DATA MINING AND TEXT MINING (CS 583)

SENTIMENT ANALYSIS USING ASPECT TERM

RESEARCH PROJECT REPORT

DINESH ANAND SIVASUBRAMANIAN S
GAUTAM

UNIVERSITY OF ILLINOIS AT CHICAGO
SPRING-2018

Table of Contents

ABSTRACT

Sentiment analysis the process of computationally identifying and categorizing opinions expressed in a piece of text, especially in order to determine whether the writer's attitude towards a particular topic, product, etc., is positive, negative, or neutral. Sentiment analysis and opinion mining play an important role in judging and predicting people's views. Target-dependent sentiment classification remains a challenge. Target-dependent sentiment classification means modeling the semantic relatedness of a target with its context words in a sentence. In this research project, we discuss how to perform aspect term based sentiment analysis using classifiers like Bayesian and then using deep learning techniques.

1.INTRODUCTION

Most of our decisions in the real world are influenced by the thinking that, how other people would perceive their decision. With the popularity of internet, the big data explosion and the ability of the people to learn and exploit the web has made tremendous amount of information available, which can be used to objectively make important decisions. The change in the customer choice influenced by different opinions, has outsmarted the traditional opinion monitoring methods and these methods have been left behind because of the fast-growing information available through various media.

Sentiment analysis is a series of methods, techniques, and tools about detecting and extracting subjective information, such as opinion and attitudes, from language. Sentiment analysis has been about opinion polarities, whether someone has positive, neutral, or negative opinion towards something. Opinion can be viewed as triplet containing the target, the aspect term and the polarity (positive negative or neutral). Given a sentence and an aspect term mention, the task calls for inferring the sentiment polarity (e.g. positive, negative, neutral) of the sentence towards the target. For example, let us consider the sentence: “I bought a new IPhone. The touch screen is amazing but the battery life is too short”. If the aspect term is touch screen, the expected sentiment polarity is “positive” as the sentence expresses a positive opinion towards touch screen. If we consider the target as battery life, the correct sentiment polarity should be “negative”. The challenge in sentiment analysis using aspect term is how to effectively model the semantic relatedness of a target word with its context words in a sentence. Supervised classification algorithms need labelled data sets to train the model and test on unlabeled data. The accuracy is

calculated for three different classes and F-score is determined to evaluate the results.

2. DATA-PREPROCESSING

Data preprocessing is essential to build a classification model. The following are the preprocessing steps we performed:

2.1 Stop-words Removal

A stop word is a commonly used word (such as “the”, “a”, “an”, “in”) that a search engine has been programmed to ignore, both when indexing entries for searching and when retrieving them as the result of a search query. NLTK (Natural Language Toolkit) in python has a list of stop words stored in 16 different languages. It is found in the nltk_data directory. Other than these the special characters, punctuation marks are also removed.

2.2 Stemming

The goal of both stemming and lemmatization is to reduce inflectional forms and sometimes derivationally related forms of a word to a common base form. Stemming usually refers to a crude heuristic process that chops off the ends of words in the hope of achieving this goal correctly most of the time, and often includes the removal of derivational affixes. For example, if a word ends with a consonant other than s, followed by an s, then delete s, if a word ends in es, drop the s. Stemming reduces the feature space as many derived words are reduced to the same root form. Multiple features now point to the same word, thus increasing the probability of the word.

2.3 TF-IDF

In information retrieval, TF-IDF (term frequency–inverse document frequency) is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. It is often used as a weighting factor in searches of information retrieval, text mining, and user modeling. The tf-idf value increases proportionally to the number of times a word appears in the document and is offset by the frequency of the word in the corpus, which helps to adjust for the fact that some words appear more frequently in general.

For predictive modelling, the text must be parsed to remove words, called tokenization. Then the words need to be encoded as integers or floating point values

for use as input to a machine learning algorithm, called feature extraction (or vectorization). Tfidfvectorizer from the scikit-learn package is used for this. The TfidfVectorizer tokenizes documents, learns the vocabulary and inverse document frequency weightings, and allows us to encode new documents, which can then be fed as input to the ML algorithms.

3. CLASSIFIERS USED

1. Multinomial Naïve Bayesian - This is the classifier which we planned to implement at first. Chi square is a calculation used to determine how closely the observed data fit the expected data. We used Chi Square to select a subset of relevant terms to be used in the construction of Naïve Bayes model. We got a relatively low accuracy of 57%. Later we planned on using POS taggers. POS tagging is the process of marking up a word in a text (corpus) as corresponding to a particular part of speech based on both its definition and its context i.e., relationship between its adjacent words. The parts of speech are noun, verb, adjective, adverb, pronoun, preposition, conjunction and interjection. Since the aspect term is given, all forms of noun is ignored, therefore adjectives and adverbs play a key role. Using this as feature Multinomial Naïve Bayesian Classification was performed and accuracy was increased to 69%.

2. Support Vector Machine - Support Vector Machines is another popular classification technique. It constructs a hyper plane or set of hyper planes in a high- dimensional space such that the separation is maximum. The hyper plane identifies certain examples close to the plane, which are called as support vectors. SVM can be used by correct estimation of class weight and penalty. The class weight is calculated by the inverse of the number of class samples in the dataset. The penalty of each sample is then weighted by the class weight (approach by Yangyang Yu). “sklearn.svm.LinearSVC”.

3. Deep Learning method using LSTM – Deep learning is the application of artificial neural networks to learning tasks using networks of multiple layers. Deep learning uses a cascade of multiple layers of nonlinear processing units for feature extraction and transformation. The lower layers close to the data input learn simple features, while higher layers learn more complex features derived from lower layer features. Finally, a softmax layer outputs the

probability of classifying the sentence as positive, negative or neutral. Using Keras and Tensorflow, we implemented the deep learning algorithm and got an average accuracy of 93%.

4. OUR APPROACH

Tools used: Python3, Keras, Tensor flow

First, the review text and the aspect term were preprocessed by removing non-alphanumeric characters.

The Tokenizer class was used to tokenize the review text and the aspect terms.
This class allows to vectorize a text corpus, by turning each text into either a sequence of integers (each integer being the index of a token in a dictionary) or into a vector where the coefficient for each token could be binary, based on word count, based on tf-idf, etc.

The tokenizer was fit on the review text of the training data. The text_to_sequences() method was used to convert all the texts as well as the aspect terms to sequences. The pad_sequences() method was used to pad all the sequences to the same length. Next two LSTM neural networks were used, one for the review text and the other for the aspect term.

Each word in the review text or aspect term is represented as a low dimensional, continuous and real-valued vector, also known as word embedding. All the word vectors are stacked in a word embedding matrix. max_features, embed_dim, input_length and lstm_out are used as hyperparameters. The embedding layer encodes the input sequence into a sequence of dense vectors of dimension embed_dim. From the embedding layer, the input is passed through the LSTM layer with a dropout, then through the dense layer in order to get intermediate output. The LSTM transforms the vector sequence into a single vector of size lstm_out, containing information about the entire sequence. Dropout consists in randomly setting a fraction rate of input units to 0 at each update during training time, which helps prevent overfitting. Dense layer is a regular densely connected NN layer.

The intermediate outputs of both the LSTM networks are then concatenated. The concatenated output is then passed through two hidden layers using relu activation.

It is then passed through the output layer using sigmoid activation.
Finally, the model was fit on the training data and 20 epochs were used to get higher accuracy. An average accuracy of 93% was obtained on the training data. The model was then used to predict the class labels of the test data.

5. RESULTS

1. The results were calculated using dataset1 (given) and Naïve Bayesian Classifier and POS tagging. The accuracy was 71%. The results were estimated using 10-fold cross validation. The below figure represents the results.

	precision	recall	f1-score
-1	0.75	0.76	0.76
0	0.64	0.12	0.21
1	0.70	0.91	0.79
avg / total	0.71	0.72	0.68

Avg. Accuracy score is 0.7170953101361573

Figure 5.1 Naïve Bayesian Classifier

2. The training data and 20 epochs were used to get higher accuracy. An average accuracy of 93% was obtained on the training data. The below figure shows the result for dataset 1 and dataset 2.

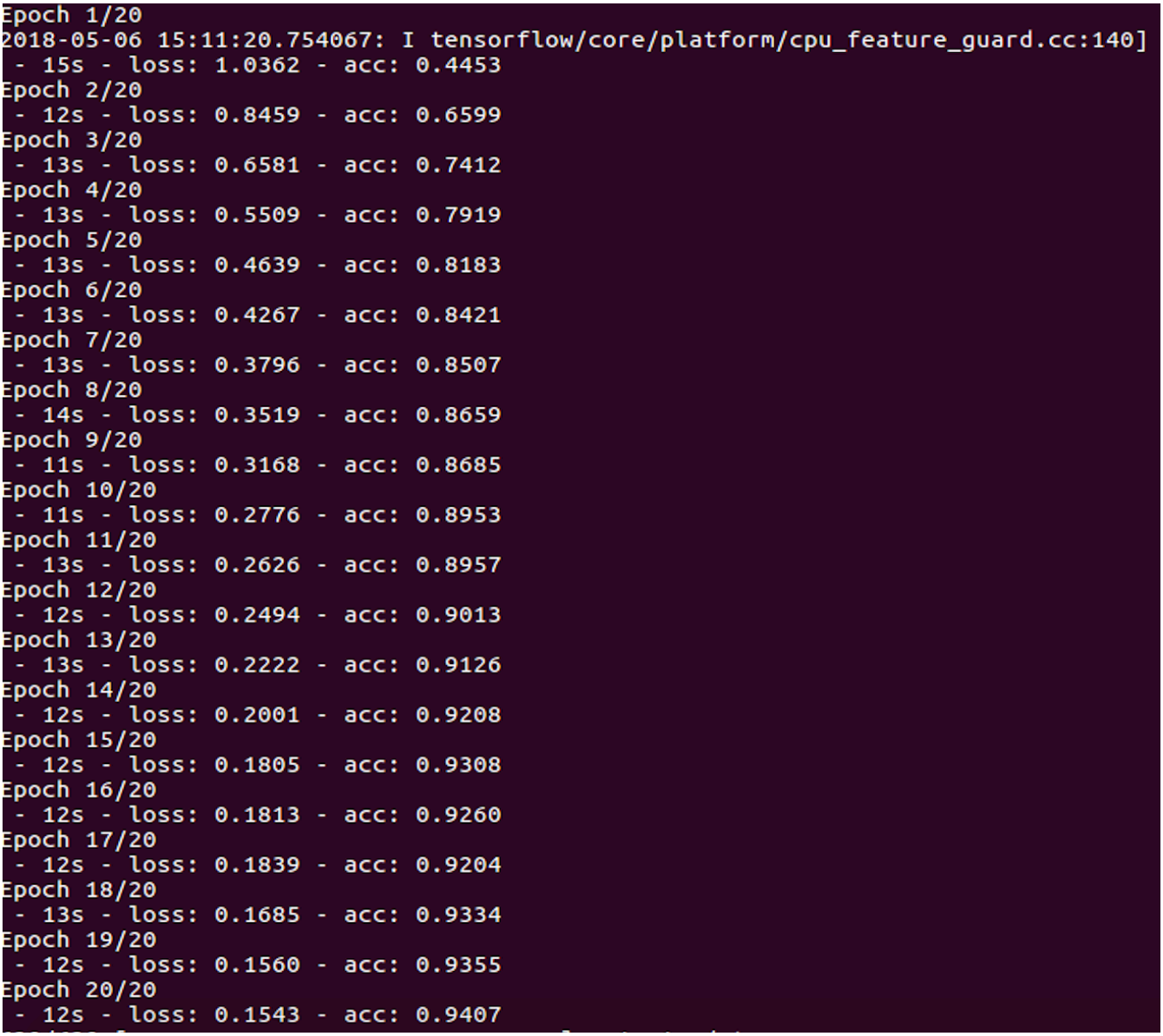

Figure 5.2 Using LSTM (Dataset 1)

```
Using TensorFlow backend.
Epoch 1/20
2018-05-06 14:55:00.307151: I tensorflow/core/platform/cpu_feature_guard.cc:140]
 - 20s - loss: 0.8908 - acc: 0.6144
Epoch 2/20
 - 18s - loss: 0.6953 - acc: 0.6957
Epoch 3/20
 - 19s - loss: 0.6073 - acc: 0.7451
Epoch 4/20
 - 23s - loss: 0.5612 - acc: 0.7629
Epoch 5/20
 - 24s - loss: 0.5176 - acc: 0.7807
Epoch 6/20
 - 25s - loss: 0.4819 - acc: 0.7971
Epoch 7/20
 - 24s - loss: 0.4476 - acc: 0.8137
Epoch 8/20
 - 25s - loss: 0.4028 - acc: 0.8370
Epoch 9/20
 - 26s - loss: 0.3697 - acc: 0.8509
Epoch 10/20
 - 26s - loss: 0.3495 - acc: 0.8615
Epoch 11/20
 - 25s - loss: 0.3090 - acc: 0.8776
Epoch 12/20
 - 24s - loss: 0.3028 - acc: 0.8817
Epoch 13/20
 - 25s - loss: 0.2826 - acc: 0.8884
Epoch 14/20
 - 25s - loss: 0.2527 - acc: 0.9034
Epoch 15/20
 - 26s - loss: 0.2397 - acc: 0.9062
Epoch 16/20
 - 21s - loss: 0.2348 - acc: 0.9062
Epoch 17/20
 - 19s - loss: 0.2211 - acc: 0.9112
Epoch 18/20
 - 19s - loss: 0.2050 - acc: 0.9187
Epoch 19/20
 - 19s - loss: 0.1996 - acc: 0.9214
Epoch 20/20
 - 19s - loss: 0.1905 - acc: 0.9248
```

Figure 5.3 Using LSTM (Dataset 2)

6. CONCLUSION

We tried various classifiers for aspect-based sentiment analysis. To achieve the best possible results the proposed methods actively use notion of semantic similarity between words, statistical measures and hand-crafted rules. We started with normal text classification using Naïve Bayesian and then used POS taggers to effectively capture various parts of speech. Since the accuracy was still low we moved to deep learning methods. We used LSTM and we were able to achieve high accuracy (up to 93%) using deep learning techniques. The major challenge was to deal with complex sentences with multiple aspect terms and domain specificity of the sentiments. In future, we would like to try dependency tree parsing to find the words that have effect on the sentiments of the opinion.

7. REFERENCES

1. "Semantic Similarity for aspect based Sentiment Analysis" http://www.dialog-21.ru/media/1111/blinovpdkotelnikovev.pdf
2. "Effective LSTMs for Target-Dependent Sentiment Classification" https://www.aclweb.org/anthology/C/C16/C16-1311.pdf
3. "Sentiment Analysis of Movie Reviews using POS tags and TF" https://pdfs.semanticscholar.org/d7b3/c50daf478667e22bf2d2f8a76df386586757.pdf
4. "Web Data mining by Bing Liu" https://www.cs.uic.edu/~liub/WebMiningBook.html
5. http://scikit-learn.org/stable/modules/naive_bayes.html (Sckit-learn)
6. https://keras.io/ (keras)
7. https://www.tensorflow.org/api_docs/python/ (Tensor flow)