# Big Data Management - Création d’un Data Lake

## Introduction :

L’objectif de ce projet était de concevoir un « mini » lac de données (Data Lake en anglais) afin de mettre en pratique les connaissances vues en cours. Pour ce faire, nous avons utilisé Neo4J afin de modéliser les données sous la forme d’un graphe. Nous avons choisi le jeu de données : Blog Authorship Corpus disponible sur la plateforme Kaggle à l’adresse suivante : <https://www.kaggle.com/datasets/rtatman/blog-authorship-corpus>. Il traite de blog écrit par des utilisateurs avant ou en 2004 comportant le texte de l’utilisateur et plusieurs métadonnées : Genre, Age, Sexe, topic, Signe astrologique, date. Pour agrémenter la quantité de métadonnées nous avons décidé d’ajouter un nom choisit par nos soins.

Le jeu de données étant beaucoup trop volumineux (611653 lignes), nous en avons gardé une dizaine que nous avons choisi unitairement afin d’en avoir de topics différents.

Cette sélection a permis de concevoir notre jeu de données (data.csv) que nous avons ensuite modélisé sous la forme d’un graphe dans Neo4j.

La prochaine étape a été de réalisé des traitements des textes sur Python afin d’explorer les données.

Une fois le Data Lake conçut, nous avons fait diverses expériences en Python sur nos données afin de distinguer des similitudes entre ces dernières.

L’ensemble des données et du code est accessible au repository GitHub à l’adresse suivante : [https://github.com/FeckNeck/data-lake-project](https://github.com/FeckNeck/data-lake-project%20).

## Données :

Comme décrit dans l’introduction, nous avons choisi des données de blog d’utilisateur que nous avons trouvé sur Kaggle. Nous extraits une dizaine de ligne car le fichier initial était trop volumineux ce qui nous a permis de concevoir le jeu de données suivant :

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Id | Gender | Age | Topic | Sign | Date | Text |
| 3581210 | male | 33 | InvestmentBanking | Aquarius | 05/08/2004 | … |
| 3539003 | female | 14 | indUnk | Aries | 05/06/2004 | … |
| 3539003 | female | 14 | indUnk | Aries | 11/08/2004 | … |
| 4172416 | female | 25 | indUnk | Capricorn | 08/08/2004 | … |
| 4030905 | female | 17 | Student | Aries | 30/07/2004 | … |
| 3705830 | male | 25 | Non-Profit | Cancer | 23/06/2004 | … |
| 4120194 | female | 17 | Arts | Aries | 18/08/2004 | … |
| 913315 | male | 25 | Communications-Media | Aquarius | 29/06/2004 | … |
| 1877178 | male | 16 | Student | Taurus | 01/04/2004 | … |
| 1107146 | female | 16 | Student | Libra | 25/05/2003 | … |

Tableau - Dataset

Le texte étant pour certain blog top long, nous avons décidé de ne pas l’ajouter dans le tableau mais il peut être visionné dans le fichier *data.csv*.

Ensuite, nous avons implémenté dans Neo4j notre dataset en créant pour chaque ligne un nœud ayant pour labels les mêmes caractéristiques que le tableau (Id, Gender etc…).

Cela nous a permis d’obtenir le graphe suivant :

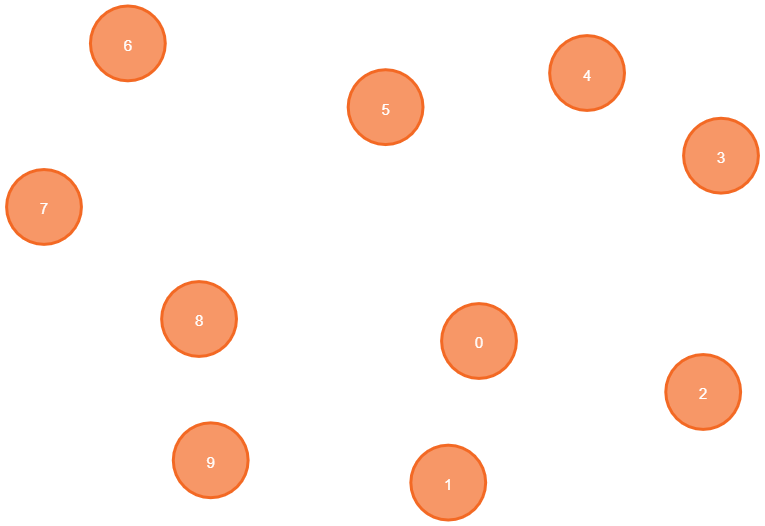


Figure - Graphe des blogs

## Métadonnées :

Pour modéliser la représentation des transformations de nos textes, nous avons choisi de concevoir un modèle similaire, peut être simplifié, de GoldModal.

Pour chaque texte, nous créons un nœud pour chaque transformation qui contient la modification apportée au texte. Un arc portant le nom de la transformation fait le lien entre la transformation et le texte.

Nous avons décidé de réaliser les 8 transformations suivantes :

* **Language Detection** : Détection de langue du texte
* **Stop words** : Les « stop words » (ou mot vide en français) sont des mots qui sont tellement commun qu'ils sont inutiles de les indexer ou de les utiliser dans une recherche. En français, des mots vides évidents pourraient être « le », « la », « de », « du », « ce ».
* **Word Count** : Comptage du nombre de mots.
* **Char Count** : Comptage du nombre de caractères.
* **Sentence Count** : Comptage du nombre de phrases.
* **Numerics** : Comptage du nombre de chiffre.
* **Sentiment** : Pour chaque texte nous estimons si ce dernier dégage un sentiment positif ou négatif représenté dans un vecteur entre -1 et 1 ou -1 exprime la négativité.
* **Subjectivity** : Pour chaque texte nous déterminons sa « subjectivité » qui est donné par un vecteur entre 0 et 1. Ou 0 représente une information factuelle et 1 est une opinion personnelle.

Pour chaque texte, nous réalisons en première transformation la détection de la langue puis la suppression des stop words. Ensuite, nous réalisons à partir de la transformation des stop words les autres cités plus hautes.

Cela donne la modélisation suivante :

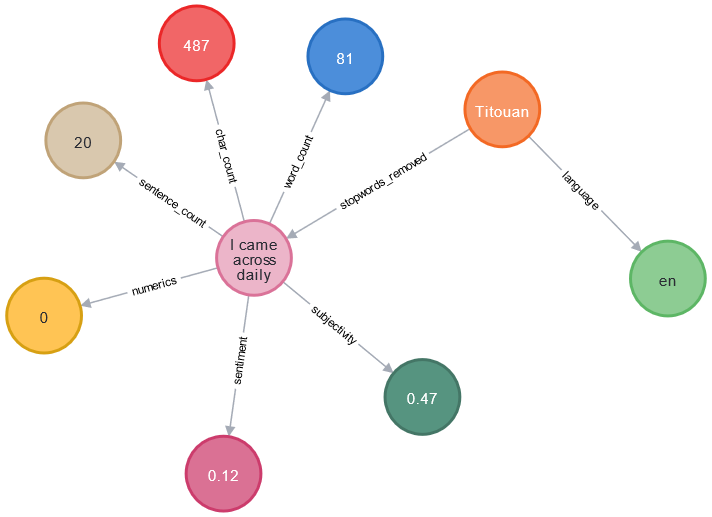


Figure - Représentation des transformations d’un texte

Nous avons réalisé les transformations en python dans le fichier *blog\_processing.py*, les résultats sont stockés dans le fichier *data\_proceseed.csv* où chaque ligne représente un texte et chaque colonne une transformation.

Pour automatiser la création des textes, des transformations et des arcs, nous avons réalisé un script en python (*init\_db.py*) qui, à partir des données prétraitées, écrit un script d’initialisation de la base de données dans un fichier txt (*init\_db.txt*).

Il suffit alors de copier le contenue du fichier et réaliser une requête d’initialisation de la base de données dans Neo4j.

Cela nous permet d’obtenir le vaste graphe des textes, de leurs transformations reliées par des arcs suivant :

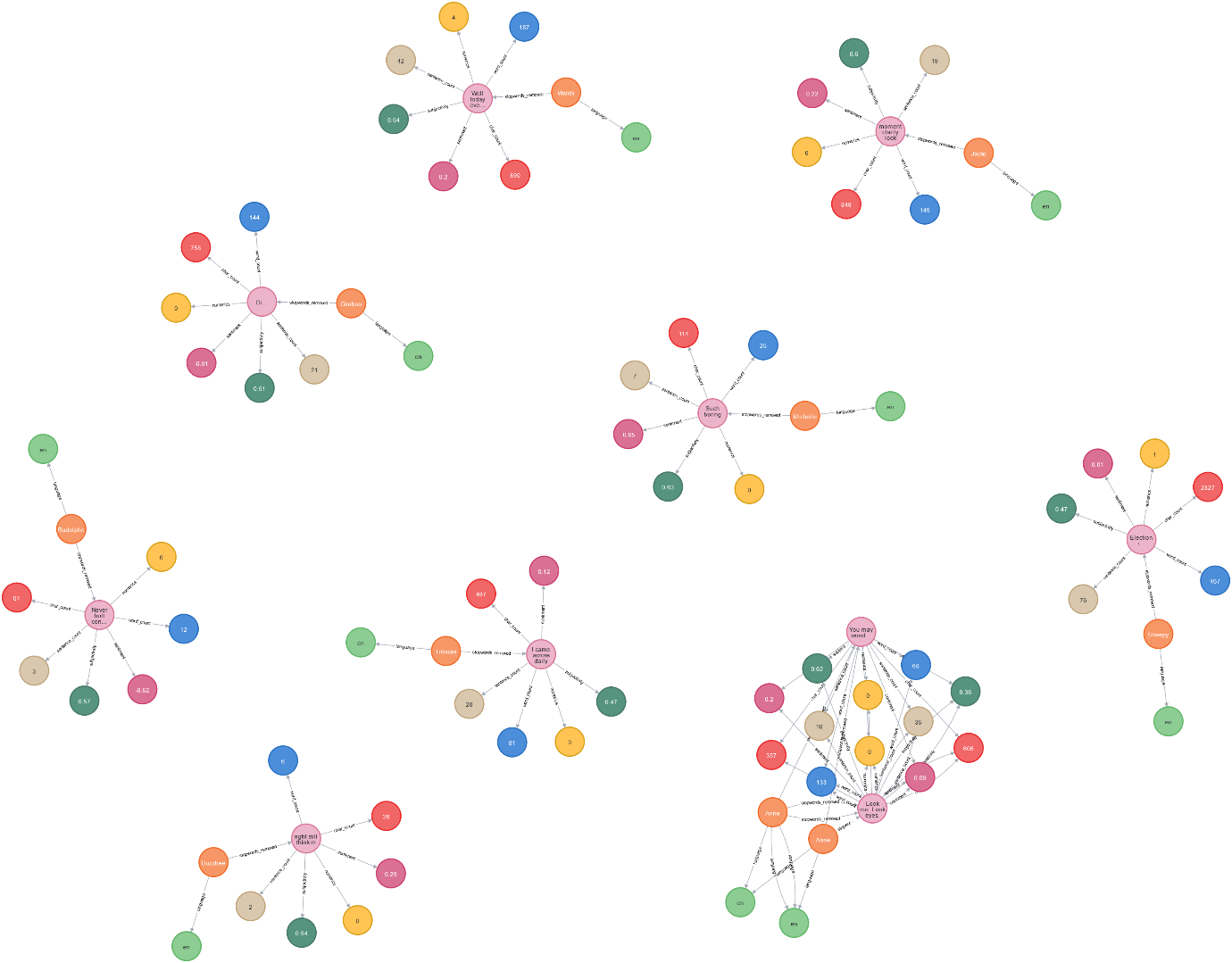


Figure - Vue d'ensemble du graphe

Sur le graph on peut voir que l’un des nœuds semblent avoir le double de transformations que les autres car c’est le même auteur qui a écrit 2 blogs.

Pour suivre le modèle GoldMedal, la prochaine étape a été de créer des groupes de similarités entre nos nœuds. Nous en avons créé 5 basés sur les critères suivants :

* Groupes de langues contenant les textes d’une même langue
* Groupe regroupant respectivement les textes qui ont des chiffres et ceux qui n’en ont pas.
* Groupe regroupant les textes qui dégagent un sentiment positif et ceux qui dégagent un sentiment négatif
* Groupe basé sur la subjectivité des textes, regroupant les textes semblant être factuels et ceux qui semblent être une opinion personnelle.

Chaque groupe est ensuite relié à ses transformations correspondantes (ex : le group « en » est relié aux transformations « language ») par un arc de valeur « yes » dans le cas ou la condition est vrai ou « no » dans le cas contraire.

Pour mieux se rendre compte, voici à quoi ressemble le groupe « is\_positive » :

A diagram of a number of circles

Description automatically generated

Figure - graph du nœud "is\_positive"

Enfin, voici à quoi ressemble le graph de l’ensemble des groupes (Le nœud du milieu étant « Grouping » le nœud reliant tous les groupes) :

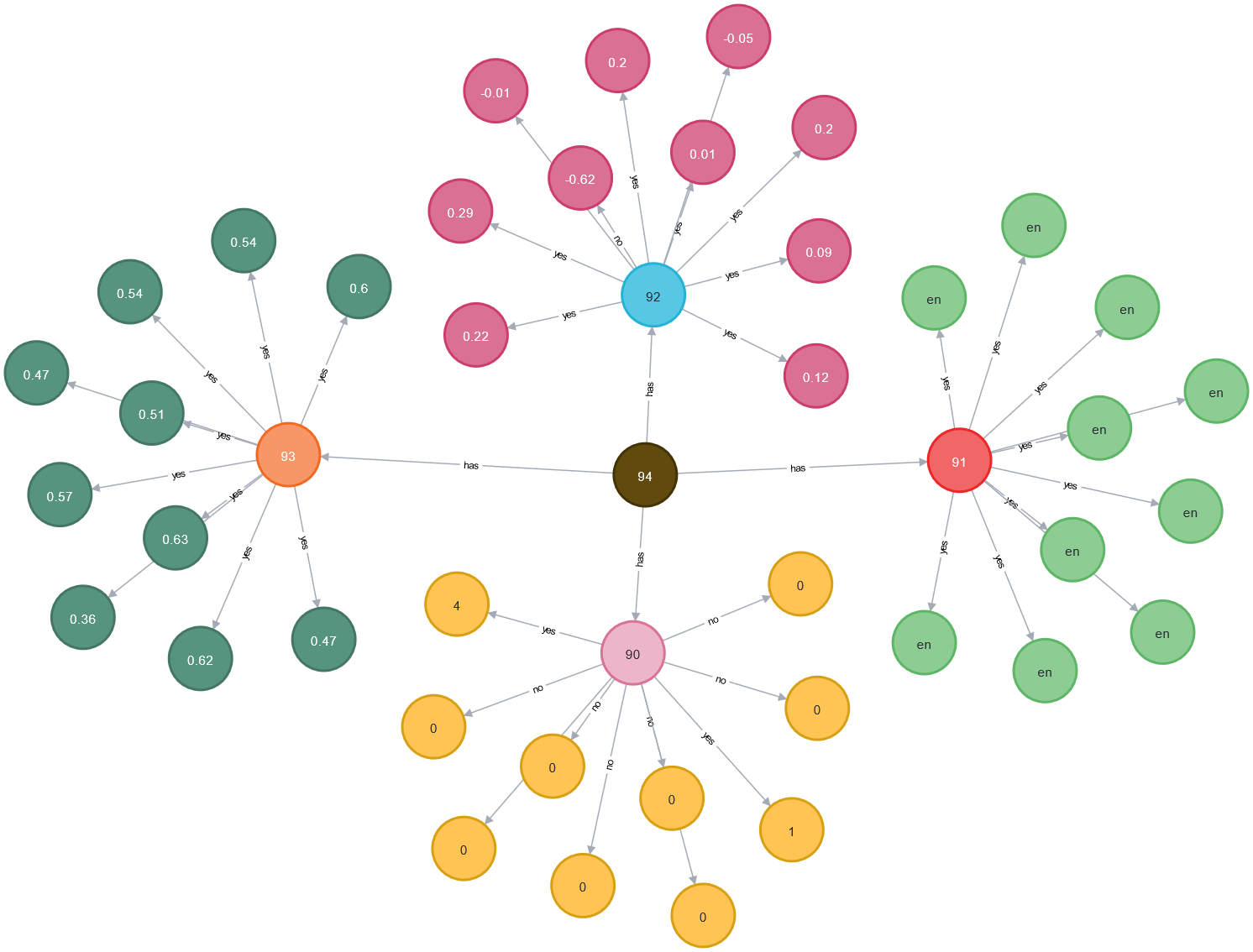


Figure - Graphe de l'ensemble des groupes et de leurs arcs.

La dernière étape a été de créée des arcs de similarités entre les métadonnées.

De la même façon que pour les groupes, nous regardons pour les transformations du texte qui nous intéresse les transformations qui sont similaire. Puis, nous les relions par un arc ayant pour valeur « similar ».

Cela nous permet d’obtenir le graphe **final** (sans les groupes) suivant :

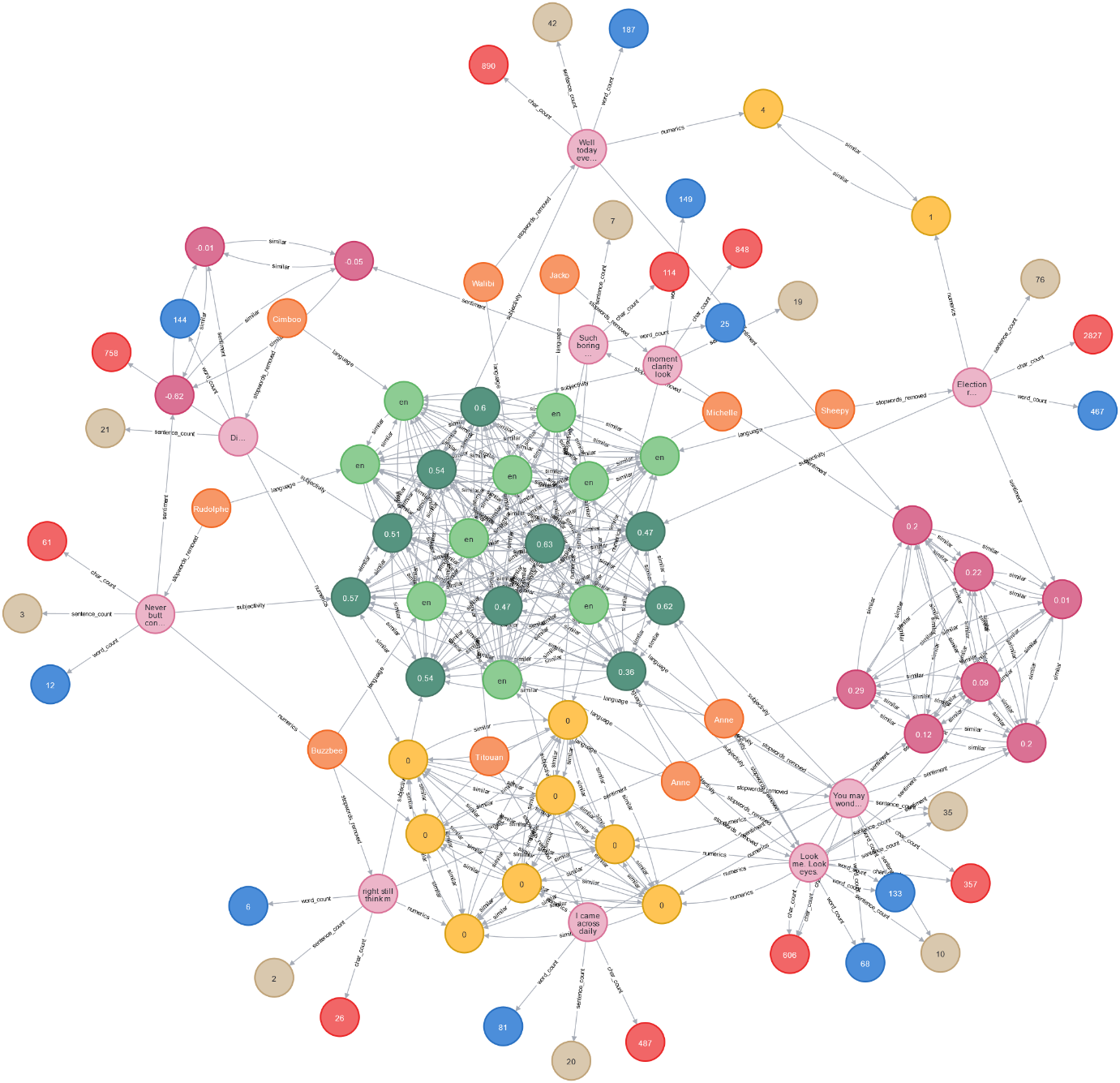


Figure - Graphe des textes, de leurs transformations ainsi que des liens de similarités

La dernière étape de la partie métadonnées a été de mettre en place un indexe inversé de notre lac de données en utilisant elastic search.

## Analyses :

Titou

## Conclusion :

Titou