Rapport du projet 1
IFT6285, A24

Shuxu LI
shuxu.li@umontreal.ca

Abstract

Dans ce projet, nous avons développé deux benchmarks de classification de textes en utilisant les données de publications issues de l'API [1] de l'ACL [2](Association for Computational Linguistics). Nous avons comparé les performances de divers classificateurs sous plusieurs approches d'extraction de "features" de librairie scikit-learn [3], notamment le TFIDF, le CountVectorizer, le Word2Vec et BERT. En combinant ces méthodes de vectorisation avec différents modèles de classification LogisticRegression, SVC, MultinonialNB etc., nous avons sélectionné deux modèles offrant les meilleures performances sur les deux benchmarks.

1 Présentation de l'Anthologie ACL et son API

L'anthologie ACL est une collection de contributions scientifiques dans le domaine de la linguistique computationnelle et du traitement automatique de langues naturelles, maintenue par l'Association for Computational Linguistics (ACL). L'anthologie comprend jusqu'à présent 100820 articles rédigés par plus de 80000 auteurs, principalement en anglais et en français. Ces articles sont répartis dans plus de 3000 volumes de 431 conférences différentes. Le tableau 1 ci-dessous résume des informations globales de cette anthologie.

Catégories	Statistiques
Nombre des articles	100820
Nombre des volumes	3036
Nombre des collections	1338
Nombre des venues	431
Nombre des auteurs	88936
Languages	fra, eng, etc.

TABLE 1 – Schéma sommaire de l'Anthologie ACL

1. https://acl-anthology-py.readthedocs.io/en/latest/guide/getting-started/
2. https://aclanthology.org
3. https://scikit-learn.org/stable/

Nous pouvons également utiliser l'API de l'anthologie sur Python pour obtenir toutes sortes de données associées, notamment :

— les métadonnées pour chaque élément de l'Anthologie, disponibles aux formats XML et YAML ;
— des guides permettant d'accéder à ces métadonnées et de les transformer, sous forme des modules de Python et des scripts ;
— et enfin, des codes et des modèles pour lancer le site web de l'Anthologie ACL.

Particulièrement, les métadonnées de tous les articles de l'anthologie sont les plus pertinentes pour ce projet, car les « features » et le « label » y sont extraits. Le tableau 2 ci-dessous présente les métadonnées des articles qui peuvent être utilisées pour ce type de tâche de classification de texte. Parmi celles-ci, le *title* et *l'abstract* sont les éléments les plus fréquemment employés comme " features".

Métadonnée	Description
id	ID de l'article
title	titre de l'article
abstract	résumé de l'article
authors	liste des auteurs de l'article
language	la langue dans laquelle l'article est principalement rédigé
pages	numéro de pages
...	...

TABLE 2 – Description des métadonnées des articles

En consultant les données, il est clair qu'un grand nombre d'articles sont associés aux quelques principales conférences, notamment ACL, EMNLP, COLING, et LREC, ce qui m'incite à effectuer une tâche de classification, voir la section 2.1. Par ailleurs, il est possible d'explorer d'autres informations à partir de ces métadonnées, comme le nombre de pages ou nombre d'auteurs, etc. Toutes ces informations explicites ou implicites sont intéressantes pour des tâches de classification.

2 Description de nos tâches distribuées

Dans ce projet, deux tâches de classification ont été développées visant à capturer des features (titre et résumé) des articles à l'aide de modèles d'apprentissage automatique, pour deux genre de classifications (binaire et multiclasses) afin de comparer les performances des différents modèles sur des tâches de classification de texte.

2.1 Tâche 1 : "lishuxu-predict_venue-title+abstract"

La Tâche 1 consiste à prédire la conférence à laquelle un article est associé en se basant sur le titre et le résumé, tout en couvrant trois conférences principales comme "label" :

— ACL (Association for Computational Linguistics) plus général du domaine linguistique computationnelle ;
— EMNLP (Empirical Methods in Natural Language Processing) met l'accent sur les méthodes de traitement automatique des langues basées sur les données, avec une forte orientation vers les méthodes expérimentales et empiriques.
— LREC (Language Resources and Evaluation Conference) principalement axé sur le développement, l'utilisation et l'évaluation des ressources et outils linguistiques.

Le tableau 3 présente la répartition des données dans le cadre de cette tâche. Il faut noter que les ensembles d'entraînement et de test sont équilibrés, avec une répartition égale des échantillons pour chaque label de conférence.

Dataset	Total	ACL	EMNLP	LREC
Train	6000	2000	2000	2000
Test	1500	500	500	500

TABLE 3 – Données pour la Tâche 1 (multi-conférence)

2.2 Tâche 2 : "lishuxu-article_lenth-title+abstract"

Tâche 2 se concentre sur la prédiction de la longueur d'un article à travers son titre et son résumé, en déterminant si l'article est "long" (9 pages ou plus) ou "short" (8 pages ou moins).

Bien que de nombreuses collections contiennent déjà des volumes divisées des articles en *Long papers* et *Short papers* se basant sur les exigences de rédaction lors de la publication, l'objectif de cette tâche est plutôt de tester un modèle qui prédisent la longueur des articles par leur nombre de pages, à partir de leur titre et de leur résumé. Un autre point à noter est que les échantillons sont sélectionnés parmi les articles rédigés de même format. En effet, le format des anciens articles est différent, ce qui permet de maintenir un meilleur équilibre dans les jeux de données.

Les données pertinentes à cette tâche est enregistrée dans le tableau 4 (après avoir éliminé les valeurs nulles) :

Dataset	Total	long	short
Train	12800	6400	6400
Test	3189	1598	1591

TABLE 4 – Données pour la Tâche 2 (binaire-long_court)

3 Classificateurs utilisés pour les tâches

3.1 Sélection de baseline

Pour établir les performances de base, nous avons sélectionné un modèle simple sur la plateforme sklearn (Pedregosa et al., 2011) comme baseline pour toutes les deux tâches :

CountVectorizer + LogisticRegression (`count_lr`) : ce modèle utilise le `CountVectorizer` pour transformer les textes en vecteurs de fréquence de mots. En se basant sur ces vecteurs, la « LogisticRegression » est appliquée pour réaliser la classification. Les paramètres principaux sont `C=1.0` pour contrôler la régularisation et `max_iter=1000` pour assurer la convergence, le n-gram est de 1 par défaut comme tous les autres paramètres. Ce modèle simple constitue la baseline comme une référence pour évaluer les performances d'autres modèles.

3.2 Modèles à comparer au baseline

Sur la base du modèle de baseline déjà sélectionné, nous privilégions les approches fournies par la librairie sklearn. Nous choisissons donc les deux méthodes de vectorisation, à savoir TfidfVectorizer et CountVectorizer, et nous les combinons avec des modèles tels que MultinomialNB, SVC, LogisticRegression, RandomForest etc, pour effectuer des comparaisons. Voici les combinaisons choisies.

— TfidfVectorizer
  — LogisticRegression (`tfidf_lr`)
  — SVC (`tfidf_svc`)
  — RandomForest (`tfidf_rf`)

— MultinomialNB (`tfidf_mnb`)
— CountVectorizer
— SVC (`count_svc`)
— MultinomialNB (`count_mnb`)
— RandomForest (`count_rf`)

Outre les huit systèmes simples mentionnés ci-dessus, par curiosité, nous avons également sélectionné plusieurs classificateurs utilisant des approches comme Word2Vec et BERT, en combinaison avec des modèles de même librairie sklearn, afin d'explorer les différences entre ces approches d'extraction de caractéristiques. Par conséquent, nous avons choisi les combinaisons suivantes comme complément :

— Word2Vec sur gesim (Řehůřek and Sojka, 2010)
— LogisticRegression (`w2v_lr`)
— SVC (`w2v_svc`)
— RandomForest (`w2v_rf`)
— DistilBert [4] de transformers
— LogisticRegression (`distilbert_lr`)
— SVC (`distilbert_svc`)
— RandomForest (`distilbert_rf`)

Au total, nous avons utilisé quatorze systèmes, y compris le baseline, pour exécuter ces deux tâches.

4 Résultats et analyses

4.1 Tâche 1 Classification multi conférences

Le tableau 5 montre les résultats de la Tâche 1 du point de vue Accuracy :

Méthode d'extraction de features. Comme cette tâche consiste à classifier un article parmi 3 conférences, autrement-dit il s'agit d'une catégorisation du topic. Pour les titres et résumés qui contiennent des informations très concentrées, les termes spécialisés jouent un rôle important dans la classification. Donc du point de vue de l'extraction de features, les méthodes basées sur la fréquence des mots et la pondération, telles que `tfidf` et `count`, permettent de mieux saisir la pertinence des mots-clés pour ce type de tâche. En revanche, la méthode `word2vec` (Mikolov, 2013), issue de l'apprentissage non supervisé, projette les mots directement dans un espace vectoriel, ce qui rendrait la distinction entre les mots moins évidente.

De plus, pour l'approche avec BERT, la recherche de Devlin 2018 ont montré que son mécanisme d'entraînement bidirectionnel offre des avantages significatifs dans les tâches de classification de texte et de compréhension sémantique, en capturant des informations globales au-delà du simple contexte local. Cela se traduit par de meilleures performances dans la tâche présentée, particulièrement lorsqu'il est associé à un classificateur svc, ce qui n'est guère surprenant. Cependant, pour des textes courts comme les titres et résumés, l'avantage de BERT dans la compréhension de textes longs ne se manifeste pas pleinement ici ; de plus, le modèle pré-entraîné ne serait pas spécifiquement destiné à l'analyse de textes académiques. Nous supposons que cela pourrait expliquer pourquoi l'avantage de BERT n'est pas totalement en position de force dans notre tâche.

4. https://huggingface.co/docs/transformers/en/model_doc/distilbert

Modèle	Accuracy
tfidf_mnb	0.597
tfidf_lr	0.611
tfidf_svc	0.610
tfidf_rf	0.614
count_lr (baseline)	0.578
count_mnb	0.602
count_svc	**0.620**
count_rf	0.609
w2v_lr	0.586
w2v_svc	0.583
w2v_rf	0.563
distilbert_rf	0.593
distilbert_svc	**0.623**
distilbert_lr	0.616

TABLE 5 – Les performances des différents modèles de la Tâche 1

Modèles pour classifier. On observe que le classificateur SVC présente une performance globale supérieure par rapport aux autres modèles, en grande partie grâce à sa capacité à gérer efficacement les caractéristiques complexes et non linéaires. D'autant plus qu'avec une extraction de caractéristiques basée sur BERT, le SVC peut tirer avantage des représentations riches et contextuelles de BERT. À l'inverse, les modèles plutôt linéaires, tels que la LogisticRegression et le MultinomialNB, montrent des résultats légèrement moins performants dans ce cadre.

Différence parmi les classes Dans cette tâche de classification multiclasses, nous avons également observé un phénomène intéressant, comme indiqué dans le tableau 6 ci-dessous : le modèle démontre

une nettement meilleure performance pour la catégorie LREC, comparée aux deux autres. En se référant aux descriptions des trois conférences mentionnées dans la section 2.1, on constate effectivement que les thèmes de recherche de LREC se distinguent sensiblement de ceux d'ACL et d'EMNLP, qui se rapprochent davantage en termes de sujets abordés. Cette différence pourrait se manifester dans les mots-clés présents dans les titres et les résumés, ce qui rendrait les features des articles LREC plus faciles à extraire et à séparer des autres catégories, expliquant ainsi la meilleure performance du modèle pour cette classe.

	ACL	EMNLP	LREC
f1_score_avg	0.50	0.50	0.80

TABLE 6 – La moyenne de f1-score des trois classes

4.2 Tâche 2 Classification de longueur (binaire)

Le tableau 7 ci-dessous reflète les performances des différents modèles pour les résultats de la Tâche 2 :

Modèle	Accuracy
tfidf_mnb	0.675
tfidf_lr	0.683
tfidf_svc	0.682
tfidf_rf	0.676
count_lr (baseline)	0.647
count_mnb	0.669
count_svc	**0.694**
count_rf	0.684
w2v_lr	0.648
w2v_svc	0.665
w2v_rf	0.665
distilbert_rf	0.688
distilbert_svc	**0.700**
distilbert_lr	**0.706**

TABLE 7 – Les performances des différents modèles de la Tâche 2

Les performances des modèles testés dans la Tâche 2 sont globalement meilleures que celles de la Tâche 1. Cela peut être attribué à la facilité d'extraction et d'analyse des features dans cette tâche de classification binaire. Bien sûr, la différence dans le nombre d'échantillons du dataset pourrait également être une raison de la variation des performances. Ce point nécessite une analyse et une discussion plus approfondies.

D'un point de vue des performances des modèles dans les résultats de cette tâche elle-même, les meilleurs modèles sont similaires par rapports à ceux de la Tâche 1. Les méthodes d'extraction de features basées sur Count, TF-IDF et BERT surpassent celles de Word2Vec. Et parallèlement, le classificateur SVC affiche des performances générales supérieures, tandis que la régression logistique montre les meilleurs résultats lorsqu'elle est combinée avec BERT dans cette tâche.

4.3 Tentative d'optimiser les performances

CountVectorizer. Après avoir exécuté toutes les deux tâches dans ces modèles, nous avons réfléchi aux faibles performances du baseline choisi(`count_lr`). De manière plus générale, cela nous amène à envisager une optimisation des méthodes d'extraction de features telles que CountVectorizer, en mettant l'accent sur la dimension des n-grammes. En effet, le par défaut du paramètre `ngram_range` de `CountVectorizer` est fixé à (1,1), ce qui correspond à un modèle unigram. Nous pourrions tout à fait envisager d'optimiser ce modèle en augmentant n à 2 ou 3.

Modèle	uni-gram	bi-gram	tri-gram
count_lr	0.578	0.610	0.606
count_svc	0.620	0.626	0.625

TABLE 8 – Accuracy en modifiant la "ngram_range=" de **Tâche 1**

Modèle	uni-gram	bi-gram	tri-gram
count_lr	0.649	0.682	0.683
count_svc	0.694	0.696	0.692

TABLE 9 – Accuracy en modifiant la "ngram_range=" de **Tâche 2**

À travers les deux tableaux 8 et 9, on observe que le passage de l'unigram au bigram améliore les performances des classificateurs. Cependant, l'ajout du trigramme n'apporte qu'une amélioration marginale, voire parfois une diminution des performances. Ce phénomène concorde avec les descriptions dans (Jurafsky and Martin, 2009), où l'on constate que la majorité des informations sémantiques se concentrent au niveau des mots et des groupes de mots. En effet, la structure grammaticale et lexicale des langues naturelles se limite souvent aux relations entre deux mots, expliquant

pourquoi le bigram parvient à équilibrer richesse des caractéristiques et complexité du modèle dans la plupart des tâches NLP.

BERT. Concernant BERT, comme mentionné dans la section précédente, le modèle pré-entraîné DistilBERT ne s'adapte pas bien aux articles scientifiques. Nous pourrions envisager d'essayer le module SciBERT, qui a été spécifiquement entraîné dans le domaine de la recherche. Nous émettons l'hypothèse qu'il pourrait donner des résultats différents. Malheureusement, jusqu'à présent, j'ai rencontré des problèmes de lenteur d'exécution avec SciBERT, ce qui a empêché son utilisation. Par conséquent, je ne peux pas fournir de données soutenant cette hypothèse dans le présent rapport.

4.4 Synthèse

En synthèse des analyses précédentes, nous avons de bonnes raisons de considérer le modèle `count_svc` comme le meilleur choix parmi les modèles non transformateurs, tandis que `distilbert_svc` se révèle être le modèle le plus performant parmi ceux testés dans la catégorie des transformateurs.

— count_svc : CountVectorizer() avec ngram_range=(1, 2), tous les autres sont par défaut.

— distilbert_svc : `batch_size=32, return_tensors='pt', padding=True, truncation=True`)

Nous appliquerons ces modèles sélectionnés sur d'autres benchmarks publiés, et les résultats peuvent être consultés dans l'Annexe A.

References

Jacob Devlin. 2018. Bert : Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv :1810.04805*.

Dan Jurafsky and James H. Martin. 2009. *Speech and language processing : an introduction to natural language processing, computational linguistics, and speech recognition*. Pearson Prentice Hall, Upper Saddle River, N.J.

Tomas Mikolov. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv :1301.3781*, 3781.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12 :2825–2830.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA. http://is.muni.cz/publication/884893/en.

Benchmarks	count_svc	distilbert_svm
felipe-long_or_findings-abstract	0.471	0.470
felipe-long_or_findings-abstract+title	0.472	0.469
felipe-long_or_findings-title	0.470	0.470
felipe-wmt_or_amta-abstract	0.887	0.830
felipe-wmt_or_amta-abstract+title	0.879	0.837
felipe-wmt_or_amta-title	0.780	0.787
kaborewe-year_range_prediction-abstract+title	0.603	0.584
lishuxu-article_length-title+abstract	0.696	0.699
lishuxu-predict_venue-title+abstract	0.626	0.623
liyuexi-high_or_low_pages-num_authors	0.727	0.727
suzukiyu-if_student_all-title	0.881	0.880
suzukiyu-if_student_all-title+abstract	0.891	0.896
tedongmu-author_prediction-title+abstract	0.628	0.526

TABLE 10 – Performance des modèles “count_svc” et “distilbert_svm” sur plusieurs benchmarks

A Annexe