Product Aspect Extraction for Sentiment Analysis without using Parsers

Narendra Roy and Samik Some

Under the guidance of Prof. Amitabha Mukherjee

E-mail: {amit,nroy,samiksom}@iitk.ac.in

Abstract

Aspect based sentiment analysis is an important task in gauging product popularity. The first step of any such algorithm is aspect extraction. Present aspect extraction algorithms are based on pre-processing the input via some kind of parser, such as POS-tagger or SRL-parser. We propose a method for aspect tagging without the use of any such parsers, for sentiment analysis. Such a methodology allows extension of the algorithm to any language, even those for which parsers are not available.

Introduction

Sentiment analysis of product reviews is an important task in opinion mining and gauging the popularity/usefulness of a certain product. However, coarse sentiment analysis such as a positive/negative sentiment for a whole review is somewhat misleading since a particular review may contain both positive and negative aspects. Aspect based sentiment analysis is much more useful in this case.

The first step in any aspect based sentiment analysis task is extracting the aspects from the

reviews. Current solutions for this problem use some form of parser such as POS tagger before extracting the aspects. This inherently creates language related dependencies, and as such these methods have limited extensibility.

We propose a new method for aspect extraction which does not depend on any kind of parser. Its an end-to-end solution based on LSTM networks. This method is advantageous in the sense that it doesn't depend on any particular language structure, and reduces the problem to a binary classification problem.

Related Works

There are a lot of aspect extraction that has already been done and is currently going on, out of which some of the remarkable ones are the following, which uses parsers :

- "Mining and summarizing customer reviews" [1]

  One of the earlier papers to use aspect extraction, it is based on POS-tagging the reviews and then extracting features from them.

- "A Rule-Based Approach to Aspect Extraction from Product Reviews, by Poria, Cambria, Wei Ku et al" [2]

  A novel rule-based approach that exploits common-sense knowledge and sentence dependency trees to detect both explicit and implicit aspects. Two popular review datasets were used for evaluating the system against state-of-the-art aspect extraction techniques, obtaining higher detection accuracy for both datasets.

- "Aspect Term Extraction for Sentiment Analysis: New Datasets, New Evaluation Measures and an Improved Unsupervised Method, by Pavlopoulos & Androutsopoulos" [3]

  They used and compared 4 methods for aspect extraction all of which use parsers. The baseline Frequency method, Hu and Liu's method, and both of these extended with Word2Vec pruning. Among these Hu and Liu extended with Word2Vec pruning was found to have the best precision.

Proposed Algorithm

Our algorithm is based on Zhou and Xu's algorithm for Semantic Role Labelling without using any parsers.[4]

We pose the aspect tagging problem as a binary classification task in which each word is either an aspect or a non-aspect. We utilise a stacked LSTM structure as shown in Fig 1.

- The input to the first LSTM layer is a sequence of words in a phrase level structure obtained by preprocessing the input. The words are represented via some word-embedding scheme.
- The next LSTM layer takes the sequence output by previous LSTM layer as input, and processes this sequence in the forward direction.
- The final LSTM layer output is fed to a Dense layer (fully-connected layer) to compress the input to a single dimension followed by a Sigmoid activation.
- The final output is thresholded at 0.5 to get either a 1 or 0, which classifies the word as an aspect or non-aspect, respectively.

The algorithm can possibly work better with bi-directional LSTMs or even by simply reversing the input sequence order for the second LSTM layer. However, we have not been able to test these structures due to lack of appropriate amount of training data.

Implementational Details

We implemented the LSTM network in python using the Keras framework.[5]

For word embedding we used python port of Word2Vec (gensim) and used the pre-trained GoogleNews model available here,[6] which provides 300 dimensional word vectors.

Two LSTM layers were used in the model each with 300 dimensional input and output, with sigmoid and hard sigmoid activation functions.

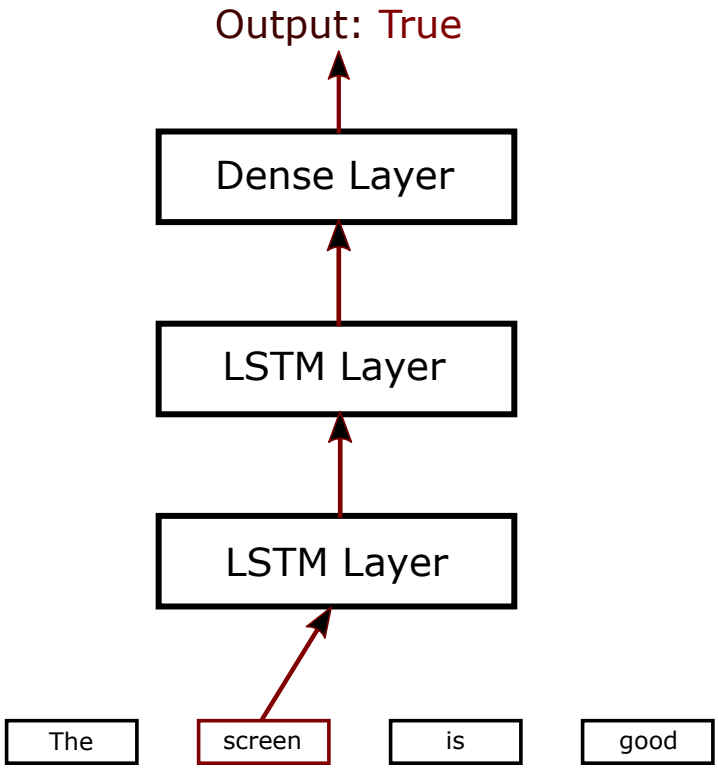

Figure 1: Structure of neural network used

The Dense layer took the 300 dimensional input (output from last LSTM layer) and gave a single dimensional output.

Training and Datasets used

For training and testing our approach we use the Amazon reviews dataset.[7] The dataset consists of reviews of five products. We use three of these, Nokia 6610, Nikon coolpix 4300 and Canon G3 for training our network. The other two are used for testing.

We preprocess the training data by splitting reviews at full-stops, commas, and whenever a aspect word is followed by 10 non-aspect words. These splits give us phrase level sequences over which we train our network. After preprocessing the data consists of only 598 samples. The network is trained for 2000 epochs. The large number of epochs were primarily required since the test set is extremely small.

We do not train Word2Vec for obtaining the word embeddings. Instead we use the pre-

trained GoogleNews model publicly available.[6]

For testing we preprocess the data by splitting it at full-stops, commas and whenever it exceeds 15 words, and pass this data to the model for prediction.

Experiments

Evaluation Measures

The method is expected to return a set A of distinct aspect terms, to be compared to the set G of distinct aspect terms the human annotators identified in the texts. TP (true positives) is $|A \cap G|$, FP (false positives) is $|A \backslash G|$, FN (false negatives) is $|G \backslash A|$, and Precision (P), recall (R), $F = (2 * P * R)/(P + R)$ are defined as usually: $P = TP/(TP + FP)$ and $R = TP/(TP + FN)$

Evaluation Results

The results obtained by our algorithm are shown in Table 1.

Product	Precision	Recall	F1-Score	Accuracy
Zen 40GB	29.30	55.57	38.37	97.22
Apex DVD	31.75	43.38	36.66	97.43

Table 1: Results

Comparison of the results[2] of a few papers mentioned earlier are as shown in Table 2.

Method	Precision	Recall
	Zen 40GB	
Hu and Liu	72.00	76.00
Poria et al.	92.25	94.15
	Apex DVD	
Hu and Liu	75.00	82.00
Poria et al.	89.25	91.25

Table 2: Results

Conclusion and Future Work

Although our results are not really up to the mark, we believe that they will get better if we a have a sizeable training dataset and if we use Bi-LSTMs or reverse the input of the second LSTM layer. We conclude that this may be a viable method for end-to-end aspect extraction given proper training set.

Also we should take into consideration that this can be extended to any language since it does not use any parsers.

The immediate future goal is to use a larger training dataset and Bi-LSTMS to get improved results and then test it on some languages for which parsers are not available.

References

(1) Hu, M.; Liu, B. Mining and summarizing customer reviews. Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining. 2004; pp 168–177.

(2) Poria, S.; Cambria, E.; Ku, L.-W.; Gui, C.; Gelbukh, A. A rule-based approach to aspect extraction from product reviews. Proceedings of the Second Workshop on Natural Language Processing for Social Media (SocialNLP). 2014; pp 28–37.

(3) Pavlopoulos, J.; Androutsopoulos, I. *Proceedings of LASMEACL* **2014**, 44–52.

(4) Zhou, J.; Xu, W. *Proc. 53rd ACL* **2015**, 1127–1137.

(5) Keras framework. `https://github.com/fchollet/keras`.

(6) Word2Vec. `https://code.google.com/p/word2vec/`.

(7) Amazon product reviews dataset. `http://curtis.ml.cmu.edu/w/courses/index.php/Amazon_product_reviews_dataset`.