Analyse de sentiments de critiques de films

Antoine Brisebois-Roy, Monssaf Toukal, Samuel Arseneault, Marc Anhoury
Intelligence artificielle : techniques probabilistes et d'apprentissage, TP4

Abstrait

Nous présentons un modèle d'apprentissage supervisé pour faire l'analyse de sentiments de critiques non professionnelles de films se trouvant sur IMDb. Notre modèle repose sur le traitement simple du langage naturel. Nous comparons quelques méthodes d'extraction de caractéristiques ainsi que quelques classificateurs afin de déterminer quelles combinaisons de ces méthodes fournissent les meilleurs résultats. Nous atteignons une exactitude de 90.506% avec de la *tokenisation*, l'utilisation d'unigrammes et de bigrammes ainsi qu'un classificateur utilisant du SGD.

1 Résumé de la littérature

L'inspiration principale à la base de ce travail est un article publié par l'Université de Stanford*: Learning Word Vectors for Sentiment Analysis* (Maas, 2011). Dans cet article, il est montré que l'extraction du sens des phrases dans un texte à l'aide de modèles est une tâche assez simple, mais que lorsque ces mêmes techniques sont utilisées pour extraire le sentiment d'une phrase elles échouent. C'est pour cela que, dans cet article, les auteurs utilisent de l'apprentissage non supervisé pour trouver les ressemblances entre les phrases et que par la suite, ils utilisent de l'apprentissage supervisé pour extraire le sentiment des phrases. Cet apprentissage supervisé est fait à partir d'une base de données de critique de films, et c'est cette même base de données que nous utiliserons pour faire la présente étude. Nous nous en tiendrons à de l'apprentissage supervisé.

2 Approche théorique

Afin de bien poser les bases de ce travail, il serait pertinent de présenter les approches théoriques qui le soutiennent. Celles-ci sont ici présentées dans l'ordre des opérations effectuées pour l'obtention des résultats finaux.

2.1 Obtention des données

Les données proviennent de la base de données établie par Maas, 2011. Avant de procéder à un prétraitement conventionnel, il a été jugé plus pratique de regrouper les données en un seul fichier CSV: sous leur format initial, chaque revue de film était dans fichier séparé, ce qui rendait leur traitement quelque peu plus complexe.

2.2 Prétraitement des données

Pour le prétraitement des données, 2 articles ont été utiles. Le premier mentionne qu'il peut être utile de retirer les *emojis* du texte, car ceux-ci introduisent souvent de l'information bruitée (Go, 2009). Par exemple, si une revue de film contient à la fois un *emoji* souriant pour une portion positive et un *emoji* fâché pour une autre portion, l'information est contradictoire. Un autre exemple pourrait être l'utilisation d'un *emoji* souriant dans un commentaire sarcastique à connotation négative. Le second article suggère cependant que les *emojis* peuvent apporter de l'information contextuelle très utile (Dickinson, 2015). Il semble y avoir ici une contradiction entre les deux. Nous avons opté pour retirer totalement les *emojis*, notre intuition suggérant que le reste de l'information contenue dans les revues serait probablement suffisante pour entrainer notre modèle avec une précision suffisante.

Plusieurs articles proposent de retirer les caractéristiques du texte qui n'apporte pas d'information utile en lien avec le sentiment du texte, c'est-à-dire les noms d'utilisateurs (souvent précédés par le caractère «@» dans le cas de Twitter; dans le nôtre, il est peu probable de retrouver ce genre de mention, mais nous avons tout de même choisi de les retirer par précaution) et les liens URLs (Dickinson, 2015) et les caractères de ponctuation (Pouransari, 2014). En effet, il est possible de constater intuitivement que ces éléments sont tous neutres: un nom d'utilisateur ou un hyperlien ne transmettent pas d'émotion pertinente pour la classification. Pour la ponctuation, on pourrait supposer que les points d'exclamation transmettent une émotion, mais ce n'est pas réellement le cas, car ils peuvent être utilisés pour amplifier tant une émotion positive que négative.

Il est aussi pertinent de retirer les éléments HTML ainsi que les *stop words* (les mots n'apportant pas d'information pertinente, telle que les déterminants), ce qui peut être réalisé par défaut par certains *tokenizers* comme ceux inclus dans NLTK (Pouransari, 2014).

L'intérêt de retirer le plus possible les symboles est d'éviter d'obtenir un trop grand nombre de *features*, ce qui pourrait avoir comme effet de ralentir l'entrainement du modèle.

2.3 Extraction des caractéristiques

Avant de passer à la classification, il est nécessaire d'extraire des caractéristiques pour l'ensemble des données. Plusieurs méthodes existent déjà pour ce faire.

N-Gram

Une des approches répandues dans le traitement de la langue naturelle est le modèle N-Gram. Pour faire court, N-Gram est un modèle probabiliste qui a pour but de prédire le prochain élément dans une chaine de N éléments (par exemple, de n mots précédents). Un élément peut être une lettre, une syllabe ou encore un mot. En fait, la méthode N-Gram est une forme de modèle de Markov: afin d'effectuer une prédiction, il faut se baser uniquement sur les éléments précédents de la chaine. Il y a plusieurs variantes de N-Gram: unigramme, bigramme, trigramme, etc. (Kapadia, 2019).

Bag of Words

Une autre approche très populaire dans le traitement de la langue naturelle est l'approche Sac de Mots ou encore *Bag of Words*. Dans l'apprentissage machine classique, il est difficile de prendre du texte tel quel comme donné parce qu'un mot n'a pas de taille fixe et cela est un problème. Pour remédier à ce problème, l'approche Bag of Words est souvent utilisée pour sa simplicité. Elle consiste à transformer chaque document texte en vecteur de booléen. Ici, un document peut être une phrase, un paragraphe ou autre. Au départ, cette approche consiste à compter tous les mots différents de tous les documents à analyser. Grâce à cela, il est possible de transformer chaque document texte en vecteur de taille n où n est le nombre de mots différents. Les vecteurs ne contiendront que des valeurs booléennes tel que chaque entrée booléenne décrit si un des n mots différents est dans le document texte. *Bag of Words* s'inspire beaucoup du fait qu'un texte, indépendamment de sa syntaxe, est similaire à un autre s'il utilise les mêmes mots (D'Souza, 2018).

TF-IDF

Cette méthode d'extraction de caractéristiques transforme les données prétraitées en un format très similaire au *Bag of Words*, c'est-à-dire un dictionnaire où les clefs sont des instances de mots et dont les valeurs sont numériques. En revanche, les valeurs numériques pour TF-IDF ne sont pas un simple compte de la fréquence des mots; ils établissent plutôt un comparatif entre les pertinences locale (par exemple, dans une phrase) et globale (dans l'ensemble du texte) d'un mot donné (Wu, 2008).

2.4 Classification

Plusieurs méthodes de classifications existent et ont été utilisées pour effectuer une analyse de sentiment en fonction de commentaires, de tweets, etc. De ce fait, il sera surtout question d'explorer trois méthodes populaires du moment, soit les SVM (support vector machines), la descente de gradient stochastique (SGD) et la régression logistique respectivement. Dans l'article *Classification of Sentiment Reviews using N-gram Machine Learning Approach*, ces trois approches ont été utilisées avec le même jeu de données, soit les données provenant du site IMDb.

SVM

Les SVM sont souvent utilisés pour résoudre des problèmes de classification. Ils ont pour fonction de pouvoir séparer les données par leur classe grâce à leur fonction noyau (leur *kernel*). Les SVM peuvent avoir différents types de noyau comme un noyau linéaire, polynomial ou encore sigmoïde.

Selon l'étude *Classification of Sentiment Reviews using N-gram Machine Learning Approach*, l'approche SVM couplée avec de l'unigramme et du bigramme a été la plus performante. Il serait donc intéressant de tenter de reproduire ces résultats et de comparer les résultats obtenus (Tripathy, 2016).

SGD

La descente de gradient stochastique est une approche intéressante à essayer puisqu'elle est moins courante dans le traitement de la langue naturelle. Elle serait intéressante à reproduire puisque cela a été la pire méthode de classification testée par les chercheurs. Il a été donc décidé de reproduire à nouveau ces résultats et de les comparer.

Régression logistique

La dernière méthode de classification est la régression logistique. Selon la littérature actuelle, les experts ont tendance à préférer les SVM plutôt que la régression logistique lorsque les variables ne sont pas explicitement définies. Au contraire, la régression logistique performe très bien pour des problèmes ou les variables explicatives sont explicites (Bassey, 2020). En général, les deux approches ont des performances similaires. Cependant, il est à penser, à priori, que l'approche SVM est plus performante. Toutefois, afin de pouvoir se prononcer avec plus de certitude, il a été décidé de tester l'approche avec régression logistique même si celle-ci ne se retrouve pas dans l'article d'origine. Toutefois, il est à noter qu'elle sera testée, d'une part avec des unigrammes et d'autre part avec des bigrammes et de des unigrammes en même temps.

3 Analyse des résultats

Dans le but de déterminer la combinaison la plus avantageuse pour le pipeline complet, toutes les combinaisons possibles de prétraitement, d'extraction de caractéristiques et de classificateurs ont été essayées. Il s'agit donc d'une recherche exhaustive de la meilleure combinaison. À noter que l'exploration d'hyperparamètres pour un modèle donné n'a pas été tentée, car le temps d'exécution aurait été trop grand. Notre implémentation est disponible à l'adresse suivante. Les cinq figures suivantes présentent les métriques de

performance (exactitude, F-mesure, précision et rappel) des combinaisons possibles pour chacun des classificateurs.

	accuracy	f1_score	precision	recall
('Our Tokenizer', (1, 2), 'tfidf')	0.900392	0.900807	0.893134	0.908613
('Default', (1, 2), 'tfidf')	0.900131	0.900209	0.895530	0.904937
('Default', (1, 1), 'tfidf')	0.897386	0.897693	0.891074	0.904412
('Our Tokenizer', (1, 1), 'tfidf')	0.893725	0.894457	0.884467	0.904674
('Default', (1, 2), 'bag of words')	0.892680	0.892694	0.888629	0.896796
('Our Tokenizer', (1, 2), 'bag of words')	0.890327	0.890627	0.884287	0.897059
('Default', (1, 1), 'bag of words')	0.887582	0.887699	0.882857	0.892595
('Our Tokenizer', (1, 1), 'bag of words')	0.881046	0.881664	0.873261	0.890231

Fig 1 : Performances du classificateur logistique

	accuracy	f1_score	precision	recall
('Our Tokenizer', (1, 2), 'tfidf')	0.904314	0.905206	0.892948	0.917805
('Default', (1, 2), 'tfidf')	0.903529	0.903630	0.898701	0.908613
('Default', (1, 1), 'tfidf')	0.898170	0.898686	0.890234	0.907300
('Default', (1, 2), 'bag of words')	0.894902	0.894544	0.893606	0.895483
('Our Tokenizer', (1, 1), 'tfidf')	0.894248	0.895356	0.882233	0.908876
('Our Tokenizer', (1, 2), 'bag of words')	0.893333	0.893667	0.886963	0.900473
('Default', (1, 1), 'bag of words')	0.883660	0.883994	0.877588	0.890494
('Our Tokenizer', (1, 1), 'bag of words')	0.882353	0.883299	0.872439	0.894433

Fig 2 : Performances du classificateur par SGD

	accuracy	f1_score	precision	recall
('Our Tokenizer', (1, 2), 'tfidf')	0.813203	0.792929	0.884578	0.718487
('Our Tokenizer', (1, 1), 'bag of words')	0.811373	0.788695	0.891427	0.707195
('Default', (1, 1), 'bag of words')	0.788627	0.816020	0.719936	0.941702
('Default', (1, 2), 'tfidf')	0.775556	0.729564	0.911452	0.608193
('Our Tokenizer', (1, 1), 'tfidf')	0.743660	0.790379	0.666486	0.970851
('Default', (1, 1), 'tfidf')	0.719477	0.776365	0.643573	0.978204
('Our Tokenizer', (1, 2), 'bag of words')	0.714902	0.763372	0.650397	0.923845
('Default', (1, 2), 'bag of words')	0.672941	0.742857	0.610267	0.949055

Fig 3 : Performances du classificateur par SVM

	accuracy	f1_score	precision	recall
('Our Tokenizer', (1, 2), 'tfidf')	0.880261	0.878675	0.886424	0.871061
('Default', (1, 2), 'tfidf')	0.877124	0.874633	0.888618	0.861082
('Default', (1, 2), 'bag of words')	0.876078	0.874967	0.878908	0.871061
('Our Tokenizer', (1, 2), 'bag of words')	0.874771	0.874509	0.872452	0.876576
('Our Tokenizer', (1, 1), 'bag of words')	0.866275	0.866187	0.862914	0.869485
('Default', (1, 1), 'bag of words')	0.865882	0.865460	0.864327	0.866597
('Default', (1, 1), 'tfidf')	0.865098	0.863347	0.870726	0.856092
('Our Tokenizer', (1, 1), 'tfidf')	0.859085	0.858382	0.858833	0.857931

Fig 4 : Performances du classificateur Bayes Naïf

	accuracy	f1_score	precision	recall
('Default', (1, 2), 'bag of words')	0.878954	0.874797	0.901616	0.849527
('Default', (1, 2), 'tfidf')	0.878954	0.874797	0.901616	0.849527
('Our Tokenizer', (1, 2), 'bag of words')	0.877516	0.874008	0.895564	0.853466
('Our Tokenizer', (1, 2), 'tfidf')	0.877516	0.874008	0.895564	0.853466
('Default', (1, 1), 'bag of words')	0.852549	0.847732	0.872222	0.824580
('Default', (1, 1), 'tfidf')	0.852549	0.847732	0.872222	0.824580
('Our Tokenizer', (1, 1), 'bag of words')	0.850327	0.846247	0.865897	0.827468
('Our Tokenizer', (1, 1), 'tfidf')	0.850327	0.846247	0.865897	0.827468

Fig 5 : Performances du classificateur par *random forest*

Il est plus intéressant de s'attarder aux meilleurs résultats pour chaque classificateur, présentés à la figure 6. Il est ainsi plus intuitif de les comparer. On observe que selon la métrique de l'exactitude (*accuracy*), le classificateur par SGD est le meilleur.

	accuracy	f1_score	precision	recall	tokenizer	ngrams	tf-idf / BoW
SGD	0.904314	0.905206	0.892948	0.917805	Our Tokenizer	(1, 2)	tfidf
Logistic	0.900392	0.900807	0.893134	0.908613	Our Tokenizer	(1, 2)	tfidf
Random Forest	0.880261	0.878675	0.886424	0.871061	Our Tokenizer	(1, 2)	tfidf
Naive Bayes	0.878954	0.874797	0.901616	0.849527	Default	(1, 2)	bag of words
SVM	0.813203	0.792929	0.884578	0.718487	Our Tokenizer	(1, 2)	tfidf

Fig 6 : Performances des meilleures configurations des classificateurs

C'est donc le classificateur SGD avec notre *tokenizer*, les unigrammes et bigrammes et TF-IDF qui sera retenu pour l'évaluation finale du modèle avec les données de test, produisant ainsi la matrice de confusion de la figure 7.

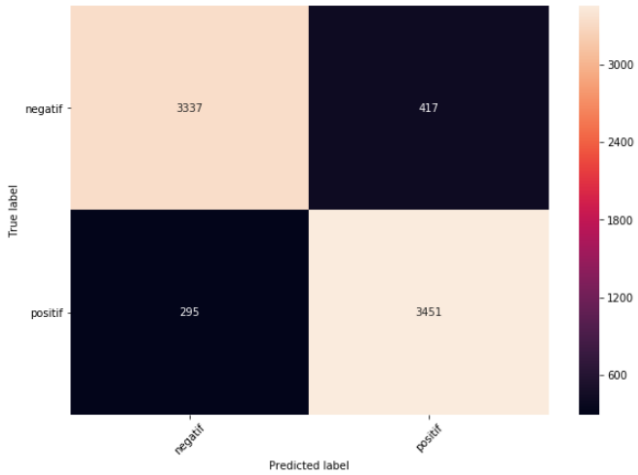

Fig 7 : Matrice de confusion du modèle sélectionné sur les données de test

La performance peut être évaluée à l'aide de mêmes métriques qui ont été utilisées plus tôt : une exactitude de 90.506%, un rappel de 92.125% et une précision de 89.219%.

4 Analyse critique de l'approche

Plusieurs méthodes de classifications ont été essayées lors de cette expérience. D'une part, il a été question d'essayer plusieurs combinaisons de différentes méthodes. Selon les résultats obtenus, la descente de gradient stochastique de gradient couplé avec le *tokenizer*, l'unigramme et le bi-gramme.

En comparant les résultats ci-haut avec ceux obtenus dans l'article *Classification of Sentiment Reviews using N-gram Machine LearningApproach*, on remarque que la méthode qui a donné la meilleure exactitude a été l'approche couplant la classification avec SVM et l'unigramme et bi-gramme. Or, ce n'est pas tout à fait ce qui a été observé au cours de cette expérience. Toutefois, une différence importante à noter est que ces chercheurs n'ont pas utilisé le *tokenizer* de la même manière : chaque critique de film est transformée en matrice de compteurs de *token*. Ceci pourrait expliquer la différence des résultats obtenus. Une autre distinction majeure est que dans notre cas aucune des différentes méthodes de classification n'a été testée avec du trigramme. C'est une distinction importante parce que c'est avec cette approche que l'exactitude du modèle a été maximisée.

Il existe plusieurs approches possibles pour tenter de résoudre ce problème. L'analyse des sentiments est un sujet de classification très connue et diverses méthodes ont été utilisées. Entre autres, il serait intéressant d'aborder ce problème en utilisant des réseaux de neurones comme les *Very Deep Convolutional Neural Network (VDCNN)*, car selon l'article rédigé par l'équipe de chercheurs de Facebook, l'utilisation d'un réseau de neurones contenant beaucoup de couches cachées améliore grandement l'exactitude du modèle (Conneau, 2017). Il serait donc intéressant d'explorer cette avenue en ce qui est à trait de la classification de textes. De plus, nous aurions pu tenter une recherche d'hyperparamètres pour le modèle que nous avons retenu, ce qui aurait possiblement pu améliorer les résultats.

References

Bassey, P., 2020. Logistic Regression Vs Support Vector Machines (SVM). [online] Medium. Available at: <https://medium.com/axum-labs/logistic-regression-vs-support-vector-machines-svm-c335610a3d16> [Accessed 25 April 2020.

Conneau, A., Schwenk, H., Barrault, L., & Lecun, Y. (2017). Very Deep Convolutional Networks for Text Classification. Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers. doi: 10.18653/v1/e17-1104

Dickinson, B., Ganger, M., & Hu, W. (2015). Dimensionality reduction of distributed vector word representations and emoticon stemming for sentiment analysis. Journal of Data Analysis and Information Processing, 3(04), 153.

D'Souza, J., 2018. An Introduction To Bag-Of-Words In NLP. [online] Medium. Available at: <https://medium.com/greyatom/an-introduction-to-bag-of-words-in-nlp-ac967d43b428> [Accessed 22 April 2020].

Go, A., Bhayani, R., & Huang, L. (2009). Twitter sentiment classification using distant supervision. CS224N project report, Stanford, 1(12), 2009.

Kapadia, S. (2019, August 19). Language Models: N-Gram. Retrieved April 23, 2020, from https://towardsdatascience.com/introduction-to-language-models-n-gram-e323081503d9

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., & Potts, C. (2011, June). Learning word vectors for sentiment analysis. In Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1 (pp. 142-150). Association for Computational Linguistics.

Pouransari, H., & Ghili, S. (2014). Deep learning for sentiment analysis of movie reviews. Technical report, Stanford University, Tech. Rep.

Tripathy A, Agrawal A, Rath SK (2016) Classification of sentiment reviews using n-gram machine learning approach. Expert Syst Appl 57:117–126

Wu, H. C., Luk, R. W. P., Wong, K. F., & Kwok, K. L. (2008). Interpreting tf-idf term weights as making relevance decisions. ACM Transactions on Information Systems (TOIS), 26(3), 1-37.