Ranking viewers comments
Machine Learning for Natural Language Processing 2020

Mansoor Malik
Ensae
mansoor.malik@ensae.fr

Youssef Chouaieb
Ensae
youssef.chouaieb@ensae.fr

Abstract

In this project, our main goal is to sort movie comments by their relevance based only on text. To do this, we scrapped the comments on the Allocine website and we combined NLP and Learning to Rank methods.

We observe that the naive method which consists on sorting by the number of words in a comment performs almost as well as the learning to rank model. However, it is possible to obtain better results by optimizing the model.

1 Problem Framing

Allocine is a reference website in the cinema industry, it is used regularly to have an overview of the quality of a film. By looking into viewers reviews, we notice that the order of reviews by relevance is strongly influenced by the number of "likes" and "dislikes", which could have different sources of bias.

- A recent film with a lot of criticism but few reactions would not organize the comments in an optimal way.
- A review posted several months / years after the release of a film will have very little chance of ending up at the top of the list despite being very useful.

Thus, we wondered if it was feasible to reorder the relevance of reviews posted by viewers based more deeply on the content of their review.

Through this project, we would like to be able to predict the relevance of a comment based only on it's content and not other parameters (author data, likes and dislikes, ...). To define the relevance of a comment we will make the big assumption that Allocine is already ranking the comments based on relevance and we will use the rankings of the website.

2 Experiments Protocol

2.1 Data

To do this, we chose to scrape the Allocine website on most popular movies figuring on the first pages of the website. Due to the pandemic, we chose to concentrate on movies released in 2019 in order to avoid the effect of the sanitary events on the industry. Taking into consideration the second point mentioned above we only take into account the comments during the 6 months following the release of the movie, (we drop recent comments just in case they may be relevant and not correctly ranked).
With the help of the BeautifulSoup package, we will firstly scrape the useful information on all the 2019 films: title, release date, synopsis, author, main actors, etc ... For computational reasons, we selected only 60 movies out of the 700+ available, and only takes the first 20 pages of reviews for each movie, which bring us to 9522 reviews.

2.2 Processing

For the processing of the comments, we start by removing stopwords and punctuations, and in order to avoid any biases coming from a movies actors names, authors names, or genre, we changed all these into "author name", "actor name", and "movie genre".
Due to the specificity of our analysis, we try to order the films and not only predict a utility score, we need a function that allows us to separate the data into a testing set and training set so that all comments for the same movie are in the same subset of data.

2.3 Modeling

We used the Word2Vec model, allowing us to turn the words in our comments into vectors. To increase the efficiency of this method, we use a

pre-trained French Word2Vec model. In order to form a vectorized comment, we do the average of the vectors of each token. A better way is to sum the vectors of each word of the comment with a weight equal to it's TF-IDF score.
In our case we try to rank the reviews based on relevance, we're not confronted to a classification or a regression problem, it's more about ranking. So we're going to use Learning to rank which is a Machine Learning framework.
The goal of Learning to Rank is to order a list. One of the common applications would be the ranking of results of an internet search.
We used the pairwise approach that consists on comparing each pair of observations, the overall structure does not intervene. This can be compared to binary classification (eg RankNet).

As previously described, the pairwise approach takes pair observations during training. Let O_i and O_j two observations and O_i is better ranked than O_j, we note that $O_i \succ O_j$, we can then define a score $s = f(x; w)$ so that $s_i > s_j$.
We then try to learn the probability that O_i is better ranked than O_j by defining it via a logistics function :

$$P_{ij} := P\left(O_i \succ O_j\right) = \frac{1}{1 + e^{-(s_i - s_j)}}$$

The associated loss function penalizes the deviation of the calculated probabilities compared to the desired probability (noted $\bar{P}_{ij}$) it's defined as follows :

$$C = -\bar{P}_{ij} \log\left(P_{ij}\right) - \left(1 - \bar{P}_{ij}\right) \log\left(1 - P_{ij}\right)$$

With

$$\bar{P}_{ij} = \begin{cases} 1 & if\ O_i \succ O_j \\ 0 & if\ O_i \prec O_j \\ \frac{1}{2} & else \end{cases}$$

With these two equations we can write :

$$C = \log\left(1 + e^{-\sigma(s_i - s_j)}\right) + \left(1 - \bar{P}_{ij}\right) \sigma\left(s_i - s_j\right)$$

In order to minimize this objective function, different methods can be used, including the stochastic gradient descent. Once learned, we can then use this function to classify the out-sample observations as follows:
$O_i \succ O_j \ \ if \ \ f\left(x_i; w\right) > f\left(x_j; w\right)$

3 Results

For the Naive model, we chose to take an intuitive approach which is to give a better ranking to the comments having more words.
In order to mesure our model performance, we use a metric that is specific to ranking problems, the Normalized Discounted Cumulative Gain (NDCG) which measures the quality of the ranking :

$$DCG = \sum_{i=1}^{N} \frac{2^{s_i - 1}}{\log_2(i+1)} NDCG = \frac{DCG}{IDCG}$$

With s_i the predicted score of observation i and $IDCG$ the ideal DCG .
The $NDCG$ allows to bring the DCG in an interval of 0 to 1, the more $NDCG$ is close to 1, the more accurate id the ranking.
With our modeling we obtain results slightly better than the results obtained with the Naive model.

Naive method	0.8022
RankNet	0.8346

Table 1: NDCG score

4 Discussion/Conclusion

In conclusion, we could see that our model slightly outperformed the naive method. However, these results remain to be qualified. Indeed, we took as target the ranking of Allociné which can be questionable.
In addition, being limited by computing power, we only took a subsample of the available data. We could also test other approaches in order to model the comments, with camemBERT for example.
And finally, we used multilayer perceptron (MLP) to predict the scores, a better approach would be to use an LSTM which would take each word one by one.

. References

- Word2vec,
  https://en.wikipedia.org/wiki/Word2vec
- Learning to Rank,
  https://en.wikipedia.org/wiki/Learning_to_rank