : marque, nom propre ...)
- séparer le corps du tweet, les hashtags
- création d'une liste de 'tokens'
- calcul du TD-IDF sur l'ensemble des tokens afin de déterminer les mots-clés majoritaires avec un point plus fort sur les mots venant des hashtags par rapport aux documents stockés en base
- requête vers l'API d'Amazon pour récuperer une liste de produits en lien avec les mots-clés calculés précédemment

La majeure partie de l'application se situe au niveau du traitement des tweets et au niveau du calcul du TD-IDF afin de pouvoir calculer les mots-clés au plus proche du contenu publié par le compte Twitter sélectionné.

Récupération des données

L'ensemble des données sont récoltées auprès des timelines d'utilisateurs de Twitter différents. Initialement, un premier set de données est récupéré afin d'avoir une base afin de proposer des produits. Cette base s'étoffe ensuite à chaque recommandation de produit par les données récupérées pour chaque utilisateur visé.

Résultats

Les résultats de notre application ont quasiment été à la hauteur de nos attentes. Plusieurs raisons existent pour expliquer les différences entre nos attentes et la réalité.

La première concerne le traitement effectué sur les tweets. Malgré l'utilisation d'un dictionnaire afin de corriger les fautes de l'utilisateur, il persiste des fautes après le passage du correcteur. Ainsi, lors du calcul du TD-IDF ces mots ressortent avec un plus fort poids par rapport aux autres mots. Une des solutions auraient été de supprimer ces mots inconnus des dictionnaires ... mais ces mots peuvent être tout simplement des marques. Dans ce cas, nous ne pouvons pas nous priver d'une information importante.

Le second élement est sur le fonctionnement du moteur de recherche d'Amazon. En effet, ce dernier n'effectue pas une recherche sur un ensemble de mots clés mais il effectue plutôt une recherche sur chacun des mots données ce qui donne un résultat moins précis.

Enfin, une dernière explication concerne le résultat retourné par Amazon. Ce dernier ne permet pas de spécifier combien d'élements sont attendus. Par conséquent, il renvoie automatiquement les 100 élements qui correspondent à la recherche effectuée ce qui ralenti le chargement des résultats.

Hormis ces différents soucis, l'application est en mesure de proposer à l'utilisateur des produits en adéquation avec le contenu qu'il a publié sur Twitter (Fig. 1 : exemple du calcul du TD-IDF d'un utilisateur).

Discussion

Analyse de notre travail

Cette application nous a amenés à travailler aussi bien sur le back-end que sur le front-end. Nous avons donc eu la possibilité de travailler les algorithmes de traitement mais aussi sur le rendu de notre site web.

Nous n'avons pas rencontré de problèmes au niveau du front-end : l'affichage et la modification du rendu des élements n'ont pas été très compliqués.

Concernant le cœur de notre application, nous avons rencontrés différents soucis techniques lors du traitement des tweets et lors du calcul du TD-IDF.

Traitements des documents

Sur les tweets que nous recevons, nous effectuons une série d'actions afin de pouvoir au maximum nettoyer la chaîne de caractères avant le calcul du TD-IDF. Le traitement le plus important consiste à corriger les éventuelles fautes de phrases et d'orthographe des différents mots.

Pour pallier à ce problème, nous avons eu recours à un dictionnaire français/anglais qui nous a permis de tester l'éxistence d'un mot mais aussi de pouvoir nous donner des synonymes et la racine d'un mot.

TD-IDF

Concernant le TD-IDF, nous étions parti sur le fait de calculer le TD-IDF d'un document qui serait constitué par l'ensemble des tweets d'un utilisateur. Or il s'est avéré que le résultat n'était pas assez correct et que nous ne satisfaisions pas toutes les contraintes imposées par cette méthode. En effet, afin d'avoir un résultat optimal et complet, il est nécessaire d'avoir d'autres documents afin de pouvoir comparer la liste des tweets de l'utilisateur à d'autres tweets venant de profils variés.

Nous avons donc mis en place une base de données afin de stocker différentes informations concernant les précédents documents déja étudiés et qui serviraient maintenant de documents de « référence ».

Améliorations/extensions possibles

Améliorations

L'algorithme de notre application pourrait être sujet à amélioration. Constituant le cœur de notre application, cette dernière est essentielle afin d'obtenir des produits en adéquation avec les tweets publiés par l'utilisateur.

D'autres traitements que ceux existant pourraient donc éventuellement être effectués sur les différents tweets afin d'affiner encore plus la requête. Par exemple, il pourrait être possible de mettre des poids en fonction du tweet : normal, retweeté, favorisé/aimé

ou encore favorisé et retweeter. En fonction de ces informations, nous pourrions éventuellement rajouter des poids sur les mots du tweet en question afin de lui donner plus d'importance lors du TD-IDF notamment.

Une autre suggestion d'amélioration pourrait être l'analyse des images associées aux tweets. Il pourrait être ainsi possible de l'analyser afin de ressortir de nouveaux mots clés en relation avec le média.

Extensions

Une des première extensions pourrait être de se baser non plus sur les utilisateurs mais sur les hashtags. Ainsi, nous pourrions, par exemple, proposer une liste de produits en relation avec un « trending topics ».

A cette époque où tout le monde cherche des cadeaux de Noël, il pourrait être possible d'ajouter un montant minimum/maximum pour les produits proposés. Ainsi, notre application recommanderait une liste de produits compris dans son budget et en adéquation avec les goûts de la personne.

La dernière extension concernerait l'ajout de Facebook. En effet, nous pourrions récuperer les contenus publiés par un utilisateur de Facebook et effectuer le même traitement que ce que nous faisons pour Twitter. Toutes les fonctionnalités précédemment évoquées seraient aussi valable pour ce réseau social.

Conclusion

Ce projet nous a permis de mettre en pratique les différentes notions et méthodes de datamining vu en cours appliqué à un cas concret. Nous avons du faire face à différents problèmes soulevés par les grandes quantités de données reçues notamment au niveau du contenu des tweets de chaque utilisateur.

Il nous a aussi permis de découvrir plus en profondeur Python et d'utiliser différentes librairies en lien avec les API de Twitter et d'Amazon.

Ce logiciel possède un grand nombres d'extensions possibles en commençant par la possibilité de récuperer du contenu d'un utilisateur depuis son compte Facebook. Il existe encore bien d'autres ajouts imaginables qui permettraient de rendre l'application encore plus précise et plus proche des recommandations que le site web pourrait leur faire.

Equipe

Répartition des tâches

Dans le cadre de notre projet, nous devions avoir recours à deux API : celle de Twitter afin de récuperer les différents tweets d'un utilisateur et Amazon pour qu'il nous renvoie une liste de produits associées aux différents mots clés que nous lui fournissions en paramètre.

Stéphane s'est chargé de mettre en place l'interface avec l'API Twitter mais aussi de nettoyer et « harmoniser »les différents tweets de l'utilisateur afin de pouvoir ensuite récuperer des mots-clés à soumettre à Amazon. De son côté, Quentin s'est occupé de l'API Amazon et du TD-IDF avec tout ce que cela implique (création de la base de donnée, calcul du TD-IDF ...).

Le site web a été développé par nous deux.

Listes des figures

pp	0.16033931567940177
nuit	0.09653898292348985
flexibility	0.09153272637347049
polyalcool	0.09153272637347049
flexible	0.09153272637347049
maintenant	0.0865264698234511
vaillante	0.08016965783970088
benchmark	0.08016965783970088
definitive	0.08016965783970088
langage	0.08016965783970088
charitable	0.08016965783970088
jouable	0.08016965783970088
Linux	0.08016965783970088
ils	0.07752854045618028
list	0.07352268885432113
season	0.07352268885432113
oie	0.07352268885432113
hmm	0.07352268885432113
feeling	0.07352268885432113
support	0.06880658930593128
window	0.06880658930593128
doute	0.06880658930593128
foncer	0.06880658930593128
love	0.0651484983007803
source	0.0651484983007803
project	0.0651484983007803
release	0.0651484983007803
ale	0.06215962032055155
open	0.06215962032055155
bout	0.06215962032055155

FIGURE 1 – Exemple du résultat d'un TD-IDF sur les tweets d'un utilisateurs