Deep Learning for NLP

Assignment 1 : BoW and CNN models

CAUDARD Joris

Pour le 21/11/2024

Résumé

Ce rapport compare les performances de différents modèles de classification de texte sur un corpus de critiques de films (IMdB). L'analyse montre que le modèle de classifieur CNN surpasse les modèles simples BagOfWord avec une précision de 80%, contre 70% pour le modèle simple. Les métriques de comparaisons utilisées incluent la précision ainsi que le F1-score.

Dépôt Github lié

Dépôt GitHub du projet : https ://github.com/JorisCaudard/M2_DL4NLP

Table des matières

1 Introduction

Dans cette étude, on cherche à construire un modèle de Deep Learning permettant la classification de critiques de films en deux classes, positive ou négative. Pour ce problème de Sentiment Analysis simple, on portera notre attention sur les performances de deux modèles (décrits en partie 2), à savoir un modèle BagOfWords ainsi qu'un modèle CNN.

On se basera pour cela sur un jeu de données publiques issus de l'Internet Movie DataBase (ImdB) contenant 300000 critiques à caractères positives et 300000 critiques négatives. Ces critiques sont de courts commentaires en anglais, de quelques mots extrait de critiques d'utilisateurs de la plateforme internet IMdB.

On cherche alors à développer un modèle de classification de critiques, permettant de classifier chaque critiques en deux classes : positives et négatives. On a donc un problème de classification binaire, prenant en entrée une phrase en anglais, et produisant en sortie la probabilité que la critique soit positive.

2 Modèles utilisés

On évaluera dans cette étude les performances de trois modèles de classifications de textes. Le **Dummy Classifier** a été utilisé comme base de référence ; On attribue à chaque phrase en entrée du modèle la classe positive ou négative de manière uniforme. On se servira de ce modèle comme base de comparaison de performances pour comparer les autres modèles.

Ensuite, on évaluera les performances d'un modèle de classification basé sur une représentation de type **Bag of Words (BOW)**, en combinaison avec un réseau MLP. Cette approche classique repose sur une vectorisation des textes, où chaque mot est considéré comme une dimension indépendante. Bien que simple, cette méthode peut s'avérer performants dans notre cas, si les textes du corpus présentent des mots-clés discriminants (classic, beautiful, best ...or despicable, bad ...).

Enfi, on implémentera un réseau de neurones convolutifs **CNN**, basé sur sa description par Yoon Kim en 2014 das l'article *Convolutional Neural Networks for Sentence Classification*. Ces modèles permettent de capturer des motifs locaux au sein des phrases, en étudiant les relations entre bigrammes, trigrammes ou plus. Les couches convolutives permettent d'extraire ces caractéristiques importantes, tandis que le réseau MLP suivant effectue la partie classification finale. Cette approche est donc particulièrement adaptée dans notre cas de phrases courtes.

On appliquera aux phrases en entrée les transformations nécessaires à leur interprétation par les différents réseaux : Une première étape de tokenisation/cleaning simple (en séparant les phrases en mots distincts, ainsi qu'en extrayant certaines terminaisons verbales), une seconde étape de vectorisation des phrases tokenisées (en utilisant le vocabulaire extrait des phrases d'entraînement), puis une étape d'embeddings. On ajou-

tera également au vocabulaire préalablement construit des tokens *<PAD>*, *<BOS>* et *<EOS>* afin de mettre en place un padding uniforme des phrases des différents datasets d'entraînement/validation/test utilisés.

3 Méthodologie

Afin de simplifier les entraînements des différents modèles, on se basera sur un extrait de 20000 critiques issues du dataset complet, répartis équitablement en 10000 critiques positives et 10000 critiques négatives. On choisira un partitionnement des données en utilisant la méthode *train_test_split*. Le corpus extrait de la base de données complète a été divisé en trois sous-ensemble : 70% des données ont été utilisées pour l'entraînement, 15% pour la validation et 15% pour le test final. On prendra soin lors du partitionnement des jeux de données de conserver un ratio de 50/50 de critiques positives et négatives dans chacun des jeux de données

En plus d'utiliser des jeux de validation et test indépendants du jeu d'entraînement, on évitera le sur-apprentissage en utilisant un mécanisme d'early stopping lors de l'entraînement des modèles.

On comparera les différents modèles en utilisant deux métriques principales : la **Binary Cross-Entropy**, qui mesure la qualité des prédictions en termes de probabilité pour des problèmes binaires, et la **précision**, indiquant la proportion de prédictions correctes. On entraînera chaque modèle avec des configurations standards, même si une optimisation des différents hyper-paramètres des modèles pourrait être effectué, par GridSearch ou différentes méthodes d'optimisation bayésienne (Optuna).

4 Résultats et analyse

On comparera les performances des trois modèles en termes de Binary Cross-Entropy Loss sur le jeu de donnée d'entraînement, ainsi qu'en terme de précisions sur les deux jeux de données de validation et de test. Comme décrit en section 3, on utilisera la mesure de la fonction de perte sur le jeu de validation afin d'implémenter un mécanisme d'early stopping, ayant pour but de minimiser l'overfitting des modèles.

En tant que référence, le modèle **Dummy Classifier** obtient une précision attendue de 50%, et ce quelque soit le jeu de données sur lequel il est évalué. Ce score est évidemment dû au partage des jeux d'entraînement, et au ratio de 50/50 de critiques de chaque classe conservé dans chacun des jeux de données.

Le modèle **BoW** couplé à un réseau de type MLP a significativement amélioré les résultats, affichant une perte de 0.38 sur le jeu de données d'entraînement et une précision finales de 68% sur le jeu de données de test. Cette amélioration s'explique par sa capacité à capturer des motifs simples, notamment la présence de certains mots-clés discriminants dans les phrases du corpus.

Le modèle **CNN** a surpassé les deux approches précédentes, en affichant une perte de 0.30 sur le jeu de données d'entraînement et une précision de 79% sur le jeu de données test. Grâce à l'utilisation de filtres convolutifs de différentes tailles, le modèle arrive a identifier des relations sémantiques plus complexes que le modèle BoW dans les données textuelles. On observe également un temps d'entraînement du modèle CNN plus important que pour le modèle BoW en raison du nombre plus importants de paramètres.

On peut représenter les résultats obtenus dans la tableau suivant :

Modèle	BCE Loss (Train)	BCE Loss (val)	Accuracy (val)	Accuracy (test)
Dummy Classifier	NaN	NaN	0.5	0.5
BoW	0.45	0.67	0.68	0.68
CNN	**0.35**	**0.42**	**0.79**	**0.78**

TABLE 1 – Résultats obtenus par les différents modèles

En étudiant de manière plus précise les performances des modèles selon les différentes classes, on confirme les résultats obtenus par le calcul plus global de la métrique de précision : le modèle CNN obtient des meilleurs résultats, quelque soit la classe réelles des phrases du corpus de test. En revanche, on peut noter que le modèle BoW obtient un score de 92% dans la précision sur des classes d'origine négatives : On peut en conclure que les mots discriminants négativement sont très identifiables par ce modèle, bien plus que des mots discriminants positivement.

5 Conclusion

Les résultats montrent donc bien une progression claire des performances avec l'augmentation de la complexité des modèles : L'analyse indépendante des mots ne suffit pas dans ce problèmes, l'analyse des bigrammes/trigrammes permettent une meilleure identifiabilité de la nature de chaque phrase. Cependant, ce gain en performance s'accompagne naturellement d'une augmentation des temps de calcul lors de l'entraînement des modèles.

Pour aller plus loin, des améliorations possibles pourraient inclure l'optimisation des hyper-paramètres des modèles, mais également l'exploration d' architectures plus avancées (LSTM, Transformers …) L'utilisation d'ensemble de données plus vaste pourraient également permettre de renforcer la généralisation des différentes prédictions.

A Annexes

A.1 Tables et figures

Liste des tableaux

A.2 Hyper-paramètres

A.2.1 Modèle Bow

```
CBOW_classifier(
  (embedding): Embedding(9885, 128)
  (MLP): Sequential(
    (0): Linear(in_features=128, out_features=64, bias=True)
    (1): ReLU()
    (2): Linear(in_features=64, out_features=1, bias=True)
  )
)
```

Listing 1 – Structure du modèle BoW

Hyper-paramètres

- `embedding_dim` : 128
- `hidden_dim` : 64
- Nombre de couches cachées : 2

A.2.2 Modèle CNN

```
CNN_classifier(
  (embedding): Embedding(9885, 128)
  (convs): ModuleList(
    (0): Conv1d(128, 10, kernel_size=(2,), stride=(1,))
    (1): Conv1d(128, 10, kernel_size=(3,), stride=(1,))
    (2): Conv1d(128, 10, kernel_size=(4,), stride=(1,))
  )
  (dropout): Dropout(p=0.5, inplace=False)
  (MLP): Sequential(
    (0): Linear(in_features=30, out_features=15, bias=True)
    (1): ReLU()
    (2): Linear(in_features=15, out_features=1, bias=True)
  )
```

```
15 )
```

Listing 2 – Structure du modèle BoW

Hyper-paramètres

— `embedding_dim` : 128
— `num_filter` : 10
— `filter_sizes` : [2,3,4]
— `dropout` : 0.5
— `hidden_dim` : 15
— Nombre de couches cachées : 2

A.2.3 Paramètres d'entraînement

— `Nombre maximal d'epochs` : 10
— `Patience` : 2
— `Optimizer` : Adam