SCIENCES SORBONNE UNIVERSITÉ

Campagne d'expérience

Avis de films & Discours Présidents Mitterrand et Chirac

Santhos Arichandra, Almehdi Krisni

Mars 2022

Enseignant: Vincent Guigue

Table des matières

Sentiments

Introduction

Nous avons à disposition des données de films et plus exactement les avis des utilisateurs sur des films. Il s'agit donc ici d'un problème de classification négative ou positive.

Pre-Processing

En ce qui concerne la ponctuation, les chiffres nous les avons enlevés avec évidence car ici elle ne permettent pas de données une information supplémentaire.

Les majuscules ne sont pas nécessaires car certes de manière générale on peut penser que les personnes mécontentent s'exprimer avec des mots forts et donc en MAJUSCULES. Sauf qu'ici c'est surtout la recherche des termes négatifs et positifs qui nous intéresse donc le fait d'avoir le memes mot en minuscule ou maj n'a pas d'intérêt. C'est le sens des mots et leur signification et non leur syntaxe qui va nous aider à déterminer le sentiment.

- Stop-Words

Les stop-words peuvent plus ou moins être utiles dans notre cas puisque certains stop-words correspondent justement à des mots sous forme négative comme par exemple don't. Justement ces mots sous forme de négation vont sûrement nous permettre de différencier les classes. Toutefois il est toujours intéressant de tester si leurs suppressions permettraient d'avoir une amélioration.

- Racinisation

La racinisation, pour notre classification, n'a pas d'intérêt car justement, on reviendrait sur la forme de base des mots sans leur forme négative/positive donc ça pose problème pour la classification.

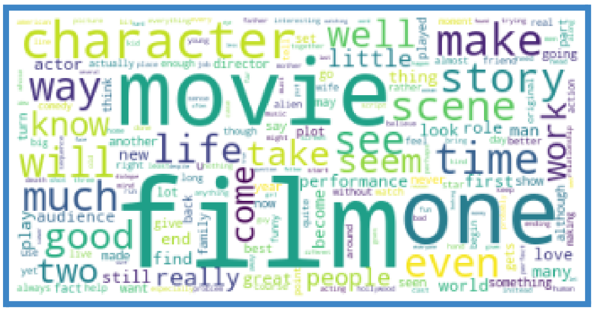
- Lemmatisation

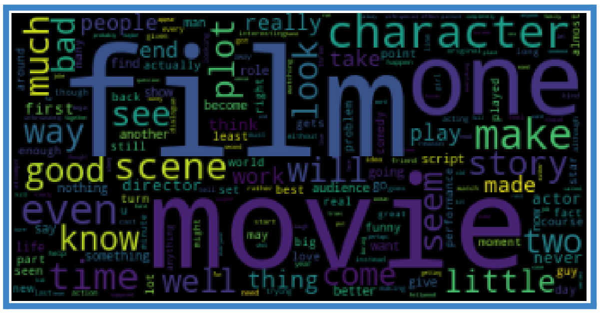
Un aspect qui pour le coup peut être intéressant c'est la lemmatisation surtout pour la négation car avec du bi-grams on aura le <'t> qui pourra surement donc nous aider à réaliser une meilleure prédiction, de même pour le tri-grams.
En plus du tri-gram, nous pouvons tester le 1-gram car on a l'apostrophe qui est la plupart du temps présent dans des mots négatifs et de manière générale le 1-gram marche bien comme dans la traduction donc a tester.

Analyses des données

Tout d'abord regardons comment les mots sont représentés sans pré-processing pour les positifs et les négatifs.

WordCloud des movies SANS suppression des similar most commons (G: Positifs D: Negatifs)





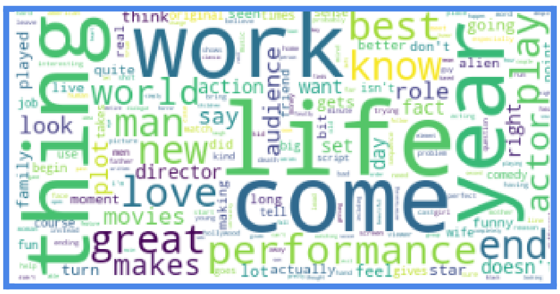
On s'aperçoit ici que dans les deux cas les mots les plus fréquents sont similaires. Il s'agit en plus du mot relatif au vocabulaire du cinéma et non en rapport avec les sentiments de l'utilisateur. Il va donc falloir enlever ces mots là pour avoir une meilleure représentation des classes par les mots.

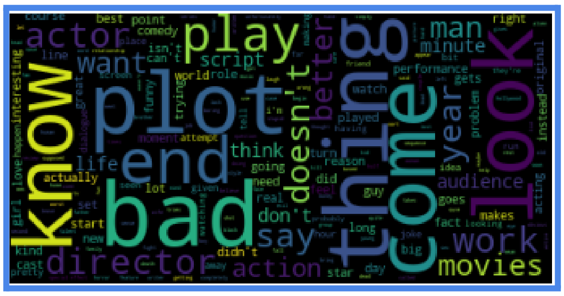
A présent, regardons ce qui se passe quand nous enlevons les mots similaires entre les 25 mots les plus fréquents de chaque classe.

Voici donc la liste des mots en communs :
['just', 'films', 'like', 'people', 'does' 'time' 'film', 'characters', 'way', 'little', 'really', 'scenes', 'movie', 'character', 'scene' 'make' 'story', 'good']

On s'aperçoit que 'good' fait partie de la liste, ce qui peut être étonnant, mais il s'agit tout simplement du fait que les avis ici ne sont pas des avis extrêmes, ou si c'est le cas ils sont très peu présents. Donc ils sont plus modérés et peuvent avoir des parties positives, mais un avis global négatif.

WordCloud des movies AVEC suppression des similar most commons (G: Positifs D: Negatifs)





Avec la suppression des mots similaires (parmi les 25 plus fréquents), on a une frequence des mots, qui représente un peu mieux les sentiments (surtout pour la classe négative).

Codage des mots

Il existe différentes manières de représenter les mots, nous allons réaliser deux types de représentation.
La première correspond à un simple comptage du nombre d'occurrences de tous les mots, pour tous les documents. (CountVectorizer)

La seconde qui sera le TF-IDF, celle-ci permettra surement d'avoir un meilleur score, car comme dit plus haut les avis ne sont pas totalement positifs ou négatifs. Il s'agit d'avis plus généraux donc il est plus préférable d'utiliser la fréquence. (TfidfVectorizer)

Tests Réalisés

- Transformations

Nous allons réaliser différents tests pour essayer de trouver quelles transformations, quels modèles avec quels paramètres, nous permettent d'avoir les meilleurs résultats, et donc une meilleure prédiction.

Tout d'abord les transformations qui seront fait dans tous les cas sont les suivants :

- Passages des majuscules en minuscules
- Suppression des chiffres
- Suppression de la ponctuation

Comme vu dans la partie pré-processing, ces trois ne vont pas permettre un gain en information pour la classification. Ensuite, nous allons faire avec les 2 types d'encodage le CountVectorizer et le TfidfVectorizer.

Pour chacun des deux, nous allons tester ces différentes transformations:

- Sans transformations supplémentaires
- Suppression des Stop Words + les mots en communs entre les 25 plus fréquents de chaque classe (StopWordAndSimilar)
- 1-gram
- Suppression des StopWordAndSimilar + 1-gram
- 2-gram
- Suppression des StopWordAndSimilar + 2-gram
- 1-gram + 2-gram
- Suppression des StopWordAndSimilar + 1-gram + 2-gram
- 3-gram
- Suppression des StopWordAndSimilar + 3-gram
- 1-gram + 2-gram + 3-gram
- Suppression des StopWordAndSimilar + 1-gram + 2-gram + 3-gram

- Modèles

Nous allons utiliser 3 modèles différents pour notre prédiction : MultinomialBayes, LinearRegression, LinearSVC.

- MultinomialBayes

Nous allons tester différentes valeurs de alpha, qui est la valeur remplacer pour les probabilités nulles.
On va tester ces valeurs : [0.1, 0.2, 0.25, 0.5, 0.75, 0.8, 1]

- LinearRegression

Dans ce modèle, on va uniquement faire varier le max_iters, car il n'y a pas de paramètres qui pourrait nous permettre d'améliorer le scoring.

- LinearSVC

Pour ce modèle, nous allons faire varier le max_iters, mais aussi le paramètre C.

Celui-ci permet d'éviter un sur-apprentissage, et donc il évident que l'on doit le faire varier et trouver la bonne valeur. Un C petit, nous permettraient d'avoir moins d'erreurs, mais donc d'avoir moins de marge entre les 2 classes et donc surement un sur-apprentissage. Et un C grand, l'inverse donc une plus grande marge, mais avec sûrement plus d'erreurs.

N'ayant pas d'a priori sur ce que serait une bonne valeur nous allons en tester pleins :
[0, 1, 2, 3, 4, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90, 95, 100]

Prediction

Comme dit plus haut, nous avons utilisé deux types de codage CountVectorizer et TfidfVectorizer, regardons l'écart entre les deux.

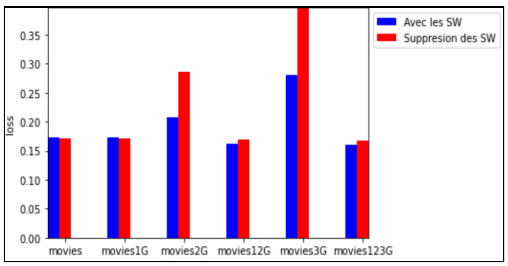
Ecart des F1-scores entre CountVectorizer et TfidfVectorizer

	NB	SVC	LR
movies	-0.003695	-0.035490	0.012975
moviesSW	-0.004353	-0.023635	0.005552
movies1G	-0.003695	-0.035490	0.012975
moviesSW1G	-0.004353	-0.023635	0.005552
movies2G	-0.003763	-0.033927	-0.007615
moviesSW2G	-0.004405	-0.026017	-0.012072
movies12G	-0.007630	-0.033780	0.013866
moviesSW12G	-0.003163	-0.015918	0.001923
movies3G	-0.000348	-0.036401	-0.035868
moviesSW3G	0.012790	-0.034584	-0.016034
movies123G	-0.004159	-0.031842	0.019442
moviesSW123G	-0.005253	-0.016021	0.004165

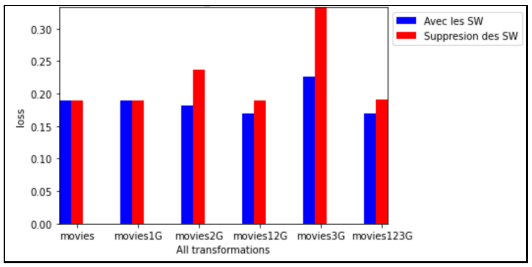
On s'aperçoit ici, un très nette avantage à l'encodage TfidfVectorizer pour NB et SVC, comme on l'avait prévu. Nous allons donc par la suite, utiliser les résultats pour les NB et SVC, obtenus avec le TfidfVectorizer et pour LR les résultats de CountVectorizer.

Regardons à présent si les StopWordAndSimilar (Stop Words + les mots en communs entre les 25 plus fréquents de chaque classe) jouent un grand rôle.

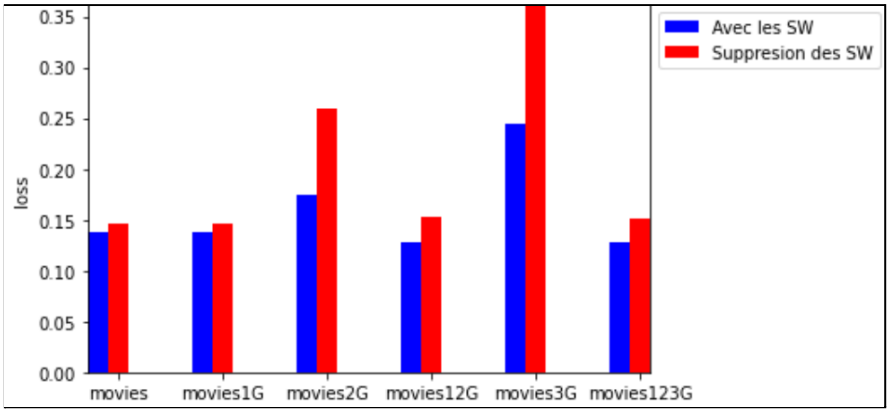
LR



NB



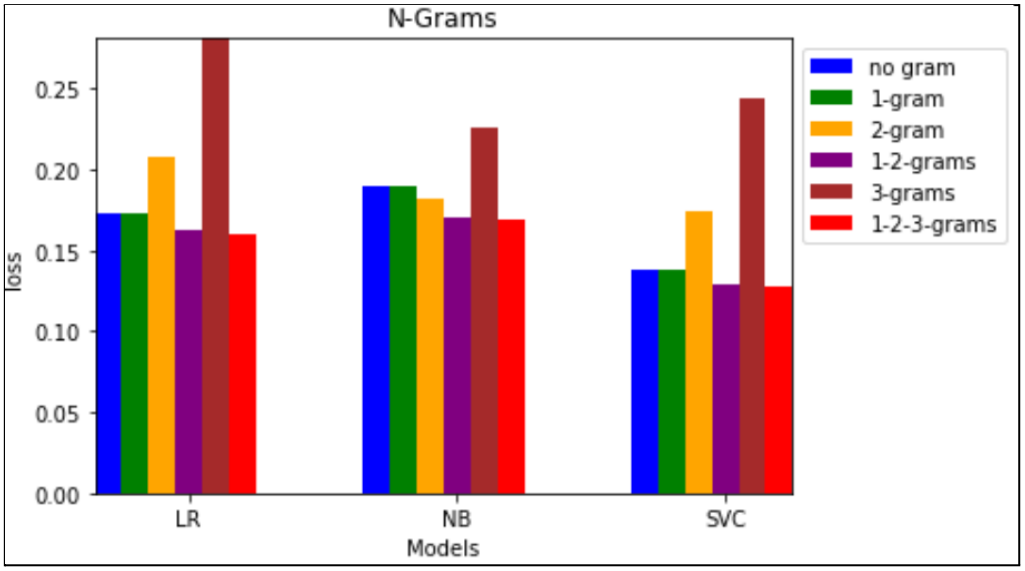
SVC



On s'aperçoit ici très clairement, que la suppression des StopWordAndSimilar ne permet pas d'avoir un gain dans notre classification, cela est sûrement dû au fait que des mots négatifs sont présents dans les stopwords, comme par exemple ''don't''. Dans la régression linéaire, certes certaines permettent de gagner en gain mais c'est très minime au vu du reste.

Donc à présent on sait qu'il ne faut pas enlever les StopWordAndSimilar, intéressons nous aux N-grams et sur le gain qu'ils apportent.

N-Grams sans la suppression des StopWordAndSimilar

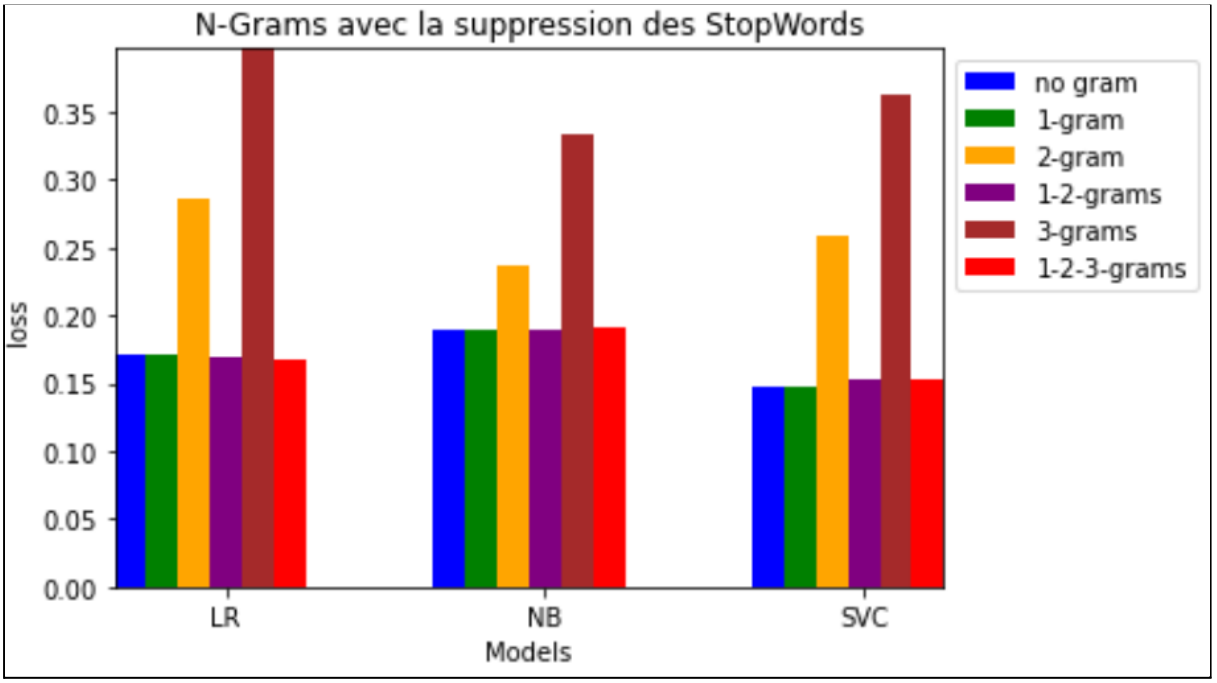


On s'aperçoit ici, que le n-grams permet d'améliorer notre classification.
Toutefois, on peut voir le bi-gram tout seul et le 3-grams tout seul ne permettent pas du tout d'améliorer le score, au contraire il est bien pire.

Pourquoi est-ce le cas ? Alors il est assez difficile de savoir la raison de cette perte, mais une hypothèse qu'on peut en sortir c'est que les termes a 2 lettres sont trop génériques et ne permettent pas d'avoir une distinction à eux seuls et donc créer une confusion. De même pour les 3-grams seul.
Le uni-gram à lui seul aussi ne permet pas d'avoir d'amélioration mais ne dégrade pas la classification également.
La combinaison de plusieurs N-grams permet la d'avoir de meilleur résultat, donc les coupes seules ne peuvent pas apporter d'information supplémentaire, mais leurs combinaisons permettant d'avoir certains patterns. Pour le 1-gram, on peut comprendre cela par le fait que les lettres sont les mêmes pour les mots négatifs et positifs avec sûrement une même fréquence.

Le tri-grams, lui, pourrait permettre de mettre en évidence les formes négatives sous la forme “ 'nt ” et pour les positifs , il pourrait nous permettre de mettre en évidence les superlatifs via le “est”.

Regardons si c'est également le cas avec la suppression des StopWordAndSimilar. Normalement il ne devrait pas y avoir de différence.



Étonnamment ce n'est pas la même chose, on toujours ce pic pour le bi-gram et le tri-gram et une valeur similaire entre le no-gram et uni-gram, ce qui confirme l'hypothèse faite.

Comment expliquer cette observation ? Alors une hypothèse, est que surement les stop words contiennent des termes négatifs, comme dit précédemment, mais que donc ces termes ont un gros poids dans la classification. Donc en les enlevant et en découpant les autres mots qui ont une importance moins grande, nous fait sûrement perdre en information et on revient à des termes génériques entre les deux classes.

Finalement, dans ces graphs on constate très clairement que le LinearSVC est bien meilleur que les deux autres modèles. En ce qui concerne les transformations, enlever les StopWordAndSimilar nous retire de l'information et le 1-2-3-grams combiner nous permet d'avoir les meilleurs résultats.

Modèle final

Transformation:

- Suppression des majuscules, ponctuation, chiffres
- uni-gram + bi-gram + tri-gram

Modèle : LinearSVC
Paramètres : C = 2, max_iters = 10000

Tableau complet des F1-score

	MultinomialB	MultinomialB_Params	SVC	SVC_params	LinearRegression	LR_params
vectorizerC_movies	0.807112	{'alpha': 1}	0.826890	{'C': 2, 'max_iter': 10000}	0.839763	{'max_iter': 3000}
vectorizerC_moviesSW	0.806599	{'alpha': 1}	0.829805	{'C': 1, 'max_iter': 10000}	0.840485	{'max_iter': 3000}
vectorizerC_movies1G	0.807112	{'alpha': 1}	0.826890	{'C': 2, 'max_iter': 10000}	0.839763	{'max_iter': 3000}
vectorizerC_moviesSW1G	0.806599	{'alpha': 1}	0.829805	{'C': 1, 'max_iter': 10000}	0.840485	{'max_iter': 3000}
vectorizerC_movies2G	0.814236	{'alpha': 0.2}	0.792063	{'C': 1, 'max_iter': 10000}	0.807623	{'max_iter': 3000}
vectorizerC_moviesSW2G	0.759183	{'alpha': 1}	0.714622	{'C': 4, 'max_iter': 10000}	0.736827	{'max_iter': 3000}
vectorizerC_movies12G	0.822753	{'alpha': 0.75}	0.837629	{'C': 1, 'max_iter': 10000}	0.843317	{'max_iter': 3000}
vectorizerC_moviesSW12G	0.807805	{'alpha': 1}	0.830889	{'C': 5, 'max_iter': 10000}	0.839443	{'max_iter': 3000}
vectorizerC_movies3G	0.773419	{'alpha': 0.2}	0.719107	{'C': 3, 'max_iter': 10000}	0.735918	{'max_iter': 3000}
vectorizerC_moviesSW3G	0.679972	{'alpha': 0.8}	0.603407	{'C': 1, 'max_iter': 10000}	0.630880	{'max_iter': 3000}
vectorizerC_movies123G	0.826811	{'alpha': 0.25}	0.840605	{'C': 2, 'max_iter': 10000}	0.846775	{'max_iter': 3000}
vectorizerC_moviesSW123G	0.804029	{'alpha': 1}	0.832036	{'C': 4, 'max_iter': 10000}	0.839695	{'max_iter': 3000}
vectorizerTFIDF_movies	0.810807	{'alpha': 0.2}	0.862380	{'C': 1, 'max_iter': 10000}	0.826789	{'max_iter': 3000}
vectorizerTFIDF_moviesSW	0.810953	{'alpha': 0.75}	0.853440	{'C': 1, 'max_iter': 10000}	0.834933	{'max_iter': 3000}
vectorizerTFIDF_movies1G	0.810807	{'alpha': 0.2}	0.862380	{'C': 1, 'max_iter': 10000}	0.826789	{'max_iter': 3000}
vectorizerTFIDF_moviesSW1G	0.810953	{'alpha': 0.75}	0.853440	{'C': 1, 'max_iter': 10000}	0.834933	{'max_iter': 3000}
vectorizerTFIDF_movies2G	0.817999	{'alpha': 1}	0.825989	{'C': 1, 'max_iter': 10000}	0.815238	{'max_iter': 3000}
vectorizerTFIDF_moviesSW2G	0.763588	{'alpha': 0.2}	0.740640	{'C': 1, 'max_iter': 10000}	0.748899	{'max_iter': 3000}
vectorizerTFIDF_movies12G	0.830383	{'alpha': 0.2}	0.871409	{'C': 1, 'max_iter': 10000}	0.829451	{'max_iter': 3000}
vectorizerTFIDF_moviesSW12G	0.810967	{'alpha': 1}	0.846807	{'C': 1, 'max_iter': 10000}	0.837520	{'max_iter': 3000}
vectorizerTFIDF_movies3G	0.773767	{'alpha': 0.1}	0.755508	{'C': 1, 'max_iter': 10000}	0.771786	{'max_iter': 3000}
vectorizerTFIDF_moviesSW3G	0.667182	{'alpha': 0.1}	0.637991	{'C': 2, 'max_iter': 10000}	0.646914	{'max_iter': 3000}
vectorizerTFIDF_movies123G	0.830971	{'alpha': 0.1}	0.872447	{'C': 2, 'max_iter': 10000}	0.827333	{'max_iter': 3000}
vectorizerTFIDF_moviesSW123G	0.809283	{'alpha': 1}	0.848057	{'C': 1, 'max_iter': 10000}	0.835531	{'max_iter': 3000}

Présidents

Introduction

Dans cette partie, nous allons nous intéresser à la prédiction de l'interlocuteur lors d'un débat présidentiel entre Chirac et Mitterrand. Entre d'autres termes, qui a dit quoi ?

Nous disposons donc d'une base de données d'apprentissage de 57000 lignes représentant un ensemble de phrases, chacune de ces dernières étiquetées par un C (Chirac) ou un M (Mitterrand), désignant la personne l'ayant prononcé. On peut alors labéliser notre ensemble d'apprentissage en 2 classes distinctes.

Notre objectif est de mettre en place des algorithmes allant nous permettre de labelliser de nouveaux débats entre Chirac et Mitterrand.

Pre-Processing

Etant donné qu'il s'agit de discours et non de texte écrit par les présidents, la ponctuation n'a pas d'importance, car celle-ci n'est pas propre à leur discours mais à celui qui a retranscrit le discours. De même les majuscules ne devraient pas avoir d'importance, pour la même raison. Nous allons tout de même essayer sans ces paramètres.

- Stop-Words

Dans ce cas d'étude, les stop words vont sûrement nous permettre de gagner en information. On ne devrait pas se retrouver avec le problème des sentiments, car ici peu importe s'il s'agit de la forme des mots, ce qui nous importe le plus sont les mots exclusifs à chaque président. Nous souhaitons alors faire en sorte que ces mots exclusifs aient un poids plus important par rapport aux autres mots lors de la classification.

- Racinisation

Comme dit juste au-dessus, un des aspects importants sont les mots exclusifs, il est donc intéressant de réaliser une racinisation. Cela va permettre de pouvoir travailler dans un espace plus réduit, mais tout en gardant cette information de mots exclusifs ou dominants.

- Lemmatisation

Ici aussi nous allons essayer différents N-grams, car il est toujours intéressant de voir s'ils ont un impact ou non sur la classification des données.

Analyses des données

Tout comme pour les sentiments, nous nous intéressons à ce que les WordCloud issus des données d'apprentissage ressemblent pour chacun des Présidents.

WordCloud SANS suppression (G: Mitterrand, D: Chirac)

WordCloud AVEC suppression suppression des similars 25 most common (G: Mitterrand, D: Chirac)

WordCloud AVEC suppression suppression des similars 50 most common (G: Mitterrand, D: Chirac)

On constate, que sans suppression supplémentaire aux Stop Words, on se retrouve avec le même problème que pendant l'étude de la partie Sentiments, des mots similaires entre les deux classes. On décide donc de supprimer les mots en communs partagés, soit les "shared most common words".

Pour 25, les mots restent assez similaires et ont la même signification, alors que pour 50 on aperçoit, une différence assez notable. D'un côté, nous avons un discours sur les problèmes, la force, etc ... et de l'autre la paix nouvelle, etc ... Deux vocabulaires qui possèdent de grandes différences . Il s'agit donc d'une nouvelle idée à mettre en place et étudier afin de comprendre si on pourrait y trouver un rapport majeur avec la classification.

Une deuxième idée qu'on peut voir et qui pourrait nous aider par la suite sont les mots exclusifs de chacun des présidents.

Voici pour Mitterrand :
['convenait', 'eureka', 'nievre', 'morvan', 'estampes', 'cee', 'nevers', 'laitiers', 'electrification', 'montlucon', 'vocabulaire', 'dedain', 'definitions', 'songez', 'disposes', 'figeac', 'guadeloupeen', 'cantal', 'banc', 'imprudent', 'plaindre', 'bougent', 'rn', 'poulenc', 'buffon', 'decentraliser', 'csce', 'legumes', 'dominent', 'surarmement']

Voici pour Chirac :
['mondialisation', 'euro', 'attentes', 'constituent', 'guyane', 'mercosur', 'sida', 'respectueux', 'internet', 'verre', 'probablement', 'superbe', 'chaleureusement', 'laicite', 'handicapees', 'e', 'gendarmerie', 'xxie', 'gouvernance', 'indien', 'multipolaire', 'mediterraneen', 'socle', 'associatif', 'pompiers', 'croissant', 'champions', 'presidente', 'evian', 'mayotte']

Les mots ont l'air simple mais cela pourrait représenter une piste pour peut-être, avoir une meilleure classification.

Première campagne d'expérience

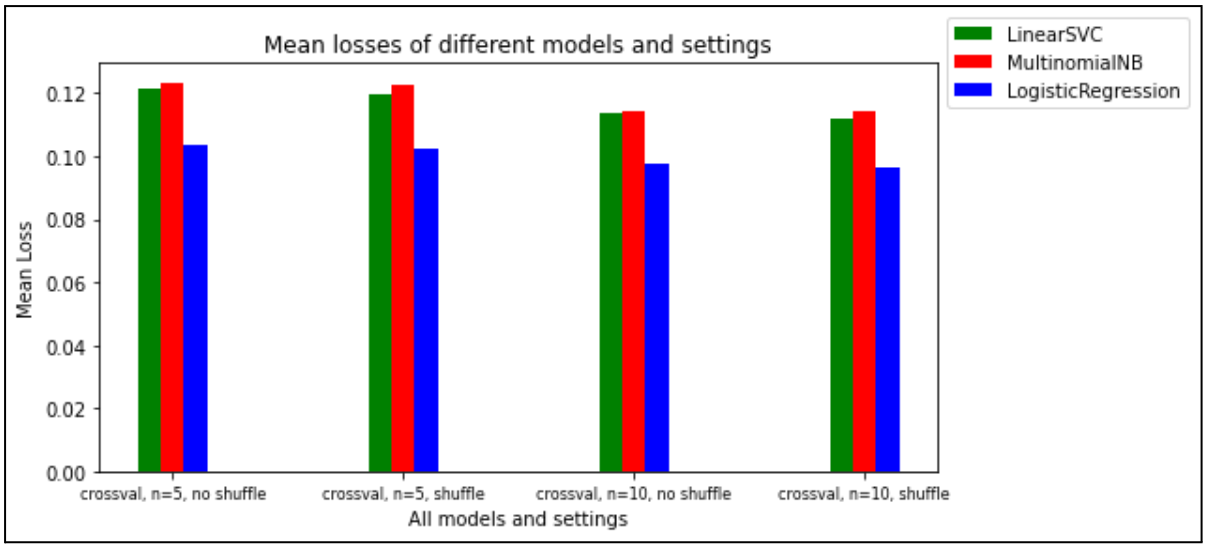
Durant notre première campagne, nous nous sommes intéressés uniquement à l'utilisation des différents modèles en faisant varier quelques paramètres afin d'obtenir les meilleurs résultats comme l'utilisation ou non de la validation croisée, le nombre d'ensemble utilisés et le mélange ou non des données d'apprentissage.

Nous avons choisi d'utiliser les principaux modèles vus en cours et en TME, c'est-à-dire :

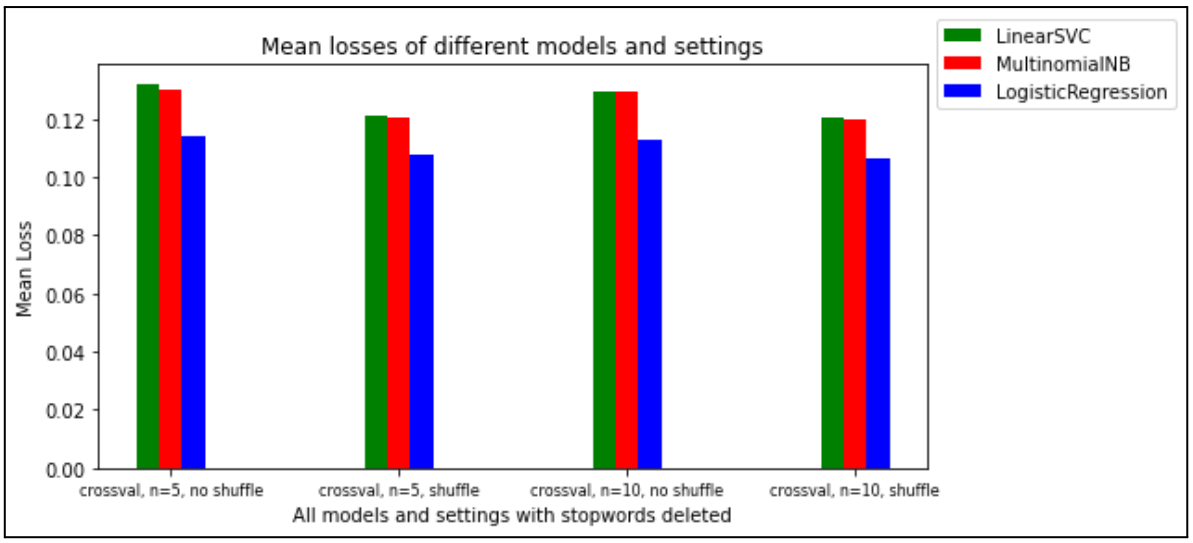
- SVM linéaire (LinearSVC)
- Naive Bayes (MultinomialNB)
- régression logistique (LogisticRegression)

On remarque que le modèle le plus précis est la Régression Logistique, puisque son taux d'erreur moyen est constamment inférieur ou égal à 10%. On remarque tout de même qu'en augmentant le nombre d'itérations sur la validation croisée et en mélangeant les données, on obtient une baisse constante de l'erreur moyenne.

La base de données utilisée n'a pas été transformée afin de supprimer toutes les majuscules, ponctuation, nombres et autres. On décide d'effectuer de nouvelles séries de tests en les supprimant et en utilisant oui ou non les racines de mots.



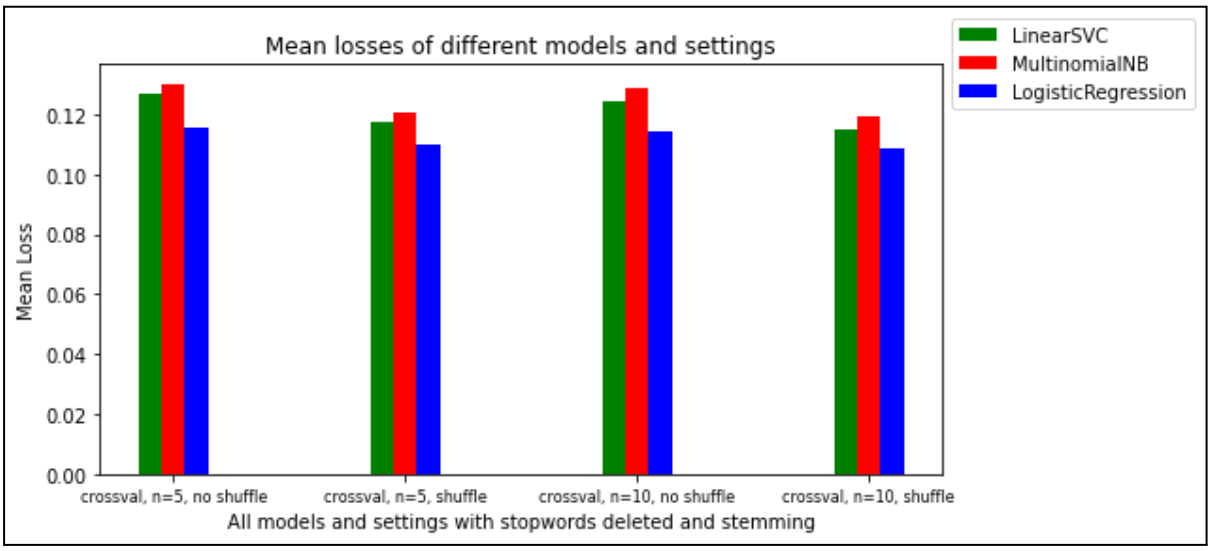
Comparaison des pertes moyennes des différents modèles avec utilisation des données brutes



Comparaison des pertes moyennes des différents modèles avec suppression des stopwords

On remarque que le modèle le plus précis est la Régression Logistique, puisque son taux d'erreur moyen est constamment inférieur ou égal à 10%. On remarque tout de même qu'en augmentant le nombre d'itérations sur la validation croisée et en mélangeant les données, on obtient une baisse constante de l'erreur moyenne.

On remarque également que le fait que les données soient traitées ou non n'a pas réellement d'influence avec le résultat final puisque les pertes moyennes sont équivalentes.. On décide d'effectuer de nouvelles séries de tests en supprimant les stopwords et en utilisant oui ou non le stemming.



Comparaison des pertes moyennes des différents modèles avec suppression des stopwords et stemming

Une nouvelle fois, on observe que le modèle dominant reste la Régression Logistique malgré les modifications apportées aux données (nouveaux stopwords et stemming). Le nombre d'itération plus important de cross-validation et le mélange des données permet toujours de réduire les pertes moyennes.

De ce fait, tous les modèles entraînés par la suite utilisent lors des séries de test une cross-validation à plus de 5 itérations et un mélange des données.

Une question inquiétante est le fait qu'il semble exister une sorte de limite aux alentours des 10% de pertes moyennes. Le fait que nos données soient représentées par Chirac à 87% pourrait expliquer cette limite. Notre deuxième campagne d'expérience aura parmi ses objectifs de répondre à cette question.

Deuxième campagne d'expérience

Nous avons évoqué précédemment le fait que chaque Président utilise plus souvent certains mots que d'autres et le fait que nous avons beaucoup plus de données d'apprentissage pour Chirac que pour Mitterrand.

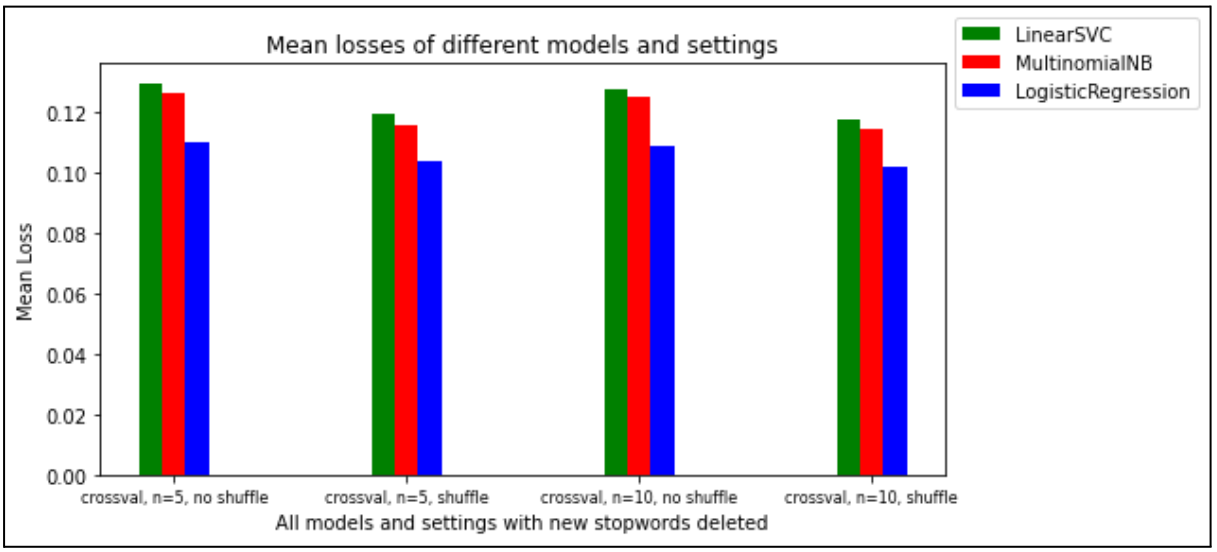
On cherche donc à mettre en place un algorithme utilisant les odds-ratio afin de mettre en avant une liaison entre le fait qu'un Président utilise plus souvent un mot que l'autre. On utilise un biais afin de représenter le fait que Mitterrand ait prononcé beaucoup moins de choses que Chirac dans la base d'apprentissage.

On pourrait penser que cette méthode de prédiction peut être intéressante vu que le score de prédiction est assez intéressant lorsqu'on joue un peu avec le biais (pour un biais de 2.5, on a un score de 0.78) et plus on le réduit, plus le score de prédiction augmente. Mais en réalité, on ne fait que mieux classer les données de Chirac et on délaisse les données de Mitterrand vu le résultat de nos expériences.

Score de prévision avec un biais de 2.5 : 0.78
Score de prévision avec un biais de 1e-05 : 0.79
Score de prévision avec un biais de 1000 : 0.13

Du fait que l'on dispose de presque six fois plus de données Chirac (87%) que Mitterrand (13%), si on continue à réduire le biais, on finira par approcher une limite de 87% de précision. De même, si on augmentait le biais à l'infini, on convergera vers une limite de 13% de précision à cause de la répartition.

Comme durant la première campagne, on effectue une comparaison des différents modèles de prédiction en utilisant cette fois-ci la nouvelle liste de stopwords créée à partir des mots les plus utilisés par les deux Présidents. On remarque que les résultats ne changent pas réellement comparé à pendant la première campagne.



Comparaison des pertes moyennes des différents modèles avec suppression des mots communs partagés

Troisième campagne d'expérience

Après avoir réalisé les deux premières campagnes, nous décidons désormais de rechercher quelles sont les configurations optimales lors de la prédiction des labels Chirac ou Mitterrand. Parmi ces configurations, on joue notamment sur :

- le modèle utilisé lors de l'apprentissage
- la méthode de vectorisation avec suppression ou non des stopwords
- le nombre d'itérations maximal
- le paramétrage

On ne peut pas essayer chaque combinaison de paramètres à la main, il faut donc mettre en place un GridSearch avec tous les paramètres utilisés afin de trouver la meilleure.

Les paramètres spécifiques sont à chaque modèle sont :

- param_grid_clf = {'alpha': [0.1,0.2,0.25,0.5,0.75,0.8,1]}
- param_grid_svc = {'max_iter': [1000,1500,2000],
  'C': [0,1,3,5,10,20,40,60,80,100]}
- param_grid_lin = {'max_iter' :[1000,2000,3000,4000,5000]}

	MultinomialB	MultinomialB_Params	SVC	SVC_params	LinearRegression	LR_params
vectorizerC_pres	0.930902	{'alpha': 0.1}	0.929691	{'C': 1, 'max_iter': 1500}	0.942732	{'max_iter': 1000}
vectorizerC_presSW	0.930121	{'alpha': 0.1}	0.923055	{'C': 1, 'max_iter': 1000}	0.936504	{'max_iter': 1000}
vectorizerC_pres1G	0.930902	{'alpha': 0.1}	0.929680	{'C': 1, 'max_iter': 1500}	0.942732	{'max_iter': 1000}
vectorizerC_presSW1G	0.930121	{'alpha': 0.1}	0.923055	{'C': 1, 'max_iter': 1000}	0.936504	{'max_iter': 1000}
vectorizerC_pres2G	0.928753	{'alpha': 0.1}	0.924684	{'C': 1, 'max_iter': 1000}	0.939680	{'max_iter': 1000}
vectorizerC_presSW2G	0.924565	{'alpha': 1}	0.923556	{'C': 1, 'max_iter': 1000}	0.931616	{'max_iter': 1000}
vectorizerC_pres12G	0.917952	{'alpha': 0.1}	0.925954	{'C': 1, 'max_iter': 1000}	0.942912	{'max_iter': 1000}
vectorizerC_presSW12G	0.924565	{'alpha': 1}	0.923556	{'C': 1, 'max_iter': 1000}	0.931616	{'max_iter': 1000}
vectorizerC_pres13G	0.913612	{'alpha': 0.1}	0.926549	{'C': 1, 'max_iter': 1000}	0.943200	{'max_iter': 1000}
vectorizerC_presSW123G	0.927688	{'alpha': 0.1}	0.923318	{'C': 1, 'max_iter': 1000}	0.936934	{'max_iter': 1000}
vectorizerC_pres3G	0.928212	{'alpha': 1}	0.926107	{'C': 1, 'max_iter': 1000}	0.936568	{'max_iter': 1000}
vectorizerC_presSW3G	0.920390	{'alpha': 1}	0.929152	{'C': 1, 'max_iter': 1000}	0.930493	{'max_iter': 1000}
vectorizerTFIDF_pres	0.936634	{'alpha': 0.1}	0.942115	{'C': 1, 'max_iter': 1000}	0.942241	{'max_iter': 1000}
vectorizerTFIDF_presSW	0.934823	{'alpha': 0.2}	0.934240	{'C': 1, 'max_iter': 1000}	0.936213	{'max_iter': 1000}
vectorizerTFIDF_pres1G	0.936634	{'alpha': 0.1}	0.942115	{'C': 1, 'max_iter': 1000}	0.942241	{'max_iter': 1000}
vectorizerTFIDF_presSW1G	0.934823	{'alpha': 0.2}	0.934240	{'C': 1, 'max_iter': 1000}	0.936213	{'max_iter': 1000}
vectorizerTFIDF_pres2G	0.939752	{'alpha': 0.1}	0.937173	{'C': 1, 'max_iter': 1000}	0.939124	{'max_iter': 1000}
vectorizerTFIDF_presSW2G	0.930810	{'alpha': 1}	0.926770	{'C': 1, 'max_iter': 1000}	0.930839	{'max_iter': 1000}
vectorizerTFIDF_pres12G	0.941013	{'alpha': 0.1}	0.944068	{'C': 1, 'max_iter': 1000}	0.943816	{'max_iter': 1000}
vectorizerTFIDF_presSW12G	0.935801	{'alpha': 0.2}	0.934187	{'C': 1, 'max_iter': 1000}	0.936561	{'max_iter': 1000}
vectorizerTFIDF_pres13G	0.940746	{'alpha': 0.1}	0.943667	{'C': 1, 'max_iter': 1000}	0.943824	{'max_iter': 1000}
vectorizerTFIDF_presSW123G	0.935839	{'alpha': 0.1}	0.934396	{'C': 1, 'max_iter': 1000}	0.936654	{'max_iter': 1000}
vectorizerTFIDF_pres3G	0.935917	{'alpha': 0.5}	0.931802	{'C': 1, 'max_iter': 1000}	0.933767	{'max_iter': 1000}
vectorizerTFIDF_presSW3G	0.929101	{'alpha': 1}	0.929744	{'C': 1, 'max_iter': 1000}	0.930386	{'max_iter': 1000}

Tableau des résultats obtenus lors du GridSearch avec les différents modèles et paramètres

On cherche alors le modèle avec la meilleure précision et on remarque qu'il s'agit du modèle suivant :

vectorizerTFIDF_pres12G	0.941013	{'alpha': 0.1}	0.944068	{'C': 1, 'max_iter': 1000}	0.943816	{'max_iter': 1000}

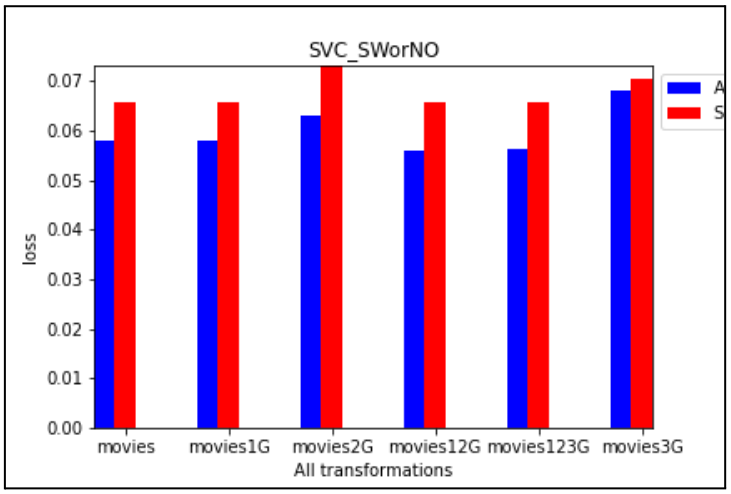
Il s'agit alors du modèle SVC avec un facteur alpha égal à 1 et avec un nombre d'itérations maximal de 1000 itérations utilisant un Vectoriser TF-IDF avec un 1-2-gram sans la suppression des stopwords.

Par rapport à la différence de vectoriseur utilisé, on remarque immédiatement que dans plus de la moitié des cas, l'utilisation d' un VectorizerTFIDF pour la mise en forme des données permet d'obtenir de meilleurs résultats comparé à lorsqu'on utilise un VectorizerCount.
La seule exception relevée est que pour la Régression Logistique, puisque dans à peu près 75% des cas, le VectorizerCount apporte de meilleurs résultats mais les différences sont toujours inférieures à 0,01.

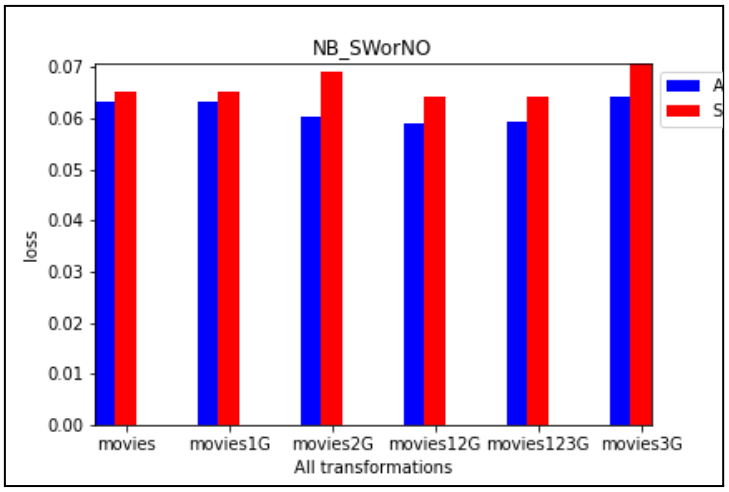
	NB	SVC	LR
movies	-0.005731	-0.012424	0.000491
moviesSW	-0.004702	-0.011185	0.000291
movies1G	-0.005731	-0.012435	0.000491
moviesSW1G	-0.004702	-0.011185	0.000291
movies2G	-0.010999	-0.012489	0.000556
moviesSW2G	-0.006245	-0.003214	0.000777
movies12G	-0.023061	-0.018115	-0.000904
moviesSW12G	-0.011236	-0.010631	-0.004946
movies3G	-0.027134	-0.017118	-0.000623
moviesSW3G	-0.008150	-0.011078	0.000280
movies123G	-0.007705	-0.005694	0.002801
moviesSW123G	-0.008711	-0.000592	0.000107

Différence des résultats pour chaque modèle avec l'utilisation d'un VectorizerCount ou TFIDF
(ici, les résultats sont égaux à resCount - resTFIDF)

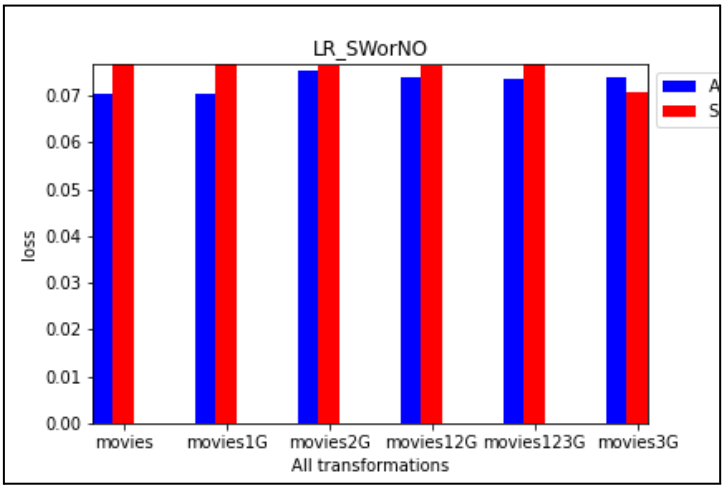
Intéressons nous à la comparaison des modèles de manière plus détaillée.
(PS. Un problème d'affichage est survenu lors de la sauvegarde des plots. Les données bleues représentent un modèle où les stopwords ont été conservés et les données rouges représentent un modèle où les stopwords ont été supprimés)



Comparaisons des pertes moyennes pour le modèle SVC avec différentes configurations



Comparaison des pertes moyennes pour le modèle NB avec différentes configurations



Comparaison des pertes moyennes pour le modèle LR avec différentes configurations

On comprend immédiatement que le modèle le moins intéressant parmi les 3 est la Régression Logistique puisque quelque soit le modèle, les pertes moyennes sont aux alentours de 0,07, tandis que pour les autres modèles, la perte moyenne est plus souvent aux alentours de 0.06 (SVC et NB).

On remarque tout de même que la suppression des stopwords n'entraîne pas une amélioration du résultat obtenu puisque le seul modèle où cette tendance n’est pas validé est le LR utilisant des données

Conclusion

Pour l'étude des sentiments, l'analyse nous a permis de faire certaines hypothèses, qui au final n'ont pas été vérifiées, comme par exemple, sur la suppression des stop words. Malgré tout, nous avons constaté ici que les autres aspects du pre-processing, nous ont permis de gagner en information, comme ça a pu être le cas avec les N-Grams.

Pour l'étude des Présidents, nous avons réalisé 3 campagnes d'expériences différentes afin de mieux comprendre les données qui nous ont été fournies et comment les prédire.

Nous avons ainsi pu réaliser la prédiction des données 'test' qui se trouve dans le fichier "presidentSentiment.txt". Nous avons ainsi prédit :

- 26859 phrases prononcées par Chirac (99,99 %)
- 303 phrases prononcées par Mitterrand (0,01 %)

Notre crainte précédente s'est donc révélée être vraie puisque l'apprentissage réalisé n'a permis d'apprendre à classer que la classe majoritaire.