Data Mining

Final Report

INFS4203/7203

Text Mining in Online Business Service

Team Members

Roshan Bagla	s4447418
Yukai Qiao	s4354963
Yangyang Xu	s4344240
Zafercan Aygurler	s4492528
Saisrikar Paruchuri	s4475157

THE UNIVERSITY OF QUEENSLAND

AUSTRALIA

Table of Contents

Problem Definition

We are interested in online peer-to-peer electronic business, both positive and negative of customer comments are valuable to analysis, and they could be the indicator of developing service quality, customer satisfaction and loyalty. Besides, thousands and millions of pieces of comments are required time and money. To reduce the cost of analysis, data mining applications are indispensable.

As we know, data mining is data driven solution and widely used in many industries.To recognise some patterns behind big data, there are massive algorithms of data mining that support auto-analysis. The positivity or negativity of a customer comment will be a binary-class of this project, data mining regards it as a classification problem. Once we triggered the concept of data mining, next thing is choosing the text mining as the data mining application.

The main task of this project is to classify the users given reviews on the websites like amazon, yelp and imdb. We are classifying the sentences into positive or negative. Each sentence has a sentiment value associated with it.

In this project, we worked on text mining by Python and used the approaches that provided by Kim's paper, which used the same database as us. According to his paper, we will use deep learning algorithms. There are also several software applications will help with. Tensorflow is an open-source deep learning library, it has full of CNN(Convolutional Neural Network) algorithms for Python. Keras 2.0 will be the Python API, used for Tensorflow. It's user-friendly, modulable, and extensible. Since the customer comments are semantic, and according to Kim's methods[1], Gensim helped transform the texts of a sentence into vector matrix, and it is adaptable to the input of word2vec algorithm, provided by Tensorflow.

Dataset

Our dataset consists of 3000 sentences from IMDB, Amazon and Yelp reviews and each source contributes 1000 sentences. Sentences are labelled as either positive or negative. Two classes are distributed equally.

Related Work

Experiments with Sentence Classification

Anthony Khoo, Yuval Marom and David Albrecht [1] explored the performance of Naive Bayes, Decision Tree and Support Vector Machine on a task of sentence classification. The dataset includes 1484 sentence from 160 email dialogues and the task is to classify each sentence to one of 14 classes.

The authors performed preprocessing to data includes stop-words removal tokenization and lemmatization. Each remaining token is then represented as word vector using Bag-of-Words.

The authors discovered that their dataset is skewed, therefore accuracy itself is not a sufficient metric and F1 score is used instead.

The results have shown the superiority of SVM over the other two machine learning algorithm.

Convolutional Neural Networks for Sentence Classification

Yoon Kim [2] presented an approach of sentence classification using convolutional neural network(CNN) on top of Word2Vec processed word vectors. The author believed that Word2Vec essentially works as a feature extractor on semantic features of words. And since words in sentence hold some temporal features which can be extracted using CNN the author believed that the architecture proposed should be a feasible approach on sentence classification.

The author used a public Word2Vec representation trained on 100 billion words on Google news. With vectorized words, sentences were represented in matrices. The author applied convolutional layers and pooling layers with different filter sizes and then concatenated the output. The output was then put into a fully connected layer to do the final classification.

The results indicate that this approach outperformed other state-of-the-art methods on 4 out of 7 competitive datasets.

Methodology

Word2Vec

There are two ways of implementing Word2Vec [3]: skip-gram model and negative sampling. Although it was using negative sampling in our implementation, we will explain how it works with skip-gram model, since the theory behind negative sampling is essentially skip-gram model.

Intuitively, Word2Vec generates word embeddings by achieving a ‘fake task’. In skip gram model, the fake task is to train an MLP to predict the probability distribution of a word’s context given the word. If such MLP achieved this task, it means the hidden layer contains sufficient information about how this specific word interacts with others and the hidden layer can work as the word’s embedding.

To give a more formal definition. We are given a corpus of words, w, and their contexts, c, within a window size. Given a corpus Text, the goal is to derive the parameters q to maximize the corpus probability [4]:

$$\arg\max_{\theta} \prod_{(w,c)\in D} p(c|w,\theta)$$

Convolutional Neural Network

Convolutional neural network(CNN) is a neural network architecture that expertise in extracting local features to recognize spatial or temporal objects [5]. A CNN classifier usually consists of convolution layers, pooling layers, and fully connected layers.

All neurons in a fully connected layer are connected to all input and output as normal neural network does. At each convolutional layer, input will go through “filters” with a shared weight similar to animal vision neural system. Then, a pooling layer will choose the most significant feature in a range.

CNN is usually used in image processing, since pixels hold spatial features which can be extracted by conv pool layers. We believed that words have similar features because words interact with each other and the temporal order of words can impact the meaning of a sentence significantly. This is why we chose to use this algorithm.

Project Tasks

The goal of this project is classifying the positivity or negativity of a comment from a CNN learner (model). An overview of whole process is illustrated in Fig.1. Assuming all comments are formatted in a bunch of sentences, since we will use supervised learning, a 0 or 1 label is marked behind each sentence (comment). The challenges are from two perspectives, one is the words chosen to clean are difficult to determine, because the comments are from three different websites which focus on different topics. Secondly, training a CNN model will come across overfitting, and altering hyperparameters can be difficult.

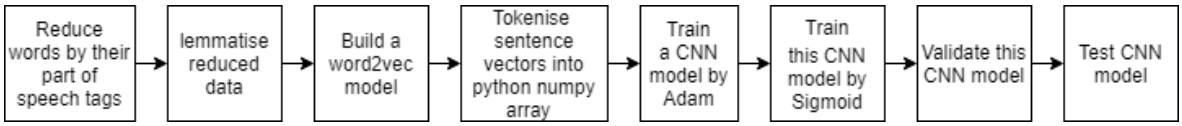

Figure 1

Preprocessing Data

At the beginning is to reduce words by tags, the tags will be identified by using the “pos_tag” library of the Natural Language Toolkit (NLTK). Since modal, adjectives and adverbs are powerful word to indicate opinions [6]. In this project, only singular proper noun, modal, adjectives, and adverbs left. To reduce the size of dictionary that used for transferring vectors, the vast forms of these part-of-speech words will be lemmatised by the corpus that provided by NLTK.

Because language is needed to be converted to vectors that machine can recognise, the semantic segmentation method will be used on the cleaned comments from above. “Word2vec” method that provided by Gensim will help with building a word2vec model, and a unique word dictionary will be yield from this model. For the word2vec model, it is trained by a neural network with a single hidden layer, the dimensionality of the feature vectors is 100, and maximum distance between the current and predicted word is 2. Since python library (Keras) of CNN only supports Numpy array (a homogeneous multidimensional array) as the input, then each

sentences will be reconstructed into vectors from the unique word dictionary, and each sentence will be tokenized into Numpy array.

Training Data

To train the CNN better, the model will be trained twice, both loss function is binary cross-entropy. Initially, “Adam”, an optimiser will be used. In this model, there are 5 layers. First one is a 1-dimensional convolutional layer, its output size is 256 dimensions, the length of convolution window is 5, activation function is the Rectified Linear Unit (ReLU), and it reshapes the input with same size. A global-max pooling layer will be after for the temporal data. The third layer and fourth layer are fully connected (dense) layer. Front dense layer has 256 dimensions, the penalisation rate is 0.0001, and the ReLU is also used as activation function. For Back dense layer, the output size is 1D and the activation function is Sigmoid. The second-time model uses Stochastic gradient descent (SGD) as the optimiser. Then the model will be validated. The data used for training, validating, and testing are randomly picked. The size will be 80%, 10%, and 10% for each of them.

Results

After doing the data pre-processing, the number of words in each sentence is reduced to at most 10.

Since the distribution of two classes in our dataset is symmetric, accuracy itself works as a good metric.

Our CNN model has the 77% validation accuracy (Fig. 2) and the 76% test accuracy. The receiver operating characteristic (ROC) curve (text color in orange) was drawn from the accuracy of each test data. The curve (Fig. 3) shows a fair accuracy, since the curve closes to top-left corner of whole axes, and the vertical axis indicates the true positive rate.

```
Epoch 200/200
 100/2399 [>.............................] - ETA: 0s - loss: 0.3452 - acc: 0.8700
 300/2399 [==>...........................] - ETA: 0s - loss: 0.3411 - acc: 0.8400
 500/2399 [=====>........................] - ETA: 0s - loss: 0.3306 - acc: 0.8560
 700/2399 [=======>......................] - ETA: 0s - loss: 0.3344 - acc: 0.8543
 900/2399 [==========>...................] - ETA: 0s - loss: 0.3426 - acc: 0.8456
1100/2399 [============>.................] - ETA: 0s - loss: 0.3474 - acc: 0.8473
1300/2399 [===============>..............] - ETA: 0s - loss: 0.3546 - acc: 0.8377
1500/2399 [=================>............] - ETA: 0s - loss: 0.3667 - acc: 0.8313
1700/2399 [====================>.........] - ETA: 0s - loss: 0.3681 - acc: 0.8324
1900/2399 [======================>.......] - ETA: 0s - loss: 0.3723 - acc: 0.8311
2100/2399 [=========================>....] - ETA: 0s - loss: 0.3693 - acc: 0.8310
2300/2399 [===========================>..] - ETA: 0s - loss: 0.3705 - acc: 0.8326
2399/2399 [==============================] - 0s - loss: 0.3714 - acc: 0.8299 - val_loss: 0.4752 - val_acc: 0.7700

Process finished with exit code 0
```

Figure 2

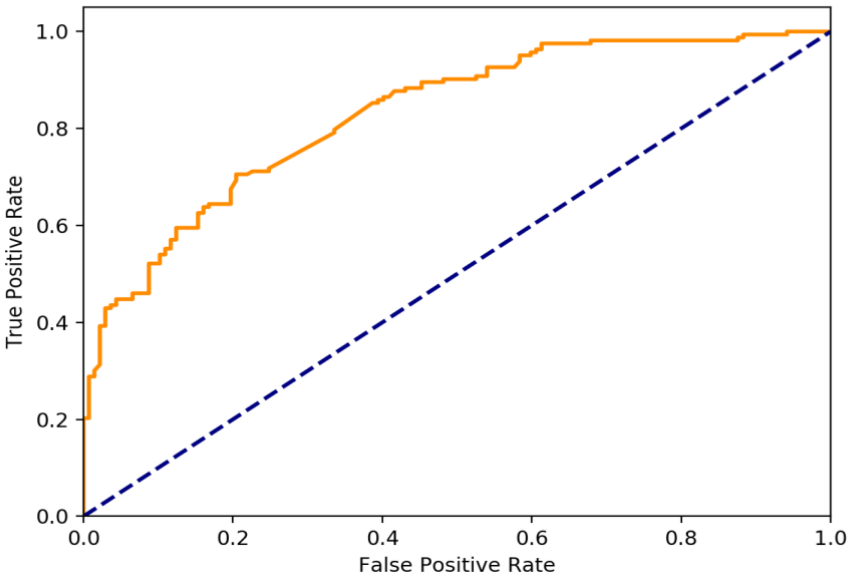

Figure 3

Discussion

The overall performance of our approach failed to match our main reference[1]. We believe this is caused by several reasons.

Instead of using filters with sizes varies from 2 to 5, we only used filters with size 2. This might cause our model to fail to capture more complex features which involve more than two words.

We trained our own Word2Vec representations using our training data. So the quality of our word embeddings might not match Yoon Kim's approach which used representations trained on 100 billion words on Google news.

During the training process we realized how different types of words can affect final result. We have found that modal, adjectives and adverbs contribute the most to our model's predicting power. Adjectives and adverbs contribute a large portion of sentiment in English. Modal words like 'does' and 'doesn't 'express positive or negative directly. So it was expected that they have big impact on prediction. However, we also found that verbs dramatically reduce the performance. After carefully looking at sentences in dataset, we can only guess that it's because ironic sentences tend to contain more verbs and they are really hard to classify. Thus, in the future, these issues will be considered and refine our python code. Hopefully, new strategies will improve current model and make it be efficient to analyse these customer comments and benefit economic use.

From this team project, except learning to implement text mining by several popular deep learning libraries and obtaining the experiences of altering the hyperparameters, we also learned to cooperate with other people. Since not

everyone is good at coding or text mining, but as a group, each person can still exert their strength.

References

[1] Kim, ‘Convolutional Neural Networks for Sentence Classification’, arXiv, 2014

[2] Khoo, Anthony, Yuval Marom, and David Albrecht. "Experiments with sentence classification." Proceedings of the 2006 Australasian language technology workshop. 2006.

[3] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations

in vector space,” arXiv preprint arXiv:1301.3781, 2013.

[4] Y. Goldberg and O. Levy, “word2vec explained: deriving mikolov et al.’s negativesampling word-embedding method,” arXiv preprint arXiv:1402.3722, 2014.

[5] Y. LeCun, Y. Bengio et al., “Convolutional networks for images, speech, and time series,” The handbook of brain theory and neural networks, vol. 3361, no. 10, p.1995, 1995.

[6] Neviarouskaya, A., Prendinger, H., & Ishizuka, M. (2011). Affect Analysis Model: Novel rule-based approach to affect sensing from text. Natural Language Engineering, 17(1), 95-135.

Appendix

Deep learning software library: TensorFlow 1.2

Website:https://www.tensorflow.org

Python Deep learning library: Keras 2.0

Website: https://keras.io

Fast Vector Space Modelling Python library: Gensim

Website: https://pypi.python.org/pypi/gensim

Data Sets Website:
http://archive.ics.uci.edu/ml/datasets/Sentiment+Labelled+Sentences?fref=gc