Projet TAL

Classification de texte

Hocine Kadem 21309534

Neil Benahmed 21200977

Date : [14/02/2024]

Table des matières

1 Analyse des sentiments

1.1 Transformation paramétrique du texte (pre-traitements)

Dans cette section, nous avons appliqué une série de prétraitements au texte afin de le nettoyer et de le préparer pour l'analyse des sentiments. Les étapes de prétraitement comprennent :

- **Suppression des caractères non normalisés** : Nous avons utilisé la fonction `unicodedata.normalize` pour normaliser les caractères non ASCII en caractères ASCII, ce qui permet de supprimer les accents et autres caractères spéciaux.
- **Transformation en minuscules** : Nous avons converti l'ensemble du texte en minuscules pour assurer une normalisation cohérente et éviter les duplications basées sur la casse.
- **Suppression de la ponctuation** : Nous avons utilisé la fonction `str.maketrans` pour créer une table de traduction qui supprime la ponctuation et les caractères de saut de ligne, de retour chariot et de tabulation.
- **Suppression des chiffres** : Nous avons utilisé l'expression régulière `re.sub` pour supprimer tous les chiffres du texte.
- **Lemmatisation (optionnel)** :on a également voulu effectuer une lemmatisation en utilisant la bibliothèque `spacy`. Le texte est lemmatisé en utilisant un modèle de langue anglaise chargé au préalable

Voici quelques exemples concret de pré-traitements pour notre base d'entrainements :

- **Avant pré-traitements** : what would you do if no one could see you ? well , if you're a super smart bio-molecular research scientist working for the military , you'd grope a co-worker and rough-up your neighbor from across the street .

- **Aprés pré-traitements sans lemmatisation** : what would you do if no one could see you well if youre a super smart biomolecular research scientist working for the military youd grope a coworker and roughup your neighbor from across the street

- **Aprés pré-traitements et lemmatisation** : what would you do if no one could see you well if you re a super smart biomolecular research scientist work for the military you d grope a coworker and roughup your neighbor from across the street

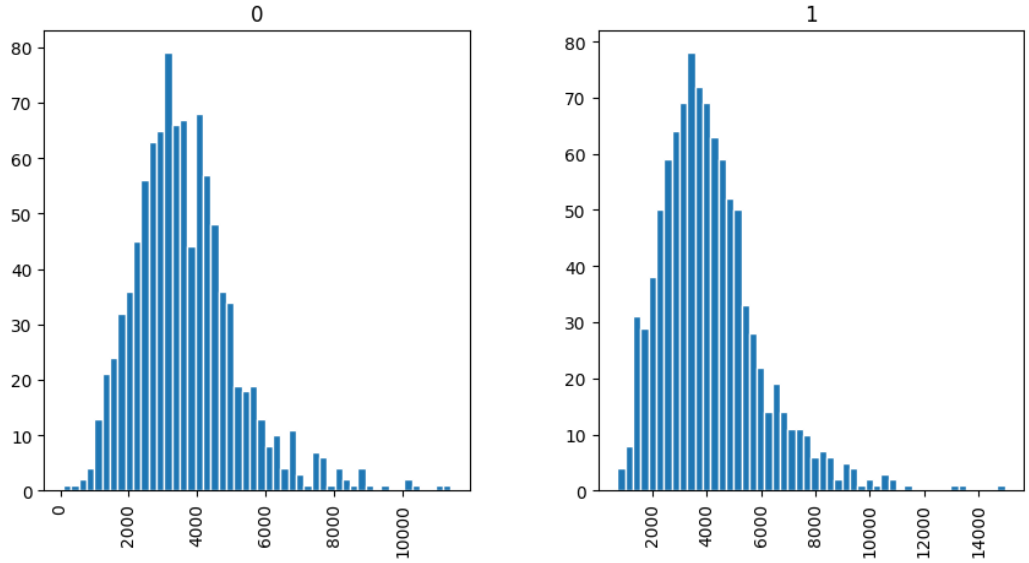
1.2 Extraction du vocabulaire (BoW)

1.2.1 Exploration préliminaire des jeux de données

Nous commençons par une exploration préliminaire de nos jeux de données en nous concentrant sur les caractéristiques du vocabulaire. Nous répondons aux questions suivantes :

- **Taille d'origine du vocabulaire** : Nous déterminons la taille initiale du vocabulaire avant toute transformation ou réduction qui est de : 39659.

— **Distribution des classes et rapartition en fonction de la taille des avis** : Nous examinons la longueur des documents en fonction de leurs classes. Cela nous permettra de mieux comprendre si la longueur des textes varie en fonction de la catégorie à laquelle ils appartiennent.



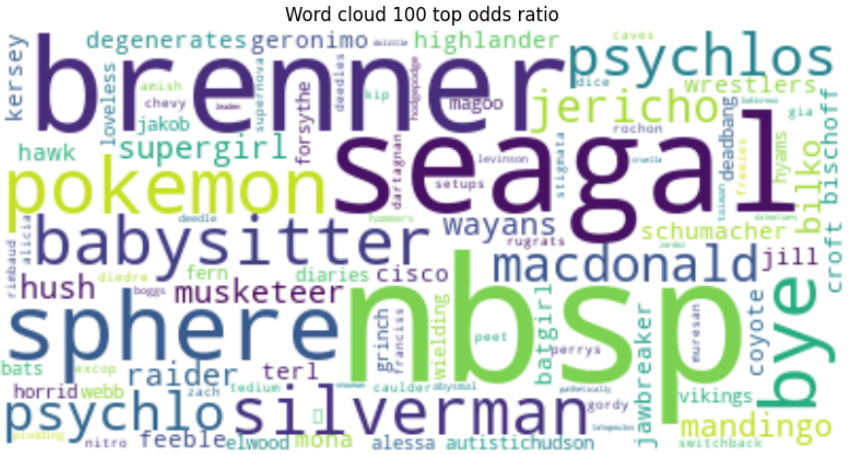
— **Les 100 mots les plus fréquents** : Nous extrayons les 100 mots les plus fréquents dans notre corpus et créons un nuage de mots pour visualiser leur importance relative avec et sans les stopwords.

FIGURE 1 – Légende de l'image

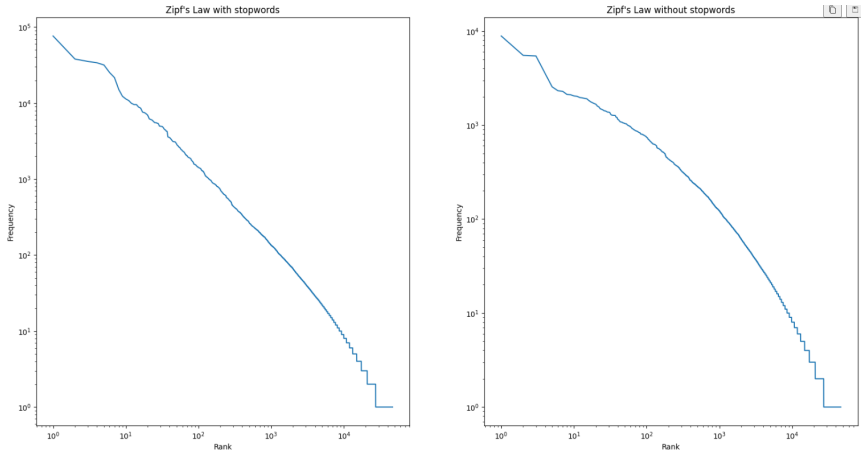
— **Les 100 mots les plus fréquents par classes** :Nous extrayons les 100 mots les plus fréquents dans notre corpus par classes : Regardant les mots les plus fréquent en commun, on peut remarquer que la grande majorité n'est pas un facteur descriminant, Sur 200 mots les plus fréquent 167 mots se retrouve etre des mots en commun.

FIGURE 2 – Légende de l'image

— **Les 100 mots les plus discriminants** : Nous calculons le rapport de cotes (odds ratio) pour chaque mot afin de déterminer ceux qui sont les plus discriminants. Les mots avec les valeurs les plus élevées de rapport de cotes sont considérés comme les plus discriminants,



— **Distribution d'apparition des mots** : Nous examinons la distribution d'apparition des mots dans notre corpus, en nous intéressant notamment à la loi de Zipf.



— **Les 100 bigrammes/trigrammes les plus fréquents** : Nous identifions les bigrammes/-trigrammes les plus fréquents dans notre corpus, ce qui peut nous donner des informations sur les combinaisons de mots les plus courantes.

1.2.2 Variantes de BoW

Après avoir exploré les caractéristiques de base du vocabulaire, nous nous intéressons à différentes variantes du sac de mots (BoW) pour améliorer notre représentation. Nous examinons les variantes suivantes :

— **TF-IDF** : Nous utilisons la pondération TF-IDF pour attribuer des poids aux mots en

fonction de leur fréquence dans le document et de leur fréquence inverse dans l'ensemble du corpus.

— **Réduction de la taille du vocabulaire** : Nous explorons différentes techniques pour réduire la taille du vocabulaire, telles que la suppression des mots qui apparaissent trop fréquemment (min_df), la suppression des mots qui apparaissent trop rarement (max_df), et la limitation du nombre total de mots (max_features).

— **BoW binaire** : Au lieu d'utiliser les fréquences des mots, nous créons une représentation binaire où chaque mot est représenté par un 1 s'il est présent et un 0 s'il est absent.

— **Bi-grammes, tri-grammes** : Nous considérons également les combinaisons de mots en utilisant des bi-grammes et des tri-grammes, ce qui peut capturer des informations supplémentaires sur la structure des phrases, on essayera aussi d'inclure les stopwords ou pas.

— **Performances et avantages/inconvénients des variantes** : Nous discuterons des performances que nous pouvons attendre de ces variantes et des avantages et inconvénients associés à chacune d'entre elles dans la phase d'apprenstissage et de tests.

1.3 Modèles de Machine Learning

1.3.1 Métriques d'évaluation

Il est essentiel d'utiliser des métriques d'évaluation appropriées en fonction de la tâche et de l'équilibrage des données. Dans notre cas, nous pouvons utiliser les métriques suivantes :

— **Accuracy (Précision)** : Il s'agit d'une mesure courante pour évaluer les performances d'un modèle de classification. L'accuracy représente la proportion de prédictions correctes par rapport au nombre total d'échantillons. Cependant, l'accuracy peut être trompeuse si les classes sont déséquilibrées, car un modèle peut être précis dans la prédiction de la classe majoritaire mais échouer dans la prédiction des classes minoritaires.

— **Courbe ROC (Receiver Operating Characteristic) et AUC (Area Under the Curve)** : Ces métriques sont couramment utilisées pour évaluer la performance des modèles de classification binaire. La courbe ROC représente la relation entre le taux de vrais positifs (TPR) et le taux de faux positifs (FPR) en faisant varier le seuil de classification. L'AUC mesure la surface sous la courbe ROC et fournit une mesure agrégée de la performance d'un modèle. Une valeur d'AUC proche de 1 indique une bonne performance de classification.

— **F1-score** : Le F1-score est une mesure de la précision et du rappel d'un modèle de classification. Il est calculé comme la moyenne harmonique de la précision et du rappel. Le F1-score est souvent utilisé lorsque les données sont déséquilibrées, car il prend en compte à la fois les faux positifs et les faux négatifs. Un F1-score élevé indique une bonne harmonie entre la

précision et le rappel du modèle.

En utilisant ces métriques, nous serons en mesure d'évaluer la performance de notre modèle de classification de manière complète et équilibrée, en tenant compte de la précision, du rappel et des spécificités des différentes classes. Cela nous permettra d'avoir une vision globale de l'efficacité de notre modèle et de prendre des décisions éclairées en fonction des résultats obtenus.

1.3.2 Gestion des hyperparametres et des differentes combinaisons

La gestion des hyperparamètres est une étape clé dans la construction et l'optimisation des modèles de machine learning. Les hyperparamètres sont des paramètres réglables qui influencent les performances et le comportement du modèle. Trouver les meilleures combinaisons d'hyperparamètres peut être un défi, car cela nécessite d'explorer un espace multidimensionnel.

Pour résoudre ce problème, nous avons utilisé une technique appelée recherche de grille (Grid Search). La recherche de grille consiste à spécifier une grille prédéfinie de valeurs pour chaque hyperparamètre que nous souhaitons ajuster. Ensuite, nous entraînons et évaluons le modèle pour chaque combinaison d'hyperparamètres de la grille.

Pour évaluer les performances du modèle, nous avons utilisé une métrique d'évaluation appropriée, telle que la précision, le rappel ou l'indice F1. Nous avons utilisé la validation croisée pour obtenir une estimation plus robuste des performances du modèle en découpant les données en plusieurs ensembles d'entraînement et de validation.

Une fois que toutes les combinaisons d'hyperparamètres ont été évaluées, nous avons sélectionné la combinaison qui a donné les meilleures performances selon la métrique choisie. Cette combinaison est considérée comme la meilleure configuration pour le modèle.

La recherche de grille est une méthode systématique pour trouver les meilleures combinaisons d'hyperparamètres, mais elle peut être coûteuse en termes de temps de calcul, car elle nécessite l'entraînement et l'évaluation de plusieurs modèles. Cependant, elle nous permet d'optimiser les performances de nos modèles en choisissant les meilleurs hyperparamètres.

On utilisera pour notre recherche en grille une pipeline sera constituer de la sorte :

— Vectorizer : Soit TFIDFVectorizer soit CountVectorizer
— Classifieur : parmi : Regression logistique, SVM, Random Forest, Naive Bayes
— optionnel : on pourra rajouter un traitement supplementaire sur les données comme reduction de dimension avec des tméthodes comme SVD ou meme PCA.

1.3.3 Méthodologie

Dans notre analyse de classification nous avons suivi les étapes suivantes :

1. **Prétraitement des données** : Avant d'entraîner nos modéles, nous avons effectué plusieurs étapes de prétraitement des données. Cela comprenait la suppression des caractères spéciaux,

des chiffres, la mise en minuscules, la suppression des stopwords (mots couramment utilisés mais peu informatifs) et la lemmatisation (réduction des mots à leur forme de base).

2. **Vectorisation des données** : Nous avons utilisé deux techniques de vectorisation différentes : CountVectorizer et TfidfVectorizer. Ces méthodes ont permis de convertir les documents textuels en vecteurs numériques, qui servent d'entrée pour les modéles. CountVectorizer compte simplement le nombre d'occurrences de chaque mot dans le texte, tandis que TfidfVectorizer pondère ces occurrences en fonction de leur importance dans le corpus.
3. **Entraînement et évaluation du modèle** : Nous avons divisé nos données en un ensemble d'entraînement et un ensemble de test. L'ensemble d'entraînement a été utilisé pour entraîner les modèles. Nous avons utilisé la validation croisée (cross-validation) pour évaluer les performances des modèles avec différentes combinaisons d'hyperparamètres.
4. **Analyse** : Nous avons exploré divers modèles d'apprentissage automatique tels que la régression logistique et le SVM, en ajustant différents ensembles d'hyperparamètres pour chaque modèle. Nous avons ensuite évalué leurs performances en utilisant un ensemble de données de test distinct afin d'analysé les résultats en examinant des métriques telles que la précision, le rappel, la F-mesure et la courbe ROC.
5. **Sélection des hyperparamètres avec GridSearch** : Nous avons aussi utilisé une recherche de grille (GridSearchCV) pour effectuer une recherche systématique des meilleures combinaisons d'hyperparamètres. Les hyperparamètres que nous avons ajustés pour les vectorizers comprenaient le nombre de mots maximum (max_features), la plage de n-grammes (ngram_range) et l'utilisation de stopwords (stop_words), et variées pour les classifieurs selont leur types
6. **Évaluation des performances** : Pour évaluer les performances des modèle, nous avons utilisé plusieurs mesures, notamment la précision, le rappel et le f1-score. Nous avons également examiné le rapport de classification pour obtenir des informations détaillées sur la capacité du modèle à classifier chaque classe.
7. **Comparaison des résultats** : Nous avons comparé les performances des modèles avec CountVectorizer et TfidfVectorizer pour identifier la méthode de vectorisation la plus efficace. Nous avons examiné les scores de f1 pour chaque classe et les métriques moyennes pondérées pour évaluer la capacité globale du modèle à classifier les données.

1.3.4 Regression logistique

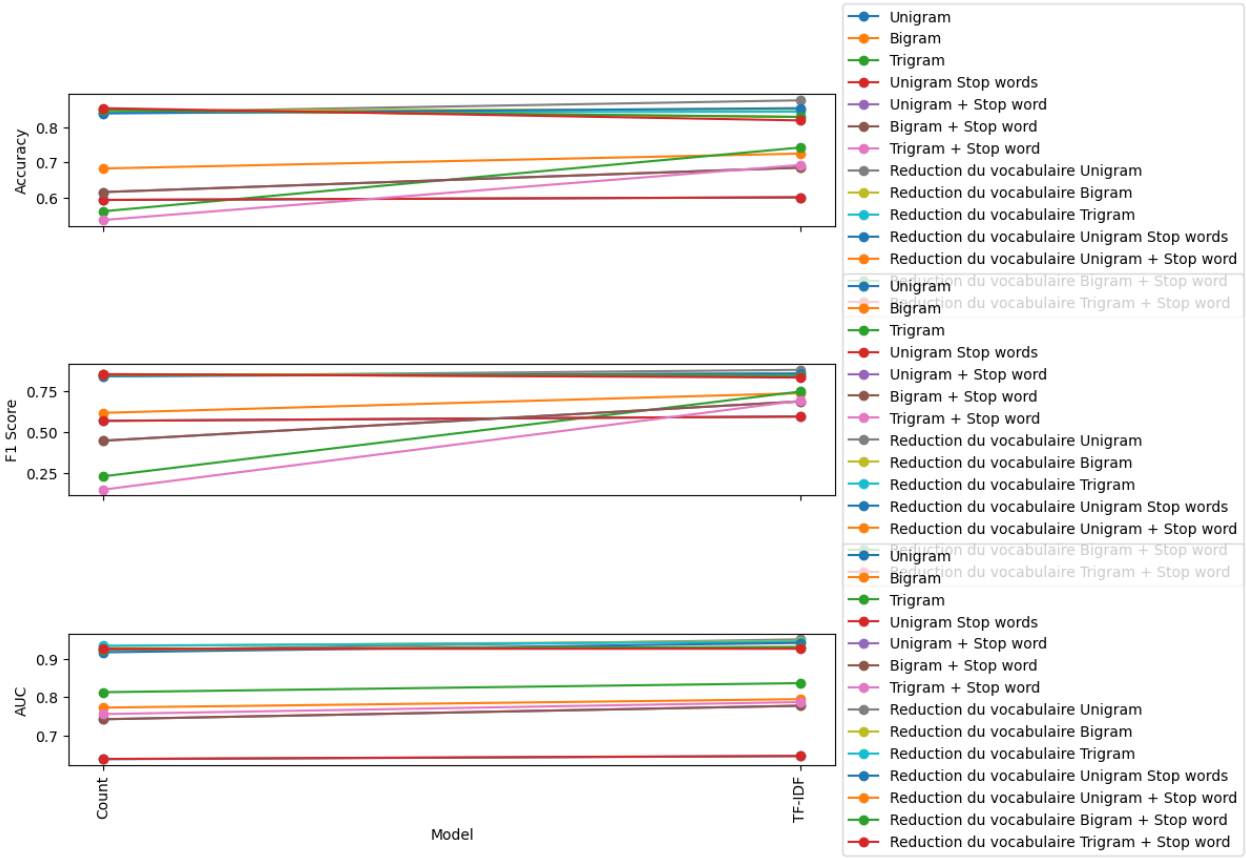
Dans cette section, nous nous concentrons sur l'analyse de classification effectuée à l'aide de la régression logistique. L'objectif de cette analyse était de développer un modèle de régression logistique capable de classifier efficacement les données dans des catégories prédéfinies

1.3.4.1 Analyse avec différents modèles et paramètres

Nous avons testé l'analyse avec différents modèles qui ont plusieurs paramètres différents. Voici les résultats de notre analyse avec différents modèles et paramètres (Concernant la réduction du vocabulaire, nous l'avons réduit a 70%) :

TABLE 1 – Résultats de l'analyse avec différents modèles et paramètres

Modèle	**Vectorizer**	**Précision**	**Rappel**	**AUC**
Unigram	Count	0.5925	0.567639	0.637753
Unigram	TF-IDF	0.6000	0.593909	0.645716
Bigram	Count	0.6825	0.616314	0.772319
Bigram	TF-IDF	0.7250	0.736842	0.794370
Trigram	Count	0.5600	0.228070	0.812545
Trigram	TF-IDF	0.7425	0.746929	0.836146
Unigram Stop words	Count	0.5925	0.567639	0.638253
Unigram Stop words	TF-IDF	0.6000	0.593909	0.646216
Unigram + Stop word	Count	0.6150	0.446043	0.741919
Unigram + Stop word	TF-IDF	0.6850	0.686567	0.777144
Bigram + Stop word	Count	0.6150	0.446043	0.741919
Bigram + Stop word	TF-IDF	0.6850	0.686567	0.777144
Trigram + Stop word	Count	0.5350	0.146789	0.755444
Trigram + Stop word	TF-IDF	0.6925	0.691729	0.786720
Réduction du vocabulaire Unigram	Count	0.8425	0.840506	0.922023
Réduction du vocabulaire Unigram	TF-IDF	0.8775	0.879607	0.950349
Réduction du vocabulaire Bigram	Count	0.8475	0.848635	0.932848
Réduction du vocabulaire Bigram	TF-IDF	0.8525	0.858513	0.946674
Réduction du vocabulaire Trigram	Count	0.8425	0.843672	0.933773
Réduction du vocabulaire Trigram	TF-IDF	0.8450	0.853081	0.944274
Réduction du vocabulaire Unigram Stop words	Count	0.8400	0.841584	0.916698
Réduction du vocabulaire Unigram Stop words	TF-IDF	0.8550	0.857143	0.941199
Réduction du vocabulaire Unigram + Stop word	Count	0.8500	0.850000	0.924848
Réduction du vocabulaire Unigram + Stop word	TF-IDF	0.8300	0.839623	0.930398
Réduction du vocabulaire Bigram + Stop word	Count	0.8500	0.850000	0.924848
Réduction du vocabulaire Bigram + Stop word	TF-IDF	0.8300	0.839623	0.930398
Réduction du vocabulaire Trigram + Stop word	Count	0.8550	0.853535	0.925848
Réduction du vocabulaire Trigram + Stop word	TF-IDF	0.8200	0.832558	0.925973



Les résultats de notre analyse avec différents modèles et paramètres sont présentés dans le tableau ci-dessus. Nous avons testé plusieurs combinaisons de modèles (Unigram, Bigram, Trigram),Vectorizer (Count, TF-IDF) et prétraitements (avec/sans stop words, réduction de vocabulaire) pour évaluer leurs performances.

Les résultats montrent que les modèles Bigram avec TF-IDF ont obtenu les meilleures performances en termes de précision, rappel et AUC, avec une précision de 0.7250, un rappel de 0.736842 et un AUC de 0.794370. Cela suggère que la prise en compte des paires de mots consécutifs (bigrammes) et l'utilisation de la méthode TF-IDF pour pondérer les termes ont été bénéfiques pour la tâche d'analyse.

D'autre part, les modèles Trigram avec CountVectorizer ont obtenu les résultats les moins performants, avec une précision de 0.5600, un rappel de 0.228070 et un AUC de 0.812545. Cela pourrait indiquer que la prise en compte des triplets de mots consécutifs (trigrammes) n'a pas été aussi efficace que les modèles unigramme et bigramme.

En ce qui concerne les prétraitements, l'ajout de stop words n'a pas eu un impact significatif sur les performances des modèles. Cependant, la réduction du vocabulaire a conduit à une amélioration notable des performances, en particulier pour les modèles unigrammes avec CountVectorizer et TF-IDF, où les précisions étaient respectivement de 0.8425 et 0.8775, les rappels de 0.840506 et 0.879607, et les AUC de 0.922023 et 0.950349.

En conclusion, les modèles Bigram avec TF-IDF et les modèles unigrammes avec réduction de vocabulaire ont été les plus performants pour notre tâche d'analyse. Cependant, il est important de noter que les performances peuvent varier en fonction du jeu de données et de la tâche spécifique, et il est recommandé de continuer à explorer d'autres combinaisons de modèles et de paramètres pour obtenir des résultats encore meilleurs.

1.3.4.2 Grid Search : Résultats de la régression logistique avec CountVectorizer

Les résultats du rapport de classification sont les suivants :

Le modèle utilisé pour la classification est LogisticRegression, avec le vectorizer CountVectorizer. Les meilleurs paramètres trouvés sont :

— **max_features** : 10000
— **ngram_range** : (1,2) representant un ensemble de 1gram et bi gram
— **stop_words** : en gardant les stopwords

Le meilleur score F1 en validation croisée est de 0.83875

Pour la classe 0, la précision est de 0.83, le rappel est de 0.87 et le F1-score est de 0.85, avec un support de 193 échantillons. Pour la classe 1, la précision est de 0.87, le rappel est de 0.84 et le F1-score est de 0.86, avec un support de 207 échantillons.

Classe	Précision	Rappel	F1-score	Support
0	0.83	0.84	0.84	193
1	0.85	0.84	0.84	207
Accuracy			0.84	400
Macro Avg	0.84	0.84	0.84	400
Weighted Avg	0.84	0.84	0.84	400

1.3.4.3 Grid Search : Résultats de la régression logistique avec TfidfVectorizer

le vectorizerr TfIdf dans tout les tests suivant on l'initialisera avec le paramétre sublinear_-tf=True graces à plusieurs éxperiences on conclu que les performances sont globalement meilleurs que sans, Lorsque l'on applique la mise à l'échelle sublinéaire (sublinear tf scaling) sur les termes fréquents, on remplace la fréquence du terme (tf) par 1+log(tf). Cette transformation vise à atténuer l'impact des termes très fréquents dans la représentation vectorielle d'un document, car ces termes peuvent dominer les autres termes moins fréquents, même s'ils ne sont pas nécessairement les plus informatifs. En prenant le logarithme de la fréquence du terme, on réduit la différence entre les

termes fréquents et les termes moins fréquents, ce qui permet une meilleure pondération des termes dans le modèle. (Voir l'article : Improving TF-IDF with Singular Value Decomposition (SVD) for Feature Extraction on Twitter)

Les résultats du rapport de classification sont les suivants :

Le modèle utilisé pour la classification est LogisticRegression, avec le vectorizer TfidfVectorizer. Les meilleurs paramètres trouvés sont :
— **max_features** : 10000
— **ngram_range** : (1,3) representant un ensemble de 1gram bi gram et tri gram
— **stop_words** : en gardant les stopwords

Le meilleur score F1 en validation croisée est de 0.8625.

Pour la classe 0, la précision est de 0.83, le rappel est de 0.87 et le F1-score est de 0.85, avec un support de 193 échantillons. Pour la classe 1, la précision est de 0.87, le rappel est de 0.84 et le F1-score est de 0.86, avec un support de 207 échantillons.

Globalement, le modèle a obtenu une précision et un rappel élevés pour les deux classes, ce qui indique une bonne performance de classification. L'accuracy globale du modèle est de 0.85, ce qui signifie que 85% des échantillons ont été correctement classés.

Ces résultats démontrent une performance solide du modèle dans la classification des données.

Classe	**Précision**	**Rappel**	**F1-score**	**Support**
0	0.83	0.87	0.85	193
1	0.87	0.84	0.86	207
Accuracy			0.85	400
Macro Avg	0.85	0.85	0.85	400
Weighted Avg	0.85	0.85	0.85	400

1.3.4.4 Comparaisons et conclusion

D'après les résultats listés précedemment, on peut constater que le modèle utilisant TfidfVectorizer a obtenu un score F1 légèrement supérieur à celui utilisant CountVectorizer.

1.3.5 Support vector machines

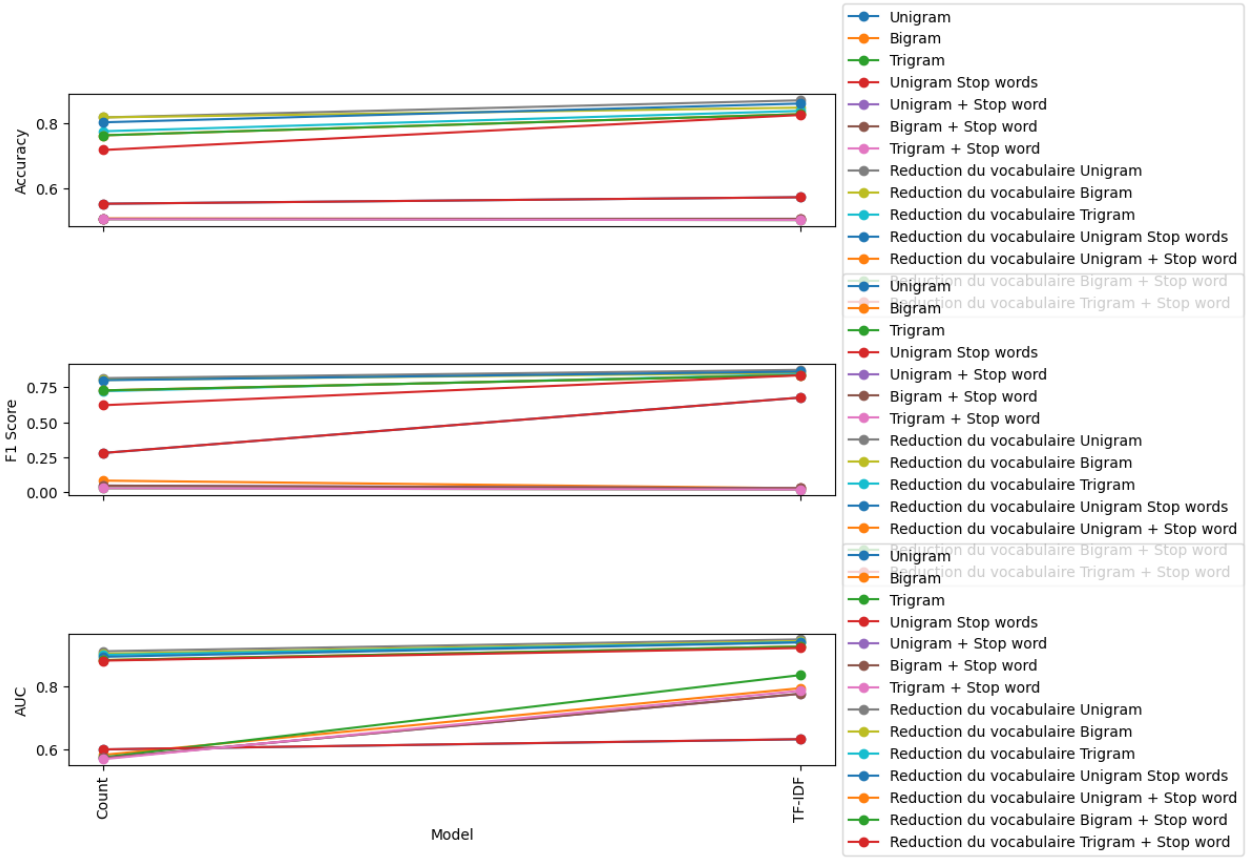
Dans cette continuité, après avoir exploré l'analyse de classification avec la régression logistique, nous nous tournons maintenant vers l'utilisation du classifieur SVM (Support Vector Machine)

1.3.5.1 Analyse avec différents modèles et paramètres

Nous avons testé l'analyse avec différents modèles qui ont plusieurs paramètres différents. Voici les résultats de notre analyse avec différents modèles et paramètres (Concernant la réduction du vocabulaire, nous l'avons réduit a 70%) :

TABLE 2 – Résultats de l'analyse avec différents modèles et paramètres

Modèle	**Vectorizer**	**Précision**	**Rappel**	**AUC**
Unigram	Count	0.5525	0.281124	0.598527
Unigram	TF-IDF	0.5725	0.676749	0.630953
Bigram	Count	0.5075	0.083721	0.581415
Bigram	TF-IDF	0.5050	0.029412	0.793720
Trigram	Count	0.5050	0.029412	0.573789
Trigram	TF-IDF	0.5025	0.019704	0.835471
Unigram Stop words	Count	0.5525	0.281124	0.598827
Unigram Stop words	TF-IDF	0.5725	0.676749	0.631378
Unigram + Stop word	Count	0.5050	0.048077	0.570414
Unigram + Stop word	TF-IDF	0.5050	0.029412	0.776019
Bigram + Stop word	Count	0.5050	0.048077	0.570414
Bigram + Stop word	TF-IDF	0.5050	0.029412	0.776019
Trigram + Stop word	Count	0.5050	0.029412	0.567764
Trigram + Stop word	TF-IDF	0.5025	0.019704	0.785545
Réduction du vocabulaire Unigram	Count	0.8175	0.815190	0.911148
Réduction du vocabulaire Unigram	TF-IDF	0.8700	0.874396	0.948949
Réduction du vocabulaire Bigram	Count	0.8175	0.803235	0.903323
Réduction du vocabulaire Bigram	TF-IDF	0.8475	0.853012	0.943549
Réduction du vocabulaire Trigram	Count	0.7750	0.722222	0.899547
Réduction du vocabulaire Trigram	TF-IDF	0.8375	0.843373	0.940399
Réduction du vocabulaire Unigram Stop words	Count	0.8025	0.801008	0.893597
Réduction du vocabulaire Unigram Stop words	TF-IDF	0.8600	0.862069	0.939348
Réduction du vocabulaire Unigram + Stop word	Count	0.7625	0.727794	0.882997
Réduction du vocabulaire Unigram + Stop word	TF-IDF	0.8275	0.836879	0.926423
Réduction du vocabulaire Bigram + Stop word	Count	0.7625	0.727794	0.882997
Réduction du vocabulaire Bigram + Stop word	TF-IDF	0.8275	0.836879	0.926423
Réduction du vocabulaire Trigram + Stop word	Count	0.7175	0.622074	0.881397
Réduction du vocabulaire Trigram + Stop word	TF-IDF	0.8250	0.835681	0.921698



Les résultats de notre analyse avec différents modèles et paramètres sont présentés dans le tableau ci-dessus. Nous avons testé plusieurs combinaisons de modèles (Unigram, Bigram, Trigram), Vectorizer (Count, TF-IDF) et prétraitements (avec/sans stop words, réduction de vocabulaire) pour évaluer leurs performances.

Les résultats montrent que les modèles Unigram avec réduction de vocabulaire et méthode TF-IDF ont obtenu les meilleures performances en termes de précision, rappel et AUC. Ces résultats suggèrent que la combinaison d'un modèle Unigram avec une réduction de vocabulaire et une pondération TF-IDF peut être efficace pour cette tâche d'analyse.

D'autre part, l'utilisation des stop words n'a pas semblé avoir un impact significatif sur les performances des modèles. Cependant, la réduction de vocabulaire a montré des améliorations notables, en particulier avec les modèles Unigram.

Il convient également de noter que les modèles Bigram et Trigram ont généralement obtenu des performances inférieures par rapport au modèle Unigram.

En conclusion, les résultats de cette analyse suggèrent que le modèle Unigram avec réduction de vocabulaire et pondération TF-IDF est une approche prometteuse pour l'analyse de texte.

1.3.5.2 Résultats du classifieur SVM avec CountVectorizer

Les meilleurs paramètres trouvés lors de l'optimisation sont les suivants :

pour le classifier :
— C : 0.1
— class_weight : None
— degree : 2
— gamma : 0.1
— kernel : linear
— probability : False
pour le vectorizer :
— max_features : None
— ngram_range : (1, 2)
— stop_words : None
Le meilleur score F1 obtenu lors de la validation croisée était de 0.828125.
Les résultats sur les données de tests :

Classe	**Précision**	**Rappel**	**F1-score**	**Support**
0	0.81	0.84	0.83	193
1	0.84	0.82	0.83	207
Accuracy			0.83	400
Macro Avg	0.83	0.83	0.83	400
Weighted Avg	0.83	0.83	0.83	400

1.3.5.3 Résultats du classifieur SVM avec TfidfVectorizer

Le classifieur SVM a été utilisé avec la méthode de vectorisation TfidfVectorizer et les meilleurs paramètres trouvés lors de l'optimisation sont les suivants :

pour le classifieur :

— C : 10
— class_weight : None
— degree : 2
— gamma : 0.1
— kernel : rbf
— probability : False
pour le vectorizer :

— max_features : 5000
— ngram_range : (1, 3)
— stop_words : None

Le meilleur score F1 obtenu lors de la validation croisée était de 0.864.
Les résultats sur les données de tests :

Classe	**Précision**	**Rappel**	**F1-score**	**Support**
0	0.83	0.87	0.85	194
1	0.87	0.83	0.85	206
Accuracy			0.85	400
Macro Avg	0.85	0.85	0.85	400
Weighted Avg	0.85	0.85	0.85	400

1.3.5.4 Comparaisons et conclusion

D'après les résultats listés précedemment, on peut constater que le modèle utilisant TfIdfVectorizer (ensemble de 1gram et bi gram et tri gram) a obtenu un score F1 légèrement supérieur à celui utilisant CountVectorizer.

1.3.6 Random Forest

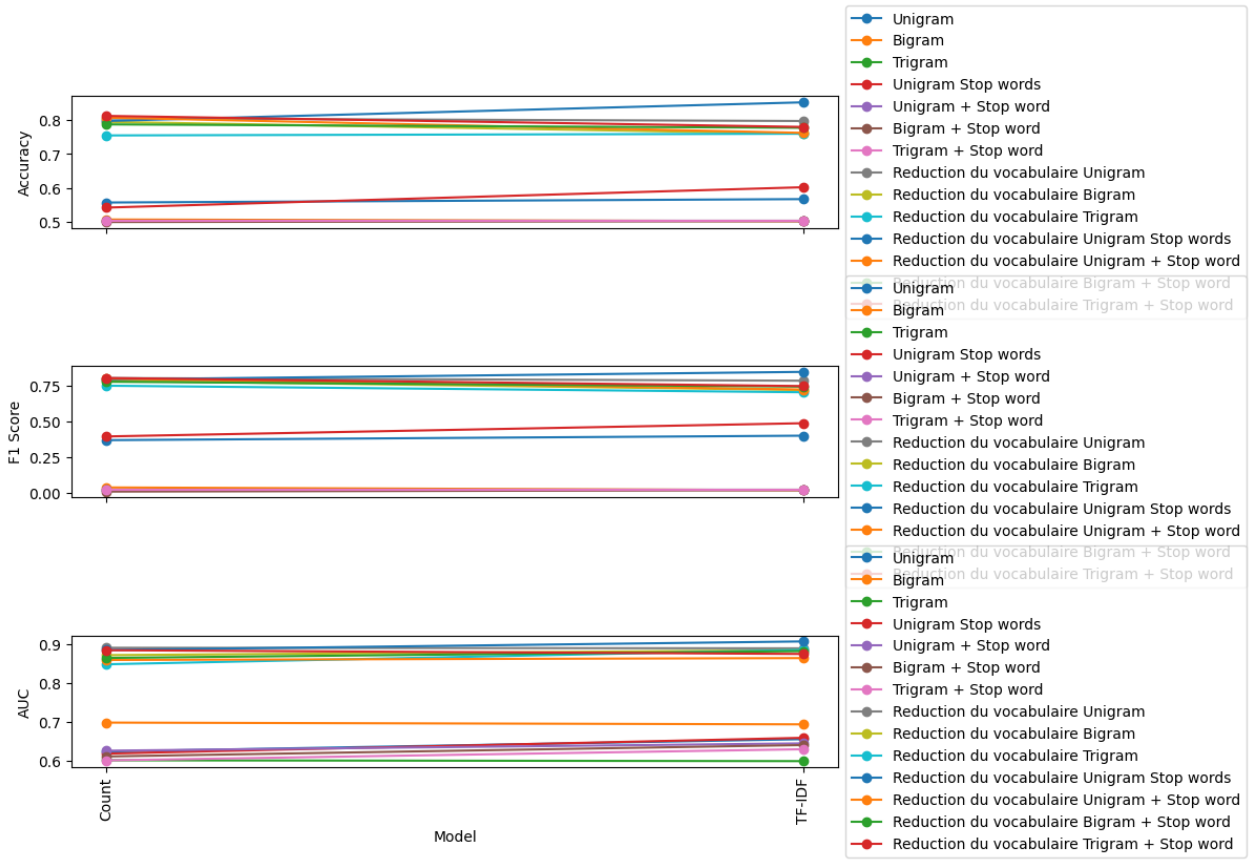
après avoir exploré l'analyse de classification avec la régression logistique et le classifieur SVM, nous nous tournons maintenant vers l'utilisation de l'algorithme Random Forest pour compléter notre étude comparative.

1.3.6.1 Analyse avec différents modèles et paramètres

Nous avons testé l'analyse avec différents modèles qui ont plusieurs paramètres différents. Voici les résultats de notre analyse avec différents modèles et paramètres (Concernant la réduction du vocabulaire, nous l'avons réduit a 70%) :

Type	Model	Accuracy	F1 Score	AUC
Unigram	Count	0.5575	0.370107	0.622853
Unigram	TF-IDF	0.5675	0.401384	0.654729
Bigram	Count	0.5075	0.039024	0.697492
Bigram	TF-IDF	0.5025	0.019704	0.692967
Trigram	Count	0.5025	0.019704	0.600353
Trigram	TF-IDF	0.5025	0.019704	0.598165
Unigram Stop words	Count	0.5425	0.396040	0.617540
Unigram Stop words	TF-IDF	0.6025	0.488746	0.658304
Unigram + Stop word	Count	0.5025	0.019704	0.625041
Unigram + Stop word	TF-IDF	0.5025	0.019704	0.643966
Bigram + Stop word	Count	0.5000	0.009901	0.610190
Bigram + Stop word	TF-IDF	0.5025	0.019704	0.639653
Trigram + Stop word	Count	0.5025	0.019704	0.600053
Trigram + Stop word	TF-IDF	0.5025	0.019704	0.628903
Reduction du vocabulaire Unigram	Count	0.8050	0.805000	0.890260
Reduction du vocabulaire Unigram	TF-IDF	0.7975	0.786280	0.888810
Reduction du vocabulaire Bigram	Count	0.7950	0.786458	0.871272
Reduction du vocabulaire Bigram	TF-IDF	0.7600	0.720930	0.884010
Reduction du vocabulaire Trigram	Count	0.7550	0.751269	0.847746
Reduction du vocabulaire Trigram	TF-IDF	0.7600	0.705521	0.884910
Reduction du vocabulaire Unigram Stop words	Count	0.7975	0.794937	0.884560
Reduction du vocabulaire Unigram Stop words	TF-IDF	0.8525	0.848329	0.906648
Reduction du vocabulaire Unigram + Stop word	Count	0.8075	0.801034	0.858859
Reduction du vocabulaire Unigram + Stop word	TF-IDF	0.7625	0.724638	0.863659
Reduction du vocabulaire Bigram + Stop word	Count	0.7875	0.779221	0.864059
Reduction du vocabulaire Bigram + Stop word	TF-IDF	0.7775	0.744986	0.881697
Reduction du vocabulaire Trigram + Stop word	Count	0.8125	0.805195	0.884072
Reduction du vocabulaire Trigram + Stop word	TF-IDF	0.7800	0.748571	0.874322

TABLE 3 – Performances des modèles et configurations



En analysant les données fournies, on peut constater que les performances des différents modèles varient en fonction du type de n-gramme utilisé, du modèle de représentation (Count ou TF-IDF), ainsi que de l'inclusion ou de l'exclusion des stop words et de la réduction du vocabulaire.

Les modèles utilisant des unigrammes semblent généralement obtenir de meilleurs résultats que ceux utilisant des bigrammes ou des trigrammes. Parmi les modèles utilisant des unigrammes, ceux qui incluent les stop words semblent légèrement meilleurs que ceux qui les excluent.

En ce qui concerne les modèles de représentation, les résultats montrent une légère préférence pour ceux basés sur TF-IDF, qui obtiennent généralement des scores d'accuracy, de F1 Score et d'AUC légèrement plus élevés que les modèles basés sur le Count.

En ce qui concerne la réduction du vocabulaire, les modèles avec réduction semblent obtenir de meilleurs résultats que ceux sans réduction. Les modèles avec réduction du vocabulaire obtiennent des scores d'accuracy, de F1 Score et d'AUC plus élevés.

1.3.6.2 Résultats du classifieur random forest avec CountVectorizer

Les résultats du rapport de classification sont les suivants :

Les meilleurs paramètres trouvés sont :

Pour le vectorizer :
— Paramètre : max_features, Valeur : 10000
— Paramètre : ngram_range, Valeur : (1, 2)
— Paramètre : stop_words, Valeur : 'english'
pour le classifier :
— Paramètre : max_depth, Valeur : 10
— Paramètre : min_samples_leaf, Valeur : 1
— Paramètre : min_samples_split, Valeur : 10
— Paramètre : n_estimators, Valeur : 500

En utilisant ces paramètres, la meilleure métrique F1 obtenue lors de la validation croisée est de 0.86.

et on retrouve ces résultats pour les données de tests :

Classe	**Précision**	**Rappel**	**F1-score**	**Support**
0	0.87	0.86	0.86	194
1	0.87	0.84	0.87	206
Accuracy			0.87	400
Macro Avg	0.87	0.87	0.87	400
Weighted Avg	0.87	0.87	0.87	400

1.3.6.3 Résultats du classifieur random forest avec TfidfVectorizer

Les résultats du rapport de classification sont les suivants :

Le modèle utilisé pour la classification est LogisticRegression, avec le vectorizer TfidfVectorizer. Les meilleurs paramètres trouvés sont :
— **max_features** : 10000
— **ngram_range** : (1,3) representant un ensemble de 1gram bi gram et tri gram
— **stop_words** : en gardant les stopwords

Le meilleur score F1 en validation croisée est de 0.841.

Classe	Précision	Rappel	F1-score	Support
0	0.85	0.81	0.83	193
1	0.83	0.86	0.85	207
Accuracy			0.84	400
Macro Avg	0.84	0.84	0.84	400
Weighted Avg	0.84	0.84	0.84	400

1.3.6.4 Comparaisons et conclusion

D'après les résultats listés précedemment, on peut constater que le modèle utilisant CountVectorizer (ensemble de 1gram et bi gram) a obtenu un score F1 légèrement supérieur à celui utilisant TfIdfVectorizer.

1.4 Arbres de décision

Nous nous tournons maintenant vers l'utilisation du classifieur d'arbre de décision

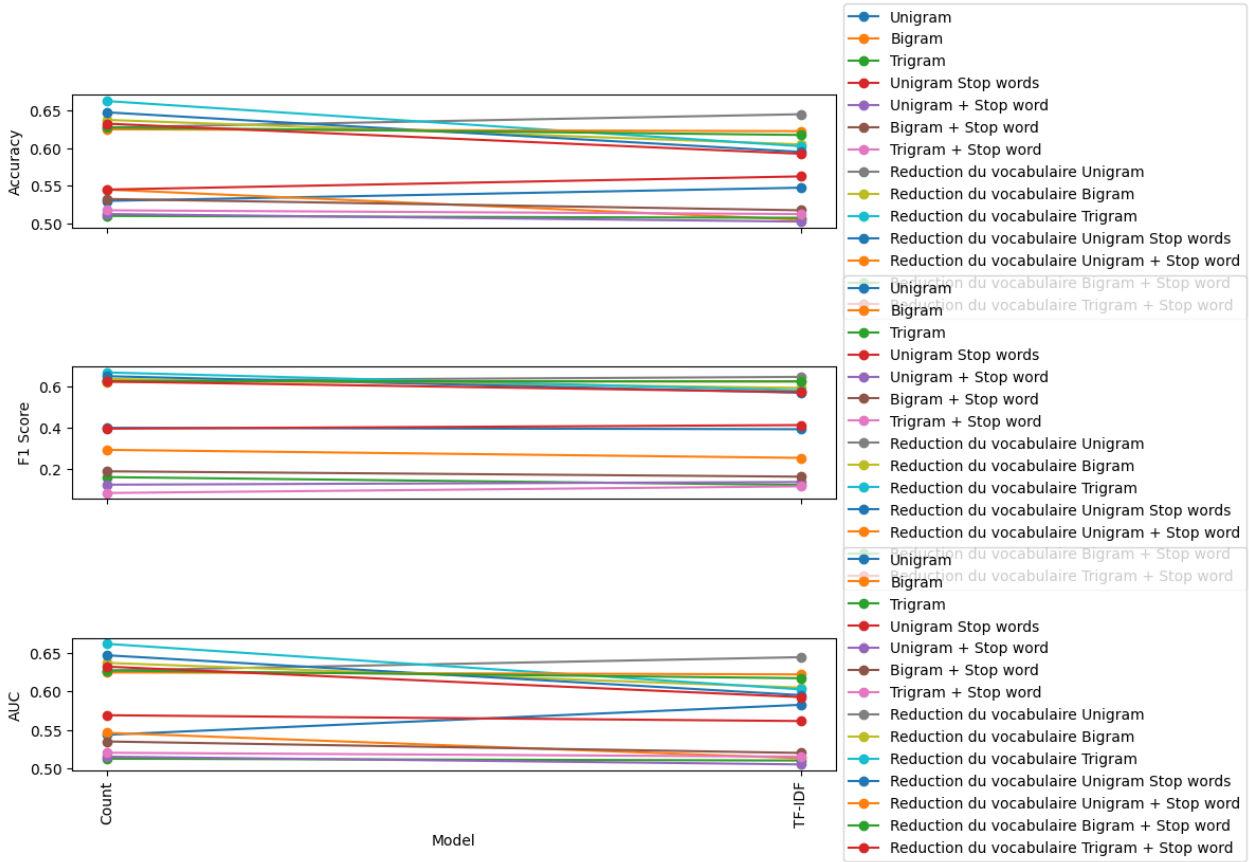
1.5 Analyse

1.5.0.1 Analyse avec différents modèles et paramètres

Nous avons testé l'analyse avec différents modèles qui ont plusieurs paramètres différents. Voici les résultats de notre analyse avec différents modèles et paramètres (Concernant la réduction du vocabulaire, nous l'avons réduit a 70%) :

TABLE 4 – Résultats de l'analyse avec différents modèles et paramètres (mise à jour)

Type	**Modèle**	**Accuracy**	**F1 Score**	**AUC**
Unigram	Count	0.5300	0.401274	0.543389
Unigram	TF-IDF	0.5475	0.394649	0.582627
Bigram	Count	0.5450	0.294574	0.545976
Bigram	TF-IDF	0.5050	0.255639	0.512663
Trigram	Count	0.5100	0.162393	0.512088
Trigram	TF-IDF	0.5075	0.124444	0.509700
Unigram Stop words	Count	0.5450	0.397351	0.568977
Unigram Stop words	TF-IDF	0.5625	0.414716	0.561364
Unigram + Stop word	Count	0.5125	0.125561	0.514725
Unigram + Stop word	TF-IDF	0.5025	0.138528	0.504625
Bigram + Stop word	Count	0.5325	0.190476	0.534626
Bigram + Stop word	TF-IDF	0.5175	0.164502	0.519625
Trigram + Stop word	Count	0.5175	0.085308	0.519875
Trigram + Stop word	TF-IDF	0.5125	0.117647	0.514750
Réduction du vocabulaire Unigram	Count	0.6275	0.630273	0.627478
Réduction du vocabulaire Unigram	TF-IDF	0.6450	0.648515	0.644966
Réduction du vocabulaire Bigram	Count	0.6375	0.638404	0.637503
Réduction du vocabulaire Bigram	TF-IDF	0.6050	0.594872	0.605140
Réduction du vocabulaire Trigram	Count	0.6625	0.669927	0.662404
Réduction du vocabulaire Trigram	TF-IDF	0.6025	0.582677	0.602753
Réduction du vocabulaire Unigram Stop words	Count	0.6475	0.651852	0.647454
Réduction du vocabulaire Unigram Stop words	TF-IDF	0.5950	0.571429	0.595290
Réduction du vocabulaire Unigram + Stop word	Count	0.6250	0.625000	0.625016
Réduction du vocabulaire Unigram + Stop word	TF-IDF	0.6225	0.627160	0.622453
Réduction du vocabulaire Bigram + Stop word	Count	0.6275	0.628429	0.627503
Réduction du vocabulaire Bigram + Stop word	TF-IDF	0.6175	0.625917	0.617403
Réduction du vocabulaire Trigram + Stop word	Count	0.6325	0.625954	0.632603
Réduction du vocabulaire Trigram + Stop word	TF-IDF	0.5925	0.574413	0.592727



Globalement, les modèles utilisant des unigrammes semblent obtenir de meilleurs résultats que ceux utilisant des bigrammes ou des trigrammes. Parmi les modèles utilisant des unigrammes, ceux qui ne tiennent pas compte des stop words semblent légèrement meilleurs que ceux qui les incluent.

En ce qui concerne les modèles de représentation, les résultats ne montrent pas de différence significative entre l'utilisation de la représentation en comptage (Count) et celle de la fréquence inverse de document pondérée (TF-IDF). Cependant, les modèles basés sur TF-IDF obtiennent généralement des scores légèrement plus élevés.

En ce qui concerne la réduction du vocabulaire, il semble que cette technique n'améliore pas considérablement les performances des modèles. Les modèles avec réduction du vocabulaire obtiennent des scores similaires à ceux sans réduction.

En prenant en compte les performances des différents modèles et configurations, il est important de noter que globalement, les résultats ne sont pas très satisfaisants. Les scores d'accuracy, de F1

Score et d'AUC obtenus pour la plupart des modèles et configurations sont relativement faibles.

1.5.1 Naive Bayes

Dans cette dernière étape de notre étude comparative, nous explorons l'utilisation de l'algorithme Naive Bayes pour compléter notre analyse de classification. Après avoir examiné la régression logistique, le classifieur SVM et l'algorithme Random Forest, nous évaluons maintenant les performances et les capacités de classification de ces quatre approches.

1.5.1.1 Résultats du Naive Bayes avec CountVectorizer

Les résultats du rapport de classification sont les suivants :

le changement de paramétes n'a que trés peu voir aucune incidence sur les performances, en gardant des paramétres générique on obtient des résultat stable

En utilisant ces paramètres, la meilleure métrique F1 obtenue lors de la validation croisée est de 0.8. Ce score représente la performance moyenne du modèle sur plusieurs plis de validation croisée et est une estimation de sa capacité à généraliser sur de nouvelles données.

et on retrouve ces résultats pour les données de tests :

Classe	**Précision**	**Rappel**	**F1-score**	**Support**
0	0.77	0.82	0.80	193
1	0.82	0.77	0.80	207
Accuracy			0.80	400
Macro Avg	0.80	0.80	0.80	400
Weighted Avg	0.80	0.80	0.80	400

1.5.1.2 Résultats du classifieur Naive Bayes avec TfidfVectorizer

Les résultats du rapport de classification sont les suivants :

Le modèle utilisé pour la classification est Naive Bayes, avec le vectorizer TfidfVectorizer. Le changement de paramétes n'a que trés peu voir aucune incidence sur les performances, en gardant des paramétres générique on obtient des résultat stable

Le meilleur score F1 en validation croisée est de 0.8.

Ces résultats démontrent une performance limité du modèle dans la classification des données.

Classe	Précision	Rappel	F1-score	Support
0	0.77	0.82	0.80	193
1	0.82	0.77	0.80	207
Accuracy			0.80	400
Macro Avg	0.80	0.80	0.80	400
Weighted Avg	0.80	0.80	0.80	400

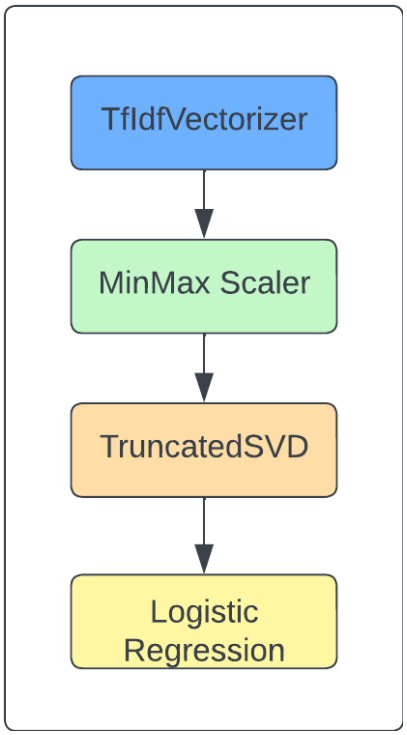
1.5.1.3 Comparaisons et conclusion

D'après les résultats listés précedemment, on peut constater que le modèle utilisant CountVectorizer a données les meme résultats en comparant avec TfIdfVectorizer et cela que ça soit pour les bi grams ou les trigramme.

1.5.2 SVD pour la regression logistique

La SVD est une technique couramment utilisée pour réduire la dimensionnalité des données dans le contexte de l'apprentissage automatique. Elle permet de représenter les données dans un espace de plus petite dimension en conservant les informations les plus significatives. Cela peut être utile pour réduire les temps de calcul, gérer les problèmes de surapprentissage et faciliter l'interprétation des données.

La technique de SVD a été utilisée pour extraire les caractéristiques les plus discriminantes des données et réduire leur dimensionnalité avant l'entraînement du modèle en utilisant la pipeline suivante :



Aprés entrainement on obtient les résultats suivant :

Classe	Précision	Rappel	F1-score	Support
0	0.91	0.86	0.88	193
1	0.87	0.92	0.90	207
Accuracy			0.80	400
Macro Avg	0.89	0.89	0.89	400
Weighted Avg	0.89	0.89	0.89	400

On remarque alors des résultats plus interessant avec une augmentation moyenne de de presque 5% en comparant l'utilisation de la regression logsitique aprés juste aprés le vectorizer

Classe	Précision	Rappel	F1-score	Support
0	0.91	0.86	0.88	193
1	0.87	0.92	0.90	207
Accuracy			0.80	400
Macro Avg	0.89	0.89	0.89	400
Weighted Avg	0.89	0.89	0.89	400

1.6 Conclusion sur l'analyse de review de films

Dans cette étude, nous avons examiné l'analyse de revues de films en utilisant différentes méthodes de Machine Learning. Nous avons commencé par effectuer une transformation paramétrique du texte en utilisant des pré-traitements tels que la suppression des caractères spéciaux et la mise en minuscule. Ensuite, nous avons extrait le vocabulaire à l'aide de la méthode Bag-of-Words (BoW), en explorant différentes variantes de cette approche.

Pour évaluer les performances des modèles de Machine Learning, nous avons utilisé des métriques d'évaluation telles que la précision, le rappel et le score F1. Nous avons également mis en place une méthodologie pour gérer les hyperparamètres et effectuer différentes combinaisons afin d'optimiser les modèles.

Nous avons appliqué plusieurs modèles, dont la régression logistique, les machines à vecteurs de support (SVM), Random Forest et le Naive Bayes. Nous avons utilisé à la fois le CountVectorizer et le TfidfVectorizer pour représenter les données textuelles.

Les résultats de la régression logistique ont montré une performance satisfaisante avec les deux vecteurs de représentation. Les SVM ont également donné de bons résultats, avec une légère amélioration en utilisant le TfidfVectorizer. Les classifieurs Random Forest et Naive Bayes ont également montré des performances prometteuses, bien que légèrement inférieures à celles de la régression logistique et des SVM.

Dans l'ensemble, nos résultats indiquent que la régression logistique a été l'une des approches les plus fiables pour l'analyse de revues de films. Elle a démontré des performances solides tant avec le CountVectorizer qu'avec le TfidfVectorizer. La régression logistique a montré sa capacité à capturer les relations linéaires entre les caractéristiques extraites des revues de films et les étiquettes de classification.

De plus, l'introduction de l'approche SVD (Singular Value Decomposition) a considérablement boosté les performances de la régression logistique. En utilisant SVD pour réduire la dimensionnalité des données, nous avons réussi à extraire des caractéristiques latentes plus informatives, ce qui a amélioré la capacité de prédiction du modèle. Cette approche a permis de mieux représenter les revues de films et d'obtenir des résultats plus précis.

En conclusion, cette étude comparative nous a permis de mieux comprendre les différentes méthodes de Machine Learning pour l'analyse de revues de films. Elle a également souligné l'importance de l'exploration préliminaire des données et de la sélection appropriée des vecteurs de représentation. En continuant à affiner et à optimiser ces modèles, il est possible de développer des systèmes d'analyse de revues de films plus précis et efficaces, contribuant ainsi à une meilleure compréhension des opinions des utilisateurs et à l'amélioration des recommandations de films.

2 Analyse de locuteurs

2.1 Analyse des données

Nous avons commencé par présenté une analyse détaillée des données.

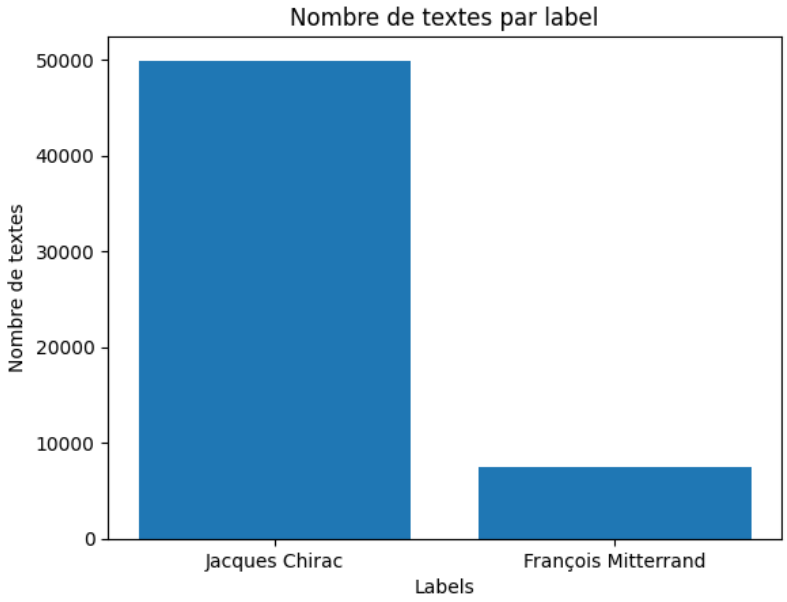
Nous avons déterminer le nombre total de textes disponibles ainsi que le nombre de labels uniques. Dans notre ensemble de données, nous avons recensé un total de 57 413 textes, répartis entre deux labels distincts.

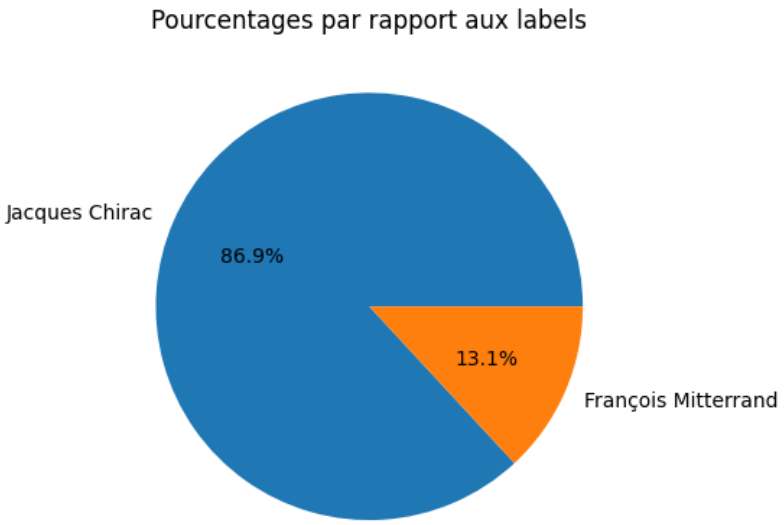
- **Le label 1** qui représente Jacques Chirac.
- **Le label -1** qui représente François Mitterrand.

Nombre de textes	57413
Nombre de labels uniques	2
Labels	[1, -1]

Pourcentages par rapport aux labels :

Jacques Chirac (label = 1)	86.90%	7523
François Mitterrand (label = -1)	13.10%	49890

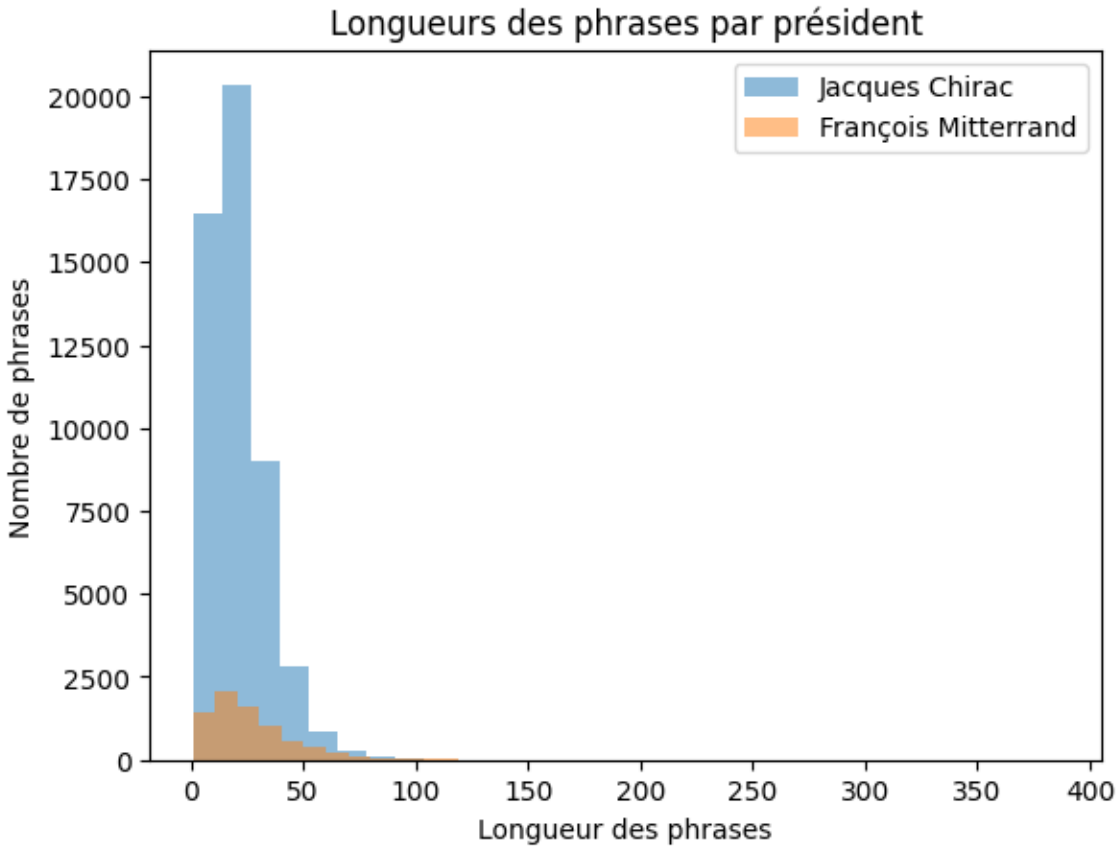




En examinant les statistiques fournies, on peut conclue :

1. François Mitterrand représente la classe minoritaire : Les données montrent que le label -1, qui est associé à François Mitterrand, représente seulement 13,10% des textes du dataset. Cela indique que François Mitterrand est moins représenté dans les données par rapport à Jacques Chirac (label 1) qui représente 86,90% des textes.

2. Les données ne sont pas équilibrées : La disparité entre les occurrences des labels 1 et -1 indique un déséquilibre significatif dans les données. La classe majoritaire (Jacques Chirac) est largement prédominante par rapport à la classe minoritaire (François Mitterrand). Ce déséquilibre peut avoir un impact sur les performances des modèles d'apprentissage automatique, en donnant potentiellement plus de poids à la classe majoritaire.



Jacques Chirac a généralement des phrases plus longues que François Mitterrand. Cette différence de longueur de phrases peut influencer la répartition des textes dans le dataset et contribuer au déséquilibre observé entre les deux présidents. Les discours de Jacques Chirac, avec leurs phrases plus longues, représentent la classe majoritaire dans les données, tandis que les discours de François Mitterrand, avec leurs phrases plus courtes, représentent la classe minoritaire.

2.2 Transformation paramétrique du texte (pre-traitements)

Nous avons écrit une fonction de prétraitement appelée preprocess et l'avons utilisée pour effectuer les plusieurs traitement parmis elles :

1. Suppression de la ponctuation : Nous utilisons `string.punctuation` pour obtenir une table de traduction qui supprime tous les caractères de ponctuation ainsi que les caractères de nouvelle ligne, de retour chariot et de tabulation. Cette table de traduction est ensuite utilisée avec la fonction `translate()` pour supprimer la ponctuation du texte.

2. Mise en minuscule : Nous utilisons la fonction `lower()` pour convertir le texte en minuscules, ce qui permet de normaliser les mots et de réduire la complexité du vocabulaire.

3. Lemmatisation : La lemmatisation est le processus de conversion des mots en leur forme de base, ce qui permet de regrouper les variantes d'un même mot. Dans la fonction `preprocess()`, si le paramètre `lemma` est défini sur `True`, nous utilisons la bibliothèque spaCy pour le français (`fr_core_news_sm`) pour lemmatiser chaque mot du texte et les reconstituer en une chaîne de texte lemmatisée.

Voici quelques exemples de transformation de texte :

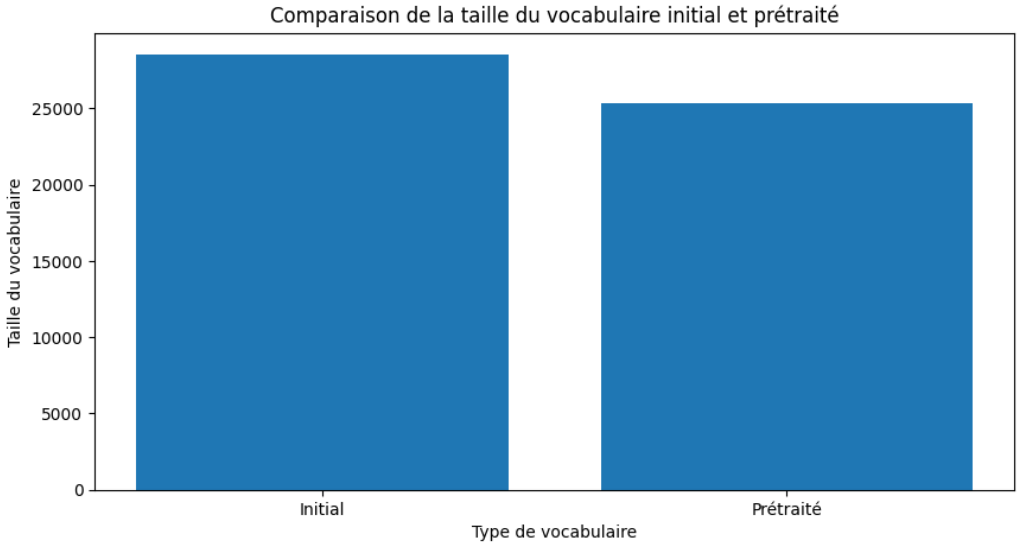
Texte initial	**Texte prétraité**
Quand je dis chers amis, il ne s'agit pas là ...	quand je dire cher ami il ne sagit pas le...
D'abord merci de cet exceptionnel accueil que...	dabord merci de ce exceptionnel accueil que...
C'est toujours très émouvant de venir en Afri...	cest toujours tre emouver de venir en afrique ...
Aucun citoyen français ne peut être...	aucun citoyen francai ne pouvoir etre
Le Congo, que naguère le <nom> qualifia de "refuge	le congo que naguere le nom qualifier de refuge...

2.3 Extraction du vocabulaire (BoW)

Afin de mieux comprendre nos données, nous avons effectué des analyses sur le vocabulaire.

Nous avons commencé par calculer la taille du vocabulaire, qui est la suivante :

— Taille du vocabulaire initial : 28524
— Taille du vocabulaire prétraité : 25377



Nous remarquons que le vocabulaire initiale est plus large. Cela peut être dû à plusieurs raisons comme la suppression des caractères spéciaux, la supression de la ponctuation, le faite de mettre tout en minuscules.

Nous avons également recensé les 10 mots les plus utilisés par chaque président, comme indiqué dans le tableau ci-dessous :

Jacques Chirac	**François Mitterrand**
le : 81535	le : 14655
de : 81441	de : 13767
avoir : 35066	avoir : 7265
et : 32457	et : 4786
un : 22282	un : 4070
être : 14448	que : 3726
ce : 13303	ce : 3270
en : 12986	qui : 2984
qui : 12650	être : 2962
que : 12579	en : 2297

TABLE 5 – Les 10 mots les plus utilisés par Jacques Chirac et François Mitterrand avec stop words

2.3.1 Strop words

Nous avons remarqué que la majorité des mots utilisés sont des stop words, ce qui pourrait avoir un impact sur l'analyse des données. Par conséquent, nous avons décidé de supprimer les stop words de nos textes. Pour cela, nous avons utilisé la bibliothèque NLTK (Natural Language Toolkit) et nous avons chargé les stop words en français à l'aide de la fonction stopwords.words('french').

Ensuite, nous avons utilisé la liste de stop words pour filtrer les mots du vocabulaire prétraité. Nous avons supprimé tous les mots présents dans la liste des stop words du vocabulaire prétraité afin d'obtenir un vocabulaire sans les stop words.

Voici la liste des stops words qu'on a supprimé :

au	en	lui	notre	sa	tu	à
étant	serai	seraient	furent	fussiez	ai	auront
avaient	aient	aux	et	ma	nous	se
un	m	étante	seras	étais	sois	fussent
as	aurais	eut	eusse	avec	eux	mais
on	ses	une	n	étants	sera	était
soit	ayant	avons	aurait	eûmes	eusses	ce
il	me	ou	son	vos	s	étantes
serons	étions	soyons	ayante	avez	aurions	eûtes
eût	ces	ils	même	par	sur	votre
t	suis	serez	étiez	soyez	ayantes	ont
auriez	eurent	eussions	dans	je	mes	pas
ta	vous	y	es	seront	étaient	soient
ayants	aurai	auraient	aie	eussiez	de	la
moi	pour	te	c	été	est	serais
fus	fusse	eu	auras	avais	aies	eussent
des	le	mon	qu	tes	d	étée
sommes	serait	fut	fusses	eue	aura	avait
ait	du	les	ne	que	toi	j
étées	êtes	serions	fûmes	fût	eues	aurons
avions	ayons	elle	leur	nos	qui	ton
l	étés	sont	seriez	fûtes	fussions	eus
aurez	aviez	ayez				

Table 6 – Liste des stop words en français

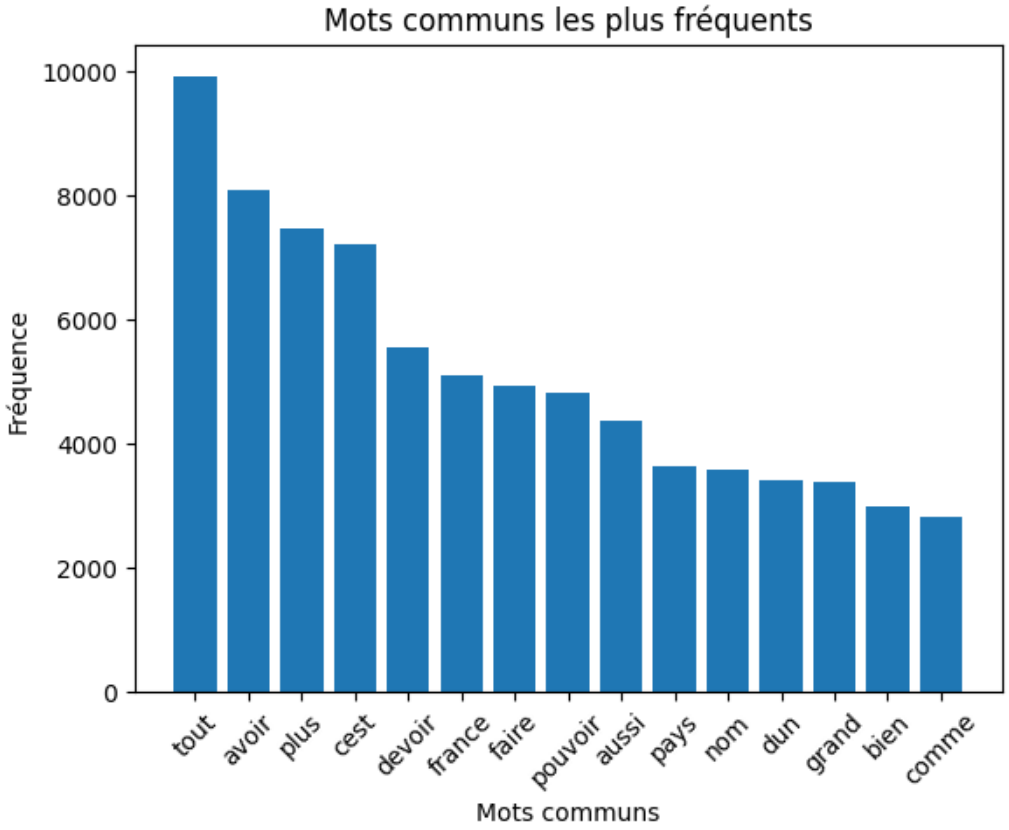
Après avoir enlevé les stops words, nous avons de nouveau rencensé les 10 mots les plus utilisés par chaque présenident, comme l'indique le tableau ci-dessous :

Jacques Chirac	François Mitterrand
tout (8362)	tout (1551)
avoir (6555)	avoir (1519)
plus (6267)	plus (1175)
cest (6042)	cest (1165)
devoir (4987)	pouvoir (1095)
france (4369)	faire (912)
faire (4003)	bien (829)
aussi (3883)	dire (814)
pouvoir (3719)	france (728)
nom (3120)	celer (716)

TABLE 7 – Les 10 mots les plus utilisés par Jacques Chirac et François Mitterrand sans stop words

Word cloud des mots les plus fréquemment utilisés par Jacques Chirac et François Mitterrand

Nous avons ensuite identifié les mots les plus fréquemment utilisés en commun par Jacques Chirac et François Mitterrand. Cela nous permet de comprendre les similitudes dans leur langage et les thématiques récurrentes dans leurs discours.



Les mots en commun les plus fréquents

2.3.2 Bigrammes et trigreammes les plus fréquents

Nous avons également recensé les bigrammes les plus fréquents ainsi que les trigrammes dans le but de mieux comprendre les constructions linguistiques utilisées par les deux présidents.
On a égallement affiché les word clouds des odds ratios.

Les odds ratios sont une mesure statistique utilisée pour évaluer l'association entre deux variables. Ils sont couramment utilisés dans les études épidémiologiques et les analyses de données textuelles.

L'odds ratio compare les chances (odds) d'occurrence d'un événement dans deux groupes différents. Il est calculé en divisant le rapport des chances dans le groupe exposé par le rapport des chances dans le groupe non exposé.

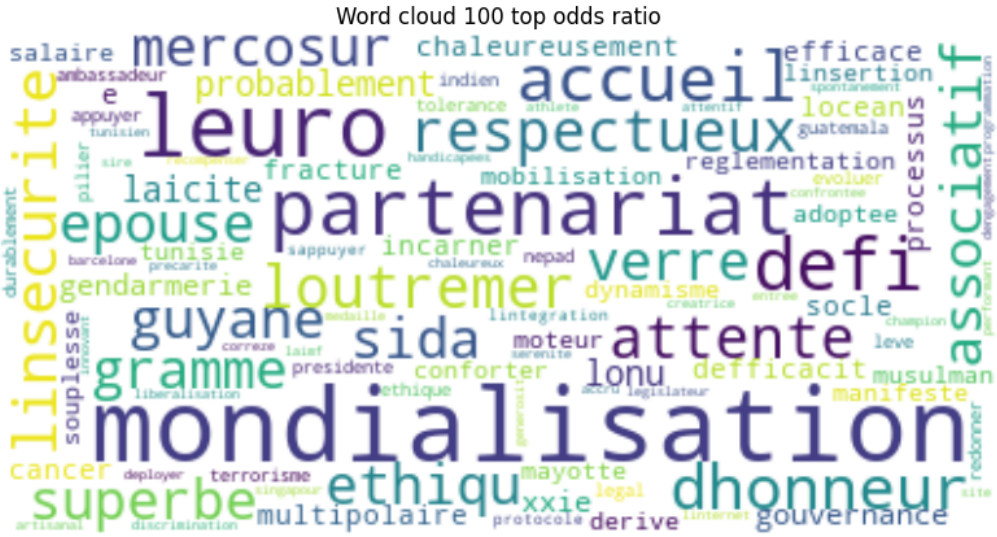
1 - 10	**11 - 20**	**21 - 30**	**31 - 40**
devoir etre	quil falloir	service public	tout fait
monsieur presider	ny avoir	monsieur maire	rendre hommage
pouvoir etre	france avoir	premier ministre	mettre place
lunion europeenne	mesdame monsieur	droit lhomm	sans doute
cest aussi	tout lheur	avoir faire	chef detat
plus grand	tout celui	quil avoir	chacun dentre
deux pays	madame monsieur	bien entendre	plus fort
plus plus	nom nom	mettre oeuvre	lunion europeenn
cest vrai	lutte contre	tout celer	cher ami
cest pourquoi	entrer deux	vouloir dire	plus jamais
41 - 50	**51 - 60**	**61 - 70**	**71 - 80**
nom avoir	plus loin	pouvoir faire	pouvoir dire
tout francai	si lon	relation entrer	communaut international
certain nombre	economiqu social	cest ainsi	quil sagisse
devoir faire	non seulement	cest tout	tout cas
nation uni	avoir besoin	faire vivre	avoir tout
chaque jour	dire tout	cest dabord	eh bien
trop souvent	tre grand	avoir permettre	premier rang
avoir savoir	homme femme	tout place	chaque annee
parce quil	pouvoir public	tout tout	derniere annees
devoir aussi	depuis longtemps	lon pouvoir	tout entiere
81 - 90	**91 - 100**		
prendre compte	tout domaine		
chacun chacun	dun nouveau		
falloir aussi	cest celer		
sans cesse	faire preuve		
beaucoup plus	partenaire social		
cest bien	tout celle		
toujours plus	general gaulle		
france devoir	tout simplement		
faire face	avoir quelque		
tout pays	sans aucun		

TABLE 8 – Bigrammes les plus fréquents

1 - 10	**11 - 20**	**21 - 30**
entrer deux pays	présider mesdame monsieur	plus plus nombreux
tout celle tout	monsieur presider mesdame	lunion europeenne avoir
celle tout celui	plus grand nombre	mieux prendre compte
chef detat gouvernement	quil ny avoir	tout doit etre
chacun chacun dentre	dire tout lheur	cest raison lequel
sans aucun doute	ministre affaire etrangere	petit moyenne entreprise
monsieur premier ministre	si lon vouloir	cours derniere annees
31 - 40	**41 - 50**	**51 - 60**
puissance economiqu monde	prendre tout part	bon heureux annee
devoir etre plus	relation entrer deux	fonds monetair international
lutte contre lexclusion	er janvier date	mise place dun
lorganisation mondial commerce	depuis si longtemps	vouloir rendre hommage
tout devoir etre	depuis plusieurs annee	celer devoir etre
vouloir tout dabord	plus fort plus	quel devoir etre
conseil economiqu social	an plus tard	lequel ny avoir
monde entier doit	faire entendre voix	presider conseil general
plus plus nombreux	falloir aller plus	avoir si longtemps
tout lheur nom	formation tout long	monsieur presider monsieur
61 - 70	**71 - 80**	**81 - 90**
monsieur presider cher	premiere puissance economiqu	politique agricole commun
france devoir etre	maintenant lever verre	premier ministre nom
cest pourquoi france	membre permanent conseil	maire mesdame monsieur
leurope tout entiere	droit lhomm citoyen	depuis quelque annee
devoir etre mettre	systeme protection social	politique etrangere securit
parce quil avoir	ny avoir plus	cooperation entrer deux
respect droit lhomm	monsieur maire mesdame	plus dun siecle
permanent conseil securit	vouloir tout lheur	presider cher ami
ny avoir plus	pourquoi jai souhaiter	chaque jour plus
tout long vie	declaration droit lhomm	savoir monsieur presider
91 - 100		
loi programmation militaire		
si jose dire		
sein lunion europeenne		
monsieur maire cher		
quil falloir faire		
cest tout sens		
savoir monsieur presider		
tout lheur monsieur		
pourquoi jai souhaiter		
lutte contre chomage		

TABLE 9 – Trigrammes les plus fréquents

2.3.3 Odds ratio

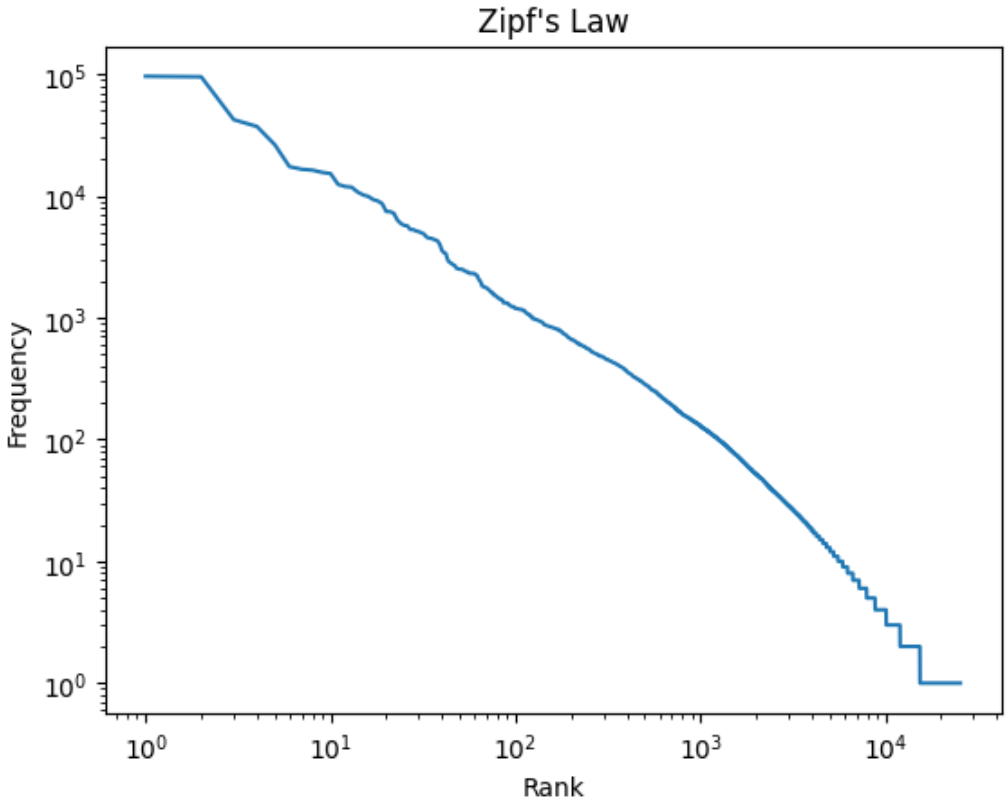


Word clouds des odds ratio

Dans notre cas, parmis les mots qui ressortent avec les plus hauts odds ratios on a "mondialisation", "partenariat", "accueil", "attente" et "Guyane". Cela suggère que ces mots sont fortement associés ou fréquemment utilisés.

2.3.4 Loi de Zipf

On a affiché la courbe de la loi de Zipf, qui représente la relation entre la fréquence des mots et leur rang dans un corpus de texte. La loi de Zipf est une loi empirique qui postule que la fréquence d'un mot est inversement proportionnelle à son rang. Cela signifie que les mots les plus fréquents ont un rang plus bas, tandis que les mots moins fréquents ont un rang plus élevé.



On remarque que la courbe correspond à la loi de Zipf, cela suggère que quelques mots sont très fréquents, tandis que la majorité des mots sont peu fréquents.

2.4 Analyse et correction de la problématique de correction d'équilibrage des classes

Lorsqu'il y a un déséquilibre dans les classes de données, il est important de prendre des mesures pour gérer ce problème lors de l'analyse des locuteurs. L'objectif de cette étude est de comparer différentes approches courantes pour gérer le déséquilibre de classe et d'évaluer leur impact sur les performances de l'analyse des locuteurs.

2.4.1 Utilisation de poids de classe

Nous avons utilisé des poids de classe pour traiter le déséquilibre de classe. Les poids de classe ont été incorporés dans le modèle de régression logistique ainsi que le modéle SVM en utilisant

l'argument `class_weight='balanced'`. Les résultats ont montré une amélioration significative du F1-score.

Sans l'ajout des poids de classes :

Classe	**Précision**	**Rappel**	**F1-score**	**Support**
0	0.82	0.25	0.39	193
1	0.90	0.99	0.94	207
Accuracy			0.89	400
Macro Avg	0.86	0.62	0.66	400
Weighted Avg	0.89	0.89	0.87	400

Avec l'ajout des poids de classes :

Classe	**Précision**	**Rappel**	**F1-score**	**Support**
0	0.45	0.72	0.55	1204
1	0.95	0.87	0.91	7982
Accuracy			0.85	9186
Macro Avg	0.70	0.79	0.73	9186
Weighted Avg	0.89	0.85	0.86	9186

2.4.2 Sur-échantillonnage de la classe minoritaire

La technique de sur-échantillonnage a été appliquée à la classe minoritaire pour équilibrer les classes. Nous avons utilisé une méthode de duplication d'échantillons pour générer des exemples supplémentaires de la classe minoritaire. Les résultats ont montré une amélioration du F1-score pour la base de test avec un resultat ressemblant au poid des classes ce qui est logique vu que dupliquer les echantillions revient en quelques sortes a appliquer des poids supplementaire

Voici les résultat obtenus durant l'entrainement (prediction sur les données de validation) :

Classe	**Précision**	**Rappel**	**F1-score**	**Support**
-1	0.87	0.97	0.92	7983
1	0.97	0.85	0.91	7982
Accuracy			0.91	15965
Macro Avg	0.92	0.91	0.91	15965
Weighted Avg	0.92	0.91	0.91	15965

voici les résultats obtenus sur la prediction des données de test

Classe	Précision	Rappel	F1-score	Support
0	0.44	0.74	0.44	1505
1	0.96	0.86	0.90	9978
Accuracy			0.84	11483
Macro Avg	0.70	0.80	0.72	11483
Weighted Avg	0.89	0.84	0.86	11483

2.4.3 Sous-échantillonnage de la classe majoritaire

Nous avons également exploré le sous-échantillonnage de la classe majoritaire comme approche de gestion du déséquilibre. Des échantillons aléatoires ont été supprimés de la classe majoritaire pour équilibrer les classes. Les résultats ont montré une amélioration du F1-score, mais avec une perte d'informations potentielles pour la classe majoritaire initiale. Il serait peut etre interessant de faire un ensemble de modéle ou chaque modéle récupére un batch de la classe majoritaire different, et de faire un vote par majorité. Voici les résultat obtenus durant l'entrainement (prediction sur les données de validation) :

Classe	Précision	Rappel	F1-score	Support
-1	0.75	0.82	0.78	1204
1	0.80	0.72	0.76	1204
Accuracy			0.77	2408
Macro Avg	0.77	0.77	0.77	2408
Weighted Avg	0.77	0.77	0.77	2408

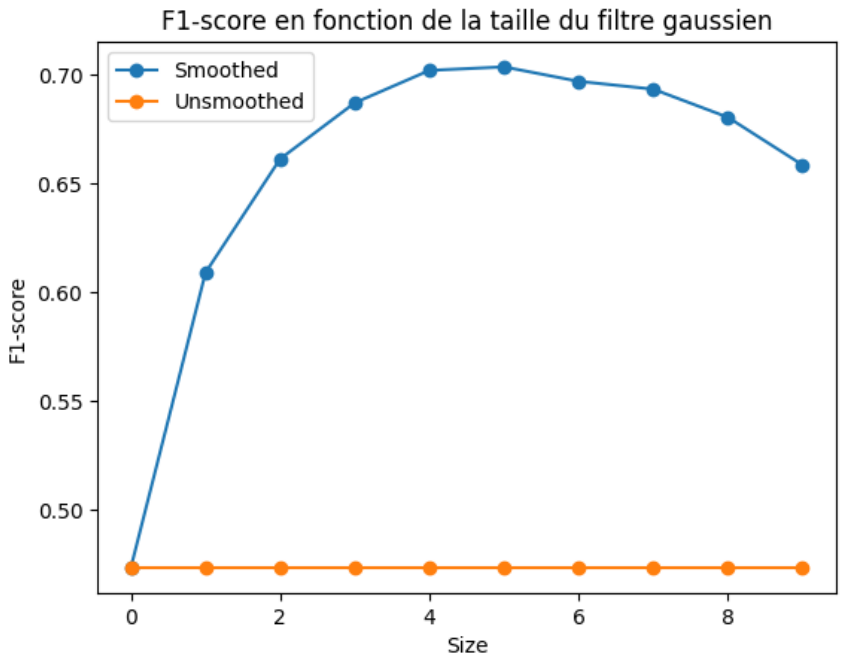
voici les résultats obtenus sur la prediction des données de test

Classe	Précision	Rappel	F1-score	Support
0	0.31	0.79	0.44	1505
1	0.96	0.73	0.83	9978
Accuracy			0.74	11483
Macro Avg	0.63	0.76	0.64	11483
Weighted Avg	0.87	0.74	0.78	11483

2.4.4 Post-processing

Lors de l'analyse des phrases des locuteurs, et comme mentionné précédemment, nous avons identifié un aspect crucial à prendre en compte dans notre analyse. Les locuteurs ont tendance à énoncer plusieurs phrases consécutives, ce qui rend plus facile la détection d'un bruit s'il est isolé. En effet, les données consistent en des phrases issues de discours. Cependant, ces phrases n'ont pas été mélangées : si on les lit de manière consécutive, on retrouve des discours complets. Ainsi, les phrases sont organisées en ce que nous appelons des "blocs de discours", qui ont été découpés en phrases et sont utilisés pour les prédictions.

Donc pour le post-traitement on décide donc appliquer un lissage gaussien pour ajuster les prédictions. Les résultats ont montré une amélioration du F1-score trés notable. Et cela en fonction de differente tailles de fenetre pour le lissage gaussien comme le montre les resultats :



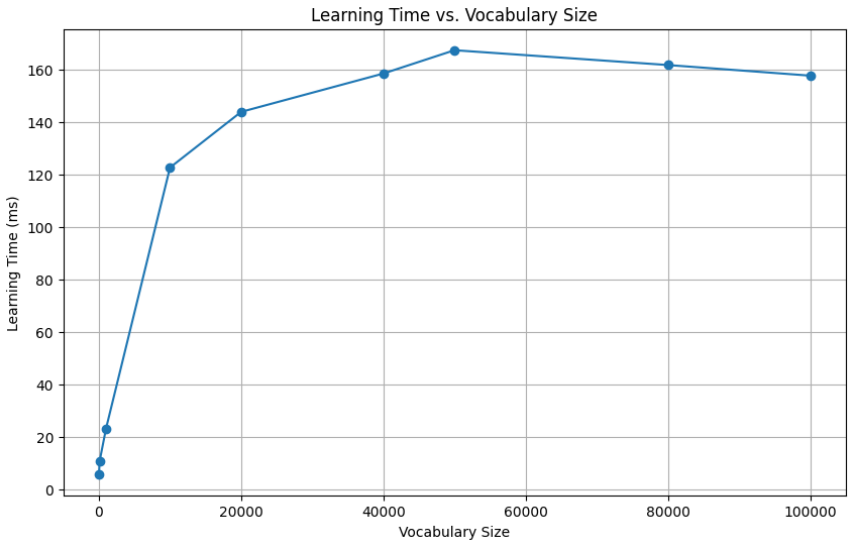
2.5 Etude de classifieur & Comparaison

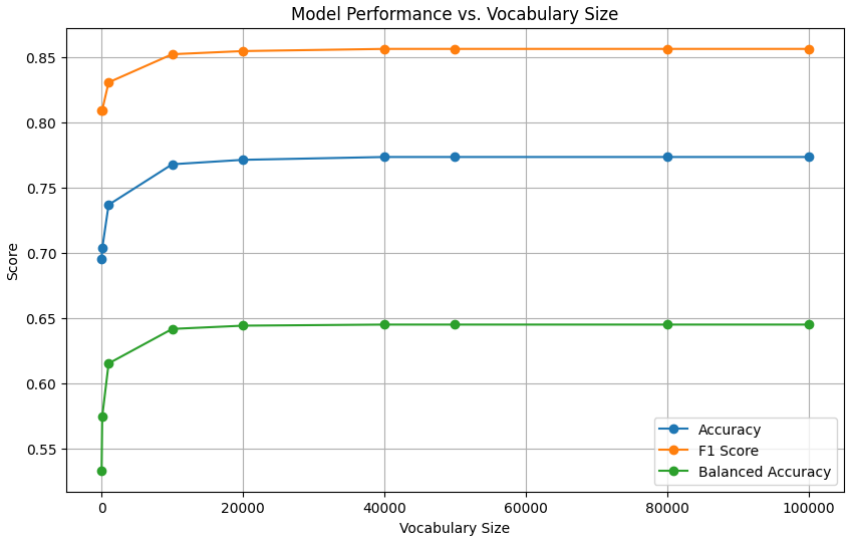
Dans notre étude, nous avons effectué une analyse comparative de différents classifieurs, à savoir le SVM (Support Vector Machine), la régression logistique et le Naive Bayes, en utilisant différentes mesures de performance. Les mesures considérées étaient le temps d'apprentissage, l'accuracy, le F1 score et la balanced accuracy.

Pour évaluer les performances de chaque classifieur, nous avons utilisé différentes tailles de vocabulaire (10, 100, 1000, 10.000, 20.000, 40.000, 50.000, 80.000, 100.000) et enregistré les valeurs correspondantes pour chaque mesure de performance.

2.5.1 Evaluation de la Regression logistique

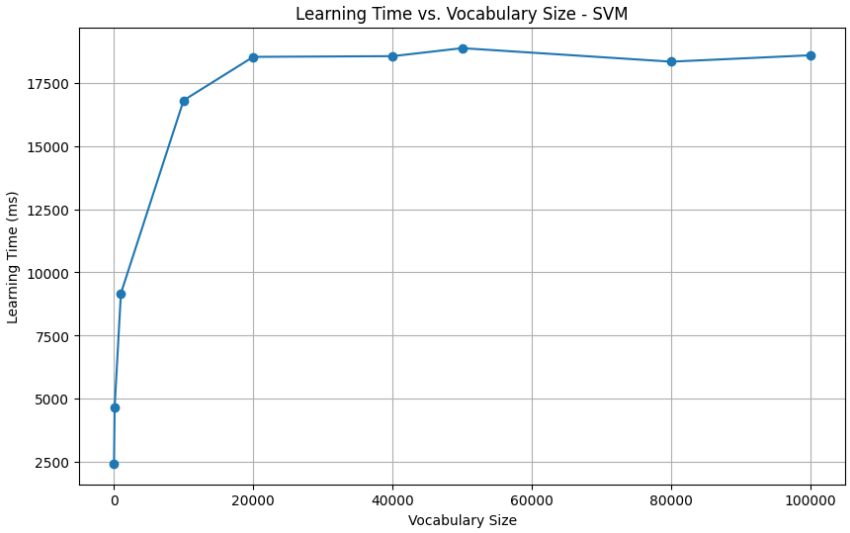
Taille du vocabulaire	Temps d'apprentissage	Précision	Score F1	Précision équilibrée
10	6.0184	0.6954	0.8093	0.5330
100	10.8588	0.7034	0.8094	0.5745
1000	23.2813	0.7367	0.8306	0.6151
10000	122.7074	0.7678	0.8522	0.6416
20000	143.8935	0.7712	0.8546	0.6441
40000	158.5019	0.7734	0.8563	0.6449
50000	167.3782	0.7734	0.8563	0.6449
80000	161.7351	0.7734	0.8563	0.6449
100000	157.6960	0.7734	0.8563	0.6449

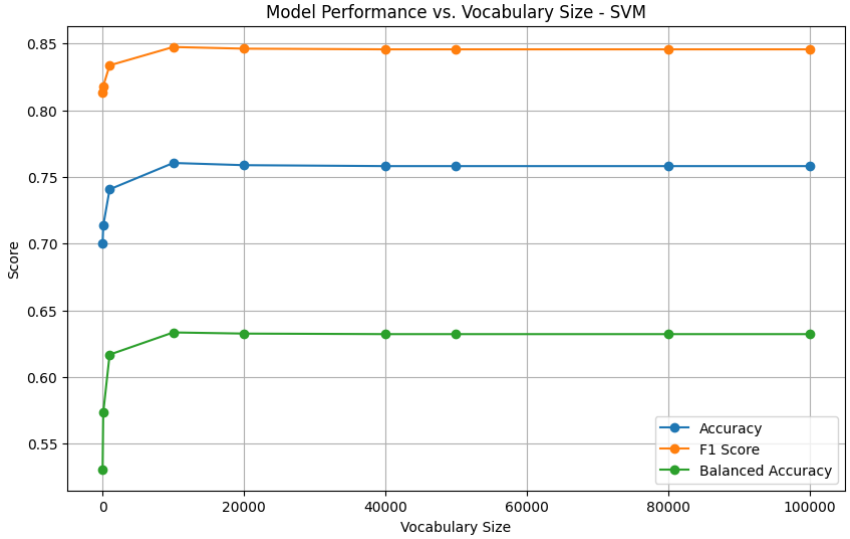




2.5.2 Evaluation de SVM

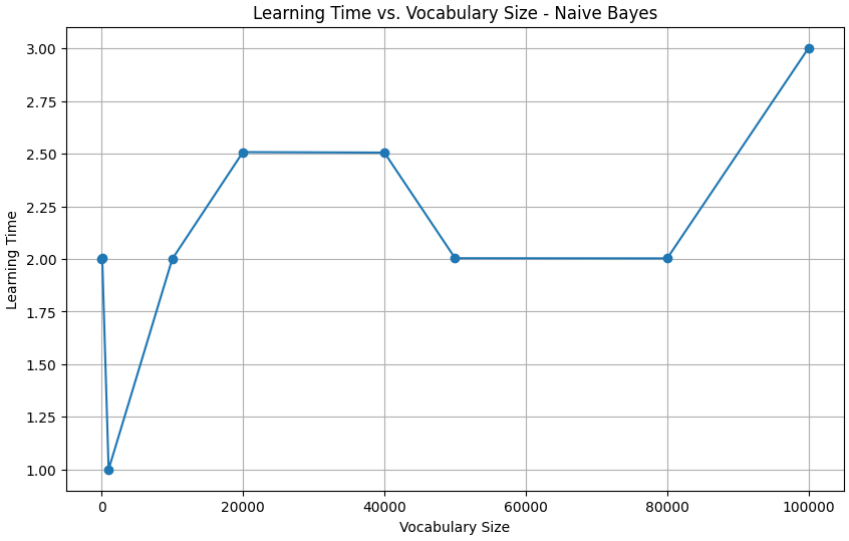
Taille du vocabulaire	Temps d'apprentissage	Précision	Score F1
10	2445.077	0.700	0.813
100	4676.778	0.714	0.818
1000	9172.198	0.741	0.834
10000	16800.772	0.761	0.847
20000	18519.655	0.759	0.846
40000	18547.269	0.758	0.846
50000	18864.136	0.758	0.846
80000	18329.767	0.758	0.846
100000	18583.161	0.758	0.846

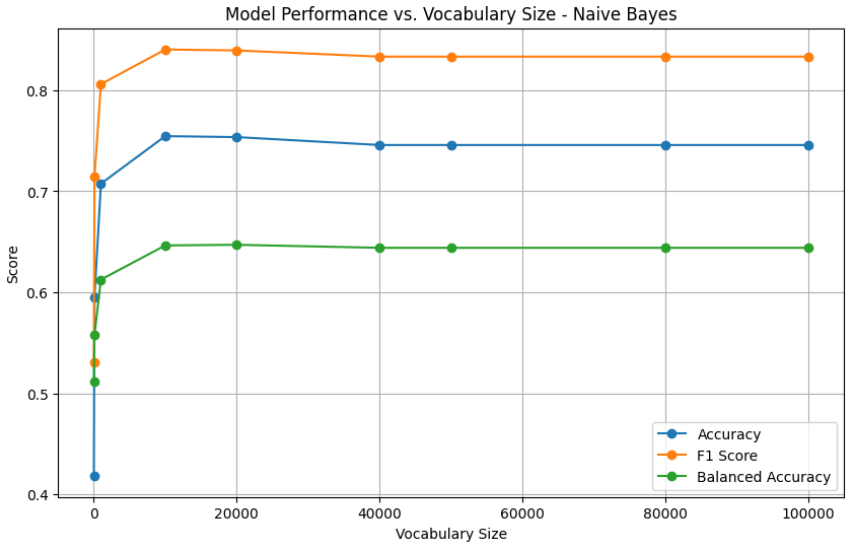

Learning Time vs. Vocabulary Size - SVM
Learning Time (ms)
Vocabulary Size
2500
5000
7500
10000
12500
15000
17500
0
20000
40000
60000
80000
100000


Model Performance vs. Vocabulary Size - SVM
Score
Vocabulary Size
0.55
0.60
0.65
0.70
0.75
0.80
0.85
0
20000
40000
60000
80000
100000
Accuracy
F1 Score
Balanced Accuracy

2.5.3 Evaluation de Naive Bayes

Taille du vocabulaire	Temps d'apprentissage	Précision	Score F1
10	2.000	0.419	0.530
100	2.003	0.595	0.715
1000	1.000	0.708	0.806
10000	2.000	0.755	0.840
20000	2.508	0.754	0.839
40000	2.506	0.746	0.833
50000	2.004	0.746	0.833
80000	2.003	0.746	0.833
100000	3.003	0.746	0.833

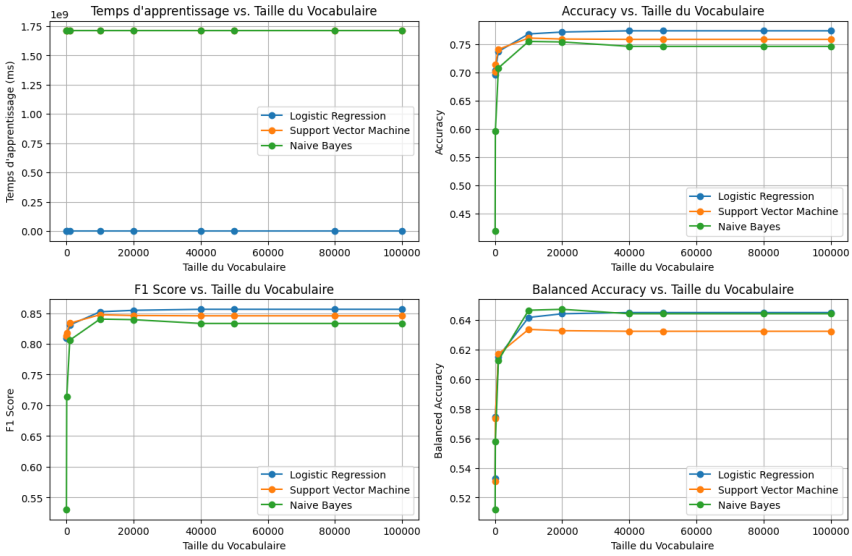




2.5.4 Comparaison

Nos résultats indiquent que le temps d'apprentissage du SVM est significativement plus long que celui de la régression logistique et de naive baies. Cependant, malgré cette différence de temps d'apprentissage, les performances des trois modèles sont presque équivalentes, avec une légère amélioration observée pour la régression logistique.

D'autre part, le modèle Naive Bayes présente un temps d'apprentissage relativement plus court que celui de la régression logistique, et ses performances sont comparables à celles de ce dernier modèle bien que légèrement moins performant.



2.6 Conclusion

En conclusion, notre étude sur l'analyse des locuteurs a démontré l'efficacité de plusieurs techniques pour remédier au déséquilibre des classes. L'utilisation de poids de classe a permis de donner plus d'importance aux classes minoritaires, ce qui a contribué à mieux capturer les exemples positifs. De plus, le sur-échantillonnage de la classe minoritaire a permis d'augmenter artificiellement le nombre d'exemples de cette classe, améliorant ainsi les performances du modèle.

En combinant ces techniques de gestion du déséquilibre des classes avec l'application d'un lissage gaussien pour ajuster les prédictions et en choisissant le classifieur approprié, nous avons réussi à obtenir des résultats satisfaisants en termes d'accuracy, de F1-score et de précision équilibrée dans notre analyse des locuteurs.

3 Conclusion générale

Dans ce projet, nous avons travaillé sur deux parties, l'analyse des sentiments et l'analyse des locuteurs. Dans le cadre de l'analyse des sentiments, différentes techniques de prétraitement et de modélisation ont été utilisées, telles que la transformation paramétrique du texte, l'extraction du vocabulaire et l'utilisation de modèles de machine learning tels que la régression logistique, les machines à vecteurs de support (SVM), le random forest et Naive Bayes. Des métriques d'évaluation ont été utilisées pour évaluer les performances des modèles.

En ce qui concerne l'analyse des locuteurs, des techniques similaires ont été appliquées, notamment le prétraitement du texte, l'extraction du vocabulaire et l'utilisation de différentes approches de correction d'équilibrage des classes. Une problématique majeure rencontrée dans cette analyse a été le déséquilibre des données entre les deux classes, ce qui a nécessité l'utilisation de techniques spécifiques pour résoudre ce problème.

Les résultats obtenus ont montré des performances variables selon les techniques utilisées et les modèles sélectionnés. Il est important de choisir les bonnes techniques de prétraitement, de gestion du déséquilibre des classes et les bons modèles pour obtenir les meilleurs résultats. L'équilibrage des classes s'est avéré crucial dans l'obtention de performances optimales.

En conclusion, ce projet TAL a permis d'explorer différentes méthodes d'analyse des sentiments et d'analyse des locuteurs, en mettant en évidence l'importance des prétraitements, de l'extraction du vocabulaire, de la gestion du déséquilibre des classes et du choix des modèles. Il offre également des perspectives intéressantes pour des travaux futurs dans ces domaines, notamment en explorant de nouvelles techniques de prétraitement, de modélisation et d'équilibrage des classes pour améliorer les performances de classification et obtenir des résultats plus équilibrés et représentatifs.