Aspect Based Sentiment Analysis

Shreyash Bali and **Arijit Chakraborty**
University of Illinois at Chicago
sbali3@uic.edu and achakr24@uic.edu

Abstract

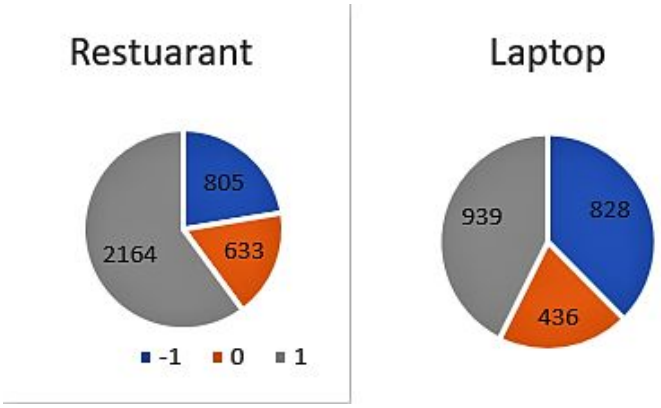
This paper discusses the aspect based sentiment analysis on Laptop reviews and Restaurant reviews dataset. Several methods were implemented to assess the ability to classify the sentiment of a review based on the aspect term associated with positive, negative and neutral class and further predict the correct sentiment. We implemented a sentence level sentiment classification model and various aspect based models to classify sentiments. The results showed that the proposed ensemble model performed better on the Laptop reviews while the recurrent neural network model performs well on the restaurant reviews data.

1 Introduction

Sentiment analysis, also known as opinion mining (Pang and Lee, 2008; Liu, 2012), is an import task in natural language processing as it helps analyze the user generated reviews on social media and product websites. Sentiment analysis in general gives an overall idea about the user's emotions but sometimes the user can be referring to more than two aspects in his review and thus an overall sentiment is not helpful. For example: "The food was good but the service was poor". Here the general sentiment analysis would return an overall neutral sentiment. However, with respect to food the sentiment is positive while with respect to service its negative. In this project we are focusing on aspect based sentiment analysis. Aspect based sentiment analysis is hard to achieve. We propose two models. One which used dependency parsing and distance of each word in the sentence with respect to the aspect term and the other, an LSTM model with the distance of words from aspect term being considered to update weights.

2 Dataset

There were 2 datasets used in this project. One was a Laptop review data set and the other was a Restaurant review dataset. The Laptop review training dataset has 2203 rows while the test dataset has 638 rows. The Restaurant review training dataset has 3602 rows and the test dataset has 1120 rows. The datasets contain the following fields - review_id, review, aspect term, aspect position and polarity. The polarity classes are +1, 0, -1.



3 Preprocessing

Stop Word and Punctuation removal

The main aim of stop words and punctuation removal is to decrease the vocabulary size and feature space as they do not contribute to anything and hinder the learning process. Removal of stop words decreases the vocabulary size on average by 30%. A list of all the stop words and punctuations which do not contribute to the sentiment of a sentence was created. And all these words are churned out from the review texts. A few spelling mistakes were corrected. The nltk

stopwords list was used as a reference to create the modified stopwords list.

Tokenization and POS Tagging

Tokenization is the process of demarcating and possibly classifying sections of a string of input characters. The resulting tokens are then further passed onto the next stage. Word-Tokenizer module provided by NLTK is used for tokenization. Also POS-Tagging module provided by NLTK was used for parts of speech tagging.

Vectorization

Bag-of-words model is a simplifying representation of text as a multiset of its own words, disregarding grammar and even word order but keeping the multiplicity. The output instances of the preprocessed data were further fed to a Bag of Words (BoW) model.
Term frequency inverse document frequency (TF-IDF) evaluates how important a word is to a collection or a corpus. TF-IDF of words from summaries were computed.

$$tfidf(w_k, d_i) = w_{ki} * log \frac{m}{\sum_{d=1}^{m} 1\{w_{kd} > 0\}}$$

where W_{ki} is the frequency of the k-th word in the i-th book's synopsis (d_i), and m is the number of training/test sets. The intuition behind calculating the TF-IDF vectors for the words was that words which have been repeated in the reviews were closely linked to the aspect and thus further help in the prediction. CountVectorizer method was tried too although TF-IDF Vectorizer performed significantly better.

Word2Vec Embedding

Pre-trained Twitter Glove embedding data(2B tweets, 27B tokens, 1.2M vocab, uncased, 25d, 50d, 100d, & 200d vectors) was used to create the word to vector lookup table for vectorization of the review text to create embeddings for the embedding layer. A maximum length of 100 words for each review is considered and all the sentences with lesser words are padded. The matrix values for each word is replaced with the vector values obtained from the lookup table.

Three words on either side of the aspect term are considered and their weight vectors added some importance to them as compared to all the other words.

4 Classification models

The following classification models were tried out for sentence level and aspect level sentiment classification.

4.1 Sentence Level Model

In the initial phase of this project, generic sentiment classification of the text sentences (without giving weights/relevance to the aspect term) using Random Forest, SVM, and Naive Bayes classification algorithms was carried out.

Each sentence from the supplied in each of the 2 dataset was taken and certain sanity checks and clean ups like removal of quotes, "[comma]" words, punctuations, corrections of mispelt words, removal of stopwords etc was performed. The next step involved vectorization of the entire set of sentences in each dataset using the TF-IDF Vectorizer. Each of the transformed dataset was then fed into Random Forest, Naive Bayes and Linear SVM model generators and 10-fold cross validation was carried out. Out of the above 3 models, SVM performed the best. For the laptop review dataset, it achieved an overall accuracy of 75.07% and F-Score of 80.7%, 52.25% and 78.11% for the positive, neutral and negative classes, respectively.

4.2 Distance Model

In the second phase, the aspect term was considered and couple of different approaches were tried out. The aspect term was also sanitized like the general sentences. The very first approach was distance based weight distribution. Words closer to the aspect term were given higher weightage. A 5-word window on either side of the aspect term was considered, with word nearest to the aspect term receiving the highest weightage (5 times higher weightage) compared to the word that's separated by 4 words from the aspect term. The aspect term itself was given the highest

weightage. This weighted sentences of each dataset were then vectorized using TF-IDF vectorizer. Post this, each of the transformed dataset were fed in to Naive Bayes and SVM model generators. SVM was found to perform better than Naive Bayes and for the laptop review dataset, it was able to achieve an overall accuracy of 66.64% with F-Scores of 76.31%, 39.68% and 67.33% for the positive, neutral and negative class respectively.

4.3 Dependency Parser model

The other approach that was tried out made use of dependency parsing. Each of the sentences was taken in its original form and only very basic level of sanity checking (punctuation and "[comma]" word removal) was carried out. These sentences were then passed to Stanford University's CoreNLP[6] dependency parser and a dependency model graph was obtained for each of them. Words that were directly related to the aspect term were assigned higher weights and the aspect term itself was assigned the highest weight. Stopwords were then removed from the sentences and the weighted sentences were fed into Naive Bayes and SVM model generators after vectorizing the datasets. Once again, SVM was found to perform better. For the laptop review dataset, it was able to achieve an overall accuracy of 70.77% with F-Scores of 78.52%, 48.57% and 72.03% for the positive, neutral and negative class respectively.

4.4 Ensemble Model (Dependency Parser and Distance Weights)

In the final phase, the above 2 approaches of distance weight and dependency parsing were combined. This was done in the following manner. Each sentence was taken and after performing very basic sanity checks, was fed into Stanford University's CoreNLP dependency parser. This time, however, words that were found to be directly as well indirectly (related to aspect term by another word) related to the aspect term were considered. For eg, if ***a*** is related to ***b,*** and ***b*** is related to ***c***, where ***c*** is the aspect term, then both ***a*** and ***b*** were considered. Directly related words were given twice the weight of indirectly related words. After this, stop words were removed from the sentence and a 5-word window was considered as described before and weights were assigned once more. The resulting sentence now had weights from both distance weight and dependency parsing approach. The aspect term was then given the highest weight like before. These sentence sets were then fed into TF-IDF Vectorizer and subsequently to SVM and Naive Bayes model generators. Once again SVM performed better with an overall accuracy of 74.58% and F-Scores of 81.08%, 56.5% and 75.93% for the positive, neutral and negative classes respectively.

4.5 Recurrent Neural Network model

A Long Short Term Network with distance weight updating model was implemented for the task.

Long Short-Term Memory(LSTM)
LSTM layer combats the vanishing gradient problem by using its gates which are used to remember the values of the gradient. The LSTM cells were set to 100 and a dropout of 0.3 was used for the model.

Dense(DEN)
The dense or Fully Connected layer is used to perform classification on the features extracted from the previous layers. In a dense layer, every node in the layer is connected to every node in the preceding layer. Usually, the final layer in a neural network is a dense layer.

Pre-Conditions
As this is multi-class classification certain parameters had to be set. The loss function had to be set to categorical cross entropy. The activation function on the final layer was set to softmax. Adam optimizer was used as the optimizer algorithm.

Architecture for the RNN
For the Recurrent neural network which usually involves many layers, the architecture is:

LSTM(dim) → DENSE → Activation → Output

Preprocessing steps like stop words and punctuation removal, converting sentences to lowercase and lemmatization were performed on

the data. Keras tokenizer was used for tokenization. Maximum length of the sentence was fixed to 100 and shorter sentences were passed using keras padding. Tokenized data was passed to create the word to vector lookup table using the pre-trained twitter glove embedding data we have. Depending on the position of a word with respect to the aspect term in the sentence the weights of the embeddings are updated with a relation as closer words get higher weights than farther words. Three words on either side of the aspect term are considered. A sequential model with one LSTM layer, one dense layer and an activation layer using keras layers. The model is fit on the training data and trained for 50 epochs before predicting the labels on the test data. Class-wise accuracy, Precision, Recall and F1 score are the metrics used for evaluation.

5 Evaluation measures

Precision:
It can be defined as the fraction of the retrieved documents that are relevant to the query.

$$\text{Precision(h)} = \frac{1}{N} \sum_{i=1}^{N} \frac{|h(xi) \cap yi|}{|yi|}$$

Recall:
It can be defined as the fraction of relevant documents that are successfully retrieved.

$$\text{Recall(h)} = \frac{1}{N} \sum_{i=1}^{N} \frac{|h(xi) \cap yi|}{|h(xi)|}$$

F1 Score:
It is defined as the harmonic mean of precision and recall.

$$F_1 = \frac{1}{N} \sum_{i=1}^{N} \frac{2 \times |h(xi) \cap yi|}{|h(xi)| + |yi|}$$

Accuracy:
It is the defined as the mean of all the comparisons between the predicted and the true labels

accuracy = mean(prediction == true_label) / 100

6 Results

The evaluation measures include precision, recall, f1-score and accuracy. The results are tabulated below. the results are evaluated using 10-fold cross validation for the baseline, distance weights and the dependency models. For the recurrent neural network model, the dataset is split into 80% for training and 20% for testing. Packages used include scikit learn, Stanford Core NLP parser, NLTK, Keras and Tensorflow. The environment used is Anaconda and Python 3.6.
It can be observed from the results that the models do significantly well with respect to the positive class as compared to the negative and the neutral class. One reason can be because of the skewed distribution of the classes in both the datasets.
The ensemble model achieves an overall accuracy of 74.58% with F-Scores of 81.08%, 56.5% and 75.93% for the positive, neutral and negative classes respectively for the Laptop Reviews dataset whereas the LSTM model achieves an overall accuracy of 73.8% with F-Scores of 81.08%, 56.5% and 58% for the positive, neutral and negative classes respectively. The results are as tabulated below.

7 Conclusion and Future work

The dependency parser with distance weights model performed better on the Laptop Reviews dataset whereas the LSTM with distance weights model performed significantly better on the Laptop Reviews dataset. This can be because of the difference in the size of the dataset and the sparse distribution of the sentiments. Larger dataset might help the LSTM models to learn the minority classes better and hence perform better. In our future work, we would like to improve our deep neural network by incorporating some more layers and tune the hyperparameters. We would also like to implement an attention-based recurrent neural network and compare the results with our model.

Classifier	Accuracy	Class = +1			Class = 0			Class = -1		
		P	R	F1	P	R	F1	P	R	F1
Sentence Level Model	75.07	80.08	81.36	80.07	68.01	42.43	52.25	72.15	85.14	78.11
Distance Weight model	66.64	75.12	77.52	76.31	46.87	34.4	39.68	64.54	71.25	67.33
Dependency Parser model	70.77	76.72	80.4	78.52	55.98	42.88	48.57	69.71	74.51	72.03
Ensemble model	**74.58**	80.69	81.46	**81.08**	61.45	52.29	**56.5**	73.52	78.50	**75.93**
LSTM model	73.48	81.64	76.89	78.4	55.43	61.89	57.81	71.26	70.97	72.01

Table 1. Accuracy Precision Recall and F1 Scores for Laptop Reviews

Classifier	Accuracy	Class = +1			Class = 0			Class = -1		
		P	R	F1	P	R	F1	P	R	F1
Sentence Level Model	63.13	65.71	93.90	77.32	37.77	10.74	16.72	52.72	21.61	30.66
Distance Weight model	67.48	75.97	84.61	80.06	44.55	36.17	39.93	54.71	46.08	50.03
Dependency Parser model	67.4	74.91	86.78	80.41	43.8	34.59	38.65	55.96	41.36	47.57
Ensemble model	68.01	75.72	87.06	80.99	43.04	30.8	35.9	56.12	46.08	50.61
LSTM model	**74.5**	87.02	81	**84.56**	51.87	55.24	**53.64**	52.18	62.87	**57.13**

Table 2. Accuracy Precision Recall and F1 Scores for Restaurant Reviews

Acknowledgement

We thank Prof. Bing Liu for encouraging us to take up a difficult problem. We would also like to thank TA Sahisnu Mazumder for assisting us with difficulties and suggesting improvements.

REFERENCES

[1] B. Liu. "Sentiment Analysis and Opinion Mining." Morgan & Claypool Publishers, May 2012.

[2] Pedregosa, Fabian, et al. "Scikit-learn: Machine learning in Python." The Journal of Machine Learning Research 12 (2011): 2825-2830.

[3] Keras Implementation URL:https://keras.io/

[4] TensorFLow
URL: https://www.tensorflow.org/

[4] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed Representations of Words and Phrases and their Compositionality.

[5] Natural Language Toolkit
URL: http://www. nltk.org

[6] Stanford CoreNLP
URL: https://stanfordnlp.github.io/CoreNLP/