Learning to Rank Reviews

Anonymous Author(s)

Abstract

In this report, we present a learning model that is trained to map from the text content of a review to a ranking score. Methods that can learn ranking functions are difficult to optimize as they are not typically smooth. We employ a pairwise approach to learning to rank called RankNet. The cost function is defined on the score difference on each pair of inputs to the model as a cross-entropy function, which permits us to employ gradient descent technique to optimize the cost function.

1 Introduction

Recent years Web 2.0 technology has grown exceptionally in terms of size and diversity of contents. Ranking text documents based on their text contents is one of the challenging tasks in information retrieval. The problem of learning to rank can be described as follows. During training, the system is presented by a set of text reviews and their corresponded labels. The goal is then to construct a ranking function which can accurately estimates the score of a new user review. In this report, we propose a method for attacking this problem: (i) we embed the data into a feature space model, i.e., display each review and its target value by vector representation. (ii) learn the pattern between reviews and then give each review a real-value score. (iii) sort the collection of reviews based on their scores.

We focus on a learning to rank method called RankNet [4], which can be trained on a supervised signal (i.e., labeled data). Learning to rank algorithms can be divided into three categories: point-wise, pairwise and list-wise. Point-wise algorithms try to find the best estimator of the scores based on the individual samples as they do not consider the relative information between documents. On the other hand, pairwise algorithms use relative labels as their target. For example, if review $i$ is better than $j$ then the label correspond to the tuple $(i, j)$ is 1; otherwise, it is -1. RankNet uses a pairwise preferences to avoid inconsistency in the labels. In our work, we present a way of learning a model using weighted word counts (e.g., tf-idf) as the feature space. Then we use RankNet as a learner and we sort the reviews based on their scores. The difficulty here is that such feature space are very large. For example, extracting the tf-idf gives a feature vector of size 62,414.

1.1 Dataset Collection

We collected our dataset from the Yelp academic dataset [1] which contains about 229,907 user reviews of different businesses on Yelp. Each review in our dataset contains structured metadata (user ID, business ID, etc.) along with a text sequence representing the user review. We divided the reviews into five categories corresponding to the five level star ratings. As we mentioned, RankNet is trained on pairs of reviews with different labels. Therefore, we must be careful not to sample an imbalanced training data as it would results in a poor model. For example, consider we use pairs of two hundred five stars reviews and six one star reviews as our training data. The consequence of such a training data is that our model would be biased. Therefore, our solution is to train the model
with maximum entropy which means collect equal data from each category and then train the model
with the training set.

2 Preprocessing on Dataset

To extract a set of features, we used the collection of all reviews available in our dataset. We first
employed some preprocessing on our original data to make it more useful for machine learning
techniques. We used functions built in Natural Language Tool Kit (NLTK) [2] for processing of
the text reviews. We first tokenized all the corpus and then eliminated all the stop words using the
standard list of stop words (such as a, about, they, who, again, etc.). We further reduced the size of
our dictionary by using the following techniques.

2.1 Part of Speech Tagging

Since our aim is to extract features and opinions from text reviews, we assigned Parts-Of-Speech
(POS) tags to the words based on the context in which they appear. We then use POS information
to extract features that we are interested in. For example, adjectives represent opinions, and adverbs
use to represent the degree of opinions, etc. By doing so, we are effectively reducing the dimension
of our feature space by excluding non-important words from our feature vocabularies. Specifically,
we have used Penn Treebank POS tags which are the most common one used in the literature.

2.2 Stemming

Stemming is the process of reducing words to their stem based on their roots. We further applied
Porter Stemmer algorithm to our features as one high rank review may contain the word “perfect”
while another may contain the word “perfection”. This will reduce the size of our dictionary and
also will provide us with a better set of features.

3 Feature Extraction

Now, we have a corpus of reviews and a dictionary of our interested words. The next step is to
extract features form these two sets. We have used weighted word counts as the value for each word
in our dictionary. The term frequency-inverse document frequency (tf-idf) is a measure of showing
the importance of a word in a document, given a corpus of documents and is computed as

\[
    tf - idf(t, d, D) = tf(t, d) \times idf(t, D)
\]

where \(tf(t, d)\) is the term frequency which reflects the normalized number of words \(t\) in a document
\(d\). \(idf(t, D)\) is the inverse document frequency of the word \(t\) in the corpus of documents \(D\) and is
given by

\[
    idf(t, D) = \log \frac{|D|}{n_t}
\]

where \(n_t\) is the number of documents in the corpus containing word \(t\). The value of \(tf - idf\) for a
given word is high when the term frequency of it is high and the inverse document frequency of the
word is low. For example, if some word appears in almost all the documents in the corpus then the
ratio in log is near one and therefore the value of tf-idf would be almost zero. This means that this
word is not important as it appears in all the documents.

The difficulty here is that such feature vectors are very large and therefore we need to deal with the
problems in terms of memory and speed. Note that the matrix representing our data has a lot of
zeros in each row, i.e., each user review contain only a small subset of the words in the dictionary.
We tackled this problem by using a “hashing trick” implemented in Scikit-Learn package [3].

4 Problem Formulation

Let \(R = \{r_1, r_2, \ldots, r_n\}\) be the set of reviews and \(L = \{l_1, l_2, \ldots, l_n\}\) be the star ratings for the
reviews where \(l_i \in \{1, 2, \ldots, 5\}\). Each review is represented by a feature vector obtained from the
contents of the reviews. We want to learn a ranking function \( f \) such that it maps the feature vector \( x \in \mathbb{R}^D \) to the real number \( f(x) \). \( D \) is the dictionary size, and the \( j^{th} \) dimension of the vector space represents the tf-idf weight of the \( j^{th} \) word. Our aim is to minimize the difference between the real score of the review and the one obtained by the model. We employ pairwise ranking learning method RankNet that optimizes the objective function defined on pairs of reviews with different ratings. To this end, we first pick any two reviews with different labels as our training pair. Assume we picked two reviews \( r_i \) and \( r_j \) where \( r_i \) has a higher star rating than \( r_j \), and is denoted by \( r_i \succ r_j \). We present these reviews with feature vectors \( x_i \) and \( x_j \), respectively. Then, we feed this two vectors to our ranking function, which computes the scores \( s_i = f(x_i) \) and \( s_j = f(x_j) \). In this case, we define a cross entropy cost function on all pairs of reviews as

\[
C_T = \sum_{(i,j)} \log \frac{e^{s_i}}{e^{s_i} + e^{s_j}} = \sum_{(i,j)} \log \sigma(s_i - s_j) \tag{3}
\]

In the training, we feed the model with tuples \((r_i, r_j)\) such that \( r_i \succ r_j \). Specifically on Yelp, reviewers can give a business an integer score between one and five stars. However, there is an uncertainty over intermediate ratings (2,3,4) as customers are often not sure whether to give, for example, a restaurant three or four stars. Therefore, in our experiments, we only used tuples such that the positive review has a five star score and the negative review has a one star score, i.e., there is a four star difference between their labels. By doing this we are essentially reducing the noise in the training labels.

5 Neural Network Construction

We used a neural network with one hidden layer as our learning model. A typical neural net is shown in Fig. 1. Note that the number of neurons at input and hidden layers shown in the figure are not representative of the actual values. RankNet defines its cost function on pairs of inputs with different labels and therefore can provide a smooth approximation of the ranking cost function. RankNet was implemented initially by neural nets, but it can also be implemented by other models such as boosted trees or any other model that can make use of gradient descent methods for learning (in this work we use neural nets). RankNet training works as follows. At run time, feature vectors in the training data are fed forwarded and mapped to their scores. Then, the weights are updated according to back propagation on each pair of the features with different labels. The cost function is defined as a log-likelihood of the logistic function on each pair of outputs with different labels. As you can imagine, updating the weights on each pair of the outputs would slow down the training time of the model. Therefore, the authors in [4] proposed the following procedure to speed up the process. Denote the neural net function by \( f(x, \omega) \) where \( \omega \) is a matrix containing all the weights of the neural net (from input to hidden and hidden to output layer). Using gradient descent and (3) we can update the weights of the neural net as

\[
\Delta \omega = -\eta \frac{\partial C_T(\omega)}{\partial \omega} \tag{4}
\]

where \( \eta \) is the learning rate for gradient descent method. It is a parameter that determines how much an updating step influences the current value of the weights. Therefore, we need to be careful about this parameter as if we set it to a large value it will overshoot the minimum in our cost function. Finally, the derivative of the cost function with respect to the weights is

\[
\frac{\partial C_T(\omega)}{\partial \omega} = \sum_{k \in \text{training data}} t_k \frac{\partial f(x_k, \omega)}{\partial \omega} \tag{5}
\]

\[
t_k = \begin{cases} \sum_{j \in r^-} t_{k,j}, & \text{if } k \in r^+ \\ -\sum_{i \in r^+} t_{i,k}, & \text{if } k \in r^- \end{cases}
\]

with \( t_{i,j} = \sigma(f(x_i, \omega) - f(x_j, \omega)) - 1 \).

\( r^+ \) and \( r^- \) are the elements of each pair \((r^+, r^-)\), where \( r^+ \) has a higher star rating than \( r^- \). One can verify the derivation of the (5) by using the chain rule and considering the following equation for sigmoid function.
In this section, we describe the experiments that we carried out on Yelp dataset. The statistics of the dataset is shown in Table 1. We divided the data into three groups for training, validation, and testing. RankNet was trained using the training set, fine tuned using the validation set, and then tested on the test set. This is a general practice when training a multilayer network. The training set is used for computing the gradient and updating the network weights. Then the error on the validation set is monitored during the training process. The model that gives the best validation error is chosen for testing.

<table>
<thead>
<tr>
<th>Star rating</th>
<th>Number of reviews</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>17516</td>
</tr>
<tr>
<td>**</td>
<td>20957</td>
</tr>
<tr>
<td>***</td>
<td>35363</td>
</tr>
<tr>
<td>****</td>
<td>79878</td>
</tr>
<tr>
<td>*****</td>
<td>76193</td>
</tr>
</tbody>
</table>

Table 1: Statistics of the Yelp dataset

We used pair-wise correct as the performance measure of our ranking model. Pair-wise correct counts the number of tuples in a correct order out of maximum possible number of such pairs. For example, assume we have \( n \) tuples of the form \((r^+, r^-)\) (i.e., \( r^+ \) has a higher rating star than \( r^- \)). Then, the ranking model gets all the test reviews and produces an ordered list of the test reviews. Assume number of miss-ordered tuples in the output is \( m \), then the pair-wise correct is given by \( 1 - \frac{m}{n} \). The labels (target values) come with the noise as users usually tend to have uncertainty when it comes to give a rating to a business. Therefore, we are interested in using reviews with difference labels as high as possible. For example, in our training we took tuples in the form of \((*****, *, +)\). Thus, our training set consists of 17,516 samples of one and five stars reviews. We trained our neural net with the training set and saved the parameters that gave the smallest error on the validation set. We then shuffled the remaining dataset to pick samples for the test data. We computed the pair-wise
correct in a way that we do not consider the miss-ordering of the reviews with one star difference as an error. With this assumption, our model gives pair wise correct of about 90 percent for different cases examined. Note that by using more complex processing on the input data (text reviews) we would probably get the good results for pair-wise correct (where we also count the miss-ordering of reviews with one star difference).

Now, assume our goal is to predict the star rating of the user review. This task is straightforward now since we have a sorted list, i.e., reviews with higher star rating are set on the top of the reviews with lower ranks. Therefore, the problem of predicting the star reviews can be solved by only determining the threshold of the discrimination lines between different star ratings.

7 Conclusion

This report focuses on user review ranking. Our approach is a two-stage method. At the first stage, we extract review features from the collection of reviews. At the second stage, we employ our learning model to the input features and employ gradient descent to learn the parameters of the model. The result is a ranking function which ranks the reviews based on their contents.

References

[1] https://www.yelp.ca/dataset_challenge