*2019-2020*

*5IABD*

# Automatic Text Summarization And Predictive Classification Of General Condition Of Use With Natural Language Processing


Projet réalisé par le Groupe 3

- *COINTEPAS Aargan*
- *OLIME Anthonny*
- *BELALA Manal*
- *WAHBI Iliasse*
- *CHOUALI Chahinaz*

Projet encadré par

*Mr Rames*

## Table des matières

## Table des matières

# INTRODUCTION

Dans le cadre de notre cinquième année à l'École Supérieure du Génie Informatique, il nous a été demandé de concevoir un projet nous permettant de mettre en pratique nos connaissances et nos acquis pédagogiques au travers d'un cahier des charges ayant pour finalité la conception et le développement d'un système de traitement de langage naturel.

Notre groupe a saisi l'opportunité d'exploiter son intérêt commun pour l'interprétation des structures de données et les modèles d'apprentissage profond pour soumettre l'ébauche d'un projet innovant sur la synthèse et la classification de données textuelles présentent dans les Condition Générales d'Utilisation de sites internet.

## Objectif du projet

Actuellement les sites internet, marchand ou non, doivent contenir certaines informations légales, des conditions générales d'utilisation (CGU) figurent à ce titre sur la majorité des sites web, la loi impose certaines mentions obligatoires sur un site Internet, elles doivent définir l'utilisation et encadrer les modalités d'accès et de navigation sur le site Internet, et déterminent les droits et obligations respectifs de l'utilisateur et de l'éditeur dans le cadre de l'utilisation du site.

Les Conditions Générales d'Utilisation, n'ont de valeur légale qu'à condition d'être acceptées par l'internaute au moment d'accéder au site, en cochant une case. Cette action engage la responsabilité de l'internaute sur la propagation d'une importante quantité d'informations personnelles ou à caractère sensible sur son identité numérique. Mais il est souvent long et fastidieux de lire et comprendre le contenue de ses CGU.

En Europe et plus particulièrement en France, les conditions générales d'utilisation doivent être rédigées en conformité avec les dispositions légales mentionnées dans le RGPD et la CNIL, malheureusement beaucoup de sites internet reste dans la non-conformité de celle-ci ou contiennent des mentions légales très ambigus.

En pratique, en navigant sur internet la majorité des internautes ne lisent pas le contenu de ces CGU, ils cochent la case adéquate pour passer à l'étape suivante et se débarrasser du pop-up, pourtant, il existe des mentions qui méritent de s'y attarder, surtout lorsqu'il s'agit de naviguer sur des sites d'opinion, car il peut y avoir des possibilités de collecte de données sensibles.

Il est donc plus que nécessaire de trouver un moyen qui permet d'une façon ou d'une autre à l'utilisateur d'avoir connaissance de certaines clauses qui engage l'exploitation de ses données en lien avec son identité virtuelle, c'est pourquoi notre projet s'inscrit dans l'optique d'orienter les internautes à valider ou non ces conditions générales en leur proposant avant validation des termes, une synthèse significative des clauses en lien avec les données sensibles ainsi qu'un score  prédictif de conformité légale de celle-ci aux RGPD et à la CNIL.

## Description du système

Notre système est constitué d'un plugin qui s'appui sur une API, il se compose des éléments suivants :

- Un plugin google chrome qui permet aux utilisateurs de nous envoyer les urls des CGU des différents site qu'ils souhaite analyser. Il construira une url composé de l'adresse de l'API, son port ainsi qu'en paramètre l'url du site qu'ils souhaite traiter. Il ouvrira un nouvelle onglet au sein du navigateur en la faisant pointer sur l'url construite précédemment. Location : ***../plugin_chrome/***
- L'API va récupérer le contenue de l'url passé en paramètre, le sauvegarder pour enrichir notre dataset, elle va ensuite nettoyer son contenue (suppression des stop word, transformation du texte en minuscule, suppression des caractère spéciaux). Son contenu sera par la suite divisé en phrases, il va attribuer des vecteurs a ses phrases à partir de notre dictionnaire de donnée puis il construira une matrice de similarité afin d'identifier les phrases ayants le plus de sens entre elle. Location : ../***scraper/quick_termofuse_api.py***

## Architecture de l'API

Pour alimenter les différentes partie de notre API nous utilisons les scripts suivants :

- ***..modele/Aargan/check_important_word_in_sentences.py*** qui sauvegarder l'ensemble des corpus dans un seul fichier, il va ensuite nettoyer l'ensemble des corpus, récupérer les phrases en français, puis au sein de ses phrases il va compté et récupérer les 200 mots les plus redondants.

  input : ***../files/Company_folder_files_clean/****

  output : ***../files/top_word.txt***,

  ***../files/CGU_FR.txt***

- ***../modele/Aargan/check_important_word_in_sentences_en.py*** qui sauvegarder l'ensemble des corpus dans un seul fichier, il va ensuite nettoyer l'ensemble des corpus, récupérer les phrases en anglais, puis au sein de ses phrases il va compté et récupérer les 200 mots les plus redondants.

  input : ***../files/Company_folder_files_clean/****

  output : ***../files/top_word_en.txt***,

  ***../files/CGU_EN.txt***

- ***../modele/Chanez/GloVe-master/demo2.sh*** lui va créer un ensemble de vecteur à partir des mots

  input : ***../files/top_word_en.txt***,

  ***../files/CGU_EN.txt***

  output : ***../files/vectors.txt*** ou

  ***../files/vectors_en.txt***


Envoie de l'url à l'API par le plugin
Ouverture d'une page avec le contenu résumé
Récupération et sauvegarde du contenu du site
Traitements du plugin / API
Summarization - Résumé des textes
Récupération récursive du contenu des urls du site ayant un rapport avec les cookies
Nettoyage des corpus récupérés

## Les Étapes du projet :

## Étape 1 : Text summarization avec  [ NLTK]

Pour cette première partie, on a essentiellement fait appel à la bibliothèque NLTK de python L'objectif de cette étape est le raccourcissement de longs morceaux de texte.

L'objectif est de créer un résumé cohérent et fluide ayant seulement les principaux points soulignés dans les clauses mentionnant le traitement et l'exploitation des données personnelles présentent dans les CGU.

Afin de créer ce résumé sans perdre de cohérence avec ce qui est dit dans le contenue nous avons pris le partie de retourner à l'utilisateurs les 10 phrases les plus en liens avec les CGU, la RGPD et les textes de la CNIL.

Pour cela nous avons du concevoir notre propre dictionnaire de mots afin d'identifier au mieux ses phrases.

### 1.Collecte de données CGU et établissement d'un dataset

Pour la collecte des données nous somme tomber sur un gros problème. C'est que Bien que présente sur chaque site, les CGU ( ou policy , privacy … enfin toutes ses dérivés ) , Ne sont pas au même format et sont aussi imbriquer dans des site tous plus différents les un des autres . Alors le scraper que nous avons mis au point a été compliqué à mettre en place.

Nous avons Ciblé les 2000 site des plus grandes entreprise du monde pour que nous ayons une plus grande chance de trouver les cgu , mais en même temps les different site internet semble vouloir cacher ou que l'accès à cela soit le plus compliqué possible par un script.

Mais nous avons réussi à créer un script python qui est aller au départ aller chercher dans les 2000 site les liens vers les cgu (ou tout les page ayant un liens avec) puis stocker tout cela dans un fichier CSV.

Puis nous avons parcourue tout les site avec tout les pages trouver pour télécharger le texte de toute les page; en les catégorisant en 4 groupe Privacy, Policy , Cookies, Terms.

Après avoir récupéré cela on a vue que ce que l'on a récupéré avais beaucoup de "garbage". Il fallait nettoyer tout les partie inutile que l'on avait récupéré sur les site comme les header et les footer avec tous les liens vers les autre page des sites visités. Ceci a été fait grâce à spacy et python.

## 2. Identification des fréquences de mots :

Notebook de test : ***../modele/Aargan/check_important_word_in_sentences.ipynb***

Script : ***../modele/Aargan/check_important_word_in_sentences.py***

L'étapes d'identification des fréquences des mots se traduit par les processus suivants :

- Lire l'intégralité des textes dans les différent dossier et documents dans ***../files/Company_folder_files_clean/***.
- Sauvegardes de l'ensemble des corpus dans le fichier ***../files/CGU_FR.txt*** qui sera nécessaire pour la construction du dictionnaire.
- Identification des phrases dans le corpus.
- Nettoyage des phrases: transformation du texte en minuscule, suppression des caractère spéciaux, suppression des stop_word (en fr et en anglais).
- Vérification que le contenue des phrases soit différent de nul et qu'il soit en français ou anglais tous dépend de notre cas d'usage.
- Récupération des mots dans les phrases et calculs du nombre d'occurrence du mots.
- Sauvegarde du résultat dans le fichier ***../files/top_word.txt*** pour les mots en français et ***../files/top_word_en.txt*** pour les mots en Anglais.

Attention, les mots en Anglais et en Français ne sont pas calculé en même temps, se sont deux script distinct qui les génère :

- ***../modele/Aargan/check_important_word_in_sentences.py*** pour les mots en français
- ***../modele/Aargan/check_important_word_in_sentences_en.py*** pour les mots en anglais.

## 3. Etablissement d'un dictionnaire

Dans ce qui suit, pour plus de détails techniques, se référer au fichier ../***modele/Chanez/GloVe-master/README.md***

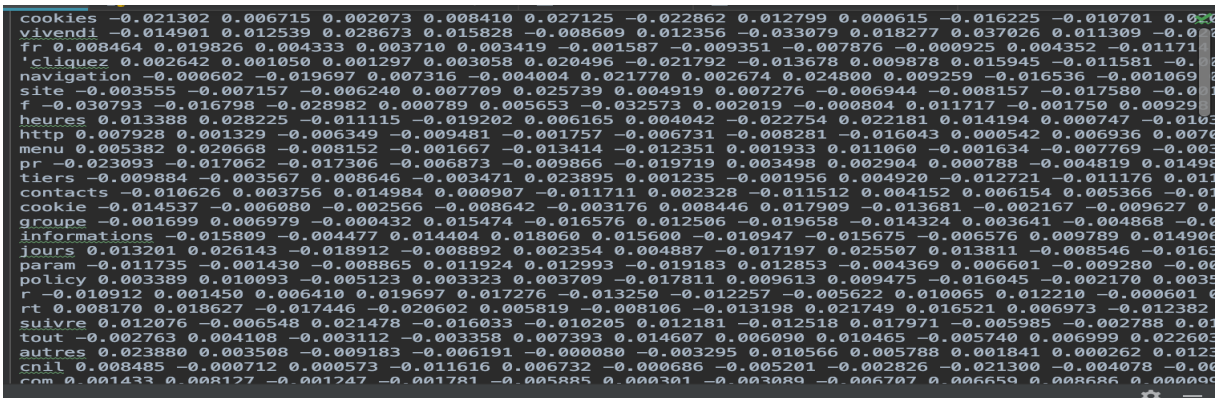
Le modèle Glove n'est que le résultat de divers méthodes récentes d'apprentissage de divers représentations spatiales de mots et qui ont par la suite réussi à capturer des régularités sémantiques et syntaxiques très précises à l'aide de l'arithmétique vectorielle, mais l'origine de ces régularités que contient le modèle Glove reste à notre niveau encore très opaque.

Cependant en appliquant ce modèle à notre dictionnaire de données, cela nous a permis l'émergence d'un espace vectoriel avec une structure significative d'ordonnancement de mots représentés par des vecteurs à valeur réelles.

Ces vecteurs réels de mots générés se reposent sur la distance entre des paires de vecteurs de mots pour évaluer les nouvelles dépendances de mots d'un nouvel ensemble de représentations de mots vectorielles c'est ainsi que ce modèle nous a permit de remonter les régularités syntaxiques existantes dans un nouvel ensemble de données .

Dans notre cas, on a formé nos propres vecteurs GloVe, en préparant un corpus de texte avec les mots dont la fréquence est la plus élevée.

Pour former nos propres vecteurs GloVe, nous avons d'abord préparer notre corpus de données CGU en tant que fichier texte unique avec tous les mots séparés par un ou plusieurs espaces ou tabulations. Une fois que notre corpus créé, on a réussi à former des vecteurs GloVe. ci-après un bref aperçu :




## 4. Générez le résumé

Notebook de test : ***../modele/Aargan/Text_Summarization_Fr-Copy1.ipynb***

Script : intégration du notebook dans l'API ***../scraper/quick_termofuse_api.py***

A l'aide des fichiers généré précédemment nous allons réaliser les actions suivantes pour effectuer un résumé :

- Récupération du corpus.
- Identifications des phrases au sein du corpus.
- Lecture du dictionnaire de donnée dans ***../files/vectors.txt***, ***../files/vectors_2.txt*** ou ***../files/vectors_en.txt.*** Attention vectors.txt et vector_2.txt sont des dictionnaires en français, mais ils ne sont pas construit de la même manière. En effet vector.txt est réalisé à l'aide des textes en français (25 fichier), au contraire vector_2.txt est calculé sur l'ensemble des fichiers (1206 fichiers) puis un filtre est appliqué pour réaliser le comptage seulement sur des phrases en Français. Cela est dû au mélanges de langues au sein de nos fichiers, en effet certains fichiers ont des parties en français et en anglais. Pour récupérer l'ensemble du contenue en français, j'ai pris la décision de tester en récupérant l'ensemble des phrases en français mais en conservant l'ensemble des corpus (du coup avec de l'anglais et du français) nous nous retrouvons donc avec un dictionnaire qui comporte des mots en français ainsi que des mots en anglais.
- Nettoyages des phrases : suppression des caractères inutile, transformation du texte en minuscule et suppression des stop_words.
- Transformation des phrases en vecteur à l'aide du dictionnaire.

- Calcul du score de similarités entre les vecteurs.
- Utilisation de l'algorithme pagerank afin de calculer le score entre les différentes phrases du corpus.

-Trie et récupération des phrases correspondant au scores calculé précédemment et ajouts des phrases avec le meilleur score

# Étape 2 : Détection, analyse et classification des clauses CGU, Justice Prédictive avec [SPACY]

Cette partie a pour but d'orienter l'utilisateur dans son choix de validation des termes d'utilisation d'un site quelconque en lui indiquant un score de prédiction en % de conformité du CGU aux lois légales sur le respect de la collecte des données personnelles.

Dans ce qui suit : voir le script **/modele/Chanez/Spacy Prediction .ipynb**

## 1°Extraction et entraînement des entités nommées

### 1.Etablissement d'un Dictionnaire :

Dans notre cas, la tâche de reconnaissance des entités nommées spaCy a été entraînée sur un jeu de données qui prend en charge plusieurs types de CGU contenant principalement des phrases ou paragraphes en lien ou pas avec les données personnelles, ce dictionnaire a été constitué de façon à inclure les différentes façon d'interpréter ou de renvoyer à un texte de loi provenant du RGPD ou de la loi du 6 janvier 1978 relative à l'informatique et libertés, on y incluant également dans quelques passages de montions légales en lien avec la CNIL.

On a choisi de nous concentrer sur des CGU de langues françaises, on a donc fait le choix de charger un modèle français, ce modèle permet une représentation vectorielles de mot communiquant leur relations avec d'autres mots.

```
nlp = spacy.load('fr_core_news_sm')
```

### 2.tokenisation

La première chose que nous allons faire est de « tokeniser » une phrase ou paragraphe présente dans le dictionnaire afin de les découper grammaticalement, pour cela nous avons utilisé la classe token de SpaCy

### 3.reconnaissance d'entités nommées (NER)

SpaCy dispose d'un système de reconnaissance d'entités statistiques (NER ou Named Entity Recognition) très performant et qui va assigner des étiquettes à des plages contigues de tokens.

```
[100]: for ent in doc.ents:
           print(ent.text, ent.label_)
```

```
[101]: from spacy import displacy

       displacy.render(doc, style="ent")
```

## 4.L'étiquetage (Part-of-Speech (POS) Tagging)

Le but de cette partie est d'attribuer une étiquette à chaque mot d'une phrase mentionnant la fonctionnalité grammaticale d'un mot (Nom propre, adjectif, déterminant…), ci-après quelques étiquettes identifiées dans un exemple de CGU :

```
partagera VERB__Mood=Ind|Number=Sing|Person=3|Tense=Fut|VerbForm=Fin partager
des DET__Definite=Ind|Number=Plur|PronType=Art un
information NOUN__Gender=Fem|Number=Sing information
personnelle ADJ__Gender=Fem|Number=Sing personnel
avec ADP___ avec
des DET__Definite=Ind|Number=Plur|PronType=Art un
tiers NOUN__Gender=Masc|Number=Plur tiers
à ADP___ à
des DET__Definite=Ind|Number=Plur|PronType=Art un
fins NOUN__Number=Plur fin
de ADP___ de
marketing NOUN__Gender=Fem|Number=Sing marketing
```

## 5.Dépendances

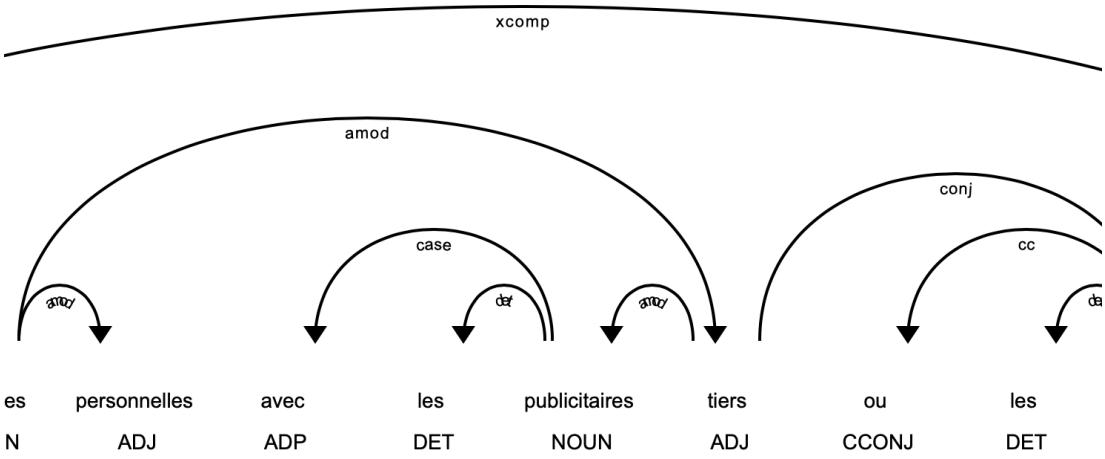
C'est la fonction la plus intéressante de Spacy , car grâce à elle nous avons réussi à pouvoir recomposer une phrase en reliant les mots dans leur contexte. Dans l'exemple ci-dessous nous reprenons notre phrase et indiquons chaque dépendance avec les propriétés dep_

```
In [87]: tokenized_text = pd.DataFrame()

         for i, token in enumerate(parsed_review):
             tokenized_text.loc[i, 'text'] = token.text
             tokenized_text.loc[i, 'lemma'] = token.lemma_,
             tokenized_text.loc[i, 'pos'] = token.pos_
             tokenized_text.loc[i, 'tag'] = token.tag_
             tokenized_text.loc[i, 'dep'] = token.dep_
             tokenized_text.loc[i, 'shape'] = token.shape_
             tokenized_text.loc[i, 'is_alpha'] = token.is_alpha
             tokenized_text.loc[i, 'is_stop'] = token.is_stop
             tokenized_text.loc[i, 'is_punctuation'] = token.is_punct

         tokenized_text[:20]
```

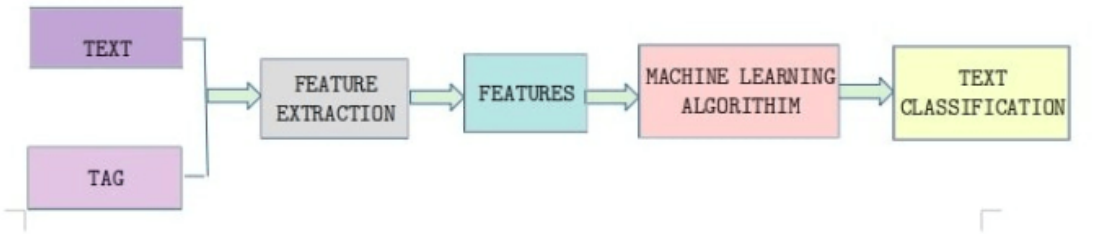
Out[87]:

| | text | lemma | pos | tag | dep | shape | is_alpha | i |
|---|---|---|---|---|---|---|---|---|
| 0 | Nous | nous | PRON | PRON__Number=Plur\|Person=1 | nsubj | Xxxx | True | T |
| 1 | ne | (ne,) | ADV | ADV__Polarity=Neg | advmod | xx | True | T |
| 2 | partageons | (partager,) | VERB | VERB__Mood=Ind\|Number=Plur\|Person=1\|Tense=Fut\|... | ROOT | xxxx | True | F |
| 3 | pas | (pas,) | ADV | ADV__Polarity=Neg | advmod | xxx | True | T |
| 4 | vos | (votre,) | DET | DET__Number=Plur\|Poss=Yes | nmod:poss | xxx | True | T |
| 5 | donnees | (donnee,) | NOUN | NOUN__Gender=Fem\|Number=Plur | obj | xxxx | True | F |
| 6 | personnelles | (personnel,) | ADJ | ADJ__Number=Plur | amod | xxxx | True | F |

Spacy dispose également d'une option de visualisation qui nous permet simplement d'afficher les étiquettes identifiées ainsi que les dépendances entre ces étiquettes.




## 2°Classification ML




La libraire Spacy propose plusieurs modèles de réseaux de neurones pour le marquage, l'analyse et la reconnaissance d'entités.

### Architecture Spacy TextCategorizer

L'architecture de notre modèle de classification textuelle est  Spacy TextCategorizer, son architecture exacte n'est pas rendu disponible en ligne actuellement, mais il s'agit d'un modèle de réseau de neurones  où les vecteurs jetons sont calculés à l'aide d'un CNN, les vecteurs sont regroupés en moyenne et utilisés comme caractéristiques dans le réseau  de neurones.

### La fonction spacy_tokenizer ()

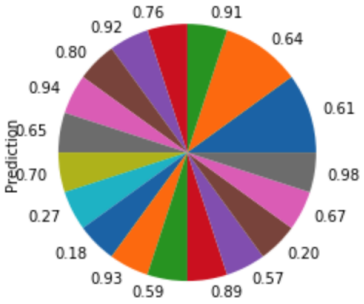
Cette fonction accepte une phrase en entrée et traite la phrase en jetons, effectuant la lemmatisation, la minuscule et la suppression des mots vides. Ceci est similaire à ce qu'on peut faire en plusieurs sous fonction en utilisant NLTK comme pour l'étape 1,  mais le succès de spacy réside dans le fait que nous pouvons tous regrouper tout cela dans une seule fonction pour prétraiter chaque avis d'utilisateur que nous analysons.

## Visualisation des résultats

Grâce à la bibliothèque Matplotlib de python on a réussi à visualiser les différents scores des CGU classifiés.

**Data Visualisation**

```
Entrée [257]: data['Prediction'].value_counts().plot.pie()

Out[257]: <matplotlib.axes._subplots.AxesSubplot at 0x1271e95f8>
```




# Axes d'améliorations futures

-Généraliser ces processus de synthèse et de prédiction sur la totalité des clauses du CGU.

-Alimenter nos dictionnaires pour l'obtention d'un contenu synthétique significatif et un score de précision meilleur.

-Actuellement il n'existe pas de dictionnaire juridique français avec Spacy, l'un des objectifs futurs de notre système est d'entraîner notre dictionnaire sur de nouveau NER en français en utilisant spacy. SpaCy offre la possibilité de définir un label et ensuite de l'entraîner, cette fonctionnalité va nous permettre d'ajouter une nouvelle entité de type "Loi " à un modèle NER vide ou existant.

-Chargement d'un vocabulaire juridique en anglais et l'entrainer sur un dictionnaire spacy en lien avec les lois américaines ou anglo-saxon.