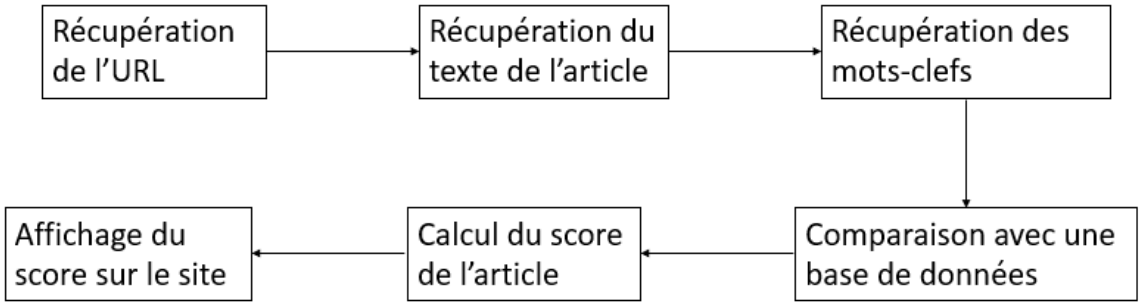
Lucas Méchinaud
Nolwenn Smith

COMPTE RENDU MI-PARCOURS

IN104

Pour ce projet, nous nous sommes intéressés au classement d'articles de journaux, selon leur degré de fiabilité. Le but de notre code est de créer un site internet à qui l'on fournit l'URL de l'article et renvoie un score permettant de déterminer si l'article est une *« fake news »* ou non.

Pour cela, plusieurs étapes sont nécessaires :



Pour le site, nous avons utilisé Django afin d'avoir un modèle de site « basique ». Nous avons notamment utilisé la librairie *Forms* pour pouvoir inclure la case de recherche où l'utilisateur donne l'URL. Pour l'apparence du site – et notamment le bouton « submit » – nous avons dû utiliser des squelettes HTML.

Pour récupérer l'article, nous avons utilisé la librairie *Newspaper*, qui permet, à partir d'un URL, de télécharger l'article et de travailler dessus.

Afin de récupérer les différents mots-clefs, nous avons entrepris deux méthodes différentes, afin de pouvoir avoir une précision supplémentaire. Ces deux méthodes utilisent le NLP.

1. Méthode « automatique »

Dans la librairie *Newspaper*, on utilise la fonction « keywords » qui prend en argument l'article et renvoie tout simplement un tableau avec les mots-clefs.

Lucas Méchinaud
Nolwenn Smith

2. Méthode « manuelle »

Nous utilisons la librairie *Sklearn*, et notamment la fonction « TfidfVectorizer », puis « fit_transform » (qui renvoie une matrice). Néanmoins, nous avons remarqué que pour avoir une matrice compréhensible, le texte devait être sur une seule ligne. Nous avons donc dû travailler le texte, afin de supprimer tous les retours à la ligne.

La matrice renvoyée par « fit_transform » est composée des poids de chaque mot par ligne (d'où l'importance que le texte soit sur une seule ligne). En comparant avec la matrice renvoyant tous les mots du texte (récupérée grâce à « get_feature_names »), nous pouvons alors récupérer les 10 mots de poids prédominant : ce sont nos 10 mots-clefs !

La comparaison avec la base de données se fait avec le module *Newsapi*. Pour cela, nous avons dû créer un compte afin de récupérer une clef. Avec un simple « get_everything » en donnant en argument les mots-clefs, nous pouvons trouver le nombre d'articles possédant ces mots-clefs.

Le score est ensuite calculé en fonction du nombre d'articles trouvés à l'étape précédente. A ce jour, nous n'avons pas encore établit les seuils qui donneront le score de l'article.

Tout du long de ce projet, nous avons rencontré une multitude de problèmes. En voilà quelques-uns :

- Au début, nous avons eu beaucoup de difficultés à nous familiariser avec les outils proposés tels que Django, et PyCharm. Néanmoins, après quelques heures de travail, nous avons réussi à maitriser les fonctionnalités qui nous sont nécessaires (views.py, manage.py, le routage url, le code html).
- Pour l'instant, la méthode « manuelle » ne renvoie pas des mots pertinents. A ce jour, elle renvoie des mots tels que « je », « nous », « le » : ce qui ne sert à rien puisqu'ils sont présents dans presque tous les articles. Ce problème est présent pour des articles en anglais et en français.
- La méthode « automatique », quant à elle, renvoie des mots pertinents pour des articles anglophones, mais pas des articles en français.

Pour la suite, nous allons nous atteler à plusieurs choses :

- Le perfectionnement des méthodes pour trouver les mots-clefs, quitte à trouver une autre méthode, toujours en s'appuyant sur le NLP.
- Déterminer les seuils afin d'obtenir des résultats cohérents
- Améliorer le design du site

Lien du git : https://github.com/Lucasmechinaud/FakeNewsDetector.git