RITAL – Projet - TP

Revues de films et reconnaissance de locuteur

Saliou BARRY, Carine MOUBARK

Analyse de sentiment appliquée à des critiques de films

Dans cette partie du projet, on s'intéresse à l'application de l'analyse de sentiment à un corpus de critiques de films, qui contient 2000 critiques, dont 1000 positives et 1000 négatives. Notre objectif est de construire un modèle capable de prédire la polarité d'une critique, c'est-à-dire de déterminer si elle est positive ou négative. Pour ce faire, nous explorons différentes méthodes de représentation du texte et d'apprentissage automatique

Dans ce projet, nous avons appliqué l'analyse de sentiment à des critiques de films, qui contient 2000 critiques, dont 1000 positives et 1000 négatives. Notre objectif étant de construire un modèle capable de prédire la polarité d'une critique de film, en utilisant différentes méthodes de représentation du texte et de classification.

Nous avons opté pour l'utilisation de la méthode du sac de mots (BoW) afin d'extraire le vocabulaire présent dans les critiques de films. Nous avons ensuite procédé à une visualisation des mots les plus fréquents, des mots les plus discriminants et des mots les plus représentatifs du vocabulaire, en employant des techniques de nuages de mots.

Pour illustrer cette exploration, le graphique suivant présente les nuages de mots correspondant aux 100 termes les plus fréquents, en utilisant à la fois une représentation BoW binaire et une représentation TF-IDF, tout en excluant les stopwords. À travers cette représentation, nous pouvons observer que les mots les plus fréquemment rencontrés dans les critiques sont des termes généraux tels que "film", "movie", "like", etc. Ce constat initial a constitué une base essentielle pour notre analyse ultérieure.

Fig 1 : Bow Binaire, Taille du vocabulaire est de : 39354

Fig 2 : TF-IDF, Taille du vocabulaire est de : 39354

- **Prétraitement du texte**

Dans cette première phase de notre projet, nous avons initié le processus en prétraitant les textes des critiques de films afin de les simplifier et de les normaliser. Pour ce faire, nous avons conçu une fonction nommée preprocess_text, qui a pour rôle d'appliquer une série de prétraitements sur chaque texte en entrée. Ces prétraitements incluent :

- Conversion en minuscules pour assurer une uniformité dans la casse des mots.
- Suppression de la ponctuation, des chiffres et des caractères spéciaux pour ne conserver que les mots pertinents.
- Élimination des stop words pour réduire le bruit dans les données.
- Application du stemming, une technique visant à réduire les mots à leur racine, afin de regrouper les variantes d'un même mot.
- Lemmatisation, une méthode qui ramène les mots à leur forme canonique pour une représentation plus cohérente du vocabulaire.

De plus, nous avons ajouté un paramètre supplémentaire nommé keep_part, permettant de conserver uniquement une partie spécifique du texte, comme la première ou la dernière ligne, afin de réduire la longueur du texte ou de se concentrer sur les informations essentielles.

- **Extraction du vocabulaire**

Dans cette deuxième étape de notre démarche, nous avons procédé à l'extraction du vocabulaire à partir des textes prétraités en utilisant une représentation TF-IDF afin de réduire l'impact des mots trop fréquents ou trop rares, susceptibles de biaiser notre modèle. Cette méthode, bien qu'elle ne capture pas la syntaxe ou la pragmatique du texte, est efficace pour représenter le contenu sémantique des documents.

- *binary=True* : Pour obtenir des vecteurs binaires, indiquant simplement la présence ou l'absence des mots dans chaque document plutôt que leur fréquence.

- *stop_words=l_stop_words* : Pour exclure les mots vides en anglais, tout en conservant quelques exceptions jugées pertinentes et d'autres mots jugés nécessaires à conserver pour le contexte spécifique de notre étude.

- *ngram_range=(1, 2)* : Pour considérer à la fois les uni-grammes et les bi-grammes, c'est-à-dire les combinaisons de un ou deux mots consécutifs, afin de capturer davantage de contexte.

- *min_df=2* : Pour ne conserver que les mots qui apparaissent au moins deux fois dans le corpus, réduisant ainsi le bruit.

- *max_features=12500* : Pour limiter la taille du vocabulaire à 12500 mots, garantissant ainsi une représentation efficace sans compromettre les performances du modèle.

Cette sélection d'un ngramme de taille 1 à 2 a été faite dans le but de capturer un contexte plus large que les simples uni-grammes, tout en évitant une augmentation excessive de la taille du vocabulaire. En effet, au cours de nos expérimentations, nous avons observé que l'utilisation des features trop importantes sur

des ensembles de données de petite taille conduisait à des résultats médiocres. Ainsi, en fixant le nombre maximal de fonctionnalités à 12500, nous avons constaté des améliorations significatives dans les performances de notre modèle.

Nous avons également exploré d'autres variantes de la méthode du sac de mots pour affiner notre représentation du vocabulaire. Ces variantes comprennent :

- CountVectorizer: Cette méthode consiste en une représentation basique où chaque document est représenté par un vecteur de fréquences de mots.
- Réduction de la taille du vocabulaire : Nous avons testé différe0.ntes valeurs pour les paramètres min_df, max_df et max_features afin de réduire la taille du vocabulaire et améliorer les performances du modèle.
- CountVectorizer binaire : Cette approche indique simplement la présence ou l'absence des mots dans chaque document, sans considérer leur fréquence.
- Différents N-grammes : Nous avons expérimenté l'utilisation de diverses tailles de N-grammes pour capturer différents niveaux de contexte dans les documents.

- **Classification du texte**

La troisième étape cruciale de notre projet a été dédiée à la classification des textes des critiques de films, en utilisant la polarité comme variable cible. Pour ce faire, nous avons implémenté trois modèles de classification supervisée, chacun apprenant à prédire la classe d'un texte à partir de son vecteur de représentation :

- Le modèle Naïf Bayésien
- La Régression Logistique
- Le SVC Linéaire

Dans un souci de rigueur méthodologique, nous avons divisé nos données en deux ensembles distincts : un ensemble d'apprentissage, regroupant 80% des données, et un ensemble de test, contenant les 20% restants. Ce découpage des données a été effectué de manière aléatoire, à l'aide de la fonction train_test_split. Par la suite, nous avons employé la classe GridSearchCV de la bibliothèque scikit-learn pour rechercher les hyperparamètres optimaux de chaque modèle, en recourant à une validation croisée sur l'ensemble d'apprentissage.

- Pour le modèle Naïf Bayésien, nous avons exploré différentes valeurs du paramètre alpha, permettant de contrôler le lissage de Laplace afin d'éviter les probabilités nulles.

- Concernant la Régression Logistique, nous avons examiné diverses valeurs du paramètre C, qui régule la force de la régularisation L2, visant ainsi à prévenir le surapprentissage, ainsi que du paramètre solver, qui contrôle l'algorithme d'optimisation.

- En ce qui concerne le SVC Linéaire, nous avons évalué différentes valeurs des paramètres C et penalty, influençant respectivement la marge d'erreur et le type de régularisation.

Pour évaluer la performance de notre modèle, nous avons défini une fonction, evaluer_modele, qui utilise l'ensemble de test. Cette fonction calcule plusieurs métriques essentielles :

- La précision, qui mesure la proportion de prédictions correctes parmi l'ensemble des prédictions.

- La matrice de confusion, permettant de croiser les classes réelles et prédites, offrant ainsi une visualisation des erreurs de classification.

- Le rapport de classification, fournissant les mesures de précision, de rappel et de F1-score pour chaque classe, ainsi que la moyenne pondérée.

- La courbe ROC, qui trace le taux de vrais positifs en fonction du taux de faux positifs pour différents seuils de décision, permettant ainsi de calculer l'aire sous la courbe (AUC), mesurant la performance globale du modèle.

- L'interprétation des poids, permettant de comprendre quels sont les mots les plus influents sur la prédiction, en fonction de leur coefficient dans le modèle.

- Analyse des résultats de la recherche des hyperparamètres optimaux

Dans cette section, nous exposons en détail les résultats de la recherche des hyperparamètres optimaux pour chaque modèle de classification supervisée utilisé dans notre étude

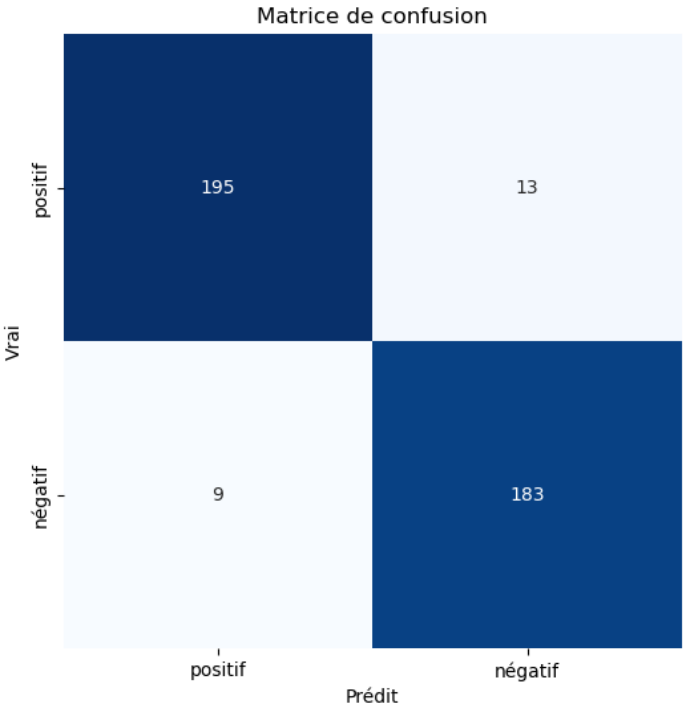
Naïf Bayésien : Lors de la recherche des hyperparamètres, le meilleur score de validation croisée a été de 0.816, correspondant à un paramètre alpha égal à 0.1.

Régression Logistique : Pour ce modèle, nous avons atteint un meilleur score de validation croisée de 0.91. Les paramètres optimaux associés sont C=1, penalty='l2', et solver='liblinear'.

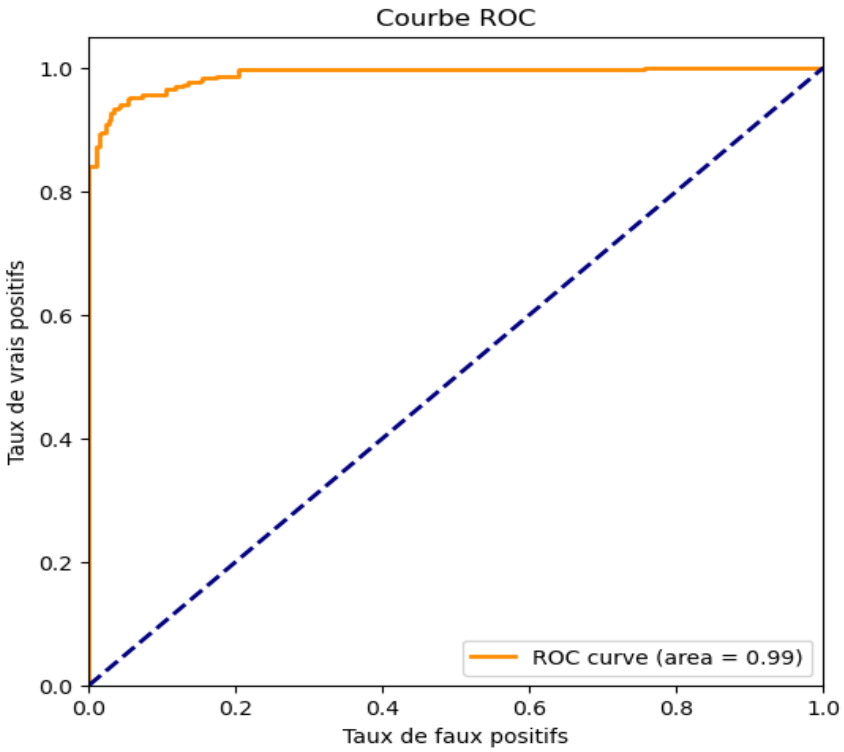
SVC Linéaire : Le modèle SVC linéaire a obtenu un score de validation croisée de 0.90, avec les paramètres C=1 et penalty='l2'.

Bien que le modèle de Régression Logistique ait présenté un score de validation croisée similaire à celui du SVC Linéaire, il convient de noter que ce dernier nécessite moins de paramètres à optimiser. Ainsi, afin de favoriser la simplicité et la facilité de mise en œuvre du modèle, nous avons choisi le SVC Linéaire comme modèle final pour notre étude.

Ci-dessous les métriques pertinentes telles que la précision, la matrice de confusion, le rapport de classification, la courbe ROC et l'interprétation des poids ont été calculées pour analyser la qualité des prédictions du modèle.



	precision	recall	f1-score	support
positif	0.96	0.94	0.95	208
négatif	0.93	0.95	0.94	192
accuracy			0.94	400
macro avg	0.94	0.95	0.94	400
weighted avg	0.95	0.94	0.95	400



Analyse de la reconnaissance de locuteur

Introduction

Dans cette seconde partie du projet, nous nous intéressons à reconnaître les énoncés prononcés par deux personnalités politiques françaises : Jacques Chirac et François Mitterrand. Nous disposons d'un jeu de données composé de 57 413 énoncés, dont 49 890 (86.9%) de Chirac et 7 523 (13.1%) de Mitterrand. Chaque énoncé est étiqueté avec le nom du locuteur ("M" pour Mitterrand, encodé à -1, ou "C" pour Chirac, encodé à 1). Notre objectif est de construire un modèle de machine learning capable de prédire correctement l'interlocuteur à partir d'un énoncé inconnu.

Pour cela, nous avons utilisé le modèle BoW dans un premier temps, qui représente les énoncés sous forme de vecteurs de fréquences de mots, pondérées par l'inverse de la fréquence documentaire (tf-idf). Nous avons ensuite appliqué des modèles de machine learning supervisés, comme la régression logistique et les SVM, pour apprendre à classifier les énoncés selon l'interlocuteur. Nous avons aussi exploré différentes stratégies d'entraînement, comme le sur-apprentissage, l'équilibrage des données, la courbe ROC et le lissage gaussien, pour améliorer la performance de notre modèle.

Ce rapport présente les étapes et les résultats de notre projet, en décrivant le prétraitement des données, les stratégies d'entraînement, les modèles de machine learning, les résultats expérimentaux

- **Prétraitement des données**

Avant d'appliquer les modèles de machine learning, nous avons effectué un prétraitement des données textuelles, afin de les rendre plus adaptées à l'analyse. Nous avons utilisé les mêmes pretraitement que sur la partie classification de sentiment a savoir la Stemminsation, la suppression des stopwords, les ponctuation et les caractère alphanumérique

Comme pour l'analyse précédente nous avons utilisé un TF-IDF en utilisant des n-grammes de taille 1 et 2 et un min_df de 1 et max_df de 95% et nous avons également limité la taille du vocabulaire a 25000 mots

- **Stratégies d'entraînement**

Pour entraîner les modèles de machine learning, nous avons dû faire face à quelques défis, comme celui d'éviter de prédire la classe majoritaire plus que celle de la classe minoritaire du fait du déséquilibre des données, comment splitter les données et comment éviter le sur-apprentissage. Nous avons donc exploré différentes stratégies d'entraînement, que nous décrivons ci-dessous.

- **Stratégies de split**

Pour garantir une évaluation précise de la performance de notre modèle de reconnaissance de locuteur, nous avons dû soigneusement considérer la manière de diviser nos données en ensembles d'apprentissage et de test. Cette étape est cruciale pour assurer la fiabilité des résultats obtenus.

Initialement, nous avons envisagé d'utiliser la méthode **train_test_split** en utilisant le paramètre **stratify** pour maintenir les proportions des classes dans l'ensemble de test. Cependant, nous avons constaté que l'utilisation de ce paramètre entraînait automatiquement un mélange des données, ce qui aurait pu compromettre la structure temporelle des énoncés. Cette structure temporelle est importante dans le contexte de la reconnaissance de locuteur.

Pour éviter de mélanger les énoncés et ainsi préserver leur ordre chronologique, nous avons décidé de ne pas utiliser le paramètre **stratify** avec **train_test_split**. Cependant, cela a posé un problème, car le jeu de données test risquait de ne contenir qu'une seule classe, ce qui aurait pu fausser l'évaluation de la performance du modèle.

Pour résoudre ce dilemme, nous nous sommes tournés vers une approche différente en utilisant la méthode **StratifiedKFold**. Cette méthode nous a permis de réaliser une division stratifiée de nos données en k plis, en veillant à ce que chaque pli maintienne les proportions des classes de manière équilibrée. En utilisant un k de 5, nous avons divisé nos données en cinq sous-ensembles, ce qui nous a permis de réaliser une validation croisée et d'obtenir une estimation robuste de la performance du modèle.

En outre, nous avons veillé à ne mélanger que les données d'entraînement lors de l'utilisation de **StratifiedKFold**, préservant ainsi l'ordre temporel des énoncés dans le jeu de test. Cette approche nous a permis de garantir une évaluation juste et impartiale de la performance de notre modèle, en utilisant un ensemble de test représentatif et en préservant la structure temporelle des énoncés. Cette préservation de la structure temporelle est cruciale, notamment dans l'optique de l'application ultérieure d'un filtre gaussien, où la chronologie des énoncés est essentielle.

- **Équilibrage des Données**

Un des défis majeurs que nous avons rencontrés dans notre projet de reconnaissance de locuteur est le déséquilibre significatif entre les classes. En effet, la classe Chirac représente environ 87% des énoncés, tandis que la classe Mitterrand ne représente que 13%. Ce déséquilibre peut entraîner un biais du modèle en faveur de la classe majoritaire et une mauvaise performance sur la classe minoritaire. Pour surmonter ce défi, nous avons exploré différentes stratégies d'équilibrage des données.

- **Sous-Échantillonnage et Sur-Échantillonnage**

Initialement, nous avons expérimenté le sous-échantillonnage en utilisant différentes méthodes telles que RandomUnderSampling, SMOTEENN et TomekLinks. Bien que le sous-échantillonnage ait permis d'obtenir de bons résultats sur l'ensemble de test, nous avons constaté une perte d'information significative. Lorsque nous avons soumis nos résultats sur le serveur, le rappel pour la classe minoritaire était d'environ 0.20, ce qui était insatisfaisant.

De même, le sur-échantillonnage n'a pas été efficace, car il aurait nécessité de dupliquer la classe minoritaire environ 7,6 fois, ce qui aurait été peu pratique et inefficace en termes de performance.

- **Combinaison de Sous-Échantillonnage et Sur-Échantillonnage**

Face à ces limitations, nous avons opté pour une approche combinée, consistant à sur-échantillonner la classe minoritaire avec un certain facteur défini, puis à sous-échantillonner également avec un seuil spécifique. Cette approche nous a permis de réduire la perte d'information tout en équilibrant les proportions des classes.

En plus du sous et sur echantillonnage des données, nous avons utilisé la pondération des classes pour pénaliser les erreurs de classification sur la classe minoritaire avec le paramètre **balanced** appliqué à **class_weight**

- **Paramètres de Régularisation**

Pour éviter le sur-apprentissage, nous avons utilisé la technique de la régularisation, qui consiste à ajouter un terme de pénalité à la fonction de coût. Nous avons ajusté les paramètres de régularisation pour chaque modèle afin de limiter la complexité et d'améliorer la généralisation du modèle.

- **Modèles de machine learning**

Nous avons utilisé deux types de modèles, la régression logistique et les SVM. Nous avons utilisé la classe GridSearchCV pour rechercher les meilleurs hyperparamètres pour chaque modèle, en utilisant la validation croisée. Nous avons défini une grille de paramètres pour chaque modèle, et nous avons choisi les paramètres qui maximisent le score moyen sur les jeux de test. Voici les paramètres les meilleurs paramètres que nous avons trouvés :

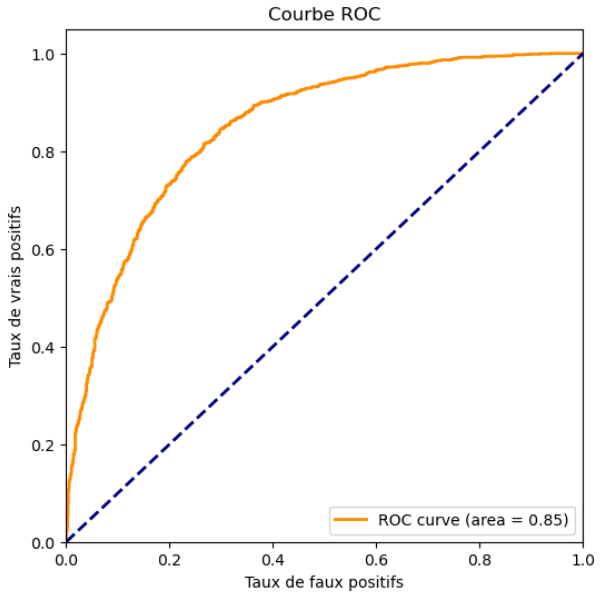
- Pour la régression logistique C=10, penalty='l2', class_weight='balanced'.
- Pour les SVM C=2, kernel='rbf', gamma='scale', class_weight='balanced'.

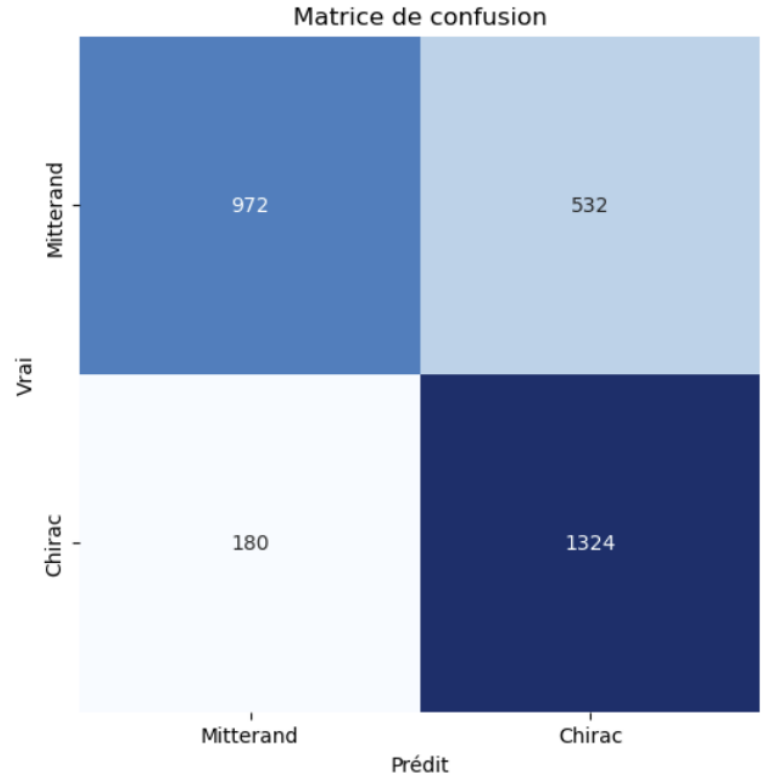
- **Résultats expérimentaux**

Nous avons commencé par entraîner nos modèles de machine learning, en utilisant les meilleurs hyperparamètres identifiés lors de la phase de recherche des hyperparamètres, principalement ceux de la régression logistique. Ensuite, nous avons évalué leurs performances sur un jeu de test distinct, composé de 11 482 énoncés, dont 9 978 de Chirac et 1 504 de Mitterrand.

Pour assurer une évaluation robuste, nous avons d'abord équilibré le jeu de test en utilisant la technique de sous-échantillonnage. Par la suite, nous avons employé plusieurs métriques pour évaluer la performance de nos modèles, notamment la précision, le rappel, le F1-score et l'AUC. Pour une meilleure compréhension des résultats, nous avons également généré des graphiques, incluant la matrice de confusion et la courbe ROC.

Les résultats obtenus avant l'application du lissage par un filtre gaussien sont les suivants :





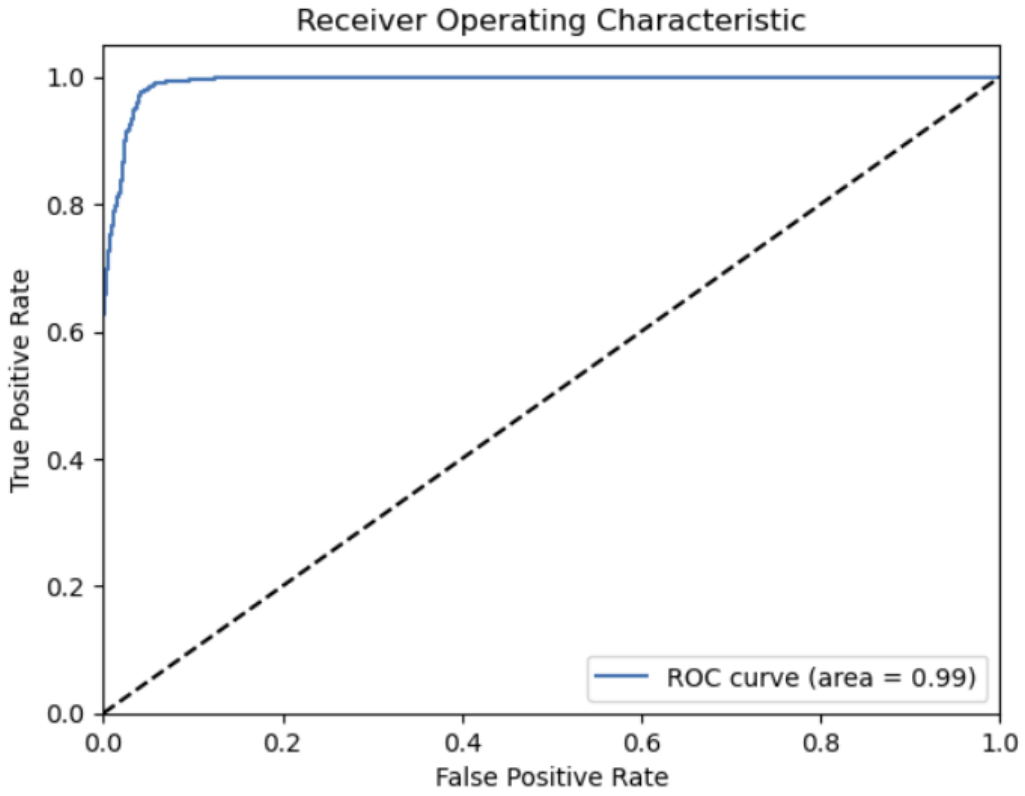
	precision	recall	f1-score	support
Mitterand	0.84	0.65	0.73	1504
Chirac	0.71	0.88	0.79	1504
accuracy			0.76	3008
macro avg	0.78	0.76	0.76	3008
weighted avg	0.78	0.76	0.76	3008

Nous avons observé un F1-score de 0.73 et une AUC de 0.85.

Suite à cela, nous avons appliqué un lissage gaussien sur les probabilités prédites par nos modèles. Cette étape visait à réduire le bruit et les fluctuations dans les prédictions. Nous avons utilisé la fonction **gaussian** de la bibliothèque Scipy pour générer un noyau gaussien de taille 11, avec un écart-type de 5. Après avoir normalisé le noyau pour qu'il somme à 1, nous avons effectué une convolution entre le noyau et les probabilités à l'aide de la fonction **np.convolve** de Numpy. Enfin, les probabilités lissées ont été transformées en prédictions binaires en utilisant un seuil de 0.5.

Les résultats après l'application du lissage gaussien ont été significativement améliorés, comme indiqué ci-dessous :

	precision	recall	f1-score	support
Mitterand	1.00	0.81	0.89	1504
Chirac	0.84	1.00	0.91	1504
accuracy			0.90	3008
macro avg	0.92	0.90	0.90	3008
weighted avg	0.92	0.90	0.90	3008

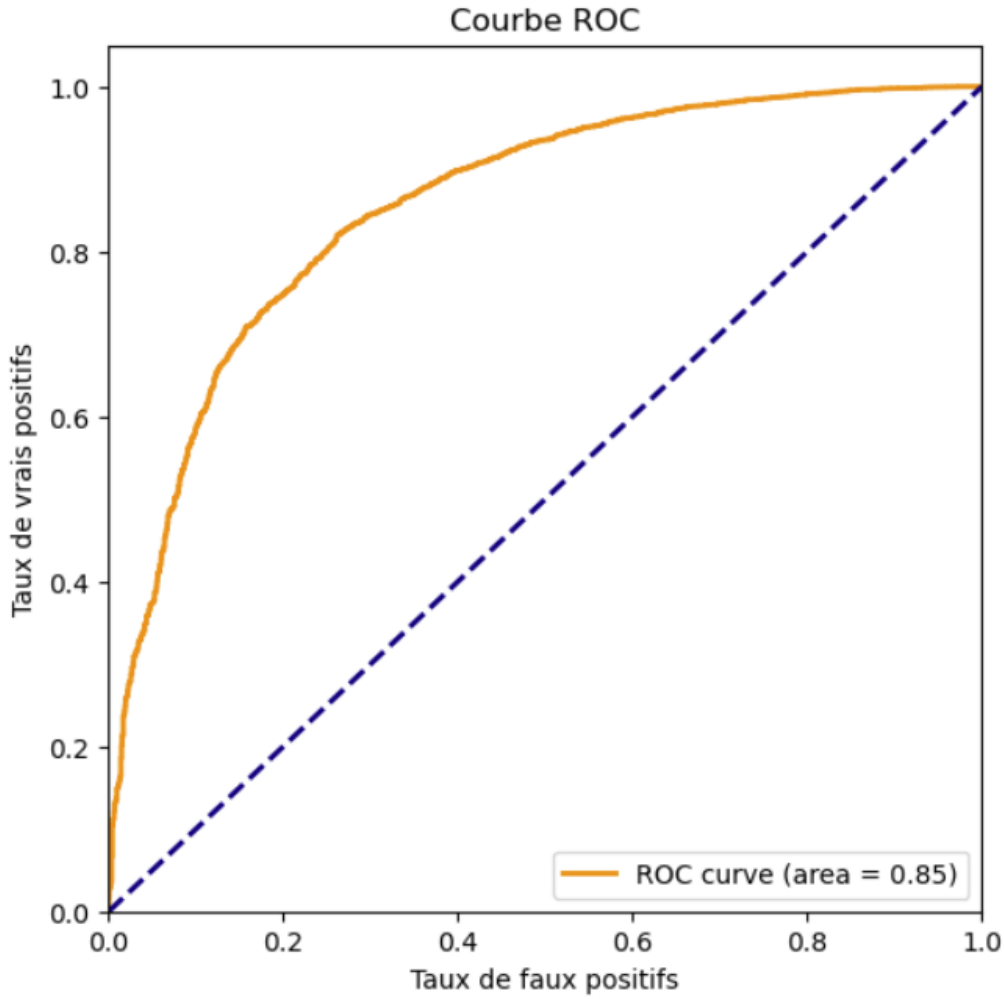


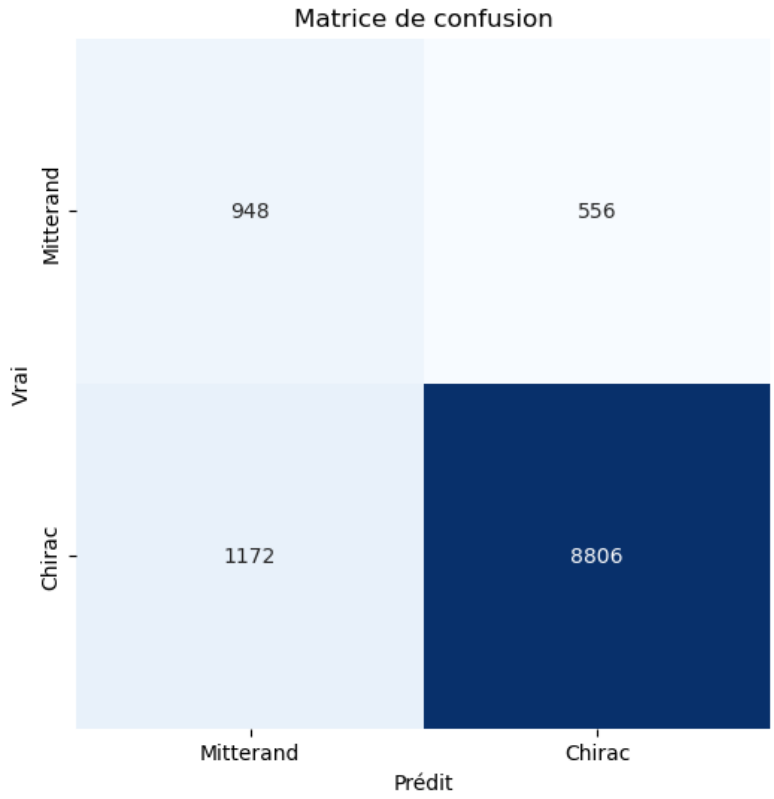
Le F1-score est passé de 0.73 à 0.89 et l'AUC de 0.85 à 0.99, démontrant ainsi l'efficacité du filtrage gaussien pour améliorer les performances de nos modèles.

Cependant, il est crucial de noter que les résultats précédents ont été générés à partir d'un jeu de test équilibré, ce qui peut ne pas refléter fidèlement la réalité, étant donné que le jeu de test final sur lequel notre modèle sera évalué présente un déséquilibre significatif entre les classes. Afin de mieux appréhender les conditions réelles, nous avons également testé nos modèles sur un jeu de test déséquilibré, comme décrit ci-dessous : Nous avons utilisé un jeu de test composé de 11 482 énoncés, dont 9 978 de Chirac et 1 504 de Mitterrand.

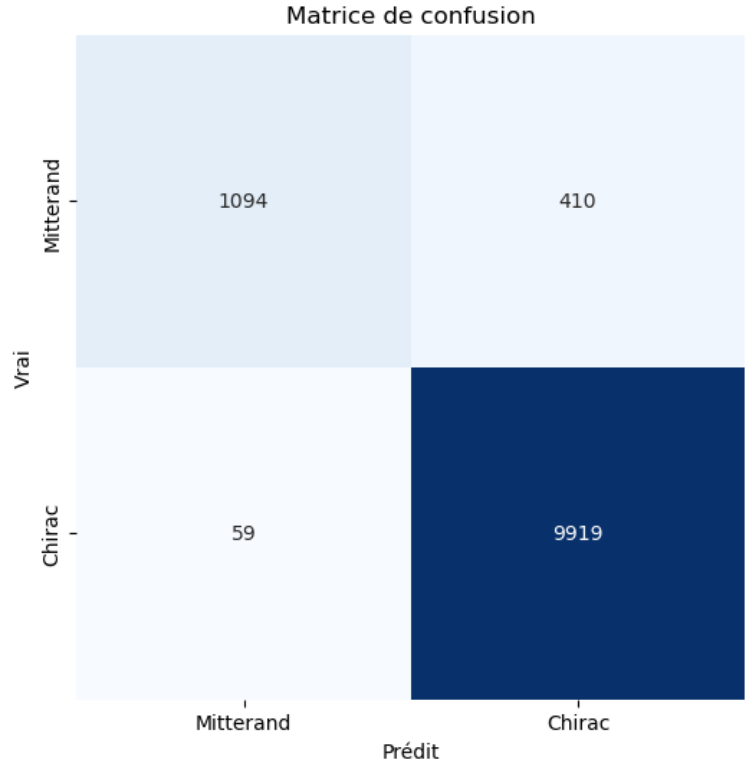
Les performances sur ce jeu de test déséquilibré ont montré un F1-score de 0.52 pour Mitterrand.

	precision	recall	f1-score	support
Mitterand	0.45	0.63	0.52	1504
Chirac	0.94	0.88	0.91	9978
accuracy			0.85	11482
macro avg	0.69	0.76	0.72	11482
weighted avg	0.88	0.85	0.86	11482

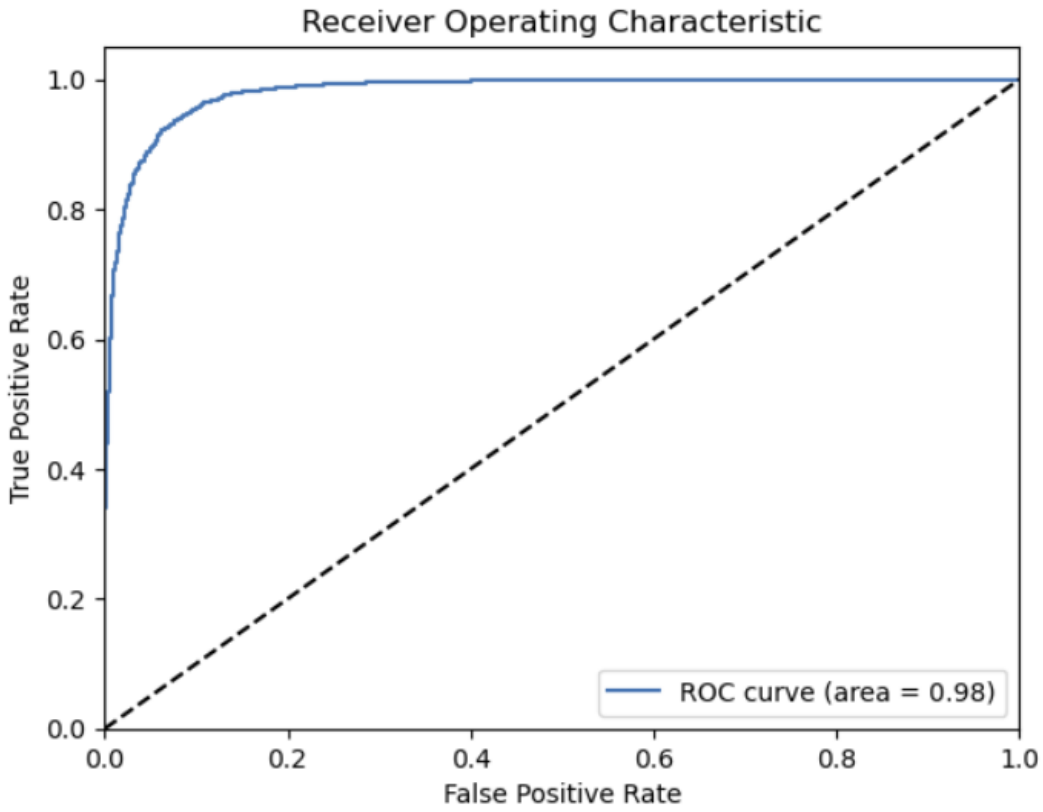




Toutefois, après l'application du lissage gaussien, les performances se sont améliorées, avec un F1-score passant de 0.52 à 0.82 et une AUC de 0.85 à 0.98 (voir les résultats ci-dessous).



	precision	recall	f1-score	support
Mitterand	0.95	0.73	0.82	1504
Chirac	0.96	0.99	0.98	9978
accuracy			0.96	11482
macro avg	0.95	0.86	0.90	11482
weighted avg	0.96	0.96	0.96	11482



Variante 2 : utilisation de Doc2Vec

Dans cette variante, nous avons exploré l'utilisation du modèle Doc2Vec pour représenter les énoncés. Ce modèle, une extension de Word2Vec, apprend des représentations vectorielles non seulement des mots, mais aussi des documents, en associant à chaque document un vecteur unique. Voici les étapes que nous avons suivies :

1. Création des documents pour Doc2Vec : Chaque énoncé a été transformé en un objet TaggedDocument, comprenant une liste de mots et un identifiant unique. Nous avons appliqué la même fonction de prétraitement que pour le modèle BoW pour normaliser les mots et éliminer les éléments indésirables.
2. Entraînement du modèle Doc2Vec : Nous avons instancié un objet Doc2Vec en définissant plusieurs paramètres :

- **vector_size** : la dimension des vecteurs de document et de mot, fixée à 500.
- **window** : la taille de la fenêtre de contexte, fixée à 20.
- **min_count** : la fréquence minimale des mots, fixée à 5.
- **sample** : le seuil de fréquence pour le sous-échantillonnage des mots fréquents, fixé à 0.001.
- **workers** : le nombre de threads utilisés pour l'entraînement, fixé à 3.
- **cbow_mean** : le mode de calcul de la moyenne des vecteurs de contexte, fixé à 1 (moyenne arithmétique).
- **epochs** : le nombre d'itérations sur le corpus, fixé à 5.

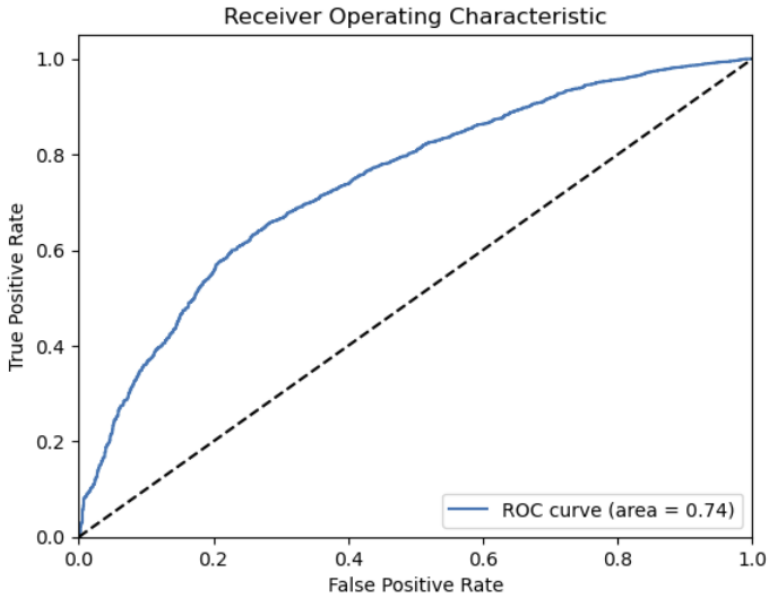
Nous avons ensuite construit le vocabulaire du modèle à partir des documents étiquetés et entraîné le modèle.

3. Obtention des vecteurs de document : Nous avons utilisé la méthode **infer_vector** du modèle pour obtenir le vecteur de document correspondant à chaque énoncé prétraité. Cela nous a donné une matrice X_doc2vec de taille (n_samples, n_features), où n_samples est le nombre d'énoncés et n_features est le nombre de caractéristiques (500).

Nous avons ensuite suivi le même processus que pour le modèle BoW pour diviser les données, équilibrer les données, entraîner les modèles de machine learning, rechercher les hyperparamètres optimaux, évaluer le modèle et prédire les énoncés. Les modèles utilisés étaient les mêmes (régression logistique et SVM) avec les mêmes paramètres que pour le modèle BoW. Les mêmes métriques et graphiques ont également été utilisés.

Malheureusement, les résultats obtenus avec Doc2Vec étaient nettement inférieurs à ceux de BoW. Même après le filtrage, les performances demeuraient insatisfaisantes par rapport au modèle BoW, comme le montre le résultat ci-dessous :

	precision	recall	f1-score	support
Mitterand	0.25	0.64	0.36	1504
Chirac	0.93	0.71	0.81	9978
accuracy			0.70	11482
macro avg	0.59	0.68	0.58	11482
weighted avg	0.84	0.70	0.75	11482



Variante 3 : Utilisation de BERT

Dans cette variante, nous avons exploré l'utilisation du modèle BERT pour représenter les énoncés. Cependant, en raison de contraintes de temps liées à l'utilisation d'un CPU, nous n'avons pas pu obtenir de résultats pour comparer avec les autres modèles.

Conclusion

Ce projet d'analyse de sentiment et de reconnaissance de locuteur a été une exploration passionnante des différentes techniques de traitement du langage naturel et de modélisation de machine learning. En utilisant le modèle Bag-of-Words (BoW), nous avons pu obtenir des résultats prometteurs dans la classification des énoncés prononcés par Jacques Chirac et François Mitterrand. Les performances obtenues avec BoW, notamment après l'application du lissage gaussien, ont démontré la capacité de cette approche à identifier efficacement l'interlocuteur à partir des énoncés.

Cependant, notre tentative d'explorer une variante en utilisant le modèle Doc2Vec s'est soldée par des performances décevantes. Bien que nous soyons conscients que cela puisse être attribuable à des erreurs de mise en œuvre ou à des choix de paramètres suboptimaux, il est clair que des recherches supplémentaires et une meilleure compréhension de ce modèle seraient nécessaires pour en tirer pleinement profit dans le cadre de cette tâche de reconnaissance de locuteur.

Concernant le modèle BERT, bien que nous n'ayons pas pu obtenir des résultats concrets en raison de contraintes de temps, nous croyons en son potentiel pour cette tâche. En tant que modèle de langage profond pré-entraîné sur de larges corpus, BERT est capable de capturer des nuances sémantiques et contextuelles complexes, ce qui en fait une option prometteuse pour la représentation des énoncés dans notre contexte. Une future exploration de BERT pourrait ainsi offrir des résultats plus précis et informatifs, contribuant ainsi à enrichir notre compréhension de la reconnaissance de locuteur.

Pour les axes d'amélioration, nous suggérons d'approfondir notre compréhension des modèles de représentation tels que Doc2Vec et BERT. Une optimisation plus fine des paramètres et des stratégies d'entraînement pourrait être nécessaire pour améliorer les performances de ces modèles. De plus, une exploration plus approfondie des méthodes d'équilibrage des données et de régularisation pourrait aider à atténuer les défis posés par le déséquilibre entre les classes et le sur-apprentissage.

Enfin, il est important de souligner l'importance de ce projet dans notre parcours académique et professionnel. En plus de nous avoir permis d'explorer diverses techniques de traitement du langage naturel et de modélisation de machine learning, ce projet nous a également sensibilisés aux défis et aux opportunités de la reconnaissance de locuteur. Ces connaissances et expériences acquises seront précieuses dans notre parcours futur, que ce soit dans le domaine de la recherche ou dans la résolution de problèmes réels dans le domaine du traitement du langage naturel.

Rapport sur les TME (2-4)

TME 2

- **Partie A:**

Ce TME se concentre sur le traitement de séquences en utilisant des modèles HMM (Hidden Markov Models) et CRFs (Conditional Random Fields) dans le domaine du traitement automatique du langage naturel. Dans la section sur les HMMs, les données d'étiquetage de parties du discours (POS) sont chargées à partir du corpus, et un modèle de base est créé en utilisant un dictionnaire simple pour mapper chaque mot à son étiquette POS la plus fréquente dans l'ensemble d'entraînement. L'algorithme de Viterbi est ensuite utilisé pour le décodage des séquences de test, et les performances du modèle HMM sont évaluées en termes de précision de classification. Une analyse qualitative des HMMs est ensuite entreprise, comprenant la visualisation des paramètres du modèle, des matrices de confusion et l'identification d'exemples correctement corrigés par le décodage de Viterbi. Dans la section sur les CRFs, les modèles discriminatifs sont introduits pour la distribution conditionnelle de probabilité des étiquettes étant donné les observations. La bibliothèque NLTK est utilisée pour entraîner et évaluer un modèle CRF sur les données d'étiquetage POS.

- **Partie B:**

La représentation en sac de mots des données textuelles est structurée sous forme d'une matrice où chaque ligne correspond à un document et chaque colonne correspond à un mot unique du vocabulaire. Nous avons examiné comment prétraiter et vectoriser les données textuelles en utilisant l'approche TF-IDF.

Nous explorons divers algorithmes de regroupement, notamment K-means, l'Analyse Sémantique Latente (LSA) et l'Allocation de Dirichlet Latente (LDA), pour regrouper les documents similaires en fonction de leurs représentations en sac de mots.

Après avoir appliqué K-means, nous analysons les clusters qualitativement en examinant les mots les plus importants pour chaque cluster et quantitativement en calculant des métriques telles que la pureté, le score Rand et le score Rand ajusté.

Ensuite, nous utilisons LSA, qui implique de décomposer la matrice en sac de mots en utilisant la Décomposition en Valeurs Singulières (SVD). Nous effectuons un regroupement sur la représentation de basse dimension résultante et évaluons les clusters en utilisant des analyses qualitatives et quantitatives similaires à celles de K-means. Enfin, nous appliquons LDA à la matrice en sac de mots et effectuons un regroupement, suivi d'une évaluation à l'aide de métriques telles que la pureté et les scores Rand.

Évaluation des Performances :

Nous pouvons observer que les performances des différents algorithmes varient selon les métriques quantitatives évaluées. En termes de pureté, qui mesure la qualité des clusters en fonction de la distribution des étiquettes de classe, LDA a obtenu le score le plus élevé, suivi de près par LSA sans normalisation L2, puis par K-Means et LSA avec normalisation L2. En ce qui concerne les scores Rand et Rand Ajusté, qui mesurent la similarité entre les clusters obtenus et les étiquettes de classe réelles, LDA a également surpassé K-Means et LSA, indiquant ainsi une meilleure qualité de regroupement dans l'ensemble.

TME *3*

- **Partie A:**

Dans ce tme, nous avons exploré l'utilisation de modèles de Word2Vec pour la classification de sentiment des avis de films en anglais. Tout d'abord, nous avons chargé un ensemble de données de 25 000 avis et avons effectué une analyse rapide pour déterminer le nombre de critiques positives et négatives. Ensuite, nous avons entraîné un modèle Word2Vec sur ces avis afin de capturer les relations sémantiques entre les mots. Nous avons évalué la similarité sémantique et syntaxique du modèle en utilisant un ensemble de données spécial contenant des analogies de mots. De plus, nous avons utilisé le modèle Word2Vec pré-entraîné sur le corpus Google News pour représenter les avis de films sous forme de vecteurs. Ces vecteurs ont ensuite été utilisés pour entraîner un modèle de régression logistique pour prédire le sentiment des avis. Le modèle de régression logistique a obtenu une précision d'environ 83,38% sur le jeu de données de test, ce qui indique une performance satisfaisante dans la classification des avis de films en fonction de leur sentiment.Enfin, nous avons comparé les performances des modèles Word2Vec Skip-gram (SG) et Continuous Bag of Words (CBOW) dans la classification des avis de films.Avec les méthodes de moyenne et de sommation, le modèle Skip-gram montre une précision légèrement supérieure à celle du modèle CBOW, atteignant respectivement 82,06 % et 82,68 % contre 77,54 % et 77,20 %. Cependant, avec les méthodes de maximum et minimum, le modèle CBOW affiche une meilleure performance par rapport au modèle Skip-gram. Ces résultats soulignent l'importance de comprendre les nuances des modèles Word2Vec et des méthodes d'agrégation pour choisir la meilleure approche en fonction du contexte spécifique de la tâche.

- **Partie B:**

Nous avons utilisé un modèle d'embedding pré-entraîné pour représenter les mots dans nos documents. En cas de mots inconnus, nous avons généré des vecteurs aléatoires et les avons intégrés à un dictionnaire pour une utilisation ultérieure. Ensuite, nous avons traité les étiquettes associées à nos données, les reliant à des identifiants uniques pour simplifier l'entraînement de modèles d'apprentissage automatique. Nous avons ensuite utilisé un modèle de régression logistique pour classer les mots en fonction de leurs vecteurs d'embedding, et avons évalué la précision de ce modèle sur un ensemble de données de test. Nous avons défini des fonctions pour extraire des caractéristiques des données textuelles, telles que la capitalisation et la position des mots, afin de préparer nos données à l'entraînement de modèles de TALN. Enfin, nous avons évalué les performances d'un marqueur CRF en utilisant différentes fonctions de caractéristiques. Les résultats montrent que l'ajout de caractéristiques structurales aux embeddings de mots améliore nettement les performances du marqueur CRF. Avec cette combinaison, la précision atteint environ 95,83%, surpassant celle obtenue avec les embeddings de mots seuls (91,40%) ou les caractéristiques structurales uniquement (92,93%). Ces résultats soulignent l'importance de prendre en compte à la fois les informations sémantiques et contextuelles pour obtenir une meilleure précision dans le marquage CRF.

TME 4

- **Partie A:**

Nous explorons des différentes architectures de réseaux de neurones, y compris les RNN, GRU et LSTM. Nous expérimentons avec les dimensions d'embedding et la longueur des chunks de texte pour évaluer leur impact sur les performances du modèle. Ensuite, nous procédons à l'entraînement du modèle, où nous observons comment il apprend progressivement à prédire les caractères suivants en se basant sur les caractères précédents. Enfin, nous testons la capacité du modèle à générer du texte en lui fournissant un début de phrase et en lui permettant de compléter le reste. Cette étape nous permet d'évaluer la qualité de la génération de texte produite par le modèle entraîné.Les résultats du modèle RNN offrent un aperçu intéressant de sa capacité à reproduire des motifs linguistiques. Cependant, le texte généré présente des incohérences grammaticales et des séquences parfois peu naturelles. Pour améliorer la qualité du texte généré avec un RNN, l'utilisation d'architectures de réseaux neuronaux plus avancées, telles que les LSTM ou les GRU, ainsi que l'ajustement des hyperparamètres tels que le taux d'apprentissage, la taille du lot (batch size) ou la taille de la couche cachée, peut jouer un rôle crucial dans l'amélioration des performances du modèle.

- **Partie B:**

Dans cette partie, nous explorons plusieurs étapes de l'analyse de sentiment en utilisant BERT, un modèle de traitement du langage naturel. Tout d'abord, nous téléchargeons un ensemble de données de critiques de films et utilisons le modèle BERT, ainsi que la bibliothèque transformers, pour encoder chaque critique en une représentation vectorielle, en nous concentrant sur le jeton [CLS], fréquemment employé pour la classification de texte.Ensuite, nous entraînons un modèle de régression logistique sur les embeddings extraits afin de prédire les polarités de sentiment. Après évaluation des performances de ce modèle, le code passe à l'étape de fine-tuning de BERT pour la classification de sentiment. Pour cela, un modèle BERT est téléchargé et ajusté sur les données de critiques de films.

De plus, une étape essentielle consiste à entraîner un modèle de régression logistique sur les embeddings extraits, sans fine-tuner BERT lui-même. Cette démarche permet de comparer les performances du modèle de régression logistique avec celui de BERT fine-tuné par la suite dans le code. Enfin, la dernière partie du processus met en œuvre la boucle d'entraînement du modèle fine-tuné, en utilisant la fonction d'optimisation Adam pour ajuster les poids du modèle et évaluer sa précision sur les ensembles de données d'entraînement et de test après chaque époque d'entraînement.