Université de Sousse

Institut Supérieur d'Informatique et de Techniques de Communications-Hammam Sousse

Projet Fouille de Données :

Classification et Regroupement des Tweets

Realisé par: Bedhief Safwa

U.A :2020/2021

I. Objectifs :

- Maitriser l'API de twitter pour l'extraction des tweets
- Maitriser la partie NLP (Natural Language Processing) avec NLTK en Python

  Appliquer les principes de nettoyage des données
- Classer les tweets : regrouper ensemble les tweets qui sont similaires.

II. Traitement du Langage Naturel :

1. Qu'est-ce que NLP

1.1. Définition

Le traitement du langage naturel est une branche de l'intelligence artificielle qui aide les ordinateurs à comprendre et manipuler le langage humain. C'est une technologie dont le but est de faire communiquer les ordinateurs et les gens au même niveau.

La NLP combine la puissance de la linguistique et de l'informatique pour étudier les règles et la structure du langage, et créer des systèmes intelligents capables de comprendre et d'analyser le sens du texte et de la parole à travers des algorithmes.

1.2. Les types de NLP

La NLP semble être une chose très professionnelle, mais elle est très populaire. Vous pouvez rencontrer un système de traitement du langage naturel dans votre vie quotidienne sans vous en rendre compte.

- ➢ Répondre aux questions
- ➢ Reconnaissance vocale
- ➢ Traduction d'un langage à une autre
- ➢ Analyse de sentiment/ d'opinion
- ➢ Classification de texte

1.3. Les avantages de NLP

- ✓ Analyse à grande échelle : Le traitement du langage naturel aide les machines à comprendre et à analyser automatiquement de grandes quantités des données textuelles non structurées, telles que

les commentaires sur les réseaux sociaux, les tickets d'assistance client, les commentaires en ligne, les reportages, etc.

- ✓ Processus automatisés en temps réel : Les outils de traitement du langage naturel peuvent aider les machines à apprendre à trier et à acheminer les informations rapidement, efficacement, avec précision et 24 heures sur 24, sans aucune interaction humaine.
- ✓ Adapté à votre industrie : Les algorithmes de traitement du langage naturel peuvent être personnalisés en fonction de vos besoins et de vos normes, tels que les langages complexes spécifiques à l'industrie, même la satire et les mots mal utilisés.

1.4. Les techniques de NLP

Les deux principales techniques utilisées pour le traitement naturel du langage sont :

- ❖ **L'analyse syntaxique** : Identifie la structure syntaxique d'un texte et les relations de dépendance entre les mots, représentées sur un diagramme appelé arbre d'analyse.
- ❖ **L'analyse sémantique :** Les tâches sémantiques analysent la structure des phrases, les interactions de mots et les concepts associés, dans le but de découvrir le sens des mots et de comprendre le sujet d'un texte.

2. Le fonctionnement de traitement du langage naturel

En raison des nombreuses nuances et incohérences du langage humain, la NLP est très difficile. Déterminer comment saisir ces nuances et le contexte qui les entoure dans le langage et contraint des ordinateurs n'est pas facile.

Alors, comment fonctionne ce traitement ?

- **Prétraitement des données :** Les données doivent être nettoyées et annotées pour pouvoir être traitées par des algorithmes. Le nettoyage consiste généralement à déconstruire les données en mots ou en morceaux, à supprimer les parties du discours qui n'ont pas de signification (par exemple, StopWords comme la, du, ces, etc.), à standardiser les données (par exemple, changer tous les mots en minuscules) et grouper les mots en catégories prédéfinies, telles que le nom de la personne. Tout cela peut être fait en utilisant la bibliothèque spaCy et/ou NLTK en Python. L'annotation se résume à regarder les mots environnants et à utiliser des règles de langage ou des statistiques pour marquer une partie du discours.
- **Vectorisation :** Après le prétraitement, les données non numériques seront converties en données numériques, car les ordinateurs fonctionnent beaucoup plus rapidement sur

les vecteurs. Cela signifie que la vectorisation de texte peut être utilisée dans divers contextes sans avoir à reconstruire à chaque fois un modèle spécifique à une collection.

- **Test :** Une fois la base de référence créée, la validation est utilisée pour tester la précision de sa prédiction. La validation croisée est une technique de validation de modèle qui divise les données en sous-ensembles d'apprentissage et de test.

3. Les taches d'analyse de données

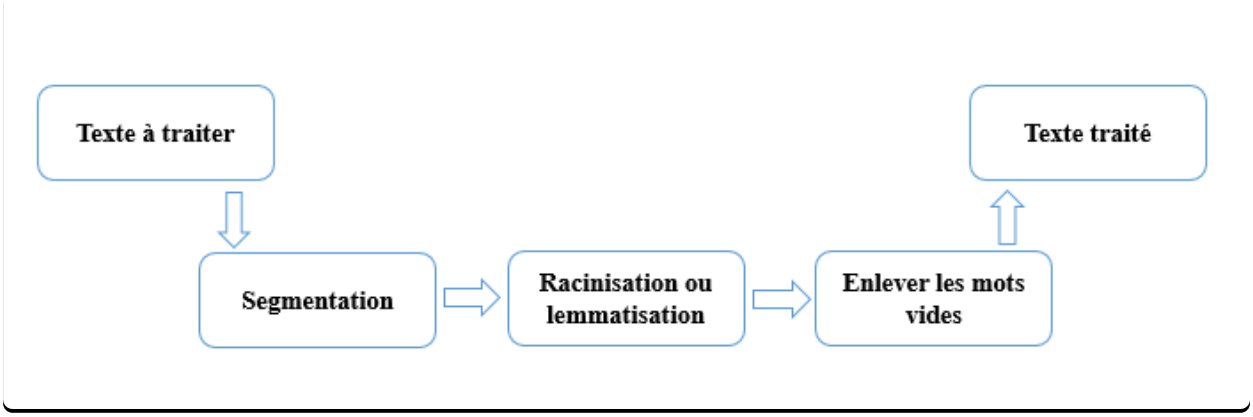
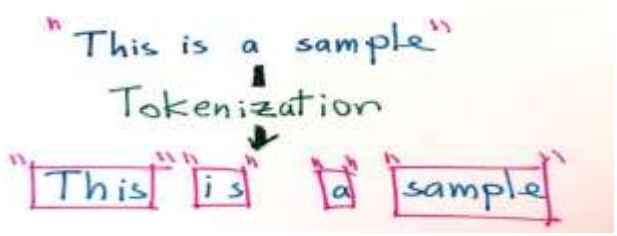


Figure 1 : tache d'analyse du texte

- Segmentation (Tokenisation) : Pour expliquer la segmentation dans le traitement automatique du langage, nous redéfinissons le concept de « mot » pour inclure le concept d'unités lexicales de base. Donc à partir de maintenant, un mot peut aussi représenter des caractères tels que ".", ",", "l'", etc.

Le texte dans son format d'origine peut être considéré comme une simple séquence de caractères. La segmentation comprend la reconnaissance des mots de cette séquence afin de les convertir ensuite, cela devient une liste de mots.

Par exemple :



Une fois que vous avez la liste de mots, la modification de ceux-ci peut être une étape utile.

- Racinisation ou lemmatisation (Lemmatisation et Stemming) : Cette étape n'est pas toujours nécessaire et dépend largement On veut terminer. Le but de la lemmatisation est de minimiser Des mots avec le même sens. Par exemple, vous souhaiterez peut-être remplacer Les mots «lenteur», «ralentir» où «lentement» sont composés de leur radical «lent». La même idée peut être appliquée à différents mois de l'année (utilisez "mois" au lieu de "juin" ou "juillet") ou nombre (remplacez "4" ou "7" par "num"). Donc nous utilisons juste un moyen d'aider à sélectionner des tâches en supprimant les informations non pertinentes c'est inutile. De même, une autre méthode de prétraitement couramment utilisé consistée à enlever la majuscule du début de phrase et de garder une majuscule aux noms propres.

- Enlever les mots vides (stop words) : Comme la lemmatisation, ce prétraitement est optionnel et son utilisation dépend de nombre des tâches à accomplir. Les mots vides font référence à aucune signification et sont généralement les plus courants dans le texte car par exemple "le", "le" ou "de".

  Vous pouvez également décider de créer une liste de mots vides utilisés pour des domaines spécifiques. Pour certaines tâches, comme la classification pour les documents, les mots vides peuvent être nuisibles et leur suppression peut aider.

III. Algorithme K-means :

1. Qu'est-ce que le clustering

Le clustering est une méthode d'apprentissage non supervisé (unsupervised Learning). Ainsi, on n'essaie pas d'apprendre une relation de corrélation entre un ensemble de features d'une observation et une valeur à prédire , comme c'est le cas pour l'apprentissage supervisé. L'apprentissage non supervisé va plutôt trouver des patterns dans les données. Notamment, en regroupant les choses qui se ressemblent.

Il existe deux types de clustering :

- Le clustering hiérarchique
- Le clustering non-hiérarchique (partitionnement)

2. Qu'est-ce que K-means

K-means est un algorithme non supervisé de clustering non hiérarchique. Il permet de regrouper en clusters distincts les observations du data set. Ainsi les données similaires se retrouveront dans un même cluster. Par ailleurs, une observation ne peut se retrouver que dans un cluster à la fois (exclusivité d'appartenance). Une même observation, ne pourra donc, appartenir à deux clusters différents.

Notion de similarité

Pour pouvoir regrouper un jeu de données en cluster distincts, l'algorithme K-Means a besoin d'un moyen de comparer le degré de similarité entre les différentes observations. Ainsi, deux données qui se ressemblent, auront une distance de dissimilarité réduite, alors que deux objets différents auront une distance de séparation plus grande.

Les littératures mathématiques et statistiques regorgent de définitions de distance, les plus connues pour les cas de clustering sont :

- **La distance Euclidienne**

$$d(x_1, x_2) = \sqrt{\sum_{j=1}^{n} (x_{1n} - x_{2n})^2}$$

- **La distance de Manhattan (taxi-distance)**

$$dist(X_i, X_j) = \sum_{k=1}^{l} \left| x_{i,k} - x_{j,k} \right|$$

- **Distance de Jaccard**

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

3. Cas d'utilisation K-means

K-Means en particulier et les algorithmes de clustering de façon générale ont tous un objectif commun : Regrouper des éléments similaires dans des clusters. Ces éléments peuvent être tous et n'importe quoi, du moment qu'ils sont encodés dans une matrice de données.

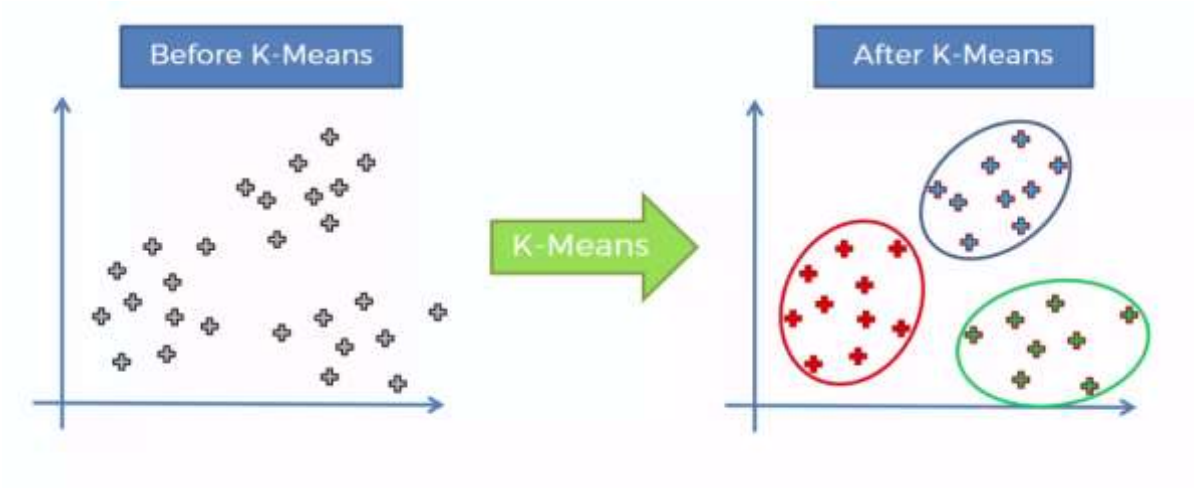
Les champs d'application de K-Means sont nombreux, il est notamment utilisé en :

- La segmentation de la clientèle en fonction d'un certain critère (démographique, habitude d'achat etc.)

- Utilisation du clustering en Data Mining lors de l'exploration de données pour déceler des individus similaires. Généralement, une fois ces populations détectées, d'autres techniques peuvent être employées en fonction du besoin.

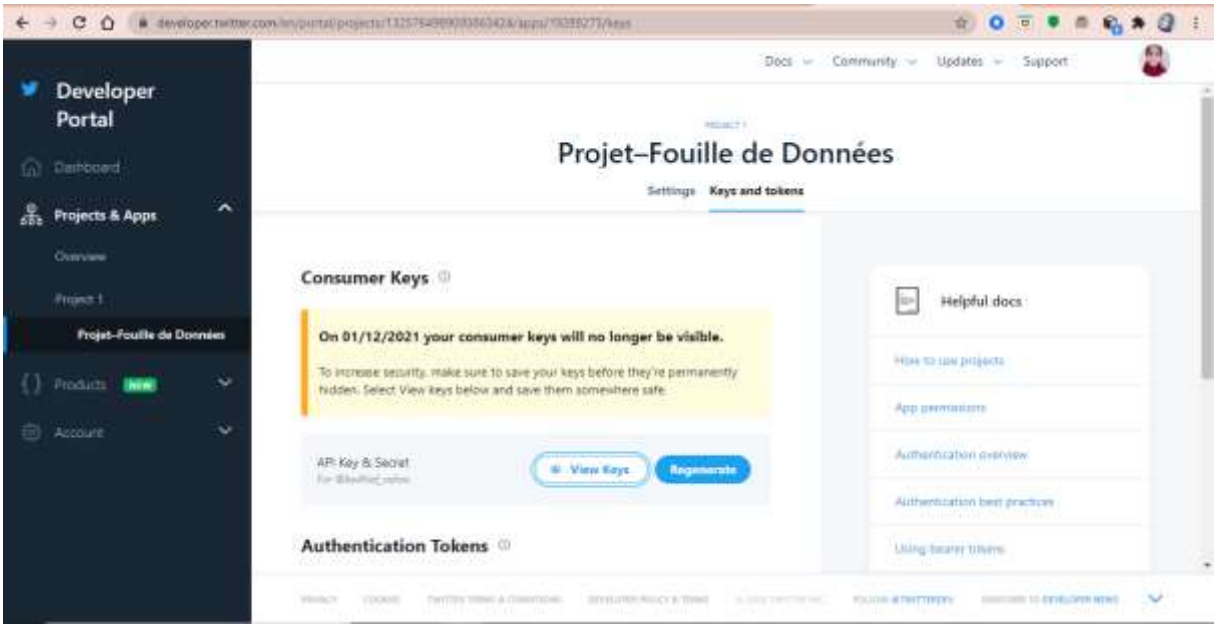
4. Fonctionnement de l'algorithme K-Means

k-means est un algorithme itératif qui minimise la somme des distances entre chaque individu et le centroïde. Le **choix initial des centroïdes conditionne le résultat final**. Admettant un nuage d'un ensemble de points, K-Means change les points de chaque cluster jusqu'à ce que la somme ne puisse plus diminuer. Le résultat est un ensemble de clusters compacts et clairement séparés, sous réserve de choisir la bonne valeur du nombre de clusters.

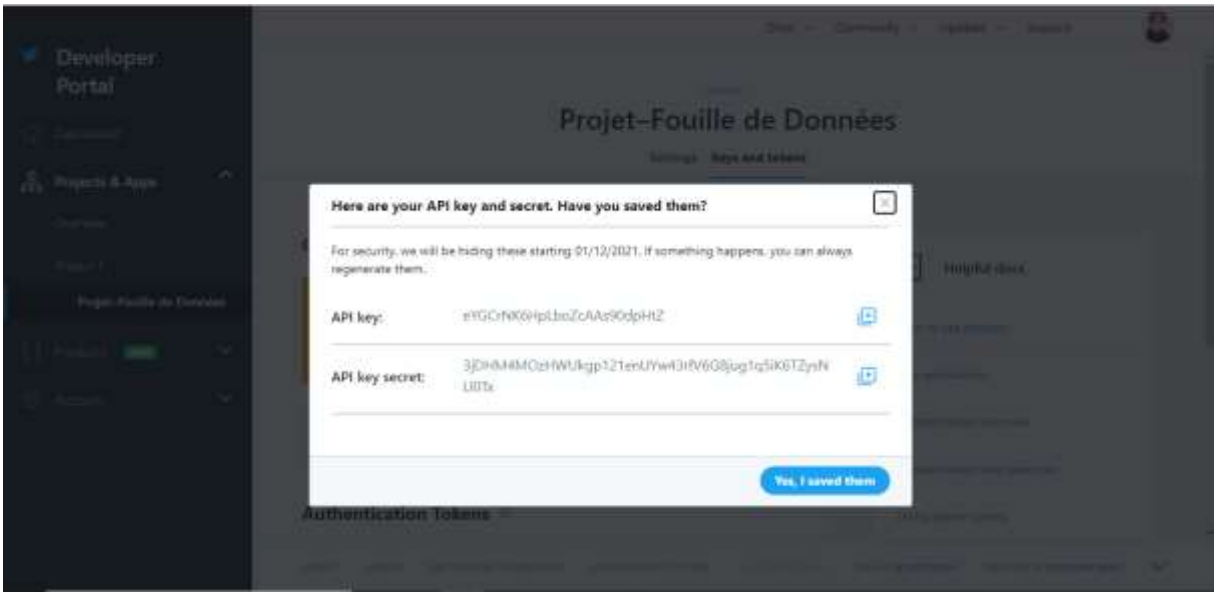


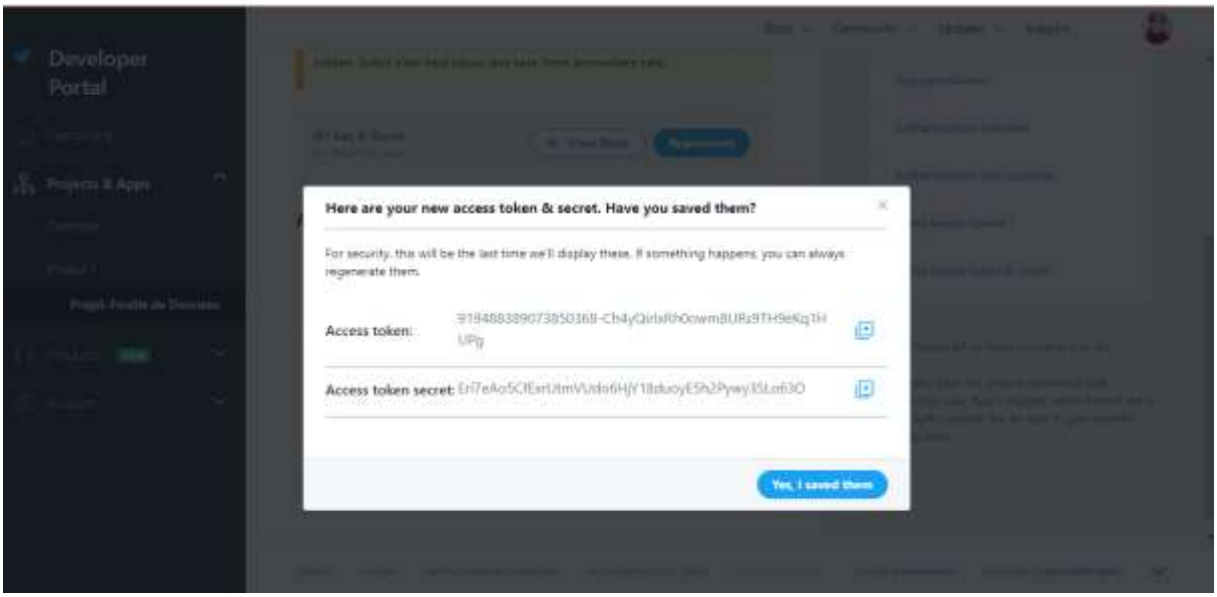
IV. Réalisation

1. Obtenir Clé API twitter

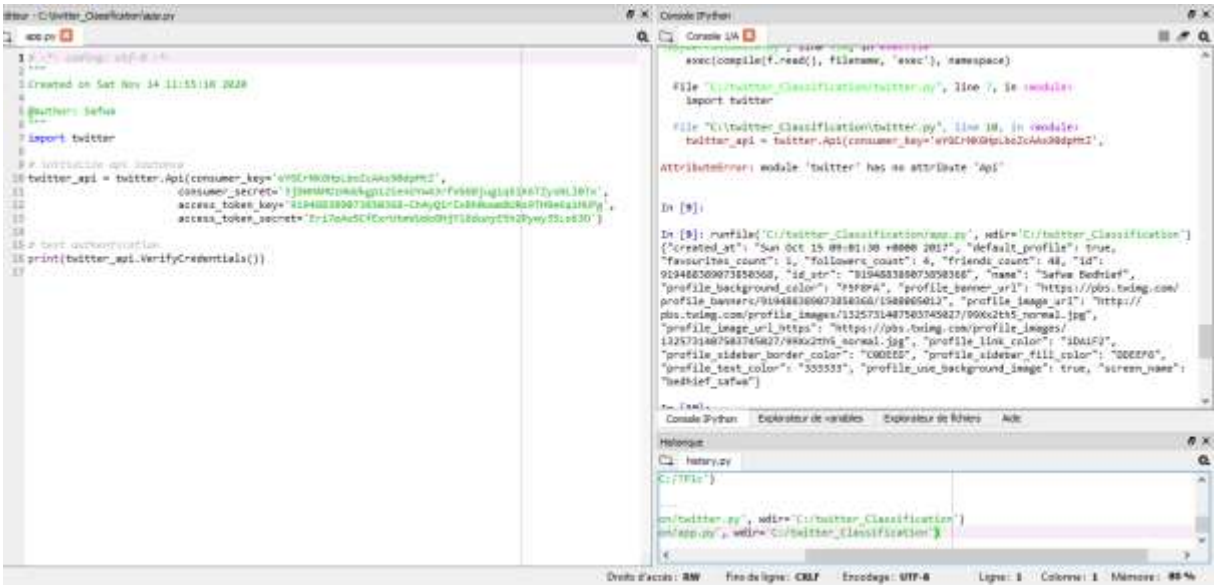


- Trouver notre clé de consommateur et les clés secrètes de consommateur.





- Vérification API Key



2. Extraction 10000 tweets

- Télécharger des tweets basés sur un mot clé de recherche

```
In [3]: userID = "NEWS"
        tweets = api.user_timeline(screen_name=userID,count=200,include_rts = False,tweet_mode = 'extended')
```

- Extraire autant de tweets passés que possible. J'ai pu extraire 126 mille Tweets grâce à ce code

```
In [5]: all_tweets = []
        all_tweets.extend(tweets)
        oldest_id = tweets[-1].id
        while True:
            tweets = api.user_timeline(screen_name=userID,count=200,include_rts = False,tweet_mode = 'extended')
            if len(tweets) == 0:
                break
            oldest_id = tweets[-1].id
            all_tweets.extend(tweets)
            print('N of tweets downloaded till now {}'.format(len(all_tweets)))
```

```
N of tweets downloaded till now 123200
N of tweets downloaded till now 123400
N of tweets downloaded till now 123600
N of tweets downloaded till now 123800
N of tweets downloaded till now 124000
N of tweets downloaded till now 124200
N of tweets downloaded till now 124400
N of tweets downloaded till now 124600
N of tweets downloaded till now 124800
N of tweets downloaded till now 125000
N of tweets downloaded till now 125200
N of tweets downloaded till now 125400
N of tweets downloaded till now 125600
N of tweets downloaded till now 125800
N of tweets downloaded till now 126000
N of tweets downloaded till now 126200
N of tweets downloaded till now 126400
N of tweets downloaded till now 126600
N of tweets downloaded till now 126800
```

- Enregistrer les tweets dans csv

```
In [6]: outtweets = [[tweet.id_str,
                       tweet.created_at,
                       tweet.user.screen_name,
                       tweet.full_text.encode("utf-8").decode("utf-8")]
                      for idx,tweet in enumerate(all_tweets)]
```

```
In [7]: df = DataFrame(outtweets,columns=["id","created_at","screen_name", "text"])
        df.to_csv('Dataset/%s_tweets.csv' % userID,index=False)
        df.head(3)
```

Out[7]:

	id	created_at	screen_name	text
0	1338770862196060161	2020-12-15 09:00:21	NEWS	Bis zu 16-Stunden-Tage, 6 Tage die Woche. Gewe...
1	1338575262850670592	2020-12-14 20:03:06	NEWS	Bundesheerhunde riechen Covid-19 - https://t.c...
2	1338554862649434113	2020-12-14 18:42:03	NEWS	Die bewegendsten Schlager-Momente 2020 - https...

3. Prétraitement des tweets

- Élimination des dédoublements

```
In [15]: df.drop_duplicates()
```

Out[15]:

	id	created_at	screen_name	text
0	1338770862196060161	2020-12-15 09:00:21	NEWS	Bis zu 16-Stunden-Tage, 6 Tage die Woche. Gewe...
1	1338575262850670592	2020-12-14 20:03:06	NEWS	Bundesheerhunde riechen Covid-19 - https://t.c...
2	1338554862649434113	2020-12-14 18:42:03	NEWS	Die bewegendsten Schlager-Momente 2020 - https...
3	1338517886973997058	2020-12-14 16:15:07	NEWS	So bleiben die Kekse zur Adventzeit länger fri...
4	1338493219181649927	2020-12-14 14:37:06	NEWS	14. Dezember: Merry Christmas, Honey! - https:...
...	...	...	...	...
196	1323979584212291586	2020-11-04 13:25:05	NEWS	Harte Zeiten für die Türkis-Grüne Regierung ht...
197	1323950670047809537	2020-11-04 11:30:12	NEWS	Was wirklich hinter "Schleich di, du Oaschloch...
198	1323923886036164610	2020-11-04 09:43:46	NEWS	Beamter über Einsatz: "Haben Verletzte geborge...

- Nettoyage des tweets

In [12]:
```
def clean_tweet(df, text_tweets):
    df[text_tweets] = df[text_tweets].str.lower()
    df[text_tweets] = df[text_tweets].apply(lambda elem: re.sub(r"(@[A-Za-z0-9]+)|([^0-9A-Za-z \t])|(\w+:\/\/\S+)|^rt|http.+?", '
    return df

clean_tweets = clean_tweet(df, 'text')
clean_tweets.head()
```

Out[12]:

	id	created_at	screen_name	text
0	1336712163914899456	2020-12-09 16:39:49	Trump	our christmas tree at is bringing a new level...
1	1334584272074760192	2020-12-03 19:44:20	Trump	unwind by an inviting fire and discover a weal...
2	1333815184108957696	2020-12-01 16:48:15	Trump	last chance to take advantage of the suitest d...
3	1333480717217636352	2020-11-30 18:39:12	Trump	bring the world of trump home with you this ch...
4	1331991068234100739	2020-11-26 15:59:52	Trump	today we give thanks with grateful hearts happ...

- Enlever les mots vides
- Ponctuations, Lemmatisation et Segmentation

In [13]:
```
nlp = en_core_web_sm.load()
tokenizer = RegexpTokenizer(r'\w+')
lemmatizer = WordNetLemmatizer()
stop = set(stopwords.words('english'))
punctuation = list(string.punctuation)
stop.update(punctuation)
w_tokenizer = WhitespaceTokenizer()
```

In [14]:
```
def furnished(text):
    final_text = []
    for i in w_tokenizer.tokenize(text):
        if i.lower() not in stop:
            word = lemmatizer.lemmatize(i)
            final_text.append(word.lower())
    return " ".join(final_text)
```

In [15]:
```
df.text = df.text.apply(furnished)
df.sample(5)
```

Out[15]:

	id	created_at	screen_name	text
1286	1217198163158802436	2020-01-14 21:33:51	Trump	home away home better
3120	1172191793473085442	2019-09-12 16:54:36	Trump	proud announce named 1 best hotel world congra...
2700	1318942734762246144	2020-10-21 15:50:27	Trump	step away bustling city street escape trump pa...
1043	1336712163914899456	2020-12-09 16:39:49	Trump	christmas tree bringing new level luxury penns...
1556	1264902210514796546	2020-05-25 12:52:42	Trump	honoring remembering men woman armed force mad...

4. Vectorisation

In [21]:
```
tv=TfidfVectorizer()
tfidf_tweets =tv.fit_transform(df.text)
```

In [22]:
```
tf_idf = pd.DataFrame(data = tfidf_tweets.toarray(), columns=tv.get_feature_names())
```

In [25]:
```
final_df.T.nlargest(5, 0)
```

Out[25]:

	0	1	2	3	4	5	6	7	8	9	...	3864	3865	3866	3867	3868	3869	3870	3871	3872	3873
bringing	0.452301	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.000000	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
level	0.401172	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.000000	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
pennsylvania	0.401172	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.000000	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
tree	0.370913	0.0	0.0	0.0	0.0	0.0	0.298712	0.0	0.222512	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
avenue	0.332574	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.000000	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

5. Cluster des Tweets

In [17]:
```
def run_KMeans(max_k, data):
    max_k += 1
    kmeans_results = dict()
    for k in range(2 , max_k):
        kmeans = KMeans(n_clusters = k
                        , init = 'k-means++'
                        , n_init = 10
                        , tol = 0.0001
                        , n_jobs = -1
                        , random_state = 1
                        , algorithm = 'full')

        kmeans_results.update( {k : kmeans.fit(data)} )

    return kmeans_results
```

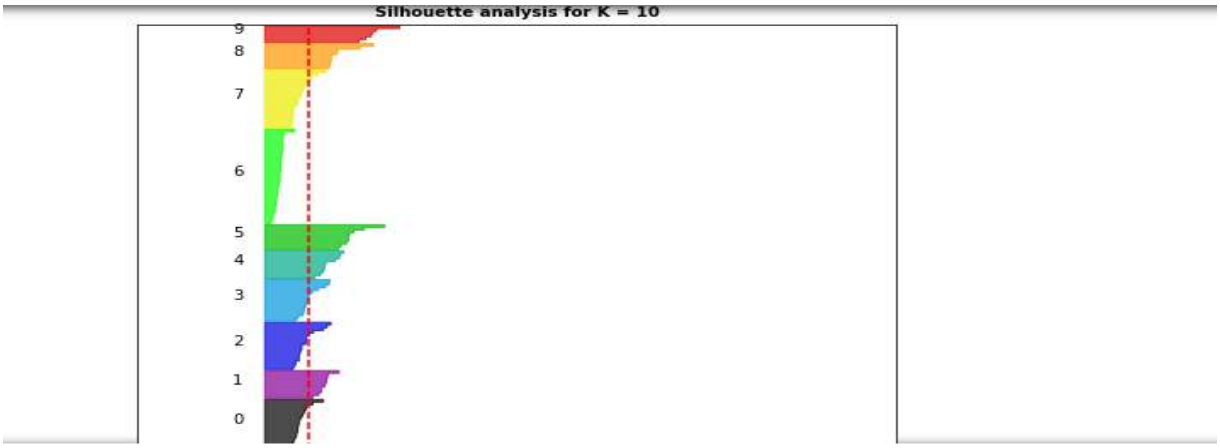
In [18]:
```
def printAvg(avg_dict):
    for avg in sorted(avg_dict.keys(), reverse=True):
        print("Avg: {}\tK:{}".format(avg.round(4), avg_dict[avg]))
```

La valeur de silhouette est une mesure de la similitude d'un objet avec son propre cluster par rapport à d'autres clusters

In [19]:
```
def plotSilhouette(df, n_clusters, kmeans_labels, silhouette_avg):
    fig, ax1 = plt.subplots(1)
    fig.set_size_inches(8, 6)
    ax1.set_xlim([-0.2, 1])
    ax1.set_ylim([0, len(df) + (n_clusters + 1) * 10])

    ax1.axvline(x=silhouette_avg, color="red", linestyle="--") # The vertical line for average silhouette score of all the value
    ax1.set_yticks([])  # Clear the yaxis labels / ticks
    ax1.set_xticks([-0.2, 0, 0.2, 0.4, 0.6, 0.8, 1])
    plt.title(("Silhouette analysis for K = %d" % n_clusters), fontsize=10, fontweight='bold')

    y_lower = 10
    sample_silhouette_values = silhouette_samples(df, kmeans_labels) # Compute the silhouette scores for each sample
    for i in range(n_clusters):
        ith_cluster_silhouette_values = sample_silhouette_values[kmeans_labels == i]
        ith_cluster_silhouette_values.sort()

        size_cluster_i = ith_cluster_silhouette_values.shape[0]
        y_upper = y_lower + size_cluster_i

        color = cm.nipy_spectral(float(i) / n_clusters)
        ax1.fill_betweenx(np.arange(y_lower, y_upper), 0, ith_cluster_silhouette_values, facecolor=color, edgecolor=color, alpha
        ax1.text(-0.05, y_lower + 0.5 * size_cluster_i, str(i))
        y_lower = y_upper + 10
    plt.show()
```

In [20]:
```
def silhouette(kmeans_dict, df, plot=False):
    df = df.to_numpy()
    avg_dict = dict()
    for n_clusters, kmeans in kmeans_dict.items():
        kmeans_labels = kmeans.predict(df)
        silhouette_avg = silhouette_score(df, kmeans_labels)
        avg_dict.update( {silhouette_avg : n_clusters} )

        if(plot): plotSilhouette(df, n_clusters, kmeans_labels, silhouette_avg)
```

```
In [32]: k=10
kmeans_results = run_KMeans(k, final_df)
silhouette(kmeans_results, final_df, plot=True)
```



6. Carte des mots

Maintenant que nous pouvons regarder les graphiques ci-dessous et voir les mots les mieux notés dans chaque cluster

```
In [48]: def centroidsDict(centroids, index):
    a = centroids.T[index].sort_values(ascending = False).reset_index().values
    centroid_dict = dict()

    for i in range(0, len(a)):
        centroid_dict.update( {a[i,0] : a[i,1]} )

    return centroid_dict

def generateWordClouds(centroids):
    wordcloud = WordCloud(max_font_size=100, background_color = 'white')
    for i in range(0, len(centroids)):
        centroid_dict = centroidsDict(centroids, i)
        wordcloud.generate_from_frequencies(centroid_dict)

        plt.figure()
        plt.title('Cluster {}'.format(i))
        plt.imshow(wordcloud)
        plt.axis("off")
        plt.show()
```

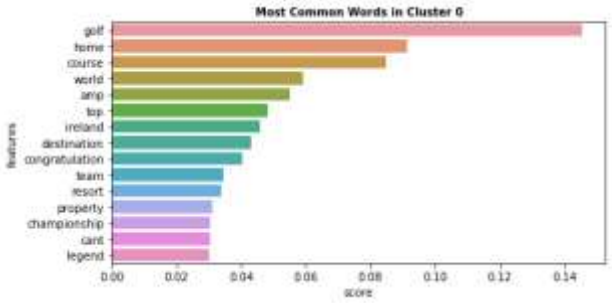
```
In [41]: centroids = pd.DataFrame(kmeans.cluster_centers_)
         centroids.columns = final_df.columns
         generateWordClouds(centroids)
```

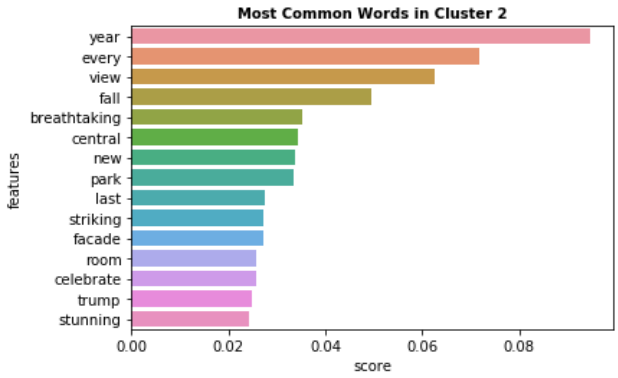
7. Mot le plus courant dans chaque groupe

```
In [37]: def get_top_features_cluster(tf_idf_array, prediction, n_feats):
             labels = np.unique(prediction)
             dfs = []
             for label in labels:
                 id_temp = np.where(prediction==label)
                 x_means = np.mean(tf_idf_array[id_temp], axis = 0)
                 sorted_means = np.argsort(x_means)[::-1][:n_feats]
                 features = tv.get_feature_names()
                 best_features = [(features[i], x_means[i]) for i in sorted_means]
                 df = pd.DataFrame(best_features, columns = ['features', 'score'])
                 dfs.append(df)
             return dfs
```

```
In [38]: def plotWords(dfs, n_feats):
             plt.figure(figsize=(8, 4))
             for i in range(0, len(dfs)):
                 plt.title(("Most Common Words in Cluster {}".format(i)), fontsize=10, fontweight='bold')
                 sns.barplot(x = 'score' , y = 'features', orient = 'h' , data = dfs[i][:n_feats])
                 plt.show()
```

```
In [57]: best_result = 6
         kmeans = kmeans_results.get(best_result)

         final_df_array = final_df.to_numpy()
         prediction = kmeans.predict(final_df)
         n_feats = 20
         dfs = get_top_features_cluster(final_df_array, prediction, n_feats)
         plotWords(dfs, 15)
```





V. Conclusion

Au cours du projet, on a couvert beaucoup de détails sur traitement du langage naturel ; comment fonctionne, Prétraitement (Nettoyage de données, standardiser, segmentation, lemmatisation et ‘Stop Word') et la vectorisation à l’aide de la bibliothèque NLTK en Python ainsi nous ont appris comment regrouper les mots similaires à l’aide de l’algorithme k-means en affiche le représentant pour chaque classe