CentraleSupélec

Rapport EI ST6

Guillaume Bertho, Marc Antoine Leny,
Armand Modjabi, Bhanujan Sivapalan,
Elisa Verde Mota

2 juin 2023

Table des matières

1 Contexte et objectifs du projet

Suite à la séquence thématique Data Web, nous avons pu appréhender les futurs défis, ainsi que les enjeux, du monde du web. Comment exploiter des données massives à grande vitesse et de manière pertinente ? Se posent alors des questions éthiques sur la collecte de ces données, ainsi que sur l'usage qui en est fait. Afin de porter un regard informé sur ces problématiques, nous avons dû dans un premier temps nous pencher sur l'aspect technique du traitement de données.

C'est dans cette démarche que s'inscrit notre projet de l'enseignement d'intégration. Les moteurs de recherche sont en effet un outil privilégié pour trier et accéder à la data, dont le fonctionnement fait appel à des systèmes d'intelligence artificielle. Comment évaluer la pertinence d'un document par rapport à la requête d'un utilisateur ? Comment optimiser la vitesse du moteur de recherche ? Comment représenter les documents dans la base de données ?

Grâce à l'aide de nos encadrants et d'experts de Headmind Partners, leader en conseil dans les secteurs de la cybersécurité et des projets numériques, nous avons pu découvrir les outils techniques à la base de la conception d'un moteur de recherche performant. Nous avons tout d'abord récupéré des données issues de https ://archive.org/details/stackexchange, que nous avons ensuite mises en forme par le biais de l'indexation. Puis, nous avons réalisé en python un moteur de recherche et testé plusieurs méthodes pour tester la pertinence des documents selon la requête (booléen, vectoriel, probabiliste et sémantique). Enfin, nous avons proposé des améliorations pour rendre notre moteur plus efficace.

2 Premiers programmes

Dans un premier temps, nos encadrants nous ont proposé de suivre une démarche pour prendre en main les outils et créer la structure du moteur de recherche. Pendant trois jours, nous avons ainsi suivi étape par étape leurs conseils.

2.1 Day 1

Le premier jour a été dédié au traitement de données massives et à la mise en forme de celles-ci. Nous avons tout d'abord écrit quelques requêtes SQL afin d'explorer la data qui nous avait été donnée.

```
# What is the Content of the smallest Post ?
SELECT Id
FROM Posts
WHERE LEN(Body) = (SELECT MIN(LEN(Body)) FROM Posts)
```

Id
89837

```
# What is the most voted post ?
SELECT Id
FROM Posts
WHERE Favoritecount = (SELECT MAX(Favoritecount) FROM Posts)
```

Id
115708

Ensuite, nous avons initialisé l'environnement et extrait la data, qui s'avérait être en format XML. Ce format peut être choisi par les développeurs pour plusieurs raisons. Il permet de décrire les données de manière hiérarchique en utilisant des balises ouvrantes, ouvrantes et fermantes, ce qui facilite la représentation des données complexes et leur organisation logique. De plus, XML est un format largement accepté et pris en charge par de nombreux langages de programmation et systèmes. Il offre une interopérabilité élevée entre différentes plateformes, ce qui facilite l'échange de données entre systèmes hétérogènes. Enfin, XML est conçu pour être lisible et compréhensible par les humains, ce qui facilite le débogage, la maintenance et la collaboration entre développeurs.

2.2 Day 2

Le deuxième jour, nous avons conçu l'architecture de base du moteur de recherche, toujours en suivant les étapes de nos encadrants. Pour cela, nous avons tout d'abord implémenté l'indexation. L'index nous a permis par la suite de comparer la requête aux documents de corpus et donc d'en déduire lesquels étaient les plus pertinents. Nous avons testé plusieurs méthodes. Si bien les premières étaient plutôt naïves et ne faisaient que compter le nombre d'occurrences des mots de la requête dans chaque document, nous avons ensuite implémenté des méthodes plus complexes (méthode booléenne et probabiliste) L'ensemble des programmes utilisés pourra être retrouvé dans le notebook joint à ce rapport.

2.3 Day 3

Le but de cette séance était d'améliorer le moteur de recherche grâce à des méthodes NLP, qui tiennent compte du contexte des mots, et non pas seulement des termes indépendamment les uns des autres. Tout d'abord, nous avons utilisé la

méthode vectorielle. Nous avons vectorisé les documents du corpus. Puis, nous avons vectorisé la requête et grâce à la similarité cosinus, la requête est comparée à l'ensemble des documents, dont la pertinence peut alors être évaluée. Ensuite, nous avons utilisé une méthode sémantique. Grâce à un modèle choisi parmi ceux choisi, et à la bibliothèque SKLearn, la pertinence d'un document dépendait aussi du contexte de chaque mot de la requête au sein de ce document. Une possibilité est alors de combiner ces deux méthodes. Ceci permet de rendre notre moteur de recherche plus robuste. Plus concrétement, on implémente chacune des deux méthodes indépendamment. On obtient deux matrices de similarité, V (méthode vectorielle) et S (méthode sémantique). La matrice de similarité finale M_{finale} est obtenue en pondérant :

$$M_{\text{finale}} = 0{,}5\ \text{V} + 0{,}5\ \text{S}$$

En utilisant l'évaluation par NDCG pour 1 requête et pour k=12, on obtient des résultats semblables pour les deux méthodes (un peu meilleurs pour le modèle purement sémantique).

Modèle sémantique : NDCG de 0.82119
Modèle mixte : NDCG de 0.81641

Nous utiliserons le modèle sémantique par la suite.

3 Amélioration de notre moteur de recherche

Après avoir suivi les étapes présentées par les encadrants, nous avons pu personnaliser notre projet et proposer des nouvelles pistes de réflexion pour rendre notre moteur de recherche plus performant et sophistiqué. Nous avons gardé la méthode sémantique, plutôt que la méthode vectorielle, car elle tenait compte du contexte des mots, et favorise donc les performances de notre moteur. Nous avons eu trois autres idées pour améliorer notre système.

Tout d'abord, nous avons implémenté un correcteur d'orthographe, de sorte à ce que les éventuelles fautes de frappe dans la requête puissent être corrigées. Ceci permet au moteur de proposer des documents pertinents à l'utilisateur, sans que celui-ci ait à taper une deuxième fois sa requête. Chaque mot de la requête sont comparés aux mots d'un dictionnaire. Si le mot n'y figure pas, le correcteur est autorisé à faire des opérations sur les lettres (suppression, transposition, ajout, etc) afin de se ramener à un mot du dictionnaire

De plus, nous avons ajouté une fonctionnalité de tri des résultats de la requête selon un critère de date. Nous nous sommes inspirés du moteur de recherche Google, où l'utilisateur peut décider de ne pas voir des documents trop anciens. Ainsi, suite à la rédaction de sa requête, l'utilisateur doit répondre à la question : "Souhaitez vous voir des documents datant d'après 2019 ?" Le moteur trie alors les documents selon la méthode sémantique étudiée dans les parties précédentes. Puis, les documents trop anciens ne correspondant pas aux attentes de l'utilisateur sont éliminés de la liste de documents renvoyés. L'utilisateur peut aussi renseigner une année précise s'il ne veut voir que des documents ed cette année-là.

Enfin, la dernière amélioration que nous avons implémenté donnait plus d'importance à un mot du titre qu'à un mot du corpus du document. Cela permettait en effet de cerner les documents dont le sujet principal était proche de la requête de l'utilisateur. Pour mener à bien cette initiative, nous procédons en deux temps. Une matrice de similarité entre la requête et le corpus est générée. Notons cette matrice C. Puis, une matrice de similarité entre la requête et le titre est générée. Notons cette matrice T. On pondère pour obtenir la matrice de similarité $M_{\text{sémantique}}$, dans laquelle les mots du titre pèsent plus que les autres mots du corpus :

$$M_{\text{sémantique}} = 0{,}1\ \text{T} + 0{,}9\ \text{C}$$

Les coefficients de pondération ont été choisis suite à quelques tests et de manière quelque peu arbitraire. Peut-être, dans une version plus sophistiquée de notre moteur de recherche, on pourrait demander à l'utilisateur l'importance qu'il

accorde aux mots du titre par rapport aux mots du corpus, afin de choisir des coefficients en accord avec ses préférences.

Toutes ces initiatives nous ont permis de personnaliser notre moteur de recherche et de le rendre plus performant. Le code sera detaillé dans le Notebook Python joint à ce rapport.

4 Conclusion

Le traitement massif de données est l'un des actuels enjeux du monde du web. Dans ce cadre-là, les moteurs de recheche sont des outils privilégiés pour explorer la data. Les performances de ces moteurs sont évaluées en fonction de leur rapidité, mais aussi de la pertinence des documents renvoyés pour une requête donnée.

Dès lors, pour l'implémentation de notre propre moteur de recherche, nous avons tenté de prendre des initiatives qui permettaient au moteur de renvoyer les documents le plus pertinents possibles, en assurant la bonne ortographe de la requête grâce à un correcteur, en permettant à l'utilisateur de ne selectionner que des documents datant d'une certaine date ou d'et en donnant plus de poids à un mot du titre qu'à un mot du corpus.