Synthèse des TP 1 - 4

- **YOUCEF KHODJA Amine 21113585**
- **KEMICHE Koceila 21114731**

Recherche d'Information et Traitement Automatique du Langage

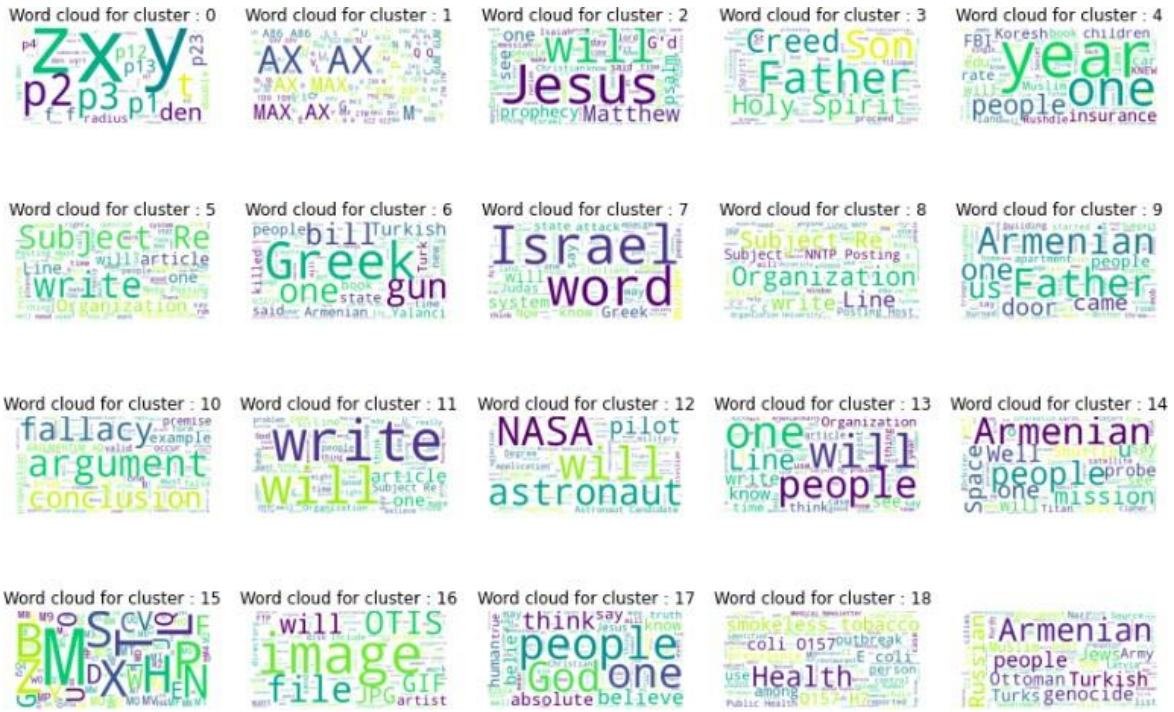
RITAL

1.1 Tutoriel et Bag of Words

Dans ce tp , nous avons pris en main les outils de base pour la traitement automatique du langage , notamment le bag of words BoW.Grâce à la base de données fournie nous avons pu tester différent Vectorizers en faisant varier leurs paramètres et en comparant les performances de ces derniers sur des modèles de machine learning. Nous avons vu aussi les principales différences entre le CountVectorizer et le TfIdfVectorizer .
Nous avons constaté aussi que la partie prétraitements des données est trés importante avant d'entamer le Bag of Words et le machine learning c'est pour cela que nous avons tester différents pré-traitements et l'un des preprocessing les plus performants était celui où on supprimait les caractères spéciaux , les chiffres et on faisait un stemming sur les documents de la base .
Enfin , on apparaît à afficher un certain nombre de features (les plus importantes) sur les vectorizer sous forme de wordcloud .

1.2 Clustering

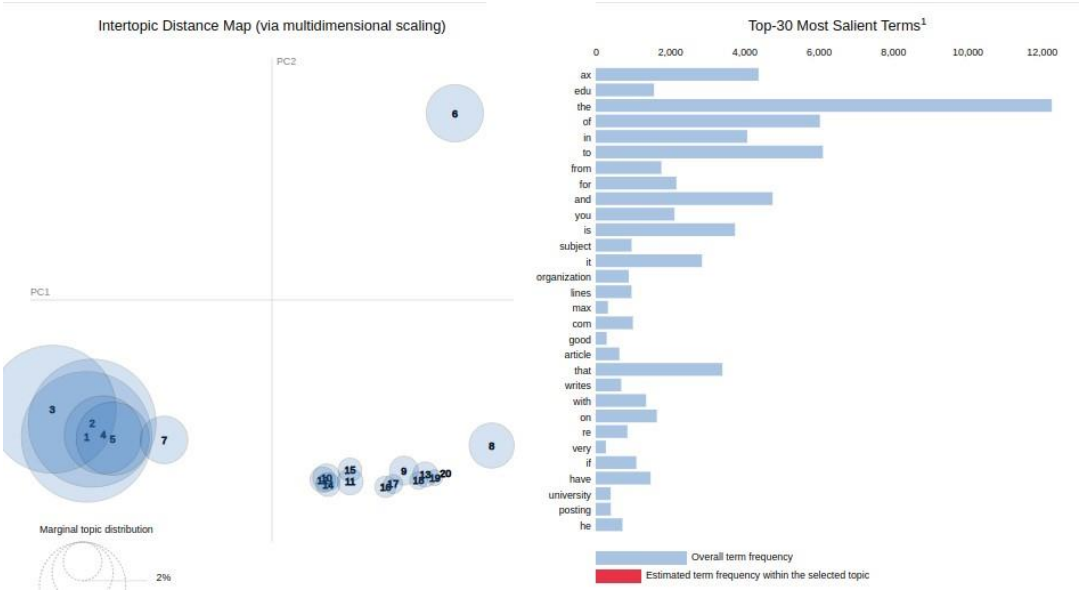
Le but de cette partie , étant de pouvoir représenter , comprendre et exploiter des documents non étiquetés . Nous étudions trois approches différentes : KMeans , Latent Semantic Analysis (LSA) , Latent Dirichlet Allocation (LDA) et en faisant des comparaisons selons des métriques qualitatifs et quantitatifs telles Rand Score , pureté et Adjusted Rand Score. KMeans étant facile à implémenter nous donne des clusters plus moins cohérents malgré sa sensibilité au choix initial des controids des clusters et par l'existence de valeurs aberrantes . LSA est plus adapté à ce contexte de traitement du langage naturel car il réduit la dimensionnalité de l'espace des documents mais reste sensible au bruits c'est pour cela qu'une étape de prétraitements peut être envisagée avant de l'exécuter. LDA nous a permis de découvrir des topics et des thèmes cachés dans le corpus de documents . Ainsi , chacun des algorithmes nous a semblé intéressant d'un côté et un peu moins de l'autre c'est pour cela que leurs utilisations est judicieuses et dépendent de problèmes à traiter.



```
 KMEAN
Purete moyenne des clusters :  0.8388676359872012
Rand Score :  0.3456096096096096
Adjust Rand Score :  0.004585756256910023


 LSA
Purete moyenne des clusters :  0.8388676359872012
Rand Score :  0.7543503503503504
Adjust Rand Score :  0.018927054103206803


 LDA
Purete moyenne des clusters :  0.8388676359872012
Rand Score :  0.8971531531531531
Adjust Rand Score :  0.09029709581759614
```



1.3 Séquences

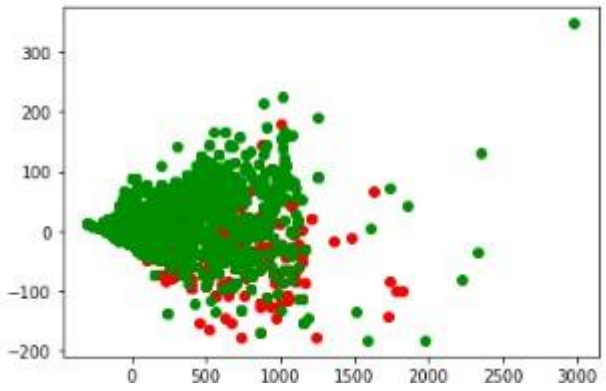
Dans cette partie de TP , nous avons manipulé les modèles séquentiels dans le contexte du traitement automatique du langage . Nous avons commencé par créer un modèle qui se base sur Part Of Speech (POS) pour l'études des séquences pour un ensemble de phrases. Ensuite nous testons un modèle probabiliste qui est le Hidden Markov Model (HMM) qui lui donne plus au moins un résultat un peu plus performant que le précèdent.
Nous passons ensuite à des modèles discriminatifs tels Conditional Random Fields (CRF) qui lui donne des performances plus intéressantes vu qu'il se base sur le principe que la sortie de chaque élément d'un séquence dépend non seulement de ses caractéristiques d'entrée mais aussi des sorties de ses voisins dans la séquence.

1.4 Embeddings Vectoriels et classification

Dans ce TP, nous nous intéressons au problème de vectorisation des documents et la capacité de capturer de l'information sémantique dans corpus . Contrairement au BoW qui lui code les

documents en matrice sparse , nous coderons dans ce tp du texte en un vecteur dense de petite taille . La première approche que nous testons c'est d'encoder les mots en vecteurs (Word2Vec) en utilisant Continuous Bag Of Word (CBOW) et Skip Gram (SG) ainsi chaque document peut représenter par une agrégation (par somme , min , max ..) des différents embeddings vectoriels de ses mots. Nous constatons aussi qu'une agrégation de ces embeddings vectoriels avec une Tf-Idf donne une représentation plus exacte du document concerné. L'inconvénient de Word2Vec est cette agrégation qui elle reste pas très représentative du document contrairement au Doc2Vect.
Nous avons remarqué aussi une légère augmentation des performances avec le (Fastext) qui lui se base sur une représentation n-gramme en caractère pour un mot donné ,ainsi un mot serra une somme de vecteurs de n-gramme caractères qui compose ce mot. La représentation doc2Vect nous a donné de bonnes performances aussi vu qu' il se base sur une représentation complète du document et évitant l'agrégation des vecteurs de mots et en prenant en compte le contenus et l'ordre d'apparitions des mots.

```
Out[24]: <matplotlib.collections.PathCollection at 0x7f179c5aaf40>
```



1.5 Embeddings Vectoriels et séquences

L'utilisation des embeddings vectoriels pour pouvoir utiliser les modèles séquentiels est plus intéressante car elle nous a permis de capturer de l'information sémantique contrairement au PoS model par exemple et permet aussi de modéliser l'ordre des mots et des phrases ce qui a augmenter d'avantages les performances de nos modèles de machine learning.

1.6 Réseaux de neurones récurrents

Nous avons conçu dans le TP un réseau de neurones récurrents pour apprendre à générer du texte en utilisant une couche d'embedding qui transforme chaque caractère du texte en un vecteur dense , un couche récurrente qui permet de prendre en compte la séquence des embeddings en entrée et de maintenir une mémoire interne et finalement une couche de prédiction qui sert à transformer le dernier état caché en une prédiction pour le caractère suivant .

1.7 Transformers

Dans ce TP , en utilisant BERT on obtient des performances assez bonnes et le modèle est capable de prendre en compte la sémantique et la syntaxe des phrases ce qui le rend plus performant que les BoWs .