RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIQUE

UNIVERSITÉ DES SCIENCES ET DE LA TECHOLOGIE HOUARI BOUMEDIENE

Faculté d'informatique Département des Systèmes Informatique (SIQ)

Filière : Informatique

Section M2 HPC

Rapport Projet NLP :

Classification de Documents avec Deep Learning sur un corpus de BBC news

---

Réalisé Par :

SERBIS	**Amani**	**191931042205**
LEMDANI	**Nour Hassiba**	**191931063501**
KHEDDACHE	**AZIZA**	**191931048870**
BEKKAR	**YOUSRA**	**192031093975**

Alger, 28 Decembre 2024

1. Introduction

Objectif du projet

Ce projet a pour objectif de développer un modèle de classification automatique de documents en plusieurs catégories thématiques en s'appuyant sur des techniques de deep learning. En utilisant un corpus d'articles "BBC News" et des architectures de réseaux neuronaux récurrents (RNN), le projet vise à démontrer la capacité des représentations vectorielles et des modèles avancés à extraire des informations pertinentes pour la classification des textes.

Plan du rapport

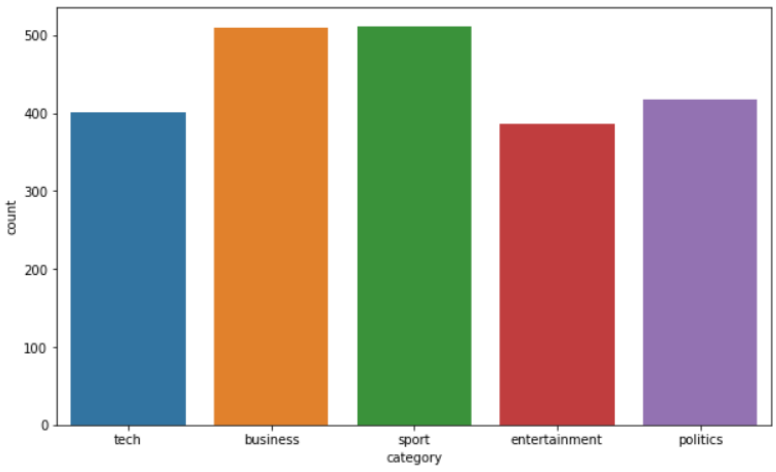
Le rapport est organisé comme suit :

1. Introduction : Présentation du projet et de ses objectifs.
2. Constitution et Préparation des Données : Description du corpus utilisé et étapes de pré-traitement.
3. Conception et Implémentation du Modèle de Classification : Détails de l'architecture et des méthodes employées.
4. Évaluation et Analyse des Résultats : Analyse des performances et limitations du modèle.
5. Développement de l'Interface Utilisateur : Fonctionnalités principales et technologies utilisées.
6. Conclusion : Résumé des résultats obtenus et perspectives d'amélioration.

2. Constitution et Préparation des Données

2.1. Collecte des données : BBC news Dataset

Pour ce projet, nous avons choisi d'utiliser le corpus "BBC News" disponible sur Kaggle. Ce dataset contient **2225 articles** extraits de la BBC et répartis en cinq catégories principales : business, entertainment, politics, sport et tech. Chaque catégorie est représentée de manière équilibrée, ce qui facilite l'entraînement d'un modèle de classification supervisée.



Les articles incluent une diversité de styles et de sujets, allant de l'économie mondiale aux événements

sportifs, en passant par la politique, la technologie, et les divertissements. Cette hétérogénéité rend le corpus particulièrement pertinent pour entraîner un modèle capable de capturer des variations thématiques et contextuelles.

Ce corpus présente également l'avantage d'être étiqueté de manière claire et d'offrir une qualité de texte élevée, sans erreurs majeures ou informations ambiguës. Cela permet de se concentrer davantage sur la performance du modèle plutôt que sur le nettoyage excessif des données.

2.2. Prétraitement

Le prétraitement des données est une étape essentielle dans la construction d'un modèle performant. Voici un aperçu détaillé des opérations de prétraitement appliquées au corpus BBC News pour assurer la qualité des données et maximiser la pertinence des résultats de classification :

- **Filtrage des catégories**

Tout d'abord, nous avons filtré les articles en supprimant la catégorie "entertainment", car elle est moins pertinente pour notre analyse. Cela nous a permis de nous concentrer sur des catégories plus représentatives de notre objectif, à savoir **sport**, **business**, **politics**, et **tech**. Ce filtrage contribue à réduire la complexité de la tâche en éliminant des catégories potentiellement confuses, et aide ainsi à améliorer la précision du modèle.

- **Suppression des doublons**

Une étape cruciale a consisté à éliminer les doublons présents dans le jeu de données, garantissant ainsi l'intégrité de notre analyse. Nous avons d'abord comptabilisé les doublons à l'aide de la méthode duplicated() de Pandas. Ensuite, nous avons utilisé la méthode drop_duplicates() pour supprimer ces doublons, ce qui a permis de maintenir un ensemble de données unique et représentatif. Après cette opération, la forme du DataFrame a été vérifiée pour nous assurer de la suppression effective des doublons.

- **Augmentation des données**

Afin d'enrichir le corpus et d'ajouter de la diversité linguistique, nous avons appliqué une technique d'augmentation des données par la substitution de mots par leurs synonymes. Cette approche aide le modèle à mieux généraliser et à éviter le surapprentissage (overfitting). Nous avons utilisé le module wordnet de NLTK pour générer des synonymes de chaque mot dans un texte. Des versions modifiées des articles ont été créées en remplaçant certains mots par des synonymes, avec l'objectif de créer trois versions uniques pour chaque article.

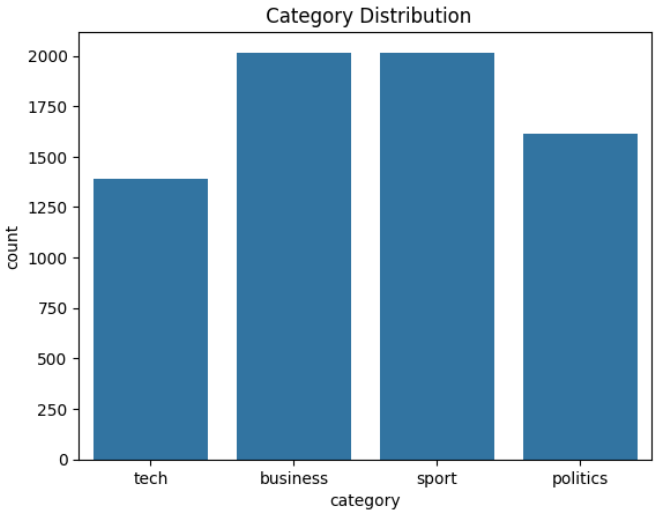
L'augmentation des données a permis de multiplier la quantité de données d'entraînement tout en maintenant leur diversité et leur représentativité.

Notons que 3 nouvelles versions du texte on été crées, ce qui augmente la taille du dataset par un facteur de 4 (1 texte original + 3 augmentations). On est passé de **2225** textes dans notre corpus initial à **7028**.

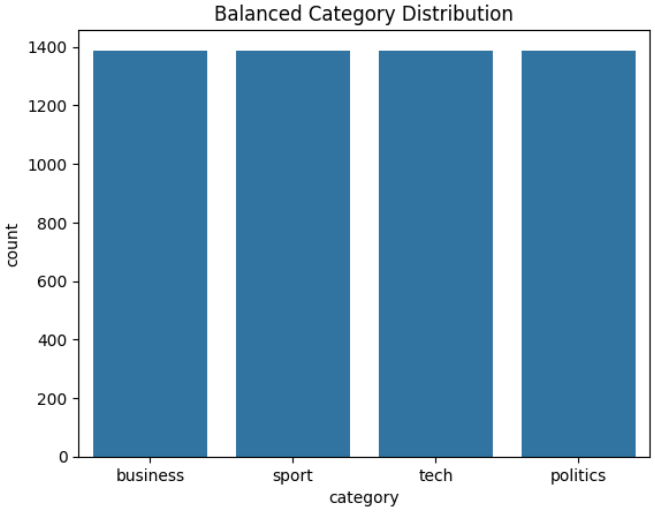
- **Équilibrage des classes**

Pour garantir que le modèle apprenne de manière équitable pour chaque catégorie, nous avons procédé à un équilibrage des classes. En effet, certaines catégories étaient sur-représentées par rapport à d'autres, ce qui pouvait entraîner un biais dans l'apprentissage. Nous avons choisi la catégorie la moins représentée comme référence, puis échantillonné les autres catégories pour avoir un nombre égal d'exemples par catégorie. Cette approche a permis de réduire les risques de déséquilibre et d'assurer que le modèle apprenne de manière optimale sur toutes les classes.

En parlant nombres on est partie de:



À :



- **Nettoyage des textes**

Les textes bruts ont été nettoyés pour en éliminer les éléments inutiles et garantir leur qualité. Plusieurs étapes ont été effectuées à ce niveau :

- **Suppression des mots vides (stopwords)** : Les mots comme "the", "is", et "and" qui n'apportent pas de valeur sémantique ont été retirés.
- **Lemmatisation** : Cette étape a permis de réduire les mots à leur forme de base (par exemple, "running" devient "run"), facilitant ainsi la compréhension du modèle.
- **Tokenisation** : Le texte a été découpé en mots (ou tokens) afin de préparer les données pour la phase suivante.

Toutes ces étapes ont été réalisées à l'aide de la bibliothèque NLTK, qui a permis de nettoyer et d'uniformiser les données de manière efficace.

- **Tokenisation et Séquençage des Textes**

Une fois les textes nettoyés, nous avons utilisé un tokenizer pour convertir les textes en une représentation numérique compréhensible pour le modèle de deep learning. Le tokenizer de Keras a été utilisé avec une taille de vocabulaire de 30 000 mots et un token spécial **<OOV>** pour gérer les mots inconnus (Out Of Vocabulary).

Les textes ont été convertis en **séquences d'entiers**, chaque mot étant remplacé par son index dans le vocabulaire. Afin de garantir une uniformité dans la longueur des textes, nous avons fixé une longueur maximale de **300** mots. Les textes plus courts ont été complétés avec des zéros (padding), tandis que ceux plus longs ont été tronqués.

- **Padding des Séquences**

Le **padding** est une étape essentielle lors du traitement des données textuelles pour les modèles de machine learning. En effet, les textes (ou séquences) peuvent avoir des longueurs variables, mais pour que le modèle puisse les traiter correctement, il est nécessaire que toutes les séquences aient la même longueur. Si les séquences sont plus courtes que la longueur cible, des zéros sont ajoutés pour les compléter. Si elles sont trop longues, elles sont tronquées à la longueur maximale autorisée.

- **maxlen=max_length** : Ce paramètre définit la longueur maximale à laquelle toutes les séquences seront ajustées. Les séquences plus longues que max_length seront tronquées, tandis que celles qui sont plus courtes seront complétées avec des zéros.
- **padding='post'** : Cette option spécifie que les zéros seront ajoutés à la fin de la séquence. Il est aussi possible de les ajouter au début avec padding='pre', mais cela dépend du type de modèle et de l'architecture de réseau de neurones utilisée.
- **truncating='post'** : Si une séquence dépasse la longueur définie par maxlen, elle sera coupée à la fin (au lieu du début, si truncating='pre').

Cette opération garantit que toutes les séquences ont la même longueur, ce qui est indispensable pour l'entrée des données dans le modèle de machine learning. Il n'y a pas de répétition ici, mais plutôt une normalisation des longueurs des séquences.

- **Encodage des catégories**

Les étiquettes des textes (les catégories) ont été encodées sous forme numérique à l'aide de la méthode astype('category').cat.codes de Pandas. Cette étape a permis de transformer les catégories textuelles en labels numériques, facilitant leur utilisation dans un modèle d'apprentissage automatique.

2.3. Prétraitement Avancé : Embedding avec Word2Vec

En plus des étapes classiques de nettoyage et de tokenisation, nous avons intégré des représentations vectorielles des mots à l'aide de **Word2Vec**, un modèle de **représentation distribuée des mots**. Word2Vec est un modèle pré-entraîné qui transforme chaque mot en un vecteur dense dans un espace de **300 dimensions**, ce qui permet au modèle de capter les relations sémantiques entre les mots.

- **Chargement du modèle Word2Vec**

Nous avons utilisé le modèle pré-entraîné **GoogleNews-vectors-negative300.bin**, qui est un modèle Word2Vec formé sur un large corpus de textes (Google News). Ce modèle contient des vecteurs pour des centaines de milliers de mots en anglais.

Le modèle a été chargé en utilisant la fonction KeyedVectors.load_word2vec_format() de la bibliothèque Gensim. Cette méthode permet de charger le modèle et d'accéder aux vecteurs de mots dans un format compatible avec l'indexation.

- **Création de la Matrice d'Embeddings**

Une fois le modèle chargé, nous avons créé une **matrice d'embeddings** qui associe chaque mot du vocabulaire de notre tokenizer à un vecteur provenant de Word2Vec. La matrice d'embeddings est utilisée pour la représentation des mots dans le modèle de deep learning.

Pour chaque mot du vocabulaire généré par le tokenizer, nous vérifions s'il existe dans le modèle Word2Vec. Si oui, nous assignons le vecteur correspondant à ce mot dans la matrice d'embeddings. Si un mot n'est pas présent dans Word2Vec, il est simplement initialisé avec un vecteur de zéros.

La création de cette matrice permet de fournir des **représentations sémantiques** des mots dès le début de l'entraînement, ce qui est particulièrement utile pour améliorer la performance du modèle sur des tâches de classification textuelle.

- **Intégration dans le Modèle**

Une fois la matrice d'embeddings créée, elle est utilisée pour initialiser les poids de la couche d'embedding de notre modèle de deep learning. Cela permet de commencer l'apprentissage avec des représentations de mots pré-entraînées, qui seront ensuite affinées pendant l'entraînement en fonction des spécificités de la tâche de classification.

3. Conception et Implémentation du Modèle

3.1. Architecture du Modèle

Le modèle de classification que nous avons conçu repose sur une architecture de type réseau de neurones récurrents (RNN) avec des couches LSTM (Long Short-Term Memory) et une couche d'attention pour améliorer l'efficacité du modèle et sa capacité à capturer les dépendances contextuelles et sémantiques des séquences textuelles. La description détaillée des différentes couches et de leur fonction est présentée ci-dessous :

1. **Couche d'entrée (Input Layer)**
    - **Description :** Cette couche attend des séquences de texte de longueur maximale définie par max_length (fixée ici à 300 mots).
    - **Raison du choix :** Elle permet de normaliser la longueur des séquences textuelles et assure une compatibilité avec les couches suivantes du modèle.
2. **Couche d'embedding (Embedding Layer)**
    - **Description :** La couche d'embedding transforme chaque mot en une représentation vectorielle dense de dimension fixe. Elle utilise les embeddings préentraînés de Word2Vec pour capturer les relations sémantiques entre les mots.
        - **input_dim :** Taille du vocabulaire.
        - **output_dim :** Dimension des vecteurs d'embedding (300).
        - **weights :** Matrice d'embeddings préentraînés (Word2Vec).
        - **trainable :** Fixée à False pour empêcher la mise à jour des embeddings pendant l'entraînement.
    - **Raison du choix :** L'utilisation des embeddings préentraînés permet au modèle de tirer parti des relations sémantiques acquises par le modèle Word2Vec sur un large corpus de texte, ce qui améliore la compréhension des textes.
3. **Couche LSTM (Long Short-Term Memory Layer)**
    - **Description :** La couche LSTM avec 128 unités cachées et un taux de dropout de 0,2 est utilisée pour capturer les dépendances contextuelles dans les séquences de texte.
    - **Raison du choix :** Les LSTM sont particulièrement adaptés pour traiter des séquences de données et gérer les dépendances à long terme, ce qui les rend particulièrement efficaces pour les tâches de classification de texte.
4. **Couche d'attention (Attention Layer)**

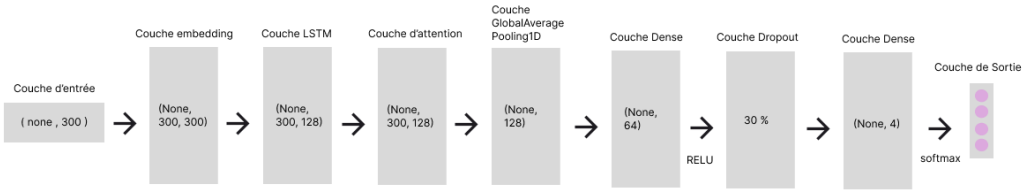
- **Description :** La couche d'attention permet au modèle de se concentrer sur les parties pertinentes de la séquence textuelle en calculant un poids pour chaque mot en fonction de son importance pour la tâche de classification.
- **Raison du choix :** L'ajout de la couche d'attention améliore la capacité du modèle à accorder plus de poids aux mots-clés ou phrases importantes, ce qui se traduit par une meilleure précision dans la classification.

5. **Couche de pooling global (Global Average Pooling Layer)**
   - **Description :** Cette couche condense les informations extraites de la couche d'attention en un vecteur de dimension fixe.
   - **Raison du choix :** Elle permet de réduire la complexité du modèle tout en conservant les informations essentielles nécessaires à la classification.
6. **Couches fully-connected (Dense Layers)**
   - **Description :**
     - **Première couche dense :** Une couche dense de 64 neurones avec une activation ReLU est utilisée pour ajouter de la non-linéarité au modèle.
     - **Dropout :** Un taux de dropout de 0,3 est appliqué pour réduire le surapprentissage.
     - **Couche de sortie :** Une couche dense avec une activation softmax est utilisée pour produire les probabilités associées à chaque catégorie (classe).
   - **Raison du choix :** Ces couches permettent de traiter et de transformer les caractéristiques extraites par les couches précédentes pour produire des prédictions sur les classes. Le dropout permet également de régulariser le modèle afin d'éviter le surapprentissage.
7. **Compilation du Modèle**
   - **Optimiseur :** L'optimiseur Adam est utilisé pour ajuster efficacement les poids du modèle.
   - **Fonction de perte :** La fonction de perte sparse_categorical_crossentropy est adaptée aux tâches de classification multi-classes.
   - **Métriques :** La métrique d'évaluation utilisée est l'exactitude (accuracy), qui mesure la proportion de prédictions correctes.
   - **Raison du choix :** Ces choix sont standards pour les tâches de classification supervisée et sont largement utilisés pour leur efficacité dans les modèles de deep learning.

L'architecture globale de ce modèle permet de capturer à la fois les dépendances contextuelles et les relations sémantiques dans les textes, tout en intégrant des mécanismes

d’attention pour améliorer la compréhension du contenu textuel, ce qui se traduit par une meilleure performance en termes de classification.

Schéma de l’architecture de notre modèle



3.2. Entraînement et Évaluation du Modèle

1. Entraînement du modèle

Le processus d'entraînement de notre modèle repose sur une stratégie visant à maximiser ses performances tout en tenant compte des spécificités de notre corpus et des contraintes liées à l'apprentissage profond.

- **Paramètres d'entraînement :**
    - **Nombre d'époques (epochs) :** Fixé à 10, ce choix permet au modèle d'apprendre suffisamment les relations dans les données sans risquer de surapprentissage (*overfitting*). Cette décision est basée sur des observations préliminaires et la taille modérée du corpus.
    - **Taille de batch (batch_size) :** Une taille de batch de 32 a été adoptée. Elle représente un compromis idéal entre la stabilité des gradients et une vitesse de convergence optimale, tout en évitant des dépassements de mémoire sur notre configuration matérielle.
    - **Fraction de validation (validation_split) :** Un pourcentage de 20 % des données d'entraînement a été réservé pour la validation. Ce paramètre garantit une évaluation continue de la performance du modèle sur des données non vues durant l'entraînement.
    - **Poids des classes (class_weight) :** Les poids des classes ont été ajustés en fonction de leur distribution dans le corpus afin de corriger tout déséquilibre éventuel. Cela permet de garantir que le modèle traite équitablement les classes majoritaires et minoritaires.
    - **Verbose :** Fixé à 1, ce paramètre affiche des détails sur la progression de l'entraînement, incluant les métriques et l'évolution de la perte à chaque époque.

2. Évaluation du modèle

Après l'entraînement, une étape clé a été l'évaluation du modèle sur un ensemble de test indépendant (X_test, y_test). Cette étape permet de mesurer la capacité du modèle à généraliser, c'est-à-dire à effectuer des prédictions précises sur des données inconnues.

- **Métriques utilisées :**
    - **Test Loss :** La perte calculée sur l'ensemble de test quantifie les erreurs faites par le modèle. Une valeur faible indique que le modèle est performant.
    - **Test Accuracy :** Le pourcentage de prédictions correctes sur les données de test reflète la capacité globale du modèle à distinguer les catégories.

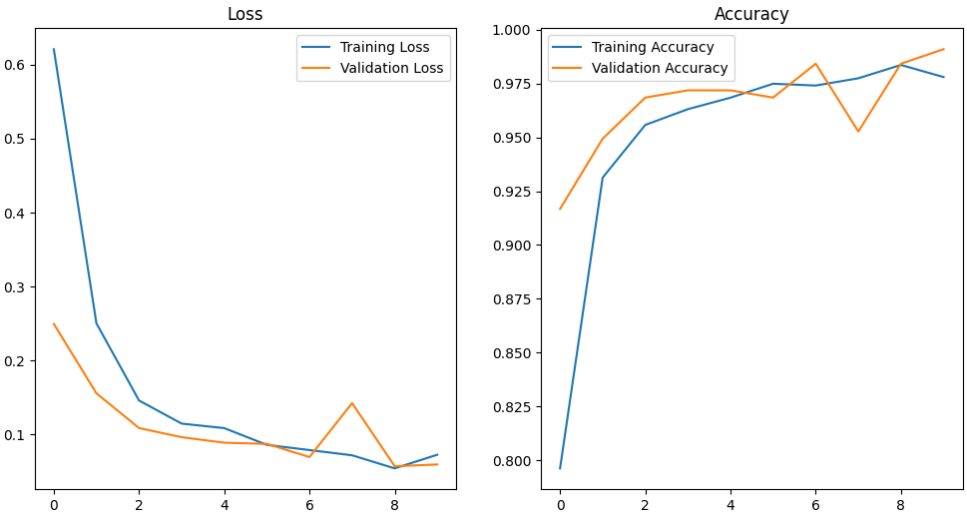
En résumé, cette phase nous a permis de valider notre approche et de mettre en évidence les performances de notre modèle dans un contexte pratique.

3.3. Visualisation des performances et commentaires

Pour mieux comprendre les performances du modèle, plusieurs visualisations ont été générées :

- **Test Loss :** Avec une perte test de 4.58 %, le modèle a montré qu'il parvient à minimiser efficacement les erreurs sur les données qu'il n'a jamais vues auparavant. Cela reflète une excellente généralisation, ce qui est crucial dans des tâches de classification.
- **Test Accuracy :** Une précision de 98.82 % sur l'ensemble de test confirme la capacité du modèle à classifier correctement les documents dans leurs catégories respectives. Ce niveau de précision, proche de celui obtenu pendant la validation (96.77 %), montre que le modèle n'a pas sur-appris les données d'entraînement.
- **Training Accuracy :** Le modèle a atteint une précision de 97.62 % pendant l'entraînement, indiquant qu'il a bien appris les relations présentes dans les données d'apprentissage tout en conservant un écart raisonnable avec la validation.
- **Validation Accuracy :** Avec 96.77 %, la validation démontre que le modèle reste performant même sur les données réservées à cet usage. Cet équilibre entre les performances d'entraînement et de validation est un indicateur d'un modèle bien régularisé.

➢ **Analyse des courbes d'apprentissage : Perte et Précision**



Les graphes ci-dessus présentent les courbes d'apprentissage du modèle, en suivant l'évolution de la perte et de la précision pour les ensembles d'entraînement et de validation au cours des époques.

Courbe de perte :

- La perte d'entraînement diminue rapidement au début, passant de 0,6 à environ 0,05, ce qui montre que le modèle apprend efficacement les caractéristiques des données.
- La perte de validation suit une tendance similaire, atteignant une valeur très faible (environ 0,1) et restant relativement stable après quelques époques. Cela indique que le modèle généralise bien sur les données de validation, sans signe évident de sur-apprentissage (overfitting).

Courbe de précision :

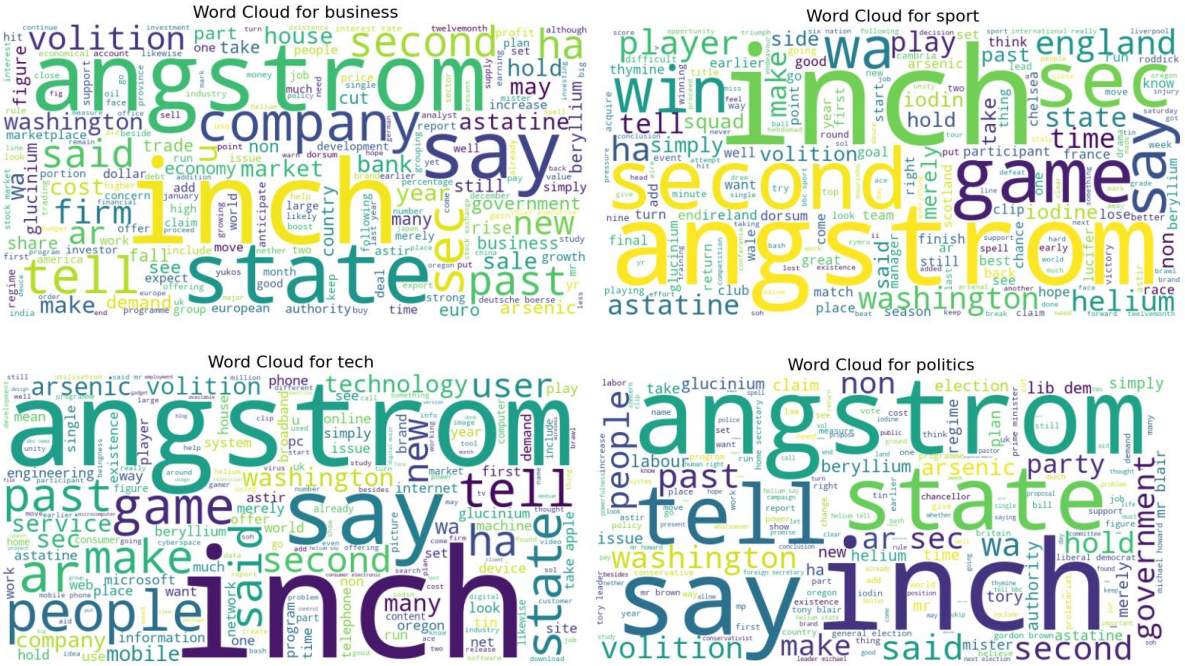
- La précision d'entraînement augmente rapidement, atteignant plus de 97 % après quelques époques. Cela reflète que le modèle s'ajuste bien aux données d'entraînement.
- La précision de validation suit une trajectoire similaire, dépassant 95 % dès les premières époques et se stabilisant autour de 97 %, ce qui confirme que les performances du modèle sur des données non vues sont robustes.

Observations générales :

- Les deux courbes (entraînement et validation) sont cohérentes, sans divergence marquée. Cela montre un bon équilibre entre l'apprentissage du modèle et sa capacité à généraliser.
- Les résultats suggèrent que le modèle est bien optimisé et qu'il n'y a pas de problème majeur comme un sous-apprentissage (underfitting) ou un sur-apprentissage (overfitting).

Ces courbes illustrent la solidité des performances du modèle et sa capacité à gérer des données inconnues efficacement.

- **Wordclouds par catégorie**
  Les wordclouds générés pour chaque catégorie (*business*, *sport*, *tech*, *politics*) mettent en évidence les mots-clés dominants.

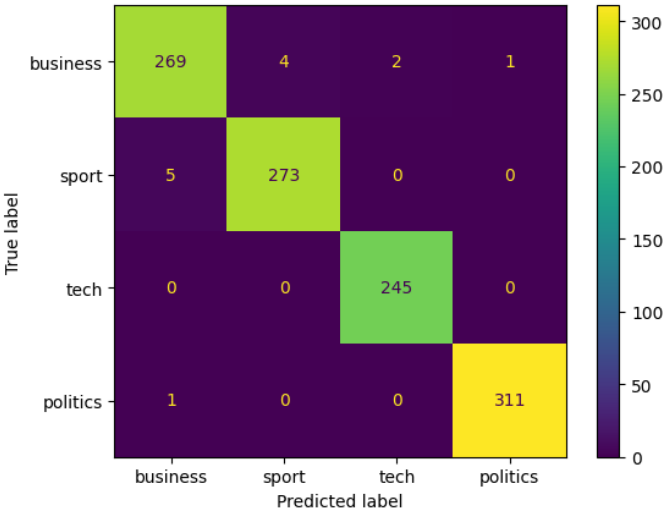


Lors de l'analyse des *wordclouds* générés pour chaque catégorie, nous avons remarqué la présence de certains mots communs apparaissant fréquemment dans toutes les catégories. Afin de mieux comprendre cette observation, une vérification approfondie a été effectuée pour quantifier ces mots partagés. Résultat : 4175 mots sont en commun entre les différentes catégories.

Bien que ce chiffre puisse sembler significatif, nous avons choisi de ne pas les supprimer pour plusieurs raisons essentielles. Tout d'abord, étant donné que notre corpus n'est pas particulièrement volumineux, retirer un grand nombre de mots aurait pu appauvrir la richesse sémantique des données. Ensuite, certains de ces mots, bien qu'apparaissant dans plusieurs catégories, peuvent conserver une importance contextuelle et contribuer indirectement à la différenciation lors de la classification.

Ainsi, cette décision reflète une approche équilibrée : conserver des données pertinentes tout en minimisant le risque de perdre des informations clés.

➢ **Matrice de Confusion**



Cette matrice de confusion reflète les performances de mon modèle sur la classification des articles dans quatre catégories : "tech", "business", "sport", et "politics". Chaque ligne correspond aux véritables étiquettes (les catégories réelles des articles), tandis que chaque colonne représente les étiquettes prédites par le modèle.

Les résultats montrent que le modèle a globalement une très bonne capacité à reconnaître les catégories :

- **Catégorie "business" :**
  Parmi 276 articles de "business", 269 ont été correctement classés. Cependant, il y a eu 4 confusions avec "sport", 2 avec "tech", et 1 avec "politics". Ces erreurs pourraient être dues à un chevauchement thématique, comme des articles portant sur des sujets sportifs dans le monde des affaires ou des technologies appliquées au business.

- **Catégorie "sport" :**
  Sur 278 articles de "sport", 273 ont été correctement classés. On observe 5 confusions avec "business". Cela peut être lié à des articles de sport ayant des dimensions économiques, comme les finances dans les clubs ou les événements sportifs.

- **Catégorie "tech" :**
  Cette catégorie montre une très forte précision avec 245 articles correctement classés sur 245. Aucun article de "tech" n'a été mal classé dans d'autres catégories.

- **Catégorie "politics" :**
  Avec 311 articles correctement classés sur 312, cette catégorie démontre aussi une grande robustesse. Une seule confusion est survenue, où un article a été classé à tort comme appartenant à la catégorie "business". Cela pourrait s'expliquer par des sujets touchant à la fois aux affaires politiques et économiques.

Observations générales :

Les erreurs restent limitées et se concentrent principalement sur les interactions entre les catégories "business" et "sport", ainsi que de légères confusions entre "business" et "politics". Ces résultats témoignent d'une forte capacité de classification du modèle, avec des performances particulièrement remarquables dans les catégories "tech" et "politics".

Ces résultats montrent que le modèle répond bien à l'objectif de classification tout en ayant des performances robustes sur la majorité des catégories**.**

4. Évaluation et Analyse

4.1. Méthodes d'évaluation

Pour évaluer les performances du modèle, les métriques suivantes ont été utilisées :

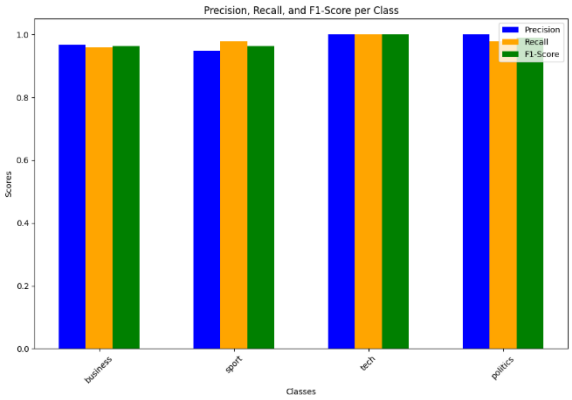
- **Précision, rappel et F1-score** : essentielles pour mesurer l'équilibre entre les prédictions correctes et les erreurs.

Pourquoi ces choix ? Ces métriques offrent une vue d'ensemble fiable pour un problème de classification multi-classes, garantissant une évaluation complète et objective.

4.2. Résultats obtenus

Le tableau et histogramme suivant résument les performances par classe :

Classe	Précision	Rappel	F1-Score	Support
Business	0.99	0.92	0.96	276
Sport	0.92	1.00	0.96	278
Tech	1.00	1.00	1.00	245
Politics	1.00	0.99	0.99	312



En analysant le tableau et l'histogramme des performances du modèle, plusieurs points intéressants émergent:

- **Business** : Avec une précision de **99%**, la catégorie *business* est bien identifiée, mais le rappel chute à **92%**, ce qui indique que le modèle a manqué certains textes pertinents. Cela pourrait être lié à une similarité contextuelle entre *business* et d'autres catégories comme *politics*.
- **Sport** : Le modèle excelle en rappel avec **100%**, ce qui signifie qu'il a capturé tous les textes classifiés comme *sport*. Toutefois, une précision légèrement plus basse (**92%**) révèle qu'il a parfois confondu des textes d'autres catégories avec celle-ci.

- **Tech** : La catégorie *tech* est remarquable avec des scores parfaits en précision, rappel, et F1-score (**100%** partout). Cela montre une distinction claire et sans ambiguïté pour cette catégorie, probablement grâce à des mots-clés très spécifiques à son domaine.
- **Politics** : Avec une précision de **100%** et un rappel de **99%**, la catégorie *politics* est également très bien gérée. Les légères erreurs restantes sont probablement dues à des recoupements thématiques avec *business*.

Analyse globale : L'exactitude globale du modèle atteint **98%**, ce qui reflète une excellente performance. Les métriques moyennes pondérées (précision, rappel, F1) confirment cette stabilité. Cependant, des ajustements ciblés sur les catégories *business* et *sport* pourraient encore améliorer l'équilibre du modèle, en réduisant les confusions.

4.3. Limites

Malgré les performances prometteuses du modèle, certaines limites méritent d'être mentionnées :

1. **Taille du Dataset**
   Le dataset utilisé reste relativement petit. Cela réduit la diversité et la richesse des données, ce qui peut limiter la capacité du modèle à généraliser efficacement sur de nouveaux textes ou des contextes moins représentés dans l'ensemble d'entraînement.
2. **Mots Répétés entre les Catégories**
   Lors de l'analyse des wordclouds, nous avons constaté un nombre important de mots communs (4 175 exactement) entre les différentes catégories. Bien que ces mots soient souvent des termes neutres ou spécifiques au contexte général du corpus, leur présence peut introduire une confusion dans la classification, notamment pour des textes ambigus.
3. **Performance sur des Textes Courts**
   Le modèle a montré des limites lorsqu'il est confronté à des textes très courts, produisant parfois des prédictions incorrectes. Cela pourrait être dû au fait que les textes courts contiennent moins de mots contextuels spécifiques à une catégorie.
   Les mots génériques ou fréquemment utilisés dans plusieurs catégories (par exemple, "team", "business") peuvent influencer de manière disproportionnée les prédictions.
   Perte de contexte :

   Les modèles comme LSTM ou Attention utilisent des dépendances temporelles pour capturer le contexte.
   Dans des textes courts, ces dépendances sont faibles, ce qui réduit leur capacité à extraire des informations significatives.
   Sous-apprentissage des classes :

   Si une classe a des textes courts dans l'ensemble d'entraînement, cela pourrait entraîner un sous-apprentissage de cette classe.

4.4. Sauvegarde du Modèle et du Tokenizer

Une fois le modèle entraîné et validé, il est crucial de le sauvegarder pour pouvoir l'utiliser ultérieurement sans avoir à le réentraîner à chaque exécution. Dans cette étape, nous avons sauvegardé à la fois le modèle et le tokenizer utilisés pour le prétraitement des données.

Le modèle a été sauvegardé au format .keras à l'aide de la fonction model.save(). Ce format est optimal pour le stockage et la réutilisation du modèle, garantissant la conservation de l'architecture, des poids et des configurations du modèle.

Le tokenizer a également été sauvegardé à l'aide de la bibliothèque pickle, ce qui permet de conserver les paramètres liés à la tokenisation (par exemple, les indices des mots dans le vocabulaire). Cette étape est essentielle pour s'assurer que le prétraitement du texte reste cohérent lors du déploiement du modèle

Cette partie est essentielle pour le déploiement du modèle sur notre interface.

5. Développement de l'Interface Utilisateur

Pour faciliter l'utilisation du modèle de classification de documents par des utilisateurs non techniques, nous avons développé une interface utilisateur intuitive et interactive. Ce développement repose sur l'utilisation de Streamlit, une bibliothèque puissante et rapide pour la création d'applications web en Python.

- **Objectifs de l'interface**

L'interface a été pensée pour :

- Rendre l'expérience utilisateur simple et accessible, même sans connaissance technique.
- Offrir un retour immédiat et visuel sur les prédictions du modèle.
- Permettre une intégration facile dans un contexte professionnel ou éducatif.

- **Fonctionnalités principales**

1. **Chargement et Préparation**
    - Le modèle de classification (*best_model.keras*) et le tokenizer nécessaire pour le traitement des séquences textuelles sont chargés au démarrage.
    - Une gestion des erreurs a été implémentée pour alerter en cas de fichiers manquants ou d'autres problèmes.
2. **Prédiction et Analyse**
    - L'utilisateur peut saisir ou coller un texte dans un champ dédié.
    - Une fois le texte soumis, l'application :
        - Prétraite l'entrée (tokenisation et padding) pour l'adapter au format attendu par le modèle.

- Fournit une prédiction de la catégorie la plus probable parmi Business, Politics, Sport, et Tech.
- Affiche la confiance du modèle sous forme de pourcentage.
- Propose un graphique interactif avec les probabilités associées à chaque catégorie, pour une visualisation détaillée.

3. **Retour Visuel**
    - Une barre de progression affiche la confiance de la prédiction principale.
    - Un histogramme interactif, construit avec Plotly, montre la distribution des probabilités sur les différentes catégories, offrant une représentation claire et esthétique.
4. **Gestion des Contraintes**
    - Pour éviter les erreurs dues à des entrées trop longues, une limite de 300 mots est imposée. En cas de dépassement, un message d'erreur clair est affiché à l'utilisateur.
5. **Design et Expérience Utilisateur**
    - L'interface est minimaliste et centrée sur les besoins de l'utilisateur, avec des icônes et des messages explicatifs pour chaque étape.
    - Les résultats sont présentés de manière organisée et professionnelle, avec des couleurs et des graphiques qui rendent l'expérience agréable.

➢ **Aperçu de l'interface**

💡 Entrez votre texte ci-dessous :

Votre texte :

An economy is an area of the production, distribution and trade, as well as consumption of goods and services. In general, it is defined as a social domain that emphasize the practices, discourses, and material expressions associated with the production, use, and management of resources.

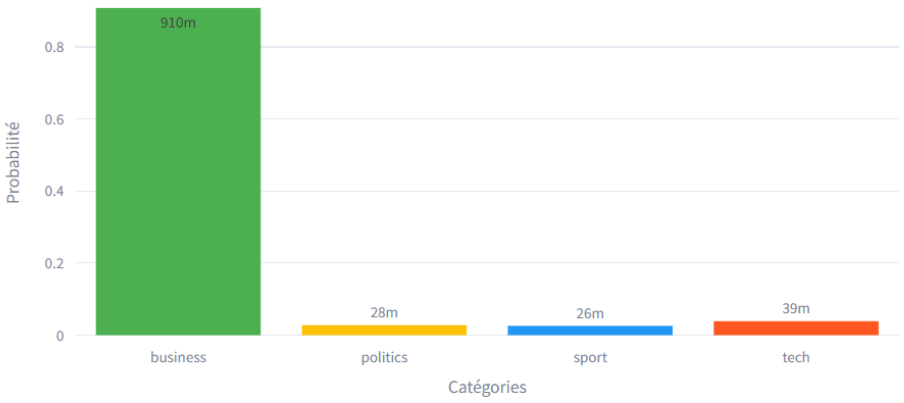
Prédire la catégorie

📋 Résultat de la Classification

Catégorie prédite : business

🔍 **Confiance : 90.76%**

Distribution des Probabilités par Catégorie



Développé par : YNAA

🌟 Projet NLP - M2 HPC

🛠️ **Technologies utilisées :** TensorFlow, Streamlit

6. Conclusion et Perspectives

6.1 Perspectives

- Collecter plus de données
- Amélioration de la gestion des mots commun entre les catégories
- Régler le problème de mal prédiction des textes courts par l'utilisation

6.2 Conclusion

En conclusion, ce travail a permis de développer un modèle de classification de texte relativement performant, malgré les défis rencontrés, notamment avec les textes courts et la gestion des mots en commun entre les catégories. Bien que le modèle affiche de bons résultats avec une précision et un rappel élevés, certaines améliorations restent possibles, notamment en ajustant l'architecture du modèle, en optimisant les hyperparamètres, et en explorant des techniques de prétraitement plus avancées. La prise en compte des textes courts, notamment, reste un point clé à travailler pour éviter les erreurs de classification. En somme, bien que des pistes d'amélioration existent, le modèle proposé constitue une base solide pour des tâches de classification textuelle, et de futures explorations permettront d'enrichir ses performances.

- **Notes sur notre travail d'équipe**
  Tout au long de ce projet, chacun d'entre nous a contribué de manière active, et on a vraiment travaillé ensemble pour surmonter les défis techniques. On se réunissait régulièrement pour discuter des progrès, résoudre les erreurs et ajuster le modèle. Dès qu'on rencontrait un problème, on s'assurait de le régler en équipe, que ce soit pour des soucis liés aux données ou pour des ajustements du modèle.

  Quand on a dû affiner le modèle, on a tous apporté nos idées et nos compétences. Certaines personnes se sont concentrées sur les hyperparamètres, d'autres ont retravaillé le prétraitement des données pour s'assurer que tout était en ordre. Chaque fois qu'on faisait un changement, on se réunissait pour échanger nos observations, tester différentes solutions et prendre les meilleures décisions ensemble.

  Concernant le rapport, chacun a rédigé une partie, mais on a aussi pris le temps de lire et de commenter les sections des autres. C'était important pour nous que tout soit cohérent et bien structuré, surtout pour les aspects techniques qu'on a dû bien expliquer. On a aussi appris beaucoup en discutant de nos approches respectives.

  En résumé, le travail d'équipe a été au cœur de ce projet. On a vraiment mis l'accent sur la collaboration et l'entraide, ce qui nous a permis de faire face aux défis et de créer un modèle solide.