Master 2 SISE Statistiques et Informatiques pour la Science des Données

Université Lumière Lyon 2
Année universitaire 2023–2024

Projet Text Mining

Analyse des offres d'emplois

Natacha Perez
Abdourahmane Ndiaye
Martin Revel

Contents

1 Description du projet

L'objectif de ce projet a été d'analyser le corpus des annonces d'offres d'emplois accessibles en ligne (pole-emploi.fr, apec.fr). Notre analyse s'est concentrée uniquement sur les postes de data scientist et data engineer, dans le but de les comparer. Ces offres d'emplois ont été extraites à l'aide de techniques de web scraping (Selenium) et via une API de Pôle Emploi. Puis, le langage de programmation python a été utilisé par la suite pour nettoyer et harmoniser ces données. Les données ont ensuite été chargées dans une base de données, modélisée sous la forme d'un entrepôt avec une table de faits, des dimensions et des hiérarchies, stockée dans un SGBD SQLite. Une application Streamlit a été développé pour servir de support d'exploration intéractif pour parcourir les analyses des offres d'emplois. L'analyse intègre notamment des dimensions territoriales sur la France métropolitaine et des techniques de NLP ont été utilisé pour analyser le corpus des offres. Enfin, la base de données et l'application ont été déployé via une image docker.

2 Extraction des données

Deux méthodes d'extraction d'offres d'emplois ont été utilisés:

- **Extraction via l'API de Pole-Emploi**
- **Extraction des offres de l'Apec par web scraping**

Le script d'extraction est contenu dans le notebook 'scraping.ipynb'.

2.1 Extraction des offres de Pole-emploi par API

Pour récupérer les données des annonces de Pole-Emploi, on a utilisé une API public de Pole-Emploi, dont la documentation est disponible : https://pole-emploi.io/data/api/offres-emploi?tabgroup-api=documentationdoc-section=api-doc-section-caracteristiques.
Pour utiliser l'API, nous avons:

- Déclarer une application: Créer un compte sur pole-emploi.io, puis créer une application depuis son espace.
- Souscription à une API: Puis on obtient un identifiant client et une clé secrète.

L'utilisation de l'API comprends 2 étapes (Figure 1):

- **Génération d'un access token:** On génère une demande d'access token à Pôle Emploi Access Management (PEAM) avec notre ID client et notre clé secrète, via une requête 'POST'. PEAM génère et fournit un access token qui expire après 24 minutes.
- **Requête de l'API:** Après obtention de l'access token on peut requêter l'API par une commande 'GET' et extraire les offres d'emplois.

Lors de la requête, nous avons choisi d'extraire les postes de 'data scientist' et 'data engineer' et des annonces triées par date de création décroissant, pertinence décroissante, distance croissante.

À l'avenir: dès février 2024, pole-emploi.io devient francetravail.io, de ce fait, les URL d'appel aux API risquent de changer et les démarches d'utilisation des API pourront être différentes de celles mentionnées ci-dessus.

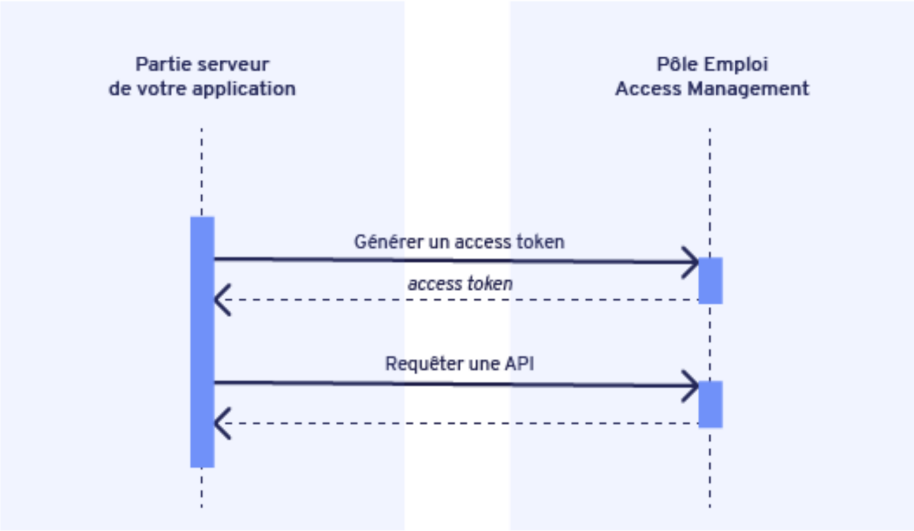


Figure 1: **Génération de token et requête de l'API Pole-Emploi**

2.2 Extraction des offres de l'Apec par web scraping

Selenium est une suite d'outils open-source conçue pour automatiser les navigateurs web. Il est principalement utilisé pour tester des applications web en simulant le comportement d'un utilisateur interactif. Cependant, il peut également être exploité à des fins d'extraction de données, comme nous l'avons fait dans notre cas.

Dans notre scénario, nous avons utilisé Selenium pour automatiser la navigation sur un site web d'emploi (Apec) afin de collecter des informations sur les offres d'emploi. Le processus d'automatisation comprend plusieurs étapes:

1. **Chargement de la page web :** Utilisation de Selenium pour ouvrir le navigateur Chrome et accéder à la page web de recherche d'emploi sur Apec.
2. **Interaction avec la page :** Selenium a été utilisé pour interagir avec les éléments de la page tels que les boutons, les liens, et les formulaires. Par exemple, nous avons cliqué sur des boutons pour accepter les cookies et avons navigué entre les pages pour accéder aux différentes offres d'emploi.
3. **Extraction de données :** À l'aide de Selenium, nous avons extrait les informations spécifiques souhaitées telles que les titres d'emploi, les salaires, les profils recherchés, etc. Pour cela il fallait récuperer le Xpath de l'information à extraire et l'integrer dans le code.
4. **Stockage des données :** Les données extraites ont été stockées dans un DataFrame pandas, une structure de données tabulaire en Python. Cela nous a permis de manipuler facilement les informations recueillies pour les nettoyer et les mettre en commmun avec le DataFrame récupéré par l'API de Pôle Emploi.

En résumé, Selenium a été un outil clé pour automatiser le processus de navigation sur un site web complexe, facilitant ainsi la collecte d'informations spécifiques à des fins d'analyse ultérieure.

3 Transformation des données

L'extraction des offres issus de l'APEC et de Pole-emploi a nécessité par la suite un ensemble d'étapes de traitements des données: 'Nettoyage.ipynb'.
Ces étapes ont consisté notamment en une harmonisation du format des dates, des lieux, des expériences requises,

des salaires et des description des entreprises. Ces variables ne sont pas renseignées de la même façon entre l'Apec et Pole-Emploi.
Des données issu de l'Open Data ont été ajouté afin d'enrichir les latitudes et longitudes des offres.
Une étape importante à été de transformer la variable salaire: initialement de type 'object', elle renseignait des intervalles de rémunération minimales et maximales, brut annuel ou mensuel. Le traitement a consisté à établir une rémunération annuelle moyenne entre la valeur minimale et maximale renseignée, afin d'obtenir une variable de type 'float' plus appropriée pour l'analyse.
La création de la variable 'profil' qui renseigne le profil du candidat recherché et les compétences exigées a été nécessaire, et servira de corpus pour l'analyse. Ces étapes ont donc été indispensables à la création d'un DataFrame, transformé ensuite en fichier CSV, regroupant les offres de data scientist et data engineer provenant des deux sources web.

4 Création de la base de données

4.1 Outils

Pour la création de notre base de donnée nous avons dans un premier temps creer des bases de données mysql. Nous avons cependant opté pour une solution plus optimale: pysqlite3. Ce dernier est une librairie Python qui permet d'interagir avec des bases de données SQLite, de requeter et gérer ces bases de donnée via des commandes en SQL.

Nous avons fait ce choix car SQLite est simple d'utilisation, a leger et ne demande pas de configuration ou administration spécifique ni même de creer de serveur. Notre base de donnée se présente alors comme un fichier facile à transféré, dockeriser.

4.2 Architecture

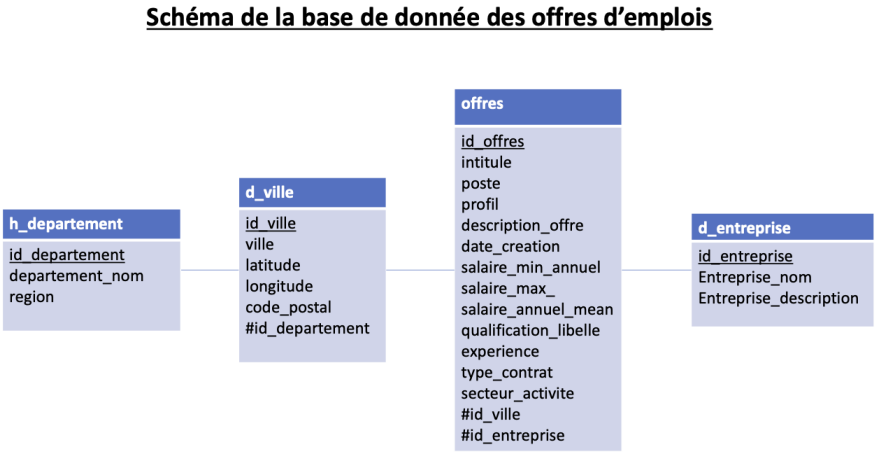
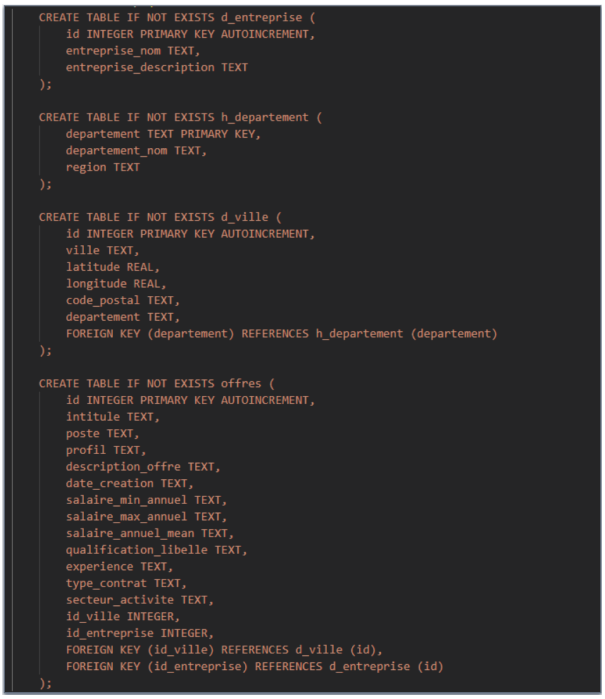


Figure 2: **Schéma de l'entrepôt de données**

4.3 Processus de création

Pour créer notre base de donnée nous avons d'abord créé sa sctructure via une requête SQL. Nous avons ensuite distribué les données nettoyées lors précédentes étapes dans cette base de donnée via Python.



```
CREATE TABLE IF NOT EXISTS d_entreprise (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    entreprise_nom TEXT,
    entreprise_description TEXT
);

CREATE TABLE IF NOT EXISTS h_departement (
    departement TEXT PRIMARY KEY,
    departement_nom TEXT,
    region TEXT
);

CREATE TABLE IF NOT EXISTS d_ville (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    ville TEXT,
    latitude REAL,
    longitude REAL,
    code_postal TEXT,
    departement TEXT,
    FOREIGN KEY (departement) REFERENCES h_departement (departement)
);

CREATE TABLE IF NOT EXISTS offres (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    intitule TEXT,
    poste TEXT,
    profil TEXT,
    description_offre TEXT,
    date_creation TEXT,
    salaire_min_annuel TEXT,
    salaire_max_annuel TEXT,
    salaire_annuel_mean TEXT,
    qualification_libelle TEXT,
    experience TEXT,
    type_contrat TEXT,
    secteur_activite TEXT,
    id_ville INTEGER,
    id_entreprise INTEGER,
    FOREIGN KEY (id_ville) REFERENCES d_ville (id),
    FOREIGN KEY (id_entreprise) REFERENCES d_entreprise (id)
);
```

Figure 3: image du script de creation de la base de donnee

5 Architecture de l'application

5.1 Requêter la BD

Pour assurer une analyse efficace et une visualisation fluide des données, notre application Streamlit utilise une approche basée sur la requête de la base de données. Cette approche présente plusieurs avantages qui contribuent à une manipulation plus aisée des informations recueillies.

Centralisation des Données :

En requêtant la base de données, nous consolidons toutes les données relatives aux offres d'emploi en un seul DataFrame. Cela facilite la gestion et l'accès aux informations, permettant une exploration plus intuitive des différentes dimensions de l'emploi, telles que les qualifications, les salaires, les régions, et bien plus encore.

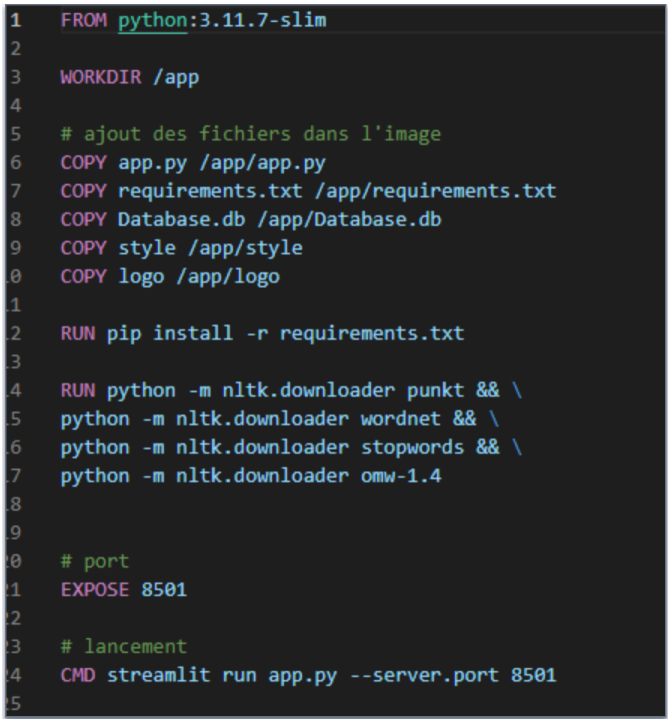
Personnalisation des Analyses :

Le DataFrame consolidé offre une flexibilité accrue pour effectuer des analyses approfondies et des visualisations spécifiques. Des filtres peuvent être appliqués en fonction des critères choisis par l'utilisateur, tels que la région, le secteur d'activité, le type de contrat, etc., pour des insights plus ciblés.

5.2 Dockerisation

Dockerization du Projet d'Application - Rapport des Étapes Suivies

La dockerization de notre application visait à encapsuler l'ensemble de notre environnement, tous les éléments qui la composent, dans un "conteneur Docker". Cela offre une portabilité de notre application et la rend assez indépendante de la configuration des ordinateurs de ceux qui l'utiliseront.



```
FROM python:3.11.7-slim

WORKDIR /app

# ajout des fichiers dans l'image
COPY app.py /app/app.py
COPY requirements.txt /app/requirements.txt
COPY Database.db /app/Database.db
COPY style /app/style
COPY logo /app/logo

RUN pip install -r requirements.txt

RUN python -m nltk.downloader punkt && \
python -m nltk.downloader wordnet && \
python -m nltk.downloader stopwords && \
python -m nltk.downloader omw-1.4


# port
EXPOSE 8501

# lancement
CMD streamlit run app.py --server.port 8501

```

Figure 4: script dans le fichier dockerfile

Configuration :

Nous avons commencé par organiser notre projet de manière à faciliter la dockerization. Cela inclut la création d'un fichier `requirements.txt` répertoriant toutes les dépendances Python nécessaires à l'exécution de notre application.

Nous avons aussi créé un fichier `Dockerfile` contenant les instructions décrivant l'environnement d'exécution de notre application, la structure du conteneur docker, ainsi que les commandes exécutées au lancement du conteneur.

Création de l'Image Docker :

À l'aide de la commande `docker build`, nous avons créé une image Docker à partir du `Dockerfile`. Cela a généré une image prête à l'emploi contenant notre application et ses dépendances.

```
docker build -t nom_de_l_image:tag .
```

Déploiement

Afin de déployer notre image sur Dockerhub, nous avons créé un espace sur la plateforme ainsi qu'un projet "nlpapp" au sein duquel nous avons pu envoyer notre image via les commandes suivantes:

```
docker tag app_nlp:latest abdouragit/nlpapp:1.0
docker push abdouragit/nlpapp:1.0
```

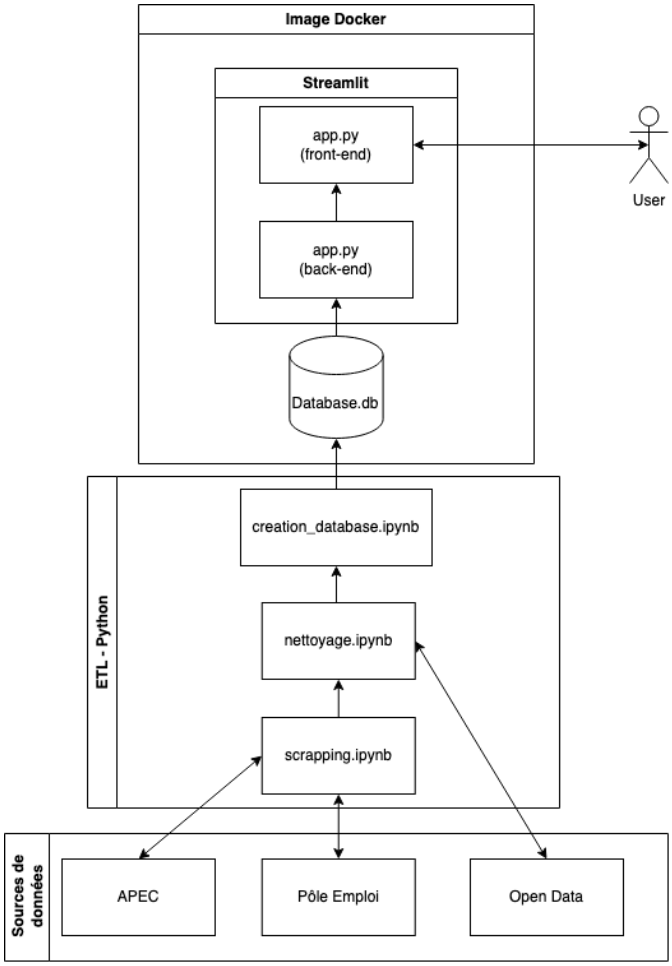


Figure 5: **Diagramme Architecture Streamlit**

6 Analyse des offres d'emplois

6.1 Analyse par territoire

Dans cet onglet, 3 cartes interractives sont affichées permettant la visualisation des emplacements géographiques des offres d'emploi. Chaques cartes possède des filtres 'Regions', 'Departement' et 'Ville', permettant d'afficher uniquement les offres correspondantes pour une visualistaion plus spécifique.

6.1.1 map 1 : Distribution des offres d'emploi

la carte adapte les zones visualisés en formant des cluster de façon automatique. On peut zoomer pour accéder au nombre total d'offres sur des clusters plus petits, jusqu'à avoir les emplacements des offres individuellement. Grâce à cette carte, on se rend compte que la grande majorité des offres se trouvent autour de Paris. On constate également une 'diagonale du vide' allant de la Meuse aux Landes.

6.1.2 map 2 : Visualisation de l'ensemble des offres d'emploi

L'intéret de cette carte est de permettre une exploration fine des offres grace aux informations affichés lorsqu'on clique sur offre tout en donnant une idée de leur dispersion/concentration. On constate les mêmes choses que sur la première carte : une forte concentration autour de Paris et une diagonale du vide, entre autres.

6.1.3 map 3 : Heatmap - Concentration des offres d'emploi

Cette carte offre une visualisation intéressante. Elle représente la concentration d'offres d'emploi par un dégradé de couleur allant du bleu au rouge. Cette carte nous permet de constater les mêmes choses que les précédentes, mais elle montre également de manière claire une deuxième zone de forte concentration dans le quart sud-est de la France.

6.2 Statistiques générales

Cette section présente une vue d'ensemble des données relatives aux offres d'emploi collectées dans notre base de données. Les principales analyses sont réparties sur deux colonnes pour faciliter la visualisation, la compréhension, et la comparaison des offres Data Engineer et Data Scientist.

Cette section vise à fournir une compréhension approfondie du paysage de l'emploi, vous permettant de prendre des décisions éclairées dans votre recherche d'opportunités professionnelles. Explorez les différentes facettes des données pour obtenir des insights pertinents selon vos besoins.

Sélection de la Granularité :

Les statistiques peuvent être explorées à différentes échelles géographiques, telles que la France entière, les régions ou les départements. Lorsque vous choisissez une région ou un département, les analyses s'adaptent pour refléter les caractéristiques spécifiques à cette localisation. Cela offre une vision détaillée des compétences et des qualifications recherchées localement.

Nombre Total d'Offres :

Un encart indique le nombre total d'offres d'emploi actuellement disponibles dans la base de données. Cela offre une vision instantanée de l'ampleur des opportunités professionnelles disponibles.

Wordcloud :

Les nuages de mots représentent visuellement les termes les plus fréquemment utilisés dans les profils de postes pour les rôles de Data Engineer et Data Scientist. Les mots sont préalablement nettoyés pour éliminer les informations redondantes, telles que les chiffres, la ponctuation, les espaces et les mots courants. Le curseur vous permet de régler le nombre de mots à afficher dans les nuages de mots, offrant ainsi une personnalisation optimale.

Histogramme des Mots les Plus Présents :

Les histogrammes vous permettent d'explorer plus en détail les mots les plus présents dans les profils de postes. Le curseur vous permet également de régler le nombre de mots à afficher dans les histogrammes.

Analyse des Salaires :

Les boxplots présentent une distribution des salaires annuels. Vous pouvez explorer ces données en fonction des différents filtres territoriales. Les valeurs aberrantes peuvent également être identifiées.

6.3 Analyse du corpus 'profil'

La variable 'profil' reseigne le profil du candidat recherché, les compétences attendues et les outils requis pour l'offre d'emploi en question. Cette variable a été utilisée comme corpus dans le but d'analyser et comparer les compétences attendues entre les postes de data scientist et data engineer.
L'algorithme Word2Vec de la librairie gensim (version 4.3.2) et la librairie nltk (version 3.8.1) ont été utilisé.

Word2Vec est un algorithme d'apprentissage profond utilisé pour représenter les mots sous forme de vecteurs dans un espace multidimensionnel. Le modèle Word2Vec a été entrainé sur le corpus spécifique aux postes 'data engineer' ou sur le corpus spécifique aux postes 'data scientist' qui ont subit des étapes de nettoyage. Nous souhaitons représenter les termes sur des graphiques, donc nous voulons un espace à 2 dimensions, renseigné par le paramètre `vector_size=2`. Le nombre de mots pris en compte lors de la création des vecteurs de mots, soit le nombre de *voisins*, a été défini par le paramètre `window=5`.
Ainsi, on a pu représenter graphiquement la représentation vectorielles des mots les plus fréquents contenu dans les profils recherchés pour les postes de 'data scientist' et 'data engineer'(Figure 2).

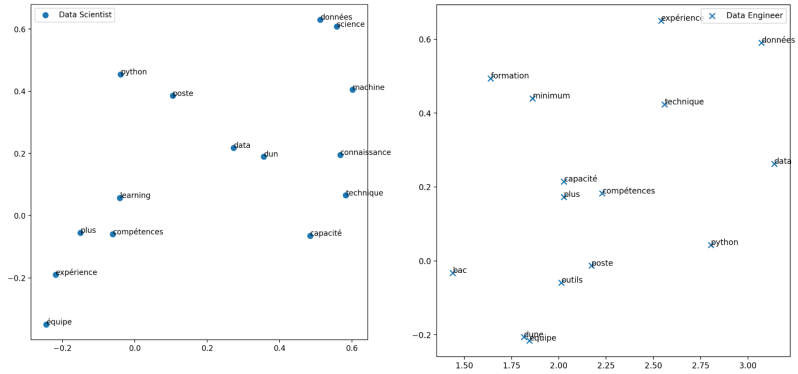


Figure 6: **Représentation des vecteurs associés aux 15 termes les plus fréquents des profils Data Scientist et Data Engineer**

Chaque point sur le graphique représente un terme, et sa position est déterminée par les valeurs des deux dimensions du vecteur associé à ce terme. On remarque que des mots comme "données", "data", "compétences", "expérience", "poste", "équipe", "capacité", "technique", "python" sont des mots fréquents autant dans les profils attendus de ces deux postes. Ils ne sont pas spécifiques à un post. Cependant, on remarque que "science" provenant certainement de "data science" et "machine", "learning" font partis des 15 mots les plus fréquents du profil de data scientist et ne sont pas retrouvés parmis ce top 15 des mots les plus fréquents dans le profil des postes data engineer. La distance entre les points sur le graphique reflète la similarité entre les termes. Des points proches indiquent des termes similaires en termes de contexte ou de co-occurrence dans les corpus. A titre d'exemple, pour les profils data scientist", à chaque fois que le mot 'données' est évoqué dans les profils attendus, il est associé au mot "science" qui est à proximité. Cela provient probablement de la combinaison "data science".

7 Tutoriel d'installation

7.1 Importation de l'image Docker et lancement de l'application

7.1.1 Étape 1: Importation de l'image Docker

Assurez-vous que Docker-desktop est installé sur votre machine. Si ce n'est pas le cas, téléchargez et installez Docker depuis: https://www.docker.com/get-startedle site officiel de Docker.
Ouvrez un terminal et exécutez la commande suivante pour télécharger l'image Docker de l'application :

```
docker pull abdouragit/nlpapp:1.0
```

7.1.2 Étape 2: Lancement de l'application

Exécutez l'application Streamlit dans un conteneur Docker en utilisant la commande suivante :

```
docker run --rm -p 8501:8501 -it abdouragit/nlpapp:1.0
```

Cette commande démarre le conteneur Docker et redirige le port 8501 de votre machine vers le port 8501 du conteneur, où l'application Streamlit est en cours d'exécution.

7.1.3 Étape 3: Accéder à l'application

Ouvrez votre navigateur web et saisissez l'URL suivante dans la barre de navigation : `http://localhost:8501`
Vous devriez maintenant pouvoir explorer l'application Streamlit depuis votre navigateur. **Note :** Assurez-vous que le port 8501 est disponible et n'est pas utilisé par une autre application sur votre machine. Si le port est déjà utilisé, vous pouvez utiliser un autre port lors de la commande `docker run`, par exemple `-p 8080:8501`.