Introduction to AI
Final Project
News Title Attractiveness Prediction with Sentiment Analysis

0713313 董柏葳 / 0713341 蔡沛樺

Github repo link:

https://github.com/potadooweii/NYCU-2021Spring-Introduction_to_AI-final_project

Introduction

With so much news being consumed online, there is a great interest in what kind of news do readers click on. The data generated in online news consumption helps us explore news content and the relation between news popularity and readers' preferences. Besides the affecting factors like title, headline and the source of news, sentiment analysis of news is also considered to be a crucial feature to predict the attractiveness of news. Therefore, we propose this project to predict the news attractiveness with news content and its sentiment analysis.

In this project, we conduct the experiment with two datasets from University of Porto and Kaggle, which is an online community of data scientists. The first dataset is "News Popularity in Multiple Social Media Platforms", as known as Social Media Data (SMD), which originally contains the sentiment scores of the titles and the headlines, and the second dataset is "New York Times Articles & Comments", as known as News Data (ND), which does not contain the sentiment scores. Therefore, we calculate the sentiment score for the dataset "New York Times Articles & Comments" by applying a built-in package of python called NLTK.

According to the paper proposed by Hardt and Rambow [2], it indicated that bag-of-word models based on headlines did indeed have predictive value concerning viewing behavior, although models based on the article body were more accurate due to the reason that readers are able to anticipate the contents of an article from a headline. Thus, we apply a pre-trained word-to-vector model to produce our word embedding from the titles and the keywords of the articles. Figure 1 is our proposed framework. Moreover, we have a better knowledge of the relations between titles by applying the doc-to-vector model. Then do PCA to those derived vectors for feature extraction.

Initially, we want to quantify the value of a title. In other words, using artificial intelligent method to regress how attractive is an article. However, it fails because the mean-square-error (MSE) is too large to verify that we have a good prediction, and we can't fix it just using the sentiment. Therefore, another idea comes to our mind, we just need to decide whether a title is attractive rather than quantify its valuation.

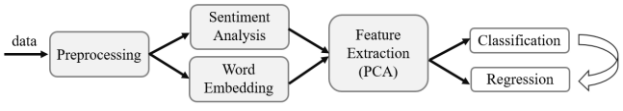

Figure1: The framework of our project.

Related Work

Attractiveness Prediction

Inspired by the paper proposed by Hardt and Rambow [2], which shows that article attractiveness prediction based on features from the article body has a better performance than features from the headline, Lamprinidis et al. [3] use an RNN with two auxiliary tasks, i.e., POS tagging and section prediction to predict attractiveness. Meanwhile, CTR is also considered to be a reliable indicator to news attractiveness. Kuiken et al. [1] analyze the relationship between CTR and the textual/stylistic features of millions of headlines, which can provide insights for how to construct attractive headlines. Nevertheless, none of the approaches mentioned above takes sentiment into consider.

Sentiment Analysis

Sentiment analysis is the task of classifying the polarity of a given text. Socher et al. [5] introduces the Recursive Neural Tensor Network (RNTN), and thus, the accuracy of single sentence positive/negative classification reaches 85.4% and the accuracy of predicting fine-grained sentiment labels for all phrases reaches 80.7%. For the dataset SST-5, which is designed to evaluate a model's ability to understand representations of sentence structure, a Bi-attentive Classification Network (BCN) with ELMo embedding achieves the accuracy of 54.7%. Moreover, using sentence-level embedding reaches the accuracy on the SST-5 dataset 64.4% [4].

Methodology

Preprocessing

First of all, we need to preprocess the dataset. It includes the cleaning of data, tokenization, and removing the stopwords. We use the NLTK package in python to do those previous things.

Sentiment Analysis

For the dataset SMD, there is a build-in sentiment score, so we just use it. However, another one (ND) does not have one, so we need to get some way to derive the sentiment score for it.

There are lots of ways to calculate sentiment score including ruled-based methods, feature-based methods, and embedding-based methods. In our research, we find a state-of-the-art attractiveness prediction using logistic regression method to calculate the score, but due to the limited time and lack of dataset, we choose a compromised way which is a build-in package in python called NLTK.

The package is VADER *(Valence Aware Dictionary and sEntiment Reasoner)*[7]. It is a rule-based sentiment analysis tool and specifically attuned to sentiments expressed in social media. We supposed to merge two datasets, but this sentiment analysis doesn't perform very well, which can be seen in the experiment part, so we just do two experiments separately.

Word Embedding

We find that there is a pre-trained word-to-vector model named *'GoogleNews-vectors-negative300'*[6][8] from Gemsim-data, which is based on Google News dataset and have about 100 billion words, and our dataset is as well as from news. Therefore, we just adapt the pre-trained model to generate our word embedding

from the titles and keywords, and we can see the dimension of vector from its name which is 300.

There is still a question that what we want is to know the relations between titles, so single word vector doesn't really meet our requirement. Therefore, doc-to-vector should be taken into account. Doc-to-vector can understand the order of words and the relation of the context, and due to the pre-trained model, we have large corpus which is useful to train a great d2v model.

However, in this application, we just want to know the similarity between the titles or keywords, so we just need to averaging the values of word vectors, which makes us available to represent sentences by vectors.

Feature Extraction

To prevent from the curse of dimensionality, we do the feature extraction. We use PCA (Principal Component Analysis) to reduce the dimensional of word(doc) vectors, and as we know, the contribution of high-dimensional feature will get lower and lower, so we find a relatively great dimensional which is 15 for it. In other words, we reduce the input vector from 300-dimensional to 15-dimensional. Moreover, in one of the two datasets we merge the titles and keywords together, because those keywords can be considered as the tag of the article which is also an important first impression of an article.

Classification

We use four kinds of classifier, and one of them is XGclassifier, which is one of the famous tools in Kaggle or some AI competition, and the others are from scikit-learn python package, which are respectively RamdomForest, KNearestNeighbor(k-means), and SVC(svm). Among those classifiers, we only fine tune the XGclassifier, and the others remain their default parameters.

The features we take are the extracted doc vector and the sentiment of titles and headlines. Timeline should be considered, but we want to select those title which is attractive regardless of date/time, so we don't take the timeline as one of our features.

We also once select the topic types section to be a candidate of our features, but we don't take it eventually. There are two reasons. Firstly, there are some overlapped topic types (eg. Style & Fashion) and missing labels which would take bunch of time to dealing with it. Secondly, there are a lot of topic types, so there might be a high chance to overfit.

Regression

Basically, all the features in regressor are as same as those in classifier, but we add one thing new for it trying to make the result better. The new feature is the average of confidence score of the previous three classifiers which SVC is exclusive due to lack of confidence level. This is inspired from Adaboost which we learn from the class, according to changing the weights of classifiers, we can make a better result.

Experiments

Before the experiment, we analyze our datasets.

Firstly, we want to know the what the pre-trained word-to-vector machine learned, so we use t-distributed stochastic neighbor embedding (t-SNE), which do dimensionality reduction on these word vectors, to visualize the patterns. In

Figure 2, we can see there are some region having strong connection. For example, we can see the top of the figure including words describing the name of a region like country names or state names. Thus, we can see that the pre-trained machine performs well. Then, as we describe in the methodology part, we averaging the word vectors to derive the doc-vector value, so we can see the result in Figure 3, it also doing quite well. Let's see the titles at the upper right, they all describe something about technics, while the titles in lower left are talking about economics.

Secondly, we want to see the distribution of the sentiment of titles/headlines, so we also visualize them. Figure 4 and Figure 5 show the title sentiment of the datasets SMD and ND respectively. Because we use two datasets to do the experiment and they have different sentiment resources, we should not compare them directly. However, we still can see some common part of them which is that neutrals title are more attractive than the bias ones.

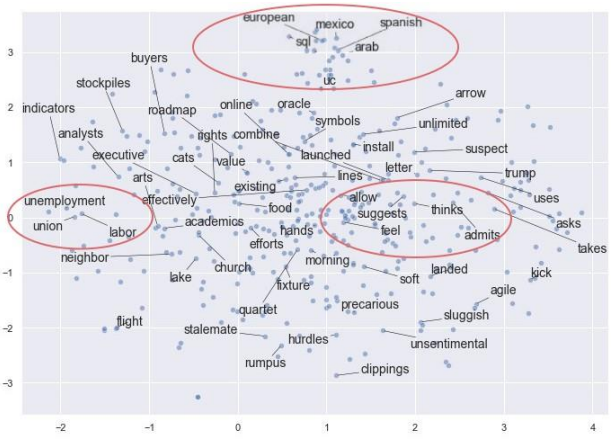

Figure 2: t-SNE analysis for word vectors of titles' words.

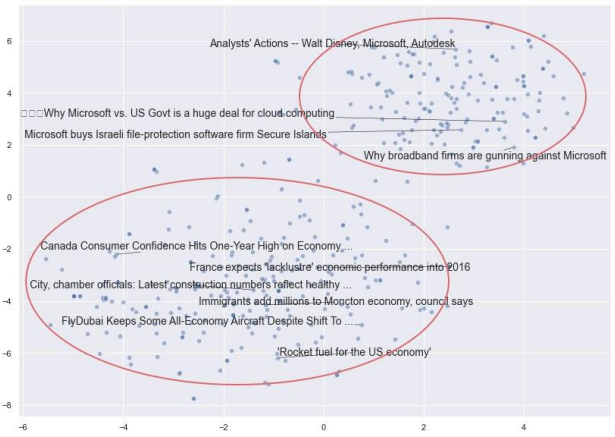

Figure 3: t-SNE analysis for word vectors of titles

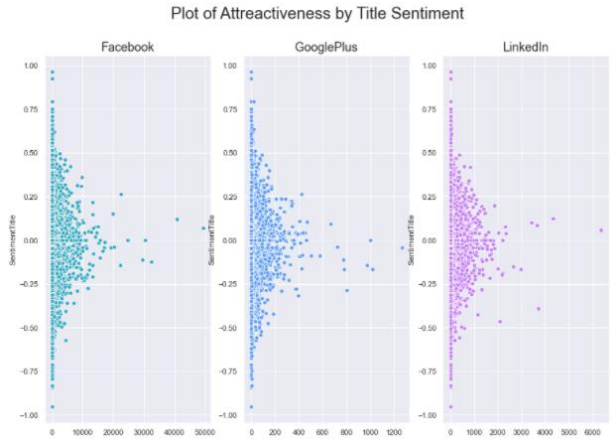

Figure 4: Plot of Attractiveness by Title Sentiment (dataset SMD)

Figure 5: Plot of Attractiveness by Title Sentiment (dataset ND)

Then, we can go to the experiment section. We would only show the results of the dataset ND, since there are lots of related papers doing experiments with another dataset.

Figure 6 shows the ROC curve of three different attractiveness classifiers, which are respectively XG classifier (orange curve), Random Forest classifier (blue curve) and k-means (green curve). Random Forest classifier has the best performance, followed by XG classifier and then k-means.

Figure 7 shows the performance of four different attractiveness classifiers, which are respectively XG classifier, Random Forest classifier, k-means, and SVM. As it shows, Random Forest classifier has the best performance with the accuracy of 80%, and the followed are XG classifier and SVM with accuracy of 79%, and the last one is k-means, which still achieves accuracy of 77%. The performance is same as the result showed in Figure 6.

With using the idea of Adaboost, we redo the regression part. It reduces the MSE slightly. Although the performance does not be improved significantly, we still note that the new feature (prob_dud) do account for a large part, which is showed in Figure 8. In other words, the new feature indeed improves the performance of the regression.

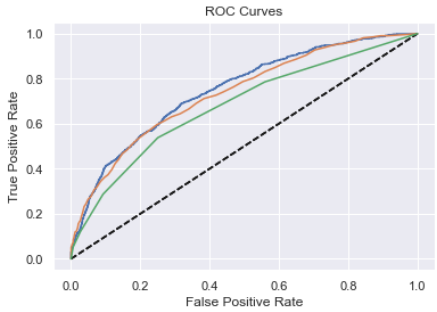

Figure 6: ROC curve of XG classifier, Random Forest classifier and k-means.

XGClassifier performance:

	precision	recall	f1-score	support
0	0.80	0.97	0.88	2615
1	0.59	0.15	0.24	742
accuracy			0.79	3357
macro avg	0.69	0.56	0.56	3357
weighted avg	0.75	0.79	0.74	3357

RandomForestClassifier performance:

	precision	recall	f1-score	support
0	0.81	0.97	0.88	2615
1	0.64	0.17	0.27	742
accuracy			0.80	3357
macro avg	0.72	0.57	0.57	3357
weighted avg	0.77	0.80	0.75	3357

KNearestNeighbors performance:

	precision	recall	f1-score	support
0	0.82	0.91	0.86	2615
1	0.47	0.29	0.36	742
accuracy			0.77	3357
macro avg	0.64	0.60	0.61	3357
weighted avg	0.74	0.77	0.75	3357

SVC performance:

	precision	recall	f1-score	support
0	0.79	0.99	0.88	2615
1	0.72	0.09	0.16	742
accuracy			0.79	3357
macro avg	0.76	0.54	0.52	3357
weighted avg	0.78	0.79	0.72	3357

Figure 7: Performance of four different attractiveness classifiers.

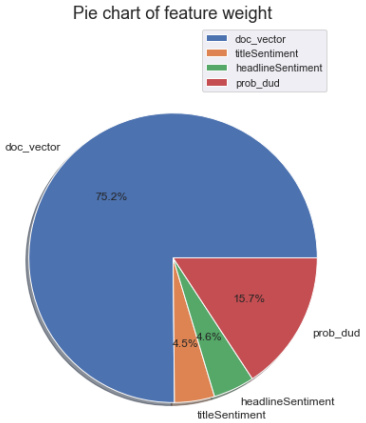

Figure 8: The pie chart of the feature weight in regressor.

Conclusion

In our project, we propose the news title attractiveness prediction with sentiment analysis. While doing data analysis, we find out that news with neutral perspective are more popular. Then we take sentiment as one of our features to train our classifiers. By using Random Forest classifier, we reach the best performance with accuracy of 80%. We consider it can be improved by using a better method to calculate sentiment scores.

Reference

[1] Badri N. Patro, Vinod K. Kurmi, Sandeep Kumar and Vinay P. Namboodiri. 2019. Learning Semantic Sentence Embeddings using Pair-wise Discriminator. *arXiv preprint arXiv: 1806.00807v5.*

[2] Daniel Hardt and Owen Rambow. 2017. Predicting user views in online news. In *Proceedings of the 2017 EMNLP Workshop: Natural Language Processing meets Journalism, pages 7–12.*

[3] Daniel Hardt, Dirk Hovy and Sotris Lamprindis. 2018. Predicting news headline popularity with syntactic and semantic knowledge using multi-task learning. In *Empirical Methods in Natural Language Processing (EMNLP), pages659–664.*

[4] Jeffrey Kuiken, Anne Schuth, Martjin Spitters and Maarten Marx. 2017. Effective headlines of newspaper articles in a digital environment. In *Digital Journalism 5(10):1300–1314.*

[5] Richard Socher, Alex Perelygin, Jean Y. Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng and Christopher Potts. 2013. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In *EMNLP*.

[6] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Proceedings of NIPS.*

[7] VADER https://github.com/cjhutto/vaderSentiment

[8] GoogleNews-vectors-negative300.bin.gz https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM/edit?resourcekey=0-wjGZdNAUop6WykTtMip30g