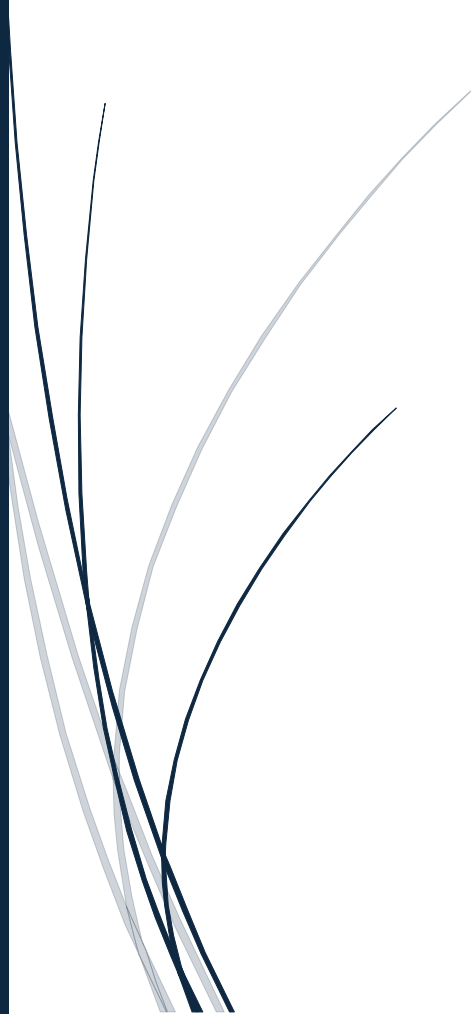
01/05/2024

Projet scala spark

Nettoyage, Analyse et Visualisation de données



Franklin ASSOGBA et Ossama CHIDMI

Table des matières

Présentation du projet 2
objectif 2
Lancer le projet 2
L'architecture du code 3
Les fichiers 3
Les dossiers 3
Les librairies 3
MLlib 3
Breeze et breeze-viz 3
Plotly 3
Le nettoyage 4
L'analyse 5
Les statistiques 5
La distributions des thèmes abordé dans les livres 6
La catégorisation des livres 7
Les visualisations 8
Le nombre de mot et phrases par livres 8
L'évolution du nombre de phrase et de mot par livre 8
L'évolution du nombre de phrase par livre 9
L'évolution du nombre de mot par livre 9
La distribution des sujets par livre 10
Le nombre de livre par catégorie 10
Le nombre d'apparition de chaque mot 11
Améliorations possibles 12

Présentation du projet

objectif

L'objectif de ce projet est de récupérer les données présentes dans 2 fichiers txt d'environ 2Go chacun afin de les analyser. Ces fichiers contiennent le texte brut de plusieurs livres à la suite de manière non structurée. Les fichiers peuvent être téléchargés à l'URL https://drive.google.com/file/d/16KCjV9z_FHm8LgZw05RSuk4EsAWPOP_z/view

Pour s'assurer de la qualité de notre analyse nous disposons de quelques références sur les informations à retrouver par notre analyse comme le nombre de livre et mots présents dans l'ensemble du jeu de donnée.

Une fois l'analyse terminée les résultats seront présentés sous forme de graphiques simples à interpréter.

Pour ce qui est des technologies le projet a été réalisé en utilisant le Template de développement disponible à l'URL https://gitlab.com/jsz4n/template-scala-spark qui permet de développer en scala en profitant des fonctionnalités de spark.

Lancer le projet

Le projet peut être lancé avec la commande **sbt run** depuis la racine

Pour améliorer le temps d'exécutions il ne lit que les X premières lignes des fichiers traités. Pour lever cette limite il suffit de retirer l'usage de **take** dans la méthode **clean_file** du fichier **pretraitement.scala** accessible depuis le chemin **projet_spark_scala\src\main\scala**

```
//On lit que les X lignes premières lignes du fichier
val text_in_single_row = file
.take(20000)
.reduce(_ + " " + _)
```

L’architecture du code

Pour ce qui est de la structure nous avons utilisé la base fourni par le Template et ajouté 3 fichiers scala assurer les différents traitements.

Les fichiers

SimpleApp.scala est notre fichier principal c’est dans celui-ci qu’on va préciser le chemin des fichiers à utiliser ainsi que faire appelle aux objets et méthodes déclaré dans les autres fichiers

Les fichiers suivants contiennent chacun des objets qui possèdent des méthodes qui permettront de réaliser le nettoyage, l’analyse et la visualisation.

- **Pretraitement.scala** Permet de normaliser et structurer le contenu des fichiers txt.
-
- **Analyse.scala** nous sert à retrouver les statistiques du jeu de données, telles que le nombre de livres. Estimer le nombre de catégorie de livres différentes. Il permet aussi de calculer la distributions des sujets dans les livres
-
- **Visualisation.scala** est responsable de créer les différents graphiques et les sauvegarder

Les dossiers

Les fichiers à traiter sont dans le dossier **bookcorpus** depuis la racine du projet. Les visualisation sont enregistrées dans le dossier **visualisation** depuis la racine du projet.

Les librairies

En plus des librairies incluse par défaut dans le Template nous avons téléchargé 3 librairies supplémentaires en les ajoutant au fichier build.sbt.

MLlib

MLlib permet de faire du machine learning nous l’avons utilisé pour estimer la distributions des thèmes. C’est à dire parmi les différents thèmes abordés dans les livres lesquelles sont les plus communs.

Breeze et breeze-viz

Breeze est une librairie spécialisée dans les calculs mathématique et breeze-viz permet de créer des représentations graphiques. Dans notre cas nous avons utilisé breeze pour permettre d’associer un numéro unique à chaque livre puis l’utiliser dans les visualisations car nous ne disposons pas des titres des livres.

Plotly

Ploty permet de faire des représentations graphiques. Contrairement à breeze-viz elle permet de créer des graphiques plus aboutis en donnant la possibilité de les enregistrer sous forme de page html sur lesquelles on peut afficher des valeurs spécifiques au survol de la souris, zoomer et comparer des valeurs. Nous l’avons utilisé dans les cas ou il n’était pas nécessaire de représenter les livres par des identifiants uniques.

Le nettoyage

Le nettoyage et l'uniformisation des données est assurée par la méthode **clean_file** de l'objet **pretraitement** . Elle retourne un dataframe qui contient sur chaque ligne le contenu d'un livre

Les données présentes dans les fichiers txt n'étaient pas structurées. De ce fait il est nécessaire de déterminer des délimiteurs entre les livres. En parcourant le jeu de donnée on constate que la fin d'un livre est souvent annoncé par ***isbn*** ou ***copyright XXXX***.

De ce fait nous avons utilisé ces termes comme séparateur de nos différents livres. Nous avons également uniformisé les données du texte en le convertissant entièrement en minuscule et en convertissant les caractères accentué en leurs équivalent sans accents. Nous avons aussi retiré les virgules pour qu'elles ne soient pas interprété comme faisant partie de certains mots

```
val text_cleaned = text_in_single_row
  // on convertit tout en minuscule
  .toLowerCase
    // on supprime la ponctuation sauf les points
    .replaceAll("[^a-zA-Z0-9\\.\\!\\?\\s]", "")
    // on remplace les caractères accentues
    .replaceAll("[àáâãäå]", "a")
    .replaceAll("[èéêë]", "e")
    .replaceAll("[ìíîï]", "i")
    .replaceAll("[òóôõö]", "o")
    .replaceAll("[ùúûü]", "u")
    .replaceAll("[ýÿ]", "y")
// les sépareateurs des livres isbn et copyright XXXX
val separators = "(isbn|copyright \\d{4})"
val text_split = text_cleaned.split(separators)
//on supprime les valeurs vides du dataframe
val df_book = spark.createDataFrame(text_split.map(Tuple1.apply)).toDF("book").na.drop()
```

Dans **SimpleApp.scala** on va appeler **clean_file** sur chacun des 2 fichiers à analyser puis combiner les 2 data frames obtenu pour obtenir un dataframe avec une colonne book qui contient l'entièreté des livres avec un livre par ligne.

```
val file_book_1 = "./bookcorpus/books_large_p1.txt"
val file_book_2 = "./bookcorpus/books_large_p2.txt"
val df_book_1 = Pretraitement.clean_file(spark, file_book_1)
val df_book_2 = Pretraitement.clean_file(spark, file_book_2)
val df_book_fuse =df_book_1.unionByName(df_book_2)
```

L'analyse

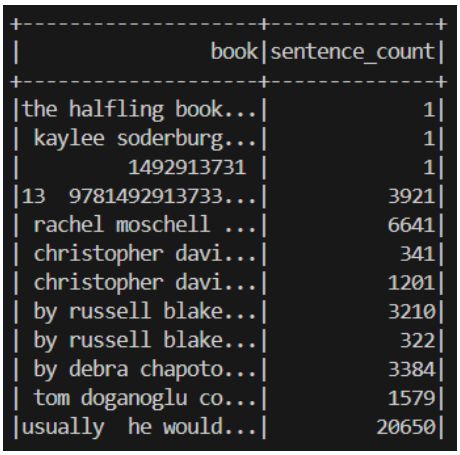
L'analyse est effectuée par les méthodes de l'objet **analyse**.

Les statistiques

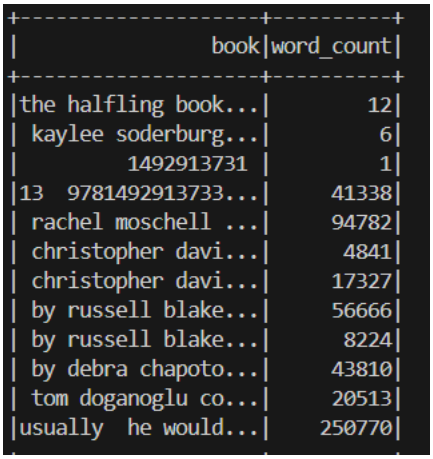
Le méthode **check_content** permet de calculer les informations suivantes

- Le nombre de phrase
- Le nombre de mot
- Le nombre de livre
- Le nombre de mots différents
- La moyenne du nombre de mot par phrase
- La médiane du nombre de mot par phrase
- Elle retourne ces informations sous forme de dataframe

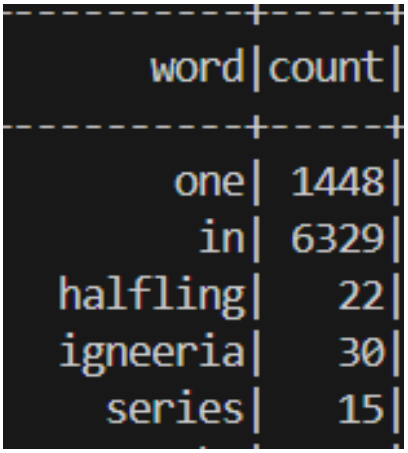
Avec les 20000 premières lignes des 2 fichiers on a



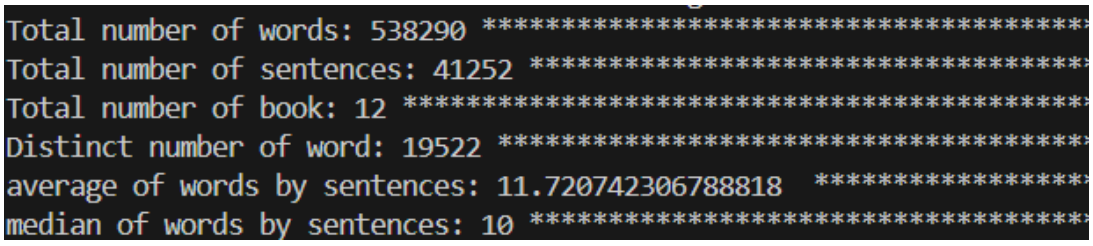
book	sentence_count
the halfling book...	1
kaylee soderburg...	1
1492913731	1
13 9781492913733...	3921
rachel moschell ...	6641
christopher davi...	341
christopher davi...	1201
by russell blake...	3210
by russell blake...	322
by debra chapoto...	3384
tom doganoglu co...	1579
usually he would...	20650



book	word_count
the halfling book...	12
kaylee soderburg...	6
1492913731	1
13 9781492913733...	41338
rachel moschell ...	94782
christopher davi...	4841
christopher davi...	17327
by russell blake...	56666
by russell blake...	8224
by debra chapoto...	43810
tom doganoglu co...	20513
usually he would...	250770



word	count
one	1448
in	6329
halfling	22
igneeria	30
series	15

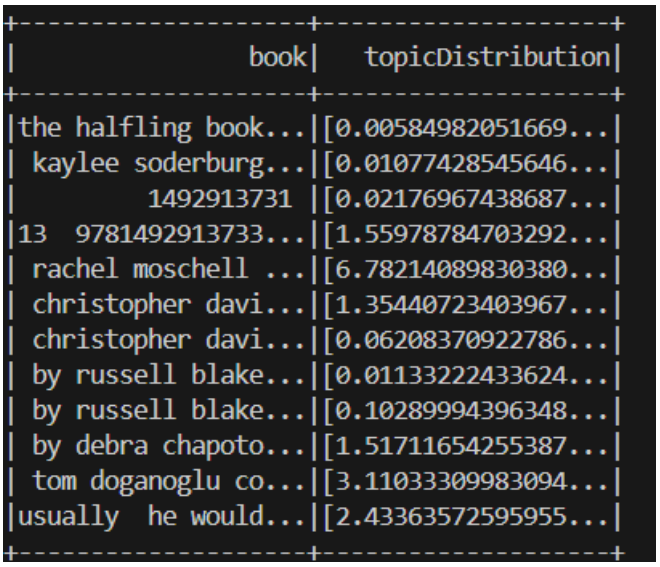


```
Total number of words: 538290 ****************************************
Total number of sentences: 41252 ****************************************
Total number of book: 12 ****************************************
Distinct number of word: 19522 ****************************************
average of words by sentences: 11.720742306788818  ****************
median of words by sentences: 10 ****************************************
```

La distributions des thèmes abordé dans les livres

La méthode **get_topic** de l'objet **analyse** se charge de classifier les livres en fonction des sujets qu'ils abordent

Pour savoir les différents sujets abordé dans chaque livre on à utiliser les fonctionnalités de traitements de langages naturels de MLlib. Pour cela on a utilisé un modèle d'Intelligence artificiel LDA qui permet de classifier chacun des livres à des thèmes. Etant donnée que le jeu de donnée ne contenait pas le sujet principal de chaque livre de façon structuré nous n'avons pas entrainé le modèle en se basant sur un jeu de donnée d'exemple puis tester son efficacité en utilisant un jeu de donnée de validation. La méthode de LDA va associer à chaque livre un vecteur avec 3 poids différents. La fonction prend en paramètre le nombre de sujets différents à utiliser pour faire la classification dans le code. Dans notre cas on a utilisé 15 pour identifier les 15 sujets les plus récurrents par exemple.



book	topicDistribution
the halfling book...	[0.00584982051669...
kaylee soderburg...	[0.01077428545646...
1492913731	[0.02176967438687...
13 9781492913733...	[1.55978784703292...
rachel moschell ...	[6.78214089830380...
christopher davi...	[1.35440723403967...
christopher davi...	[0.06208370922786...
by russell blake...	[0.01133222433624...
by russell blake...	[0.10289994396348...
by debra chapoto...	[1.51711654255387...
tom doganoglu co...	[3.11033309983094...
usually he would...	[2.43363572595955...

Cette représentation n'est pas évidente à interprété et sera explicitée dans la partie visualisation.

La catégorisation des livres

Pour estimer le type des livres on regarde le nombre de fois où certains mots apparaissent. Par exemple si le champ lexical de la cuisine est plus fréquent que les autres on le considère comme étant un livre de cuisine.

Cette classification est faite par la méthode **guessTopic** de l'objet **analyse.** La fonction contient une liste de mot clé pour chaque genre de livre

```
def guessTopic(df: DataFrame, bookColumn: String): DataFrame = {
    val genres_keywords = Map(
        "Fantasy" -> Seq("magic", "wizard", "fantasy", "dragon"),
        "Mystery" -> Seq("detective", "crime", "murder", "mystery"),
        "Romance" -> Seq("love", "romance", "heartbreak"),
        "Science Fiction" -> Seq("space", "alien", "robot", "future"),
        "Thriller" -> Seq("suspense", "thriller", "danger", "tension"),
        "Horror" -> Seq("horror", "scary", "fear", "haunted"),
        "Historical Fiction" -> Seq("historical", "era", "past", "period"),
        "Adventure" -> Seq("adventure", "journey", "quest", "explore"),
        "Poetry" -> Seq("poem", "verse", "rhyme", "stanza"),
        "Drama" -> Seq("drama", "theater", "stage", "act"),
        "Biography" -> Seq("biography", "life", "autobiography", "memoir"),
        "Autobiography" -> Seq("autobiography", "life story", "memoir", "journey"),
        "Self-Help" -> Seq("self-help", "improve", "personal growth", "success"),
        "Philosophy" -> Seq("philosophy", "thought", "theory", "belief"),
        "Business" -> Seq("business", "management", "entrepreneurship", "leadership"),
        "Travel" -> Seq("travel", "journey", "explore", "destination"),
```

Le data frame retourné contiendra le nombre de livre associé à chaque genre de livre

```
+--------+-----+
|   topic|count|
+--------+-----+
|  Poetry|    7|
| Unknown|    3|
|Thriller|    2|
+--------+-----+
```

Les visualisations

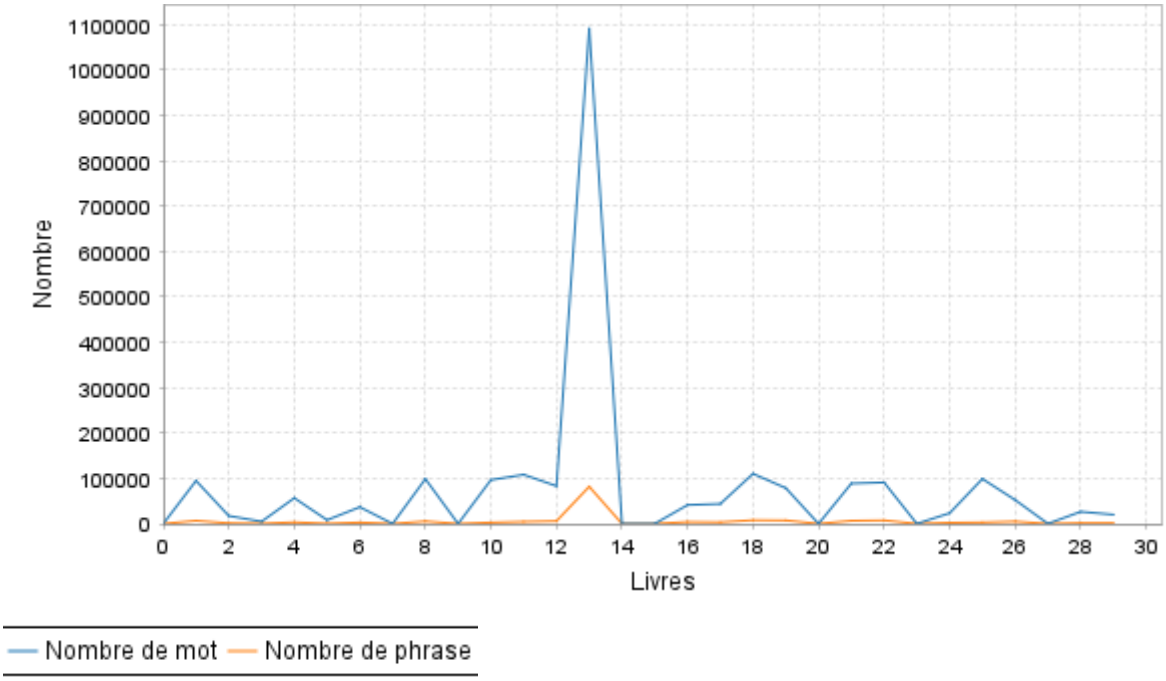
La visualisation a été faites avec les 80000 premières lignes des 2 fichiers

Le fichier **Visualisation.scala** comprend 2 objets **Visualisation_breeze** et **Visualisation_ploty** pour éviter les erreurs d'interprétation la méthode Plot ayant un fonctionnement différents dans les 2 librairies

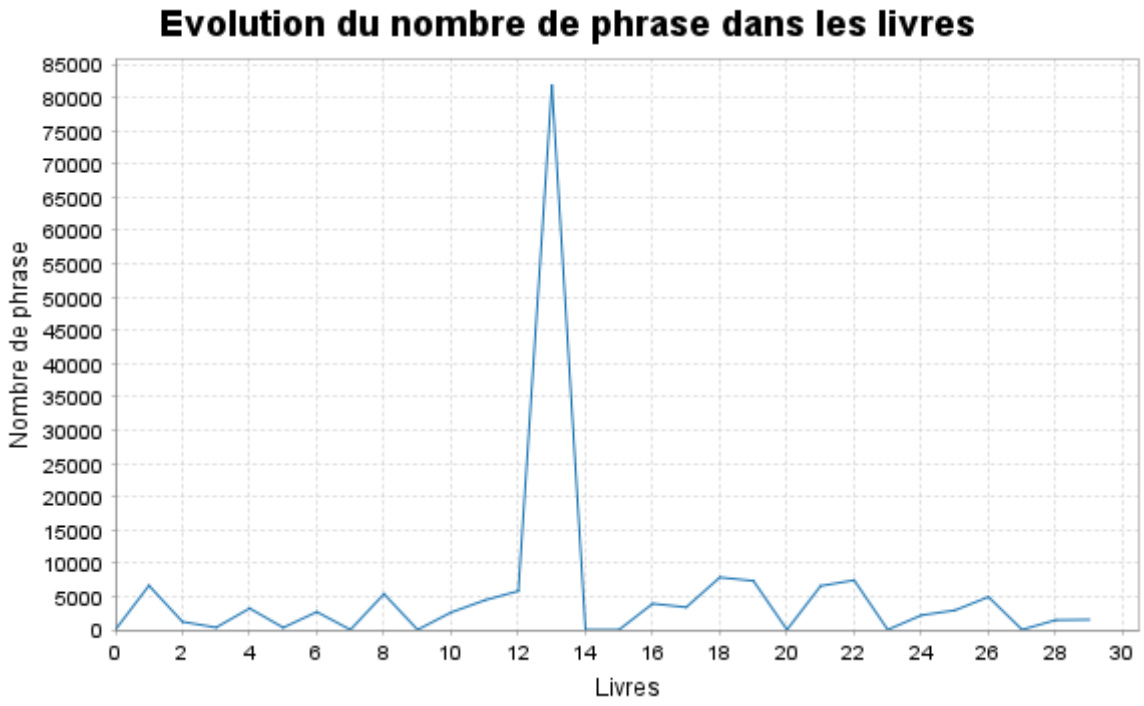
Le nombre de mot et phrases par livres

Comme on a utilisé breeze pour ajouter un id à chaque livre on se sert de breeze-viz pour visualiser ces informations. On constate que le 13ème livre a est celui avec le plus de contenu sur cette partie du dataset

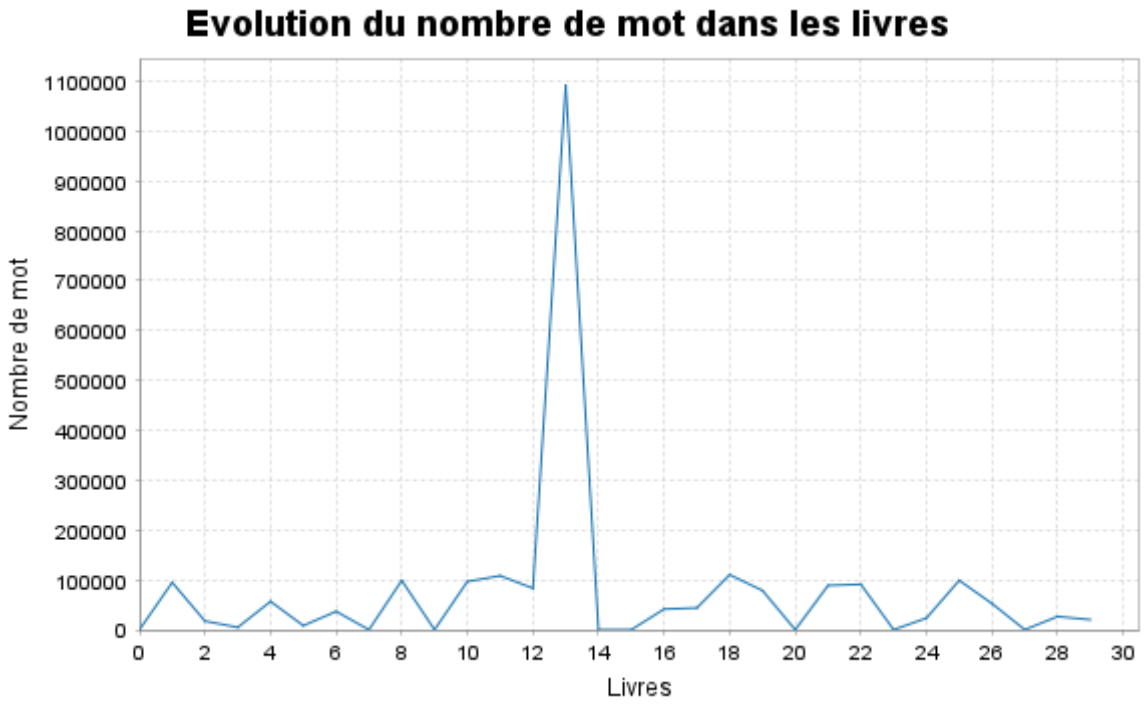
L'évolution du nombre de phrase et de mot par livre



L’évolution du nombre de phrase par livre

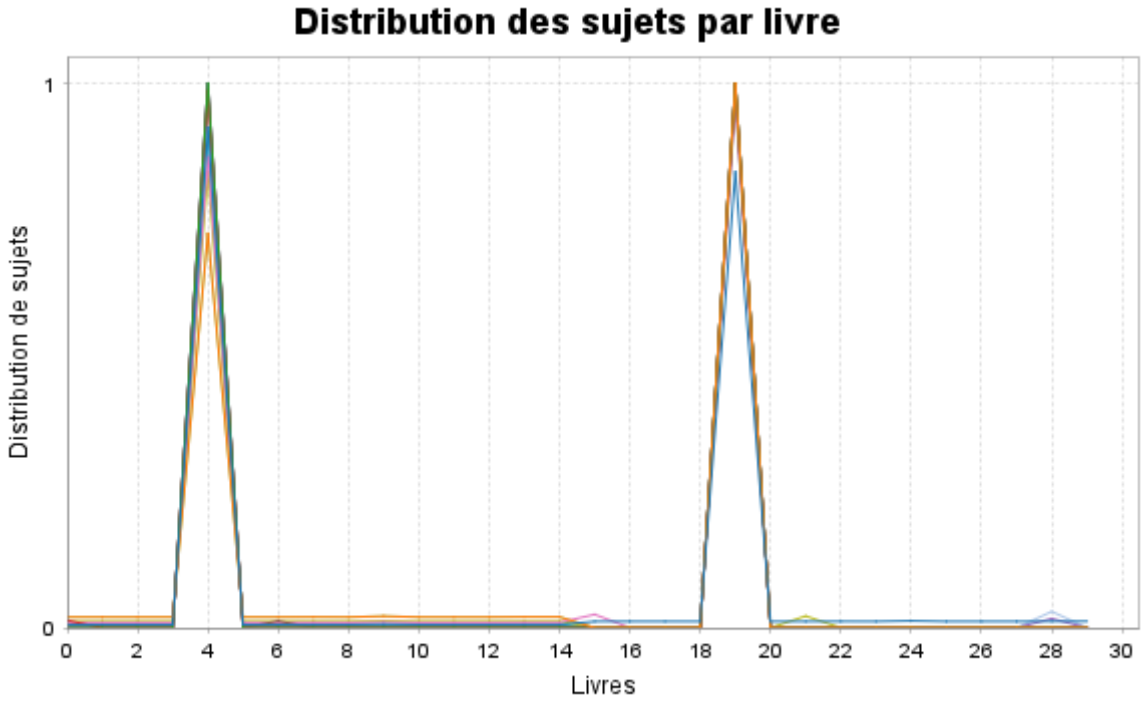


L’évolution du nombre de mot par livre



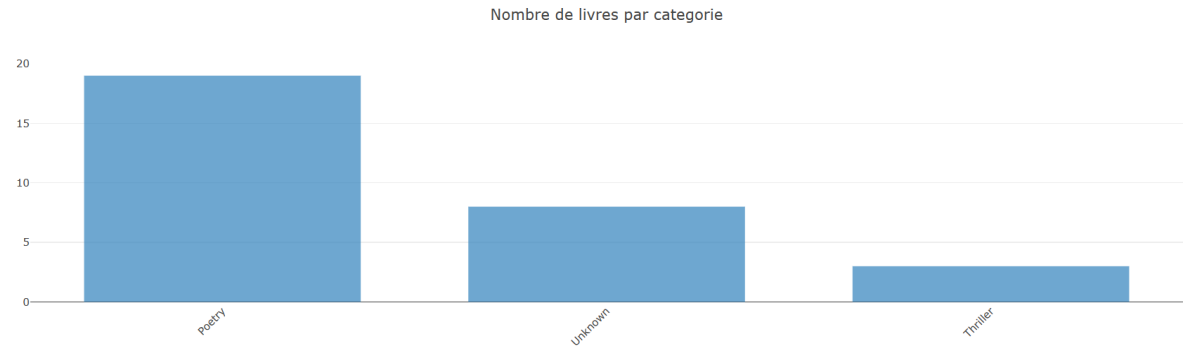
La distribution des sujets par livre

Avec 15 sujets différents à identifier on constate que 2 sujets sont majoritairement abordé représenté par les couleurs bleu et orange.



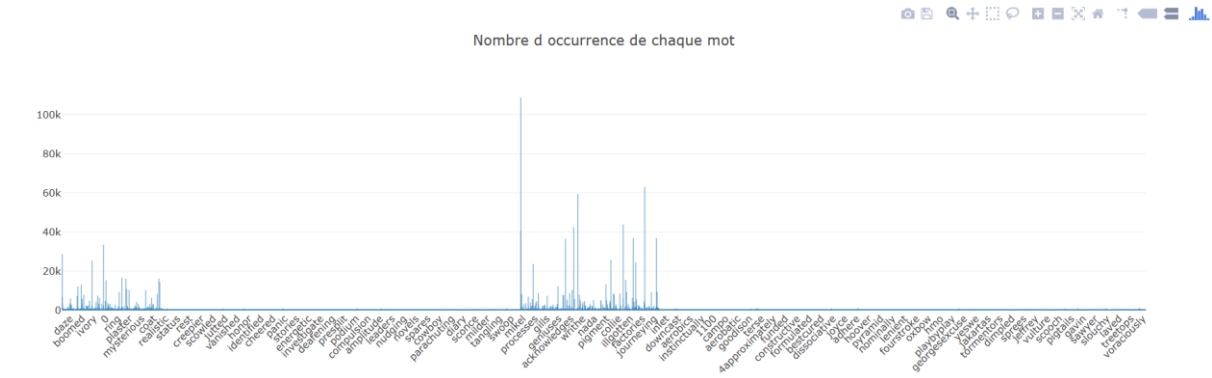
Le nombre de livre par catégorie

19 livres ont été identifié comme étant des poèmes ou poésie. 3 autres comme des thriller et les 8 autres n'ont pas de suffisamment de mots clé pour être classé dans une des catégories



Le nombre d’apparition de chaque mot

On constate que le mot le plus utilisé est **fidelity**.



Améliorations possibles

Scala et spark étant des technologies que nous avons très peu utilisé en pratique auparavant, nous nous sommes avant tout focaliser sur l'obtention des résultats plutôt que sur la manière la plus efficace de les obtenir. Il existe probablement des fonctions natives ou des librairies qui permettent de simplifier davantage les traitement que nous avons effectué. Il est aussi important de se renseigner sur les bonne pratique, nous avons utilisé des Objets avec chacun leurs méthodes respectives pour davantage de simplicité au départ mais dans la maintenance et l'évolution d'un projet utiliser des classes et davantage d'abstraction aurait été plus judicieux.

Il est également possible d'enrichir le projet en proposant davantage de type de visualisation la réalisation nuage de mot avait été testée mais abandonnée faute de temps pour corriger un problème d'installation de librairie.