Projet NLP
Promo 2025 Mai 2024

Présidents américains

DORIAN.PENSO LES.MARGERY MAXIME.BUISSON SACHA.HIBON

Table des matières

1 Jeu de données

Dans le cadre de ce projet de Traitement Automatique du Langage Naturel, nous avons choisi de travailler sur les discours et écrits des présidents américains. Nous souhaitons réaliser deux tâches distinctes. La première consiste à identifier quel président a écrit un texte donné, c'est à dire de la classification de texte. La seconde vise à générer un texte imitant le style d'un président des États-Unis.

1.1 Présentation

Pour implémenter ces différents modèles, il fallait un jeu de données qui rassemble des textes et des transcriptions des discours des présidents des États-Unis. Nous souhaitions qu'il contienne des textes allant de 1789 à 2024. D'autre part, nous nous sommes principalement concentré.e.s sur Barack Obama, Donald Trump et Joe Biden durant leur mandat. Un tel jeu de données n'existant pas actuellement, nous l'avons créé. Nous avons utilisé plusieurs ensembles de données existants (par exemple, les tweets de Donald Trump) et avons collecté des données supplémentaires (via du scraping). Puis, nous avons reformaté ces données pour les combiner en un ensemble cohérent. Celui-ci est stocké sous la forme de plusieurs fichiers en format CSV.
Chaque fichier CSV comprend les colonnes suivantes :

- **Category** : 'debate', 'tweet', 'speech', ...
- **Name** : le nom de la personne qui a dit ou écrit le texte
- **Date** : la date à laquelle le texte a été dit ou écrit
- **Text** : le texte lui-même

Voici un échantillon de notre jeu de donnés :

Category	**Name**	**Date**	**Text**
tweet	Donald Trump	2011-08-02 18 :07 :48	Republicans and Democrats have both created our economic problems.

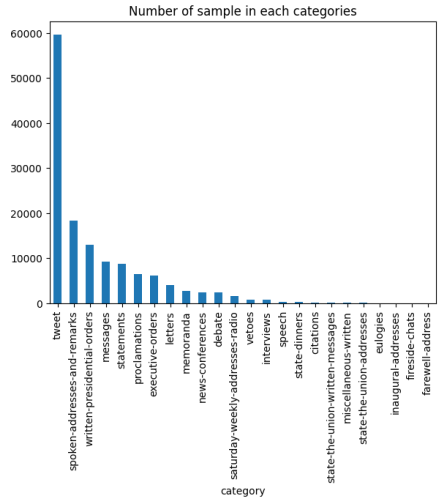
Concernant la catégorie 'debate', notre intention initiale était de conserver la chronologie des débats afin d'avoir accès aux questions et aux réponses. Cependant, après réflexion dans le cadre de notre projet, nous avons conclu que cela n'était pas nécessaire. En effet, nous avons réalisé que nous n'allions pas jusqu'à prendre en compte la question posée pour évaluer la réponse.

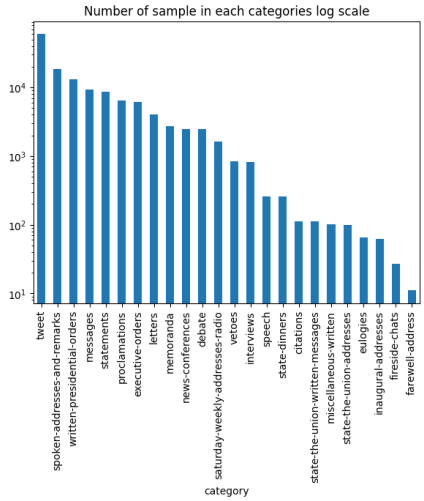
2 Statistiques descriptives des données

Notre jeu de données, que nous nommons corpus, est composé d'un total de 29 documents regroupant différentes catégories telles que les discours, débats, tweets, lettres et interviews. Ces textes ont pour auteurs tous les présidents américains.

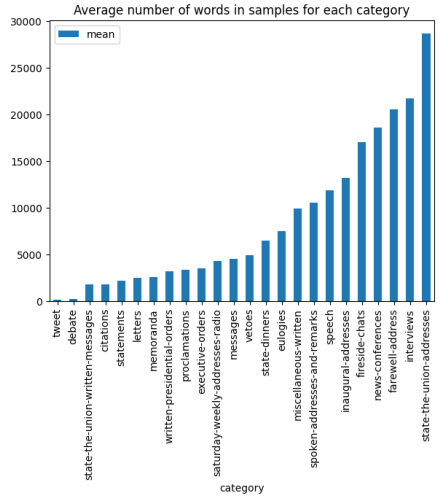
2.1 Catégories des textes

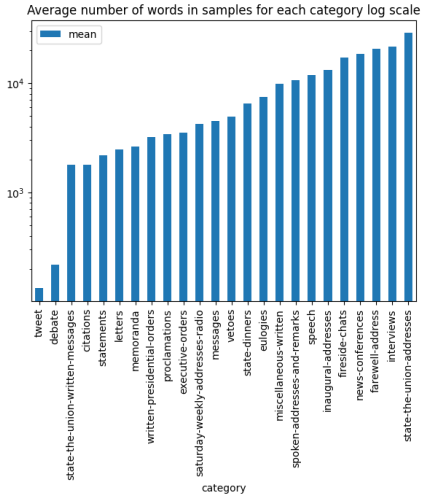
Notre corpus contient 137 685 échantillons répartis de manière inégale entre chaque catégorie.





D'autre part, le nombre de mots varie selon les catégories pour ces échantillons.





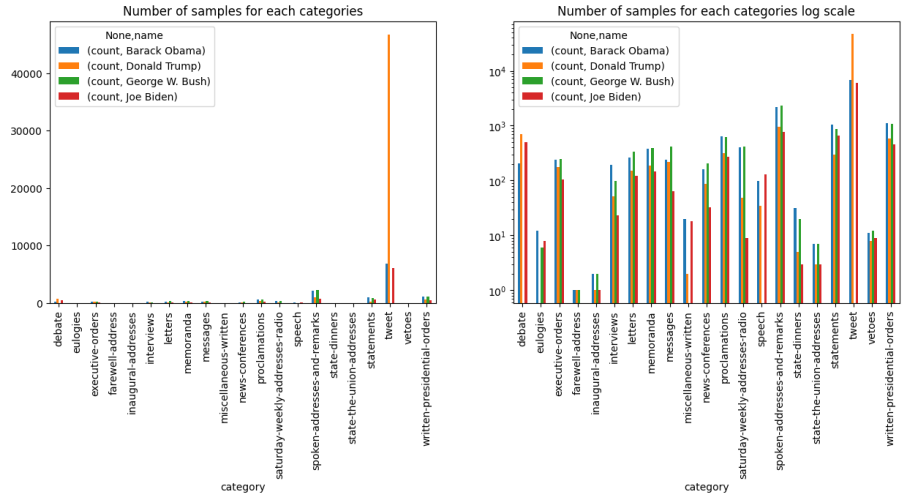
Ainsi, nous remarquons une grande disparité dans le nombre et le format de nos échantillons.

De plus, il est à noter que le contenu des textes de chaque catégorie peut être très différent. En effet, les tweets possèdent beaucoup de liens URL, les lettres comportent eux la signature de leur auteur à la fin tandis que d'autres catégories représentent exclusivement des discussions...

Nous devrons donc être particulièrement prudents quant au biais introduits lors de la classification.

2.2 Sur un sous-ensemble de présidents

Concentrons nous sur Barack Obama, Donald Trump, Joe Biden et un quatrième président à titre de comparaison, George W. Bush.



Nous remarquons que le nombre d'échantillons de tweets de Donald Trump écrase largement celui des autres.

En revanche, une fois mis à l'échelle logarithmique, chaque président possède un nombre d'échantillons par catégorie plutôt équivalent. Ainsi, nous pourrons tester nos modèles de classification de manière indépendante sur chaque catégorie, en ayant la certitude de disposer d'assez d'échantillons pour ce sous-ensemble de présidents.

2.3 Tokens

Enfin, voici les mots les plus fréquemment utilisés par chaque président :

— **Barack Obama** : United, people, States, President, American, other, right, health, every, should

— **Donald Trump** : President, people, United, great, Thank, States, other, American, know, really, @realDonaldTrump

— **Joe Biden** : United, States, people, President, American, other, including, Secretary, Federal, every

Nous remarquons que certains mots tels que 'American' ou 'President' figurent parmi les plus utilisés dans les trois classes. Il est donc probable que nos classificateurs se concentrent sur d'autres mots spécifiques, comme '@realDonaldTrump' pour Donald Trump.

3 Prétraitement du jeu de données

Pour entraîner nos modèles de langue, nous devions d'abord diviser nos chaînes de caractères en *tokens*. Néanmoins, notre jeu de données provient de différentes sources,

certaines avec des formats spécifiques (comme les tweets). C'est pourquoi, cette étape s'est avérée plus complexe qu'il n'y paraissait.

3.1 Tokénisation

3.1.1 À base d'expressions régulières

Tout d'abord, nous avons recherché les tokenisers existants. En effet, nous souhaitions évaluer leur efficacité sur nos différents modèles. Nous avons ainsi constaté que la bibliothèque **NLTK** propose plusieurs tokenisers basés sur des expressions régulières (comme '*word_tokenize*', '*wordpunct_tokenize*' et '*sent_tokenize*').

Cependant, pour obtenir un tokeniser plus adapté à notre corpus, nous en avons créé un spécifique en utilisant des expressions régulières. Pour ce faire, nous avons d'abord identifié les différents types de chaînes de caractères atypiques que les tokenisers de NLTK, ne prenaient pas en compte ou dont la tokenisation ne nous convenait pas. Nous n'avons pas trouvé de modèle capable de les gérer tous. C'est pourquoi, les expressions régulières de notre tokeniser prennent en compte les URLs, les mots contenants des apostrophes, les abréviations (comme U.S.A), les nombres (avec ou sans virgule), les tweets (hashtags et mentions), ainsi que les signatures à la fin d'un texte (par exemple "- president obama").

3.1.2 Byte Pair Encoding

En premier lieu, nous avons implémenté notre propre Byte Pair Encoding.

Afin de pouvoir manipuler nos BPE simplement, nous avons décidé de créer une classe MyBpeTokenizer. A partir de cette classe abstraite nous avons créer nos deux classes MyBpeTokenizerBytePair et MyBpeTokenizerHuggingFace.

Avant BPE : "what would this sentence look like if it was tokenized by byte-pair"
Après BPE : ['wh', 'at', ' ', 'w', 'ou', 'ld', ' ', 'th', 'is', ' ', 'se', 'n', 'te', 'n', 'ce', ' ', 'lo', 'ok', ' ', 'li', 'ke', ' ', 'if', ' ', 'it', ' ', 'w', 'as', ' ', 'to', 'ke', 'n', 'iz', 'ed', ' ', 'by', ' ', 'by', 'te', ' ', '-', ' ', 'p', 'a', 'ir']

Nous avons également exploré les tokenisers BPE existants et avons découvert la bibliothèque **Tiktoken**, développée par OpenAI. Cette bibliothèque propose des tokenisers rapides et performants, utilisés par leurs modèles avancés tels que ChatGPT-4. C'est pourquoi, nous voulions tester ces tokenisers sur nos modèles en raison de leur efficacité et de leur compatibilité avec nos besoins en terme de traitement de texte.

3.2 Normalisation du texte

Comme mentionné précédemment, certains échantillons de notre corpus contiennent des tokens particuliers, tels que les URLs dans les tweets qui renvoient vers d'autres tweets. Après réflexion et expiramentation, nous avons réalisé que la conservation de ces URLs dans le texte pourraient potentiellement biaiser les prédictions et la qualité des générations de nos modèles. En effet, ceux-ci pourraient trop se focaliser sur les caractères qui constituent ces URLs et qui sont, somme toute, plutôt aléatoires. C'est pourquoi, nous avons implémenté une fonction '*convertURL*' qui permet, grâce à une expression régulière, de convertir toutes les URLs par le mot "*TOKURL*".

4 Entrainement de plusieurs modèles

4.1 N-gram

Nous avons donc créé notre premier modèle de génération de texte, dont l'objectif est de produire un texte qui aurait pu être écrit par un président américain.

Pour ce faire, nous avons utilisé la bibliothèque NLTK, et ses différents modèles de langage tels que MLE, StupidBackoff et Laplace. Cependant, le modèle Laplace ayant donné de bons résultats rapidement, nous nous sommes principalement concentrés sur celui-ci.

Configurations	Génération (sans contexte)	Génération (avec contexte)	Perpléxité
Trigramme avec '*convertURL*' sur 'debate', 'tweets' et 'speeches'	"**URL** happy national vietnam with kim jong un, we've been president because this is your neighborhood and with no bounds of color."	"they want open borders" ⇒ "and california wants you to respond quickly in order to salvage hope !"	22425
Trigramme avec '*convertURL*' sur 'speeches'	"that's all over and over a century just five years from us - when the marchers are playing san francisco this weekend about people that"	"they want open borders"⇒ "they wouldn't be solved very quickly after they put into place."	12220
Trigramme sans '*convertURL*' sur 'debate', 'tweets' et 'speeches'	"another governor in disgrace !#buildthewall https ://t.co/s5nd6p6a0dx4fzct add your voice count for @dhsgov border patrol, but i firmly believe that."	"they want open borders" ⇒ "all republicans not realize how much more ground to declare war on cops has already told us for this beloved nation"	23358

Table 1 – Résultats de génération et perplexité pour différentes configurations

Nous remarquons que sans l'utilisation de la fonction '*convertURL*', la perplexité du modèle augmente. De plus, cela semble entraîner une augmentation de la génération d'URLs (plus ou moins bien formatées) réduisant ainsi la quantité de texte générés.

D'autre part, lorsque l'ordre du n-gram est trop élevé, le modèle semble incapable de générer du texte de manière efficace. En effet, il se contente de réciter des fragments de text présents dans le jeu de données, plutôt que de créer des phrases nouvelles et plutôt cohérentes dans leur construction.

En revanche, le fait de ne prendre en compte qu'une catégorie dont le vocabulaire est plutôt simple (contrairement aux tweets), améliore la perplexité du modèle. Toutefois, malgré nos différentes tentatives, pour améliorer la perplexité du modèle celle-ci reste relativement haute. Cela peut être dû à la complexité du corpus, qui introduirait une grande variabilité dans les données. Cette variabilité rendrait plus difficile pour le modèle de capturer les nuances spécifique à chaque type de texte.

4.2 Bayésien naïf

Le bayésien naïf est notre première méthode de classification devant ainsi reconnaître l'auteur d'un texte parmi les présidents des États-Unis.
Cette méthode a immédiatement donné de bons résultats atteignant un f1-score moyen de plus de 0.9 de f1-score en utilisant seulement les tweets. Ceci s'explique par des vocabulaires très différents entre nos trois présidents.

Prenons dans un premier temps un tweet de Donald Trump prédit comme écrit par Donald Trump : "a-rod @yankees had hip surgery &amp, will be out 6 months. do you notice all the "druggies" have bad hips."

(a) Explication influence des mots sur un texte écrit par Donald Trump

name	Barack Obama	Donald Trump	Joe Biden
rod	27.0	533	42.0
yankees	NaN	84	NaN
amp	62.0	5468	526.0
notice	3.0	63	3.0
bad	10.0	1234	46.0

(b) Nombre d'échantillons contenant les mots influents rangés par président

On remarque en effet que les mots qui ont permis de prédire que l'auteur était Donald Trump ("rod", "yankees", "amp", "notice", "bad") sont très utilisés par celui-ci et beaucoup moins par les deux autres présidents. Permettant ainsi de classifier un texte seulement par son vocabulaire.

Répétons l'expérience avec un texte de Barack Obama : "live : president obama is announcing a new plan to boost job growth for middle-class americans.TOKURL"

(c) Explication influences des mots sur un texte écrit par Barack Obama

name	Barack Obama	Donald Trump	Joe Biden
obama	2851	2877	370
president	2775	3503	1412
live	383	830	419
boost	16	20	1
growth	73	146	5

(d) Nombre d'échantillon contenant les mots influents rangés par président

Les occurrences des mots les plus importants ne sont pas à première vue discriminant mais nous avons déja vu que les tweets de Donald Trump était en plus grand nombre que ceux de Barack Obama et Joe Biden. Ainsi, gardant à l'esprit ce déséquilibre, cela permet de comprendre que Barack Obama est celui qui utilise le plus souvent ces termes.

Prenons maintenant un tweet de Joe Biden : "we are months into this crisis, and health care workers still don't have the protective equipment they need. it's a national disgrace. TOKURL"

(e) Explication influences des mots sur un texte écrit par Joe Biden

name	Barack Obama	Donald Trump	Joe Biden
crisis	18.0	89	241
protective	NaN	4	19
health	511.0	332	361
care	638.0	1248	475
workers	88.0	167	162
equipment	NaN	22	22

(f) Nombre d'échantillon contenant les mots influents rangés par président

Cette fois-ci il est intéressant de remarquer que le vocabulaire de Joe Biden est très marqué par la crise sanitaire, permettant encore une fois de faire une distinction facilement avec un sac de mots et le simple comptage de leurs occurrences.

Prenant toutes les catégories de texte (pas seulement les tweets), la moyenne pondérée du f1-score est de 0.87 prouvant ainsi que ces analyses se généralisent.

4.3 Régression logistique

Suite à cela, nous avons implémenté un second modèle de classification sur notre corpus. Tout comme le premier, celui-ci détermine si un texte provient soit de Joe Biden, Donald Trump ou Barack Obama. Pour ce faire, nous avons décidé d'utiliser la bibliothèque **scikit-learn** et ces nombreux modules. Pour évaluer notre modèle, nous avons utilisé diverses métriques telles que la précision, le f1-score, le rappel, ainsi que des outils visuels comme la matrice de confusion. En faisant un premier entraînement du modèle nous sommes d'abord parvenus à un score de 0,88.

En outre, nous souhaitions comprendre les mécanismes décisionnels de notre modèle, notamment la manière dont certains mots influençaient ses choix. Pour ce faire, nous avons utilisé les fonctionnalités offertes par les bibliothèques **eli5** et **lime**. En explorant cette dynamique, nous espérions discerner la cohérence et la fiabilité des précisions, afin de déterminer leur pertinence dans le contexte de nos analyses.

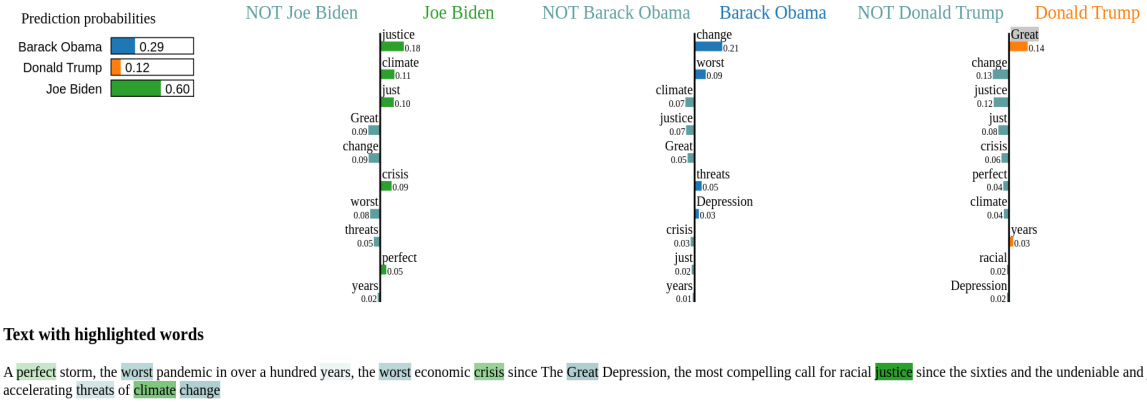
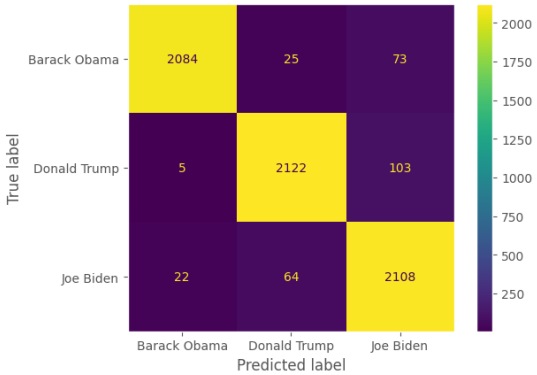


FIGURE 1 – Explicabilité des décisions du modèle

Ce modèle utilisant plusieurs hyperparamètres, nous avons décidé d'optimiser sa précision en effectuant une recherche d'hyperparamètres à l'aide de la bibliothèque Optuna. Suite à cette étude, nous en avons conclu que les meilleurs hyperparamètres sont :

- **tol** : 1,9e-4
- **C** : 0.63
- **solver** : liblinear
- **tokenizer** : gpt4.tokenize

Enfin, après avoir optimisé notre jeu de données, nous obtenons les résultats suivants :



(a) Matrice de confusion

```
Classification Report:
                precision    recall  f1-score   support

Barack Obama       0.99      0.96      0.97      2182
Donald Trump       0.96      0.95      0.96      2230
   Joe Biden       0.92      0.96      0.94      2194

    accuracy                           0.96      6606
   macro avg       0.96      0.96      0.96      6606
weighted avg       0.96      0.96      0.96      6606
```

(b) Métriques

4.4 Word2vec

Nous avons entrepris la création d'un modèle Word2vec dans le but d'améliorer nos différents classificateurs. Initialement, nous l'avons implémenté à partir de zéro, mais avons rapidement constaté que l'entraînement prenait un temps considérable pour seulement une Epoch. Pour remédier à cela, nous avons opté pour l'utilisation de la bibliothèque **Gensim**, qui nous a permis de mettre en place notre premier modèle Word2vec de manière plus efficace.

Une fois notre premier modèle créé, nous devions évaluer ses performances et explorer la possibilité de faire varier les hyperparamètres de Word2vec pour améliorer ses résultats. Pour ce faire, nous avons choisi d'évaluer les similarités entre les mots à l'aide d'un ensemble de données de similarité appelé **WordSim-353**. Celui-ci est disponible au format CSV avec les colonnes *word1*, *word2* et *similarité*, représentant la similarité entre les paires de mots. Nous avons donc calculé la similarité entre chaque paire de mots présente à la fois dans WordSim-353 et dans notre modèle Word2vec, afin de déterminer **le coefficient de corrélation de Spearman**. Ce coefficient mesure la similarité de l'ordre des valeurs entre deux variables, indépendamment de la relation linéaire, ce qui nous permet d'évaluer la performance de notre modèle. Dans un premier temps, nous avons obtenu un coefficient de 0,3, mais en faisant varier les hyperparamètres, nous sommes parvenus à atteindre un résultat maximal de 0,39.

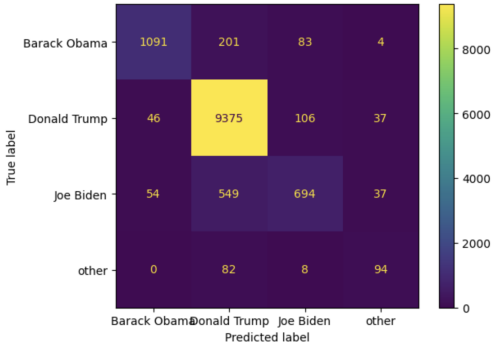
Outre l'évaluation des similarités, nous avons également testé notre modèle en effectuant des opérations entre les mots, telles que "president - minority = donald" ou encore "Obama - Trump = Billclinton". Nos tests ont révélé un résultat logique seulement une fois sur quatre, ce qui suggère que notre modèle peut encore être amélioré pour capturer ces relations de manière plus précise.

Nous avons également examiné les mots les plus proches de certains termes, tels que "président", afin d'évaluer la cohérence des résultats. Dans l'ensemble, les mots proches identifiés semblaient logiques, ce qui indique que notre modèle Word2vec est capable de capturer des relations sémantiques pertinentes.

4.5 Réseaux de neurones feedforwards

4.5.1 Classificateur

Pour notre classificateur feedforward, contrairement aux autres modèles, nous avons rajouté une quatrième classe "Other". Cette class correspond a touts les interlocuteur autres des trois presidents americains dans leurs debats. Pour se faire, avons d'abord utilisé notre tokeniser BPE et notre modèle Word2Vec, tous deux entraînés à partir de notre corpus. Avec cette configuration et en faisant varier les différents hyperparamètres, notre modèle converge bien, mais il reste limité.

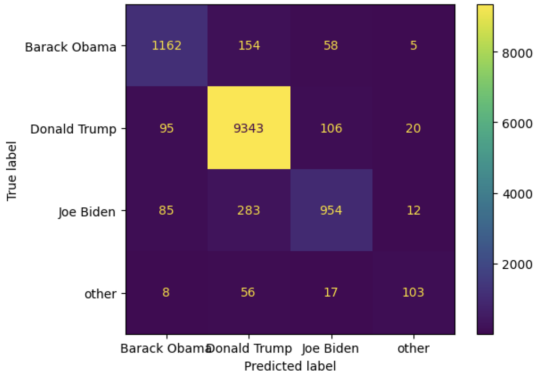


(c) Matrice de confusion

	precision	recall	f1-score	support
Barack Obama	0.92	0.79	0.85	1379
Donald Trump	0.92	0.98	0.95	9564
Joe Biden	0.78	0.52	0.62	1334
other	0.55	0.51	0.53	184
accuracy			0.90	12461
macro avg	0.79	0.70	0.74	12461
weighted avg	0.90	0.90	0.90	12461

(d) Métriques

Nous avons donc décidé de remplacer notre tokeniser par '*word_tokenize*' de la bibliothèque NLTK et de garder notre Word2vec. Cette modification a conduit à des résultats améliorés. Ensuite, nous avons poussé notre démarche plus loin en utilisant Optuna pour optimiser les paramètres, ce qui nous a donné les résultats suivants.



(e) Matrice de confusion

	precision	recall	f1-score	support
Barack Obama	0.86	0.84	0.85	1379
Donald Trump	0.95	0.98	0.96	9564
Joe Biden	0.84	0.72	0.77	1334
other	0.74	0.56	0.64	184
accuracy			0.93	12461
macro avg	0.85	0.77	0.81	12461
weighted avg	0.93	0.93	0.93	12461

(f) Métriques

4.5.2 Utilisation de Word2Vec

L'utilisation de Word2Vec dans un réseau feedforward nous a semblé être la meilleure solution pour vectoriser les tokens. Nous savons qu'il améliore à la fois les résultats et les performances du modèle. En effet, Word2Vec conserve des informations contextuelles sur les mots, ce qui permet à notre modèle de saisir des relations complexes entre les mots qui ne seraient pas immédiatement évidentes autrement.

Comparé au One-Hot Encoding, qui crée des vecteurs de dimension égale au nombre de mots dans le vocabulaire, Word2Vec convertit les mots en vecteurs de taille relativement

plus petite. Cela réduit la complexité computationnelle, améliore l'efficacité du modèle et facilite l'entraînement de notre réseau de neurones.

4.5.3 Générateur de texte

Méthode

Nous avons tenté d'implémenter un générateur de texte en usant d'un réseau neuronal feedforward.
Concentrons nous sur un 4-gram, c'est-à-dire un réseau générant un texte à partir des 4 mots précédents. Ainsi, la couche en entrée est de 4 neurones, la couche en sortie est de la taille du vocabulaire (peu important le nombre de gram) et les couches sont toujours des hyperparamètres à faire varier pour améliorer le réseau.
Pour entraîner notre réseau, chaque échantillon représentait 4 mots qui se suivaient pour être le 'X' (ce que l'on met en entrée) ainsi que le cinquième mot qui les suit désignant le 'y' (ce que l'on cherche en sortie).
Ces échantillons ont été créés à partir de chaque texte de notre corpus en extrayant par 'slidingwindow' chaque échantillon.

Résultats

Notre réseau neuronal arrive très facilement à converger en terme de 'précision' mais son 'rappel' reste, peu importe les configurations, très mauvais.
Ceci est facilement explicable car lorsque notre réseau doit générer un peu mot, il est "précis" si son mot fait partie des réponses possibles. Chaque groupe de mots en entrée pouvant être associé à beaucoup de mots différents en sortie ('I love the' ⇒ 'chocolate', 'sun', 'swimming pool'), il n'est pas si compliqué pour notre réseau d'être correct.
En revanche, pour ce qui est du 'rappel' pour le mot 'sun' par exemple, le rappel représente le nombre de fois que l'on a prédit le mot 'sun' pour tous les groupes de mots le précédent ("I love the", "because of the", "I look at"). Comme précisé plus tôt, ces groupes de mots ont plusieurs possibilités en sortie ce qui mène à un rappel faible pour le mot 'sun' ainsi que pour tous les mots.

Ensuite, d'autres remarques sont à faire sur les résultats :

- lorsque le nombre de gram est faible, le texte généré est gramaticallement bon
- sans génération aléatoire, c'est à dire toujours généré le mot qui a la meilleure prédiction, la génération peut se retrouver bloquée :
  - **3-gram** : I Romney , I Romney , I Romney , I Romney , I Romney ...
  - **5-gram** : for three month , we ' ve got to make sure that we ' ve got to make sure that we ' ve got to make sure that we ' ve got to make sure that we ' ve got to make sure that we ' ve got ...
- après génération aléatoire parmi les mots les plus probables, des tokens sont généré trop souvent : "to make to the kind , we have been , we have not to take for . we ' ve done . i said , ' s the thing that you know happened . i said we ' ve not . and the kind of american . that we are"
  - Notre réseau aime les ponctuations ',', '.' qui sont en effet des tokens extrêmement présents dans les textes.
  - Il aime les mots simples

  - Cela créé une qualité de langue très faible
- Avec un trop grand nombre de gram, le générateur produit des phrases qui n'ont plus de sens et qui subissent les mêmes problématiques précédentes. Voici des textes générés à partir d'un 8-gram :
  - 'other' 'republican' 'candidate' 'for' 'the' 'nomination' 'and' 'he' ⇒ "said you be , , you said we can have , , and that is . you want . i said i said we ' ll be , we said , you have , and that is to are the the . we said . they want ' ll have"
  - ',' 'folk' 'like' 'michelle' ',' 'kid' 'probably' 'who' ⇒ "want , , we can ' romney. . . , we have to are . i am that . you want the a of of . and you are , they said i said i said , they are a a . we ' ll be , we can do to be"

Conclusions

Notre réseau n'est pas un très bon générateur de texte.
Des possibles améliorations seraient de :

- trouver un moyen de réduire la génération de mot tel que 'you', 'a', 'to' tout en garatissant syntaxe correcte
- ajouter plus de poids au contexte proche
- utiliser la perplexité pour entrainer le modèle

5 Limitations de vos approches, difficultés rencontrées et pistes d'améliorations

Une des premières difficultés que nous avons rencontrées a été la création de notre corpus. D'une part, il fallait trouver des sites qui contenaient les données que nous voulions scraper. D'autre part, les différents formats dans lesquels nous récupérions les données nous ont contraints à formater chaque fichier selon son contenu, que ce soit pour le scraping ou pour les jeux de données déjà existants.

Une seconde difficulté rencontrée était que nos modèles de classification obtenaient de très bons résultats dès le début, sans appliquer de prétraitement ou d'optimisation des hyperparamètres, lorsqu'ils étaient testés sur nos trois cétegories principales. Pour essayer de comprendre ce qui pourrait provoquer cela, nous nous sommes penchés sur le corpus. Nous nous sommes rendus compte que la catégorie 'speech' contenait peu d'échantillons mais que leur taille était beaucoup plus grande par rapport aux autres catégories. C'est pourquoi pour uniformiser les données , nous avons décidé de diviser ces grands échantillons en plusieurs plus petits. En entraînant ensuite la régression logistique sur ce nouveau jeu de données, sans optimisation, les résultats se sont avérés moins bons, avec une précision d'environ 0,70.

Après analyse des échantillons de la catégorie 'speech', nous nous sommes rendus compte que Barack Obama et Donald Trump ont un vocabulaire quatre fois plus grand que celui de Joe Biden. Cela est dû au fait que la taille des 'speech' de ces derniers est beaucoup plus grande que celle de Joe Biden. Ainsi, on peut en conclure que le modèle se base probablement sur la taille des échantillons pour faire ses prédictions .Une amélioration possible de ce modèle consisterait donc à s'assurer que les échantillons des

présidents aient à peu près la même taille afin que le modèle ne se focalise pas sur cet aspect du texte.

Enfin, la dernière difficulté rencontrée était que nous n'avions pas un cahier des charges assez précis concernant la classification. Nous devions prédire le président ayant prononcé un certain discours ou écrit un certain texte mais, fallait-il supprimer leur signature à la fin des lettres ? Fallait-il que notre classicateur parvienne à reconnaître une portion seul d'un texte ? De quelle taille devaient-être nos textes ? En résumé, nous manquions d'échantillon de tests objectifs.