NLP
Projet: Groupe 8

Mai 2024

Auteur:
Abel ANDRY
Lyes BENACER
Leo BILLANT
Ilyas OULKADDA
Nael EL JANATI
Ilyas OULKADDA

Encadrant:
Gustave CORTAL

1 Présentation dataset

Le dataset "IMDB Spoiler Dataset" est une collection de donneés qui provient du site de recensement de film, série et émission de télévision IMDB. Il contient des informations détaillées sur les films et le émission de télévision, ainsi que les critiques de ces derniers par les utilisateurs. L'une des caractéristiques intéressantes de ce dataset est qu'il indique si une critique donnée contient ou non un spoiler.

Le fichier "IMDB_movie_details.json" contient des métadonneés sur les films et les émissions de télévision pour lesquels de critiques sont présentes dans le fichier "IMDB_reviews.json". Chaque enregistrement dans ce fichiers contient les informations suivantes :

- review_date : Date à laquelle la critique a été rédigée.
- movie_id : Identifiant unique de l'élément.
- user_id : Identifiant unique de l'auteur de la critique.
- is_spoiler : Indication si la critique contient un spoiler.
- review_text : Critique textuelle sur l'élément.
- rating : Note attribuée par l'utilisateur à l'élément.
- review_summary : Résumé court de la critique.

Le dataset "IMDB Spoiler Dataset" peut être utilisé pour différentes tâches d'apprentissage automatique, telles que l'identification de critiques contenant des spoilers, la découverte de signaux qui rendent une critique plus susceptible de contenir des spoilers, ou encore la prédiction de la note attribuée à un média en fonction de ses métadonnées.

En résumé, le dataset "IMDB Spoiler Dataset" est une collection de données provenant du site IMDB, contenant des métadonnées sur les films et les émissions de télévision, ainsi que des critiques d'utilisateurs avec des informations sur la présence ou non de spoilers. Avec un grand nombre de données et une variété d'informations, ce dataset est un choix intéressant pour les personnes qui s'intéressent à la consommation de médias et à l'impact des spoilers sur l'expérience de visionnage.

2 Preprocessing

Le prétraitement du jeu de donnée est une étape essentielle pour faciliter le travail des modèles prédictifs. On y inclut principalement deux fonctions : la tokénisation et la normalisation du texte. Nous allons passer en revue les différentes méthodes utilisées.

2.1 Tokénisation

La tokénisation est presque obligatoire lors de tâches attribuées au NLP. Son objectif est de prendre un texte et de le séparer en mots ou "tokens". Elle y inclut plusieurs défis puisque les textes fournis ne sont pas simplement des mots juxtaposés. On y notera la présence de ponctuation, de contractions de mots (ex: "can't") et dans certains cas des tokens propres à un contexte précis tels que les dates, les adresses mail, les URL, etc...

Il existe plusieurs méthodes de tokénisation, nous en verrons deux : la tokénisation à l'aide d'expressions régulières et la tokénisation byte-pair encoding.

La tokénisation à l'aide d'expressions régulière est régie par des règles prédéfinies à la main pour "matcher" des patterns reconnaissables. On peut l'utiliser dans des contextes spéciaux tels que la reconnaissance de patterns d'adresse mails. La librairie utilisée pour cette méthode est `NLTK`. Elle contient un module avec une classe permettant de lui passer une expression régulière pour ensuite tokeniser le texte.

L'expression régulière choisie pour ce dataset est la suivante :

```
\b\w+(?:[-']\w+)*|\b\w+\b|[!]+
```

Cette expression régulière fonctionne de la manière suivante :

- `\b\w+(?:[-']\w+)*` : correspond à un mot pouvant contenir des tirets ou des apostrophes internes (par exemple, "can't" ou "check-in").
- `\b\w+\b` : correspond à un mot délimité par des bornes de mots (\b).
- `[!]+` : correspond à un ou plusieurs points d'exclamation.

La tokénisation byte-pair encoding (BPE) est une méthode permettant d'apprendre un ensemble de tokens qui ne sont pas définis comme un mot ou un caractère. Certains tokens sont plus petits que la taille d'un mot, ils seront alors appelés "sous-mots". L'une des problématiques à laquelle répond cette méthode est le traitement de mots encore inconnus au tokeniseur. En effet, chaque mot peut être défini comme un ensemble de sous-mots préalablement connus. Ainsi, cette méthode permet d'élargir infiniment un vocabulaire fixé.

Le principe de la tokénisation BPE se déroule comme suit :

- On définit un vocabulaire initial composé de tous les caractères uniques.
- On parcourt le corpus d'entraînement et on fait le compte de tous les symboles faisant partie du vocabulaire étant adjacents.
- La paire étant la plus fréquemment adjacente est fusionnée puis ajoutée au vocabulaire.
- Cette paire est alors considérée comme un "caractère" dans le vocabulaire.

- On répète le procédé jusqu'à ce que k fusions aient été effectuées.

Exemple : si la paire 'A', 'B' est la plus représentée dans tout le corpus, on ajoute 'AB' au vocabulaire. À chaque fois que l'on croise 'A' et 'B' étant adjacents dans le texte, on les considérera comme 'AB'.

Pour implémenter la tokénisation BPE, on peut utiliser la librairie `tiktoken`, qui offre des outils performants pour ce type de traitement. `tiktoken` permet de créer un modèle de tokénisation basé sur BPE et de l'appliquer à des textes pour obtenir une représentation tokenisée.

Les statistiques suivantes montrent le nombre de tokens uniques générés par deux méthodes de tokénisation différentes :

- **Tokénisation BPE** : 16,099 tokens uniques
- **Tokénisation par expressions régulières** : 16,561 tokens uniques

Le nombre de tokens uniques est légèrement inférieur pour la méthode BPE comparé à la méthode par expressions régulières. Cela peut s'expliquer par le fait que BPE fusionne fréquemment des paires de caractères ou de sous-mots pour créer des tokens plus longs et plus complexes, réduisant ainsi le nombre total de tokens uniques nécessaires pour représenter le même corpus. En revanche, la tokénisation par expressions régulières ne fusionne pas les caractères ou sous-mots, ce qui entraîne un nombre plus élevé de tokens uniques car chaque occurrence individuelle est traitée comme un token distinct.

2.2 Normalisation

La normalisation de texte en NLP est une étape clé. Elle permet d'uniformiser un texte brut afin d'augmenter la similarité entre les mots. Nous utiliserons trois méthodes de normalisation: la suppression des stop-words, la lemmatisation des mots et rendre tout le texte en minuscule.

Les stop words sont des mots très fréquents dans une langue qui ont peu de valeur sémantique, tels que "et", "ou", "mais", "le" en français. Ces mots fournissent que très peu d'information sur le corpus analysé et peuvent donc être supprimés favorisant la performance du modèle entraîné ultérieurement.

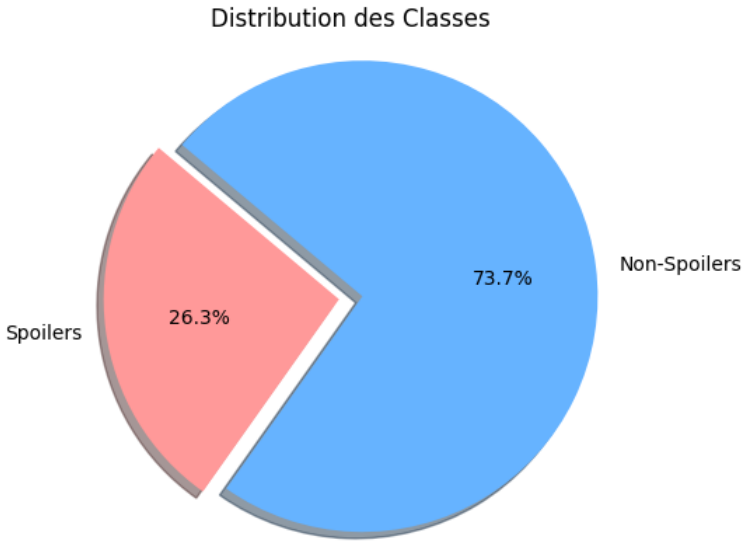
Avant la suppression des stop words, la longueur du corpus est de 872,920,167 mots. Après traitement, on obtient 542,372,378 mots. Cette réduction montre que supprimer les stop words réduit significativement le volume de données sans perdre d'informations essentielles.

La lemmatisation consiste à réduire un mot en sa forme de base appelée lemme. Elle prend en compte le sens du mot et son contexte afin d'effectuer cette réduction. Par exemple, les formes fléchies "petite", "petits", "petites" renvoient toutes au lemme "petit". Cela rempli bien l'objectif d'uniformiser le corpus.

Rendre tout le texte en minuscule est une autre méthode de normalisation courante. Cela permet de traiter les mots de manière uniforme sans tenir compte de la casse. Cette uniformisation est essentielle pour éviter les doublons et les incohérences, ce qui améliore la précision des modèles NLP en assurant une similarité maximale entre les mots.

3 Statistiques descriptives du dataset

Le dataset comprend un total de 573,913 critiques, recueillies à partir de divers films sur IMDB. Les critiques couvrent plusieurs anneés, offrant ainsi une vue d'ensemble des opinions des utilisateurs au fil du temps.



3.0.1 Statistiques descriptives

Longueur des critiques :

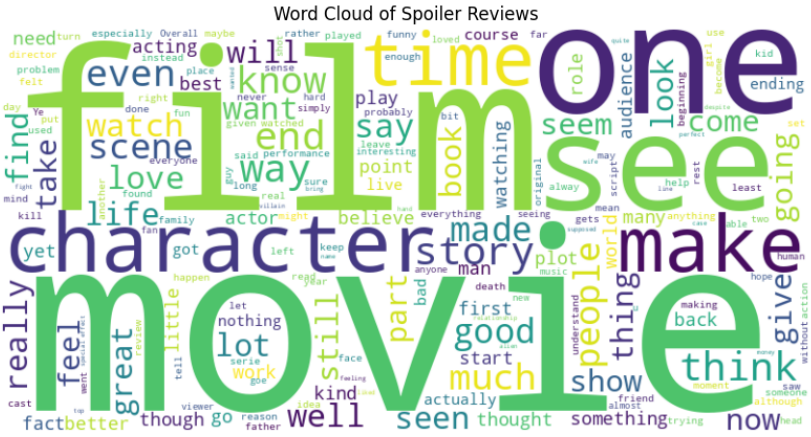
- Longueur moyenne des critiques : 296 mots
- Médiane de la longueur des critiques : 250 mots
- Longueur la plus fréquente : 200 mots

La distribution de la longueur des critiques montre une asymétrie à droite, avec la plupart des critiques étant concises mais aussi quelques critiques très longues qui étendent la moyenne.

- Proportion de spoilers : 150,924 critiques (seulement 26,3%) sont marquées comme spoilers
- Note moyenne (spoilers) : 6,8
- Note moyenne (non-spoilers) : 7,4
- Longueur moyenne des critiques (spoilers) : 350 mots
- Longueur moyenne des critiques (non-spoilers) : 280 mots

Les critiques contenant des spoilers ont tendance à avoir des notes légèrement plus basses et sont généralement plus longues, suggérant que les critiques détaillées sont plus susceptibles de contenir des spoilers.

3.0.2 Analyse textuelle



Excluant les mots vides, les termes les plus fréquents dans les critiques sont 'film', 'movie', 'story', 'characters', et 'scene'. Cela indique une focalisation sur les éléments fondamentaux de l'expérience cinématographique.

4 Entraînement des modèles

4.1 N-gram

Un des premiers modèles demandés était le N-gram, qui sous-entendait la génération de texte. Dans ce rapport, il a été décidé de créer deux modèles génératifs : l'un générant des critiques à caractère positif, l'autre à caractère négatif, qui sera mentionné un peu plus tard.

Le modèle génératif est assez simple : l'idée est de créer des 4-grams en se basant sur toutes les critiques d'IMDb ayant attribué une note de 10/10 aux films.

En termes de métrique, il est intéressant de s'intéresser à la perplexité, qui est de :

- Perplexité d'entraînement : 1.5178801571978164
- Perplexité de test : 7859.655869811526

Maintenant, pour tester le modèle et avoir un aperçu du type de critique positive qu'il peut proposer, nous pouvons partir d'un début de critique classique, du genre : *"It is a story of one man."* Le modèle commencera alors par générer quelque chose de ce style : *"It is a story of one man's journey to discover his identity, value, and purpose in life. It is not defined by the things and positions he seeks and pursues in society, but by how he is motivated to serve and help others, no matter how much people view her as a villain..."*

Il s'agit d'une simple description de l'histoire, donc il peut être difficile de percevoir le côté positif. Cependant, la fin de la critique démontre bien l'idée d'un modèle basé sur des critiques notant 10/10 : *"...I have never seen a movie like this before. It is a great movie, and it exceeded my expectations."*

4.2 Bayésien naïf

Le but de nos modèles de classification est de prendre une critique de film et de vérifier si ce texte contient des spoilers.

Le modèle de Bayesien naïf va d'abord vectoriser les textes sous forme de 4-grammes avant de réaliser l'implémentation classique d'un Bayesien naïf.

En termes de performance, c'est malheureusement un des modèles les moins performants. En effet, bien qu'il parvienne à reconnaître assez correctement les critiques qui ne contiennent pas de spoilers, il est plus intéressant de deviner qu'un texte contenant des spoilers est effectivement un spoil plutôt que de prédire qu'un texte n'en contenant pas est un non-spoil.

Une des raisons de cette faible performance est principalement due à la disproportion des critiques non-spoil par rapport aux critiques spoil dans le dataset.

Classe	**Précision**	**Rappel**
Non-spoil	0.77	0.96
Spoil	0.65	0.20

4.3 Régression logistique

Nous avons testé l'efficacité de différents modèles de régression logistique pour la classification des données. Les modèles produits utilisent trois approches de représentation des caractéristiques

textuelles : N-gram, TF-IDF et Word2Vec. De plus, nous avons aussi utilisé la méthode Stratified K-Fold pour traiter les déséquilibres des datasets.

4.3.1 Régression Logistique avec N-gram

Dans cette étude, nous avons exploré diverses approches de régression logistique pour la classification des spoilers dans les critiques de films. Tout d'abord, nous avons utilisé une méthode de bag-of-words (BoW) avec des N-grams pour représenter les critiques sous forme de vecteurs de comptage de mots.
Le modèle a été entraîné avec différentes configurations, notamment l'utilisation de solver lbfgs, liblinear et saga, avec des paramètres ajustés pour garantir la convergence.

4.3.2 Régression Logistique avec TF-IDF

Une autre approche que nous avons explorée était d'utiliser TF-IDF (Term Frequency-Inverse Document Frequency) pour pondérer les mots en fonction de leur importance dans le corpus. Cela a permis de mieux représenter la sémantique des critiques, en mettant l'accent sur les mots les plus informatifs tout en réduisant l'importance des mots courants.Les performances ont été évaluées de manière similaire à l'approche N-gram

4.3.3 Régression Logistique avec Word2Vec

Nous avons exploré l'utilisation de Word2Vec pour représenter les critiques sous forme de vecteurs de mots. Initialement, nous avons utilisé des modèles pré-entraînés tels que GloVe.
Cependant, nous avons remarqué un problème : de nombreux mots n'étaient pas présents dans ces modèles, ce qui limitait la qualité des représentations. Pour résoudre ce problème, nous avons entraîné notre propre modèle Word2Vec sur l'ensemble des critiques du dataset. Bien que cette démarche ait amélioré la situation en fournissant des embeddings plus pertinents, les performances de la régression logistique avec Word2Vec étaient encore inférieures à celles des approches précédentes. Néanmoins, d'autres explorations avec des modèles Word2Vec ou un affinement de notre propre modèle pourraient potentiellement améliorer les résultats.

4.3.4 Régression Logistique avec Stratified K-Fold

Enfin, nous avons utilisé la validation croisée stratifiée (Stratified K-Fold) pour évaluer la robustesse de nos modèles. Cela nous a permis d'estimer les performances du modèle sur différents plis de données, en garantissant une répartition équilibrée des classes entre les ensembles d'entraînement et de test.

4.3.5 Comparaison des résultats

Les résultats montrent que l'utilisation de la méthode TF-IDF offre le meilleur compromis entre précision (moyenne de 0.67) et rappel (moyenne à 0.70).
Bien que les approches de régression logistique avec TF-IDF et Word2Vec aient montré des performances acceptables en termes de précision, le rappel pour la classe des spoilers reste relativement faible. Dans notre contexte, où la détection des spoilers est prioritaire, il est essentiel de rechercher des améliorations pour augmenter ce rappel. Cela pourrait passer par l'exploration de modèles Word2Vec alternatifs ou par un ajustement plus fin des paramètres des modèles existants.

Modèle	Classe	Précision	Rappel
N-gram	0 (non-spoil)	0.83	0.78
	1 (spoil)	0.48	0.56
TF-IDF	0 (non-spoil)	0.86	0.74
	1 (spoil)	0.48	0.67
Word2Vec	0 (non-spoil)	0.82	0.63
	1 (spoil)	0.37	0.62
K-Fold	Moyenne Générale	0.74	0.41

Table 1: Performances des modèles de régression logistique

4.4 Réseaux de neurones

Dans le cadre de ce projet de traitement automatique du langage naturel (NLP), nous avons exploré et comparé les performances de trois modèles distincts, chacun implémenté et entraîné pour des tâches spécifiques. Les modèles étudiés incluent des réseaux de neurones développés from scratch ainsi que des modèles fine-tunés, exploitant les bibliothèques PyTorch, KerasNLP et Transformers. Deux des modèles développés sont dédiés à des tâches de classification. Le troisième modèle est orienté vers la génération de texte.

4.4.1 Réseau Feedforward

Dans un premier temps, nous avons utilisé un modèle de réseau de neurones feedforward simple. Ce modèle est constitué de plusieurs couches fully connected, également appelées couches denses.

L'architecture du modèle est la suivante :

1. **Première couche dense** : Cette couche reçoit en entrée un vecteur de taille égale au vocabulaire et le transforme en une représentation dense de dimension spécifiée.
2. **Couche de dropout** : Après la première couche dense, nous appliquons un dropout avec un taux de 50% pour prévenir le surapprentissage (overfitting).
3. **Deuxième couche dense** : Cette couche prend en entrée la sortie de la première couche dense (après le dropout) et la transforme en une autre représentation dense de la même dimension cachée.
4. **Troisième couche dense** : Enfin, cette couche prend en entrée la sortie de la deuxième couche dense et réduit la dimensionnalité à un nombre de classes, correspondant au nombre de catégories dans notre tâche de classification.

Pour l'embedding des mots, nous avons utilisé le modèle Word2Vec, déjà employé antérieurement dans nos modèles de régression logistique.

Le modèle a été entraîné sur un split de 80/20 des données en utilisant une batch size de 64. Nous avons expérimenté avec plusieurs optimizers et taux d'apprentissage pour trouver la meilleure configuration pour notre tâche. Les trois principaux optimizers testés sont : SGD, Adam, et AdamW.

Le modèle feedforward a montré des performances variables selon l'optimizer utilisé. Les résultats montrent que le rappel est généralement meilleur pour la classe spoil avec l'optimizer SGD (0.73) et légèrement moins bon avec Adam (0.70) et AdamW (0.67).

Optimizer	Learning Rate	Classe	Précision	Rappel
SGD	1e-2	0 (non-spoil)	0.86	0.62
		1 (spoil)	0.40	0.73
Adam	5e-4	0 (non-spoil)	0.86	0.64
		1 (spoil)	0.41	0.70
AdamW	1e-4	0 (non-spoil)	0.85	0.67
		1 (spoil)	0.42	0.67

Table 2: Résultats des différents optimizers sur le modèle feedforward

Dans le contexte de la détection des spoilers, le rappel pour la classe spoil est particulièrement crucial. Un rappel élevé pour les spoilers signifie que le modèle est capable de détecter la majorité des spoilers présents dans le jeu de données, ce qui est essentiel pour notre tâche. En d'autres termes, il est préférable de maximiser la détection des spoilers même si cela implique d'avoir quelques faux positifs (c'est-à-dire des non-spoilers classés comme spoilers).

En conclusion, bien que le modèle feedforward soit simple, il est essentiel de choisir le bon optimizer et les bons hyperparamètres pour équilibrer la précision et le rappel selon les besoins spécifiques de la tâche. Dans notre cas, où le rappel des spoilers est prioritaire, l'optimizer SGD semble être le plus efficace, malgré une précision plus faible. Cependant, pour un compromis entre précision et rappel, Adam et AdamW offrent des performances équilibrées.

4.4.2 Fine-tune BERT

Pour la tâche de classification, nous avons utilisé le modèle DistilBERT, une version allégée de BERT (Bidirectional Encoder Representations from Transformers) nommée distilbert-base-uncased. Ce modèle est conçu pour offrir des performances comparables à celles de BERT tout en étant plus léger et plus rapide. DistilBERT conserve environ 97% des performances de BERT tout en réduisant le nombre de paramètres et la taille du modèle de 40

Le modèle a été fine-tuné sur notre jeu de données pour un total de 2 epochs, ce qui a pris environ 8 heures sur un GPU NVIDIA P100. Pour adapter DistilBERT à notre tâche spécifique, nous avons ajouté 4 couches fully connected au backbone de BERT, augmentant ainsi le nombre total de paramètres de 1,5 million aux 66 millions initiaux de DistilBERT.

Nous avons utilisé l'optimizer Adam avec un taux d'apprentissage de 1e-5 pour entraîner le modèle. De plus, nous avons expérimenté l'entraînement avec et sans pondération des classes pour compenser le déséquilibre des classes dans notre jeu de données.

Les performances du modèle sans pondération des classes étaient significativement inférieures par rapport à celles obtenues avec pondération des classes. Dans le contexte de la prédiction de spoilers, les métriques de rappel (recall) et de précision (precision) sont particulièrement cruciales. Il est moins grave de prédire un spoil comme n'étant pas un spoil que l'inverse.

Sans pondération des classes, le modèle montre une bonne précision pour les non-spoils (0.82) et un excellent rappel (0.95), mais il lutte avec la classe spoil, ayant un rappel de seulement 0.41. Cela signifie que beaucoup de spoils sont incorrectement classés comme non-spoils, ce qui n'est pas souhaitable dans notre application.

Après avoir appliqué la pondération des classes, nous observons une amélioration notable dans les métriques pour les spoils. La précision de la classe spoil reste modérée à 0.52, mais le rappel passe à 0.70. Cela signifie que le modèle est maintenant bien meilleur pour identifier les spoils,

Pondération	Classe	Précision	Rappel
Sans pondération	0 (non-spoil)	0.82	0.95
	1 (spoil)	0.74	0.41
Avec pondération	0 (non-spoil)	0.88	0.77
	1 (spoil)	0.52	0.70

Table 3: Résultats du modèle BERT avec et sans pondération des classes

réduisant ainsi les faux négatifs, ce qui est crucial pour notre cas d'utilisation.

En conclusion, la fine-tuning de DistilBERT avec une architecture étendue et l'application de la pondération des classes ont significativement amélioré les performances du modèle, en particulier en ce qui concerne les métriques de rappel et de précision pour la classe spoil. Ces ajustements rendent le modèle bien plus adapté pour la détection précise des spoilers.

4.4.3 Fine-tune GPT-2

Pour la tâche de génération de texte, nous avons utilisé le modèle openai-community/gpt2, une version optimisée du modèle GPT-2 développé par OpenAI. GPT-2 (Generative Pre-trained Transformer 2) est un modèle de langage autoregressif qui utilise l'architecture Transformer pour générer du texte de manière cohérente et fluide. Il est pré-entraîné sur un large corpus de texte et peut être fine-tuné pour des tâches spécifiques.

Le modèle AigrisGPT a été fine-tuné pendant seulement une epoch, ce qui a pris environ une heure. L'entraînement a été réalisé sur un corpus de critiques de films ayant une note maximale de 2/10. L'objectif était de créer un modèle capable de générer des critiques de films négatives. Le jeu de données comprenait un total de 20 millions de tokens.

Pendant l'entraînement, le modèle a atteint une perplexité de 44.54 sur l'ensemble d'entraînement. La perplexité est une mesure de la capacité du modèle à prédire un échantillon de texte, avec des valeurs plus basses indiquant de meilleures performances.

Voici un exemple de génération en utilisant notre modèle AigrisGPT.

- *Top_p : 0.95, max_length : 100, input : 'This movie'*
- *Output : 'This movie tries too hard to be a thriller film and to say there are lots of people like me who like this kind of movies it falls apart at some points. But the thing is this: these people would probably be bored with the genre anyway. All the characters are a mix of stereotypical, racist, violent and sexist stereotypes which are supposed to fit into a mmon genre. One that I found myself thinking about after I watched it. I should have read the books first. If not, I'*

En conclusion, le modèle AigrisGPT, fine-tuné à partir de GPT-2, est spécifiquement conçu pour générer des critiques de films négatives. Son entraînement sur des critiques très négatives et sa perplexité relativement basse indiquent qu'il est bien adapté à la tâche pour laquelle il a été créé.

5 Limites et difficultés rencontrées

Dans le cadre de notre projet de détection de spoilers, nous avons rencontré plusieurs limites et difficultés impactant les performances de nos modèles.

Tout d'abord, de nombreuses critiques contiennent des informations superflues sans lien direct avec l'intrigue, ce qui complique l'identification des spoilers. Par exemple, une critique typique pourrait inclure : "J'ai adoré ce film, les acteurs étaient formidables. C'est très surprenant que Lyes meurt à la fin. Et les acteurs étaient meilleurs dans ce film que dans les autres films. Regardez-le absolument !!". Ici, la phrase "C'est très surprenant que Lyes meure à la fin" est un spoiler, mais elle est noyée parmi des commentaires non pertinents. Pour surmonter ce problème, nous avons prétraité les critiques en éliminant les phrases jugées inutiles, réduisant le bruit et concentrant les modèles sur les informations essentielles. Toutefois, cette méthode peut supprimer des contenus importants ou conserver des informations non pertinentes, rendant le prétraitement sensible et délicat.

Les variations stylistiques dans les critiques posent également problème. Les critiques peuvent varier en longueur, vocabulaire et structure grammaticale, compliquant la normalisation et la tokénisation du texte. L'ambiguïté contextuelle est un autre défi majeur. Un mot ou une phrase peut ne pas être un spoiler dans un contexte, mais peut l'être dans un autre. Par exemple, la phrase "Il découvre finalement la vérité" est banale, sans contexte, mais peut devenir un spoiler majeur dans un contexte spécifique.

Les modèles de machine learning basés sur des n-grams ou des représentations TF-IDF montrent des limitations en capturant la sémantique contextuelle. Pour améliorer cela, nous avons exploré des modèles avancés comme Word2Vec et les Transformers (BERT, GPT-2), nécessitant des ressources computationnelles significatives et des temps d'entraînement longs.

La gestion de l'équilibre des classes a été problématique. Le dataset contient un nombre disproportionné de critiques sans spoilers par rapport aux critiques avec spoilers, entraînant des biais dans les modèles. Pour atténuer cet effet, nous avons utilisé des techniques de pondération des classes et de sur-échantillonnage, sans complètement résoudre le problème.

L'évaluation des performances des modèles pose également des soucis. Les métriques classiques comme la précision et le rappel sont utiles, mais dans le contexte de la détection de spoilers, le rappel pour la classe des spoilers est crucial. Nous devons nous assurer que les spoilers sont détectés avec un taux élevé, même si cela implique d'augmenter le taux de faux positifs.

En conclusion, bien que notre approche ait permis des progrès dans la détection des spoilers, plusieurs problèmes subsistent. La complexité des critiques de films, l'ambiguïté contextuelle, les variations stylistiques, les limitations des modèles de machine learning et les déséquilibres des classes influencent les performances de nos modèles. Affiner les techniques de prétraitement, explorer des modèles plus sophistiqués et ajuster les stratégies d'évaluation seront essentiels pour améliorer nos résultats.