Actor Sentiment Analysis

Amartya Chakraborty
661188393
chakra2@rpi.edu

Tyler Shepherd
661148286
shepht2@rpi.edu

Abstract

Sentiment analysis is a vital component of text analysis, especially for reviews such as movie reviews. However, most methods have been focused on analyzing the sentiment of the entire movie review, which while useful does not accurately convey the sentiment towards fine-grained aspects of the review, such as the actors performances. As such, we develop a tool which can predict the reviewers sentiment towards each actor mentioned in the review. We implement and compare four different models: Lexicon Sentiment Analysis, Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN) and Attention-based Long Short-Term Memory (LSTM) models. The results show that the Attention-based LSTM model yielded the highest accuracy among the four chosen methods.

1 Introduction

Sentiment analysis is an increasingly studied topic in Natural Language Processing and is being used in many real-world applications (Nasukawa and Yi, 2003). The idea is that, given some user text, the computer can determine the polarity of the opinion expressed - that is, whether the opinion is positive or negative. This is incredibly useful in many applications, like opinion mining of vast social media networks (Bannister, 2016). It is also very useful for reviews: a company often wants to understand their customer's opinions to a better extent from the text itself than a star rating system can provide.

In the world of film and Hollywood, reviews are essential. Studios are spending hundreds of millions of dollars on blockbuster films and they need to carefully choose leading actors to star (Mueller, 2011). The actors themselves, and their agents, want the studios and the audience to rate the actor by their talents, and not by the quality of the whole film. A good actor can give a great performance in a poorly-reviewed movie, after all, and the actor will not want their public opinion to be tarnished due to aspects of film-making outside of their control. As such, it is to the benefit of all parties involved to introduce a tool that can find the sentiment held by reviewers towards a specific target actor in a film, regardless of the overall quality of the film.

Take, for example, the sentence "In the end, 'The Social Network' is a mess of a movie, but Jesse Eisenberg shines in the lead role." If we focus our analysis on the target of "Jesse Eisenberg," the sentiment should be positive, even though this reviewer had a negative opinion of the film overall. Fine-grained, target-dependent sentiment analysis is an important task.

1.1 Our Contributions

We introduce in this paper a tool to perform target-dependent sentiment classification. It finds all actors mentioned in a review and classifies the reviewers opinion towards each actor (positive, negative or neutral). To perform this task, we attempt different sentiment analysis techniques. We use sentiment analysis models based on lexicon look-up, convolutional and recurrent deep neural networks, and an attention-based LSTM (long short-term memory) model. We examine the results of each on test data and analyze errors and issues.

2 Related Work

Sentiment classification is a deep field with many related works. A paper by Pang, Lee and Vaithyanathan in 2002 attempted the issue of classifying movie reviews as positive or negative at

a document level (Pang et al., 2002). They used different machine learning techniques and showed the initial difficulty of the problem. Further work by Pang and Lee provided more detailed star rating predictions, rather than just a binary classifier (Pang and Lee, 2005).

A work from 2007 by Snyder and Barzilay addressed the problem of aspect targeting in restaurant reviews (Snyder and Barzilay, 2007). Similar to our problem of determining the opinion towards actors, Snyder and Barzilay predicted the opinion toward aspect targets such as food and service. They presented the "Good Grief Algorithm," which scores each aspect based on dependencies and agreement between the aspects. Further, Jiang et al. argued that up to 40% of sentiment classification errors are caused by not considering targets (Jiang et al., 2011).

More recent works tend to focus on the emerging field of deep neural networks for target-dependent aspect-level sentiment classification. Yoon Kim presented a convolutional neural network implementation for sentence classification in 2014 that achieved impressive results with little parameter tuning (Kim, 2014). The paper also showed the importance of pre-computed word embeddings and using tools such as word2vec for these tasks, especially with tuning for the specific task. A paper from 2016 by Wang, Cao and others presented another convolutional neural network for the purpose of relation classification and determining dependencies, (Wang et al., 2016a) a useful tool in sentiment analysis. Mikolov et al. presented a recurrent neural network implementation for use in various NLP tasks (Mikolov et al., 2010). LSTM (long short-term memory) models have shown promising results as well, as can be seen in Tang et al.'s Target-Dependent LSTM model (TD-LSTM) (Tang et al., 2015).

Attention mechanisms have also received much recent focus in the field. For the purpose of aspect-level sentiment classification, determining attention focus has been shown to be beneficial. An implementation of an attention-based LSTM by Wang et al. achieved up to 83.1% accuracy on restaurant data using positive/negative/neutral classifiers (Wang et al., 2016b).

3 Methodology

Figure 1 shows an overview of the system architecture. Given an arbitrary review textfile, a named entity recognizer is first run to find all entities with class PERSON. We use the ELISA Information Extraction API for this purpose (RPI). Next, the found named entities are fed into the IMDbPY API, which uses data from the Internet Movie Database (IMDb) (Malagoli). The purpose of this is to guarantee that the named entities found are actually actors and not, for example, characters in the movie.

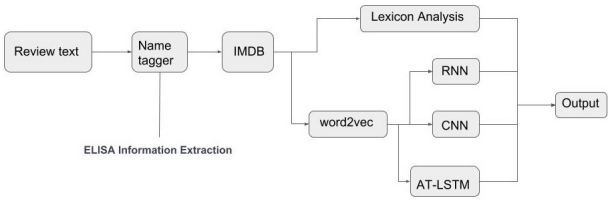

Figure 1: System Architecture Overview

From there, the sentiment analysis method specified by the user is ran. Word embeddings are needed for the RNN, CNN, and AT-LSTM, which are created using word2vec with 100 dimensions (Mikolov et al., 2013). All of these models were run using Python3 and on Windows.

3.1 Lexicon Sentiment Analysis

The lexicon sentiment analysis uses the SentiWordNet sentiment lexicon (Baccianella et al.). The method works as follows: it first extracts all sentences containing the actor name from the review to make the set of sentences S. Then the sentiment score V of the actor is calculated as:

$$V = \sum_{s \in S} \sum_{w \in s} (pos\_score(w) - neg\_score(w))$$

Where s is a sentence in S and w is a word in the sentence s. pos_score and neg_score are taken from the SentiWordNet lexicon. We compute the average sentiment score of a word $V_{avg} = \frac{V}{|S|}$ where $|S|$ is the number of words in S. We then compare V_{avg} to the average $pos\_score - neg\_score$ of a word over all words in the lexicon (which is a constant of 0.010794992). If $V_{avg} > 0.010794992$ return Positive, if $V_{avg} < -0.010794992$ return Negative, else return Neutral.

3.2 Deep Neural Networks

We use the DNN-Sentiment implementation by Arthur Juliani for the Convolutional Neural Network (CNN) and Recurrent Neural Network

(RNN) deep learning models (Juliani). Both models are trained using data provided in the implementation. We ran a total of 10000 training iterations for each model. Learning rate was 0.01 and batch size was 64. The model uses 256 hidden layer nodes for the features.

As was done in the lexicon look-up method, for a given actor we extract the set of sentences containing that actors name S. For each sentence $s \in S$, we run the model (either CNN or RNN) on that sentence, which returns a continuous value score $score(s)$. $\sum_{s \in S} score(s)$ computes the score V for that actor. To turn that continuous score into our multinomial classification problem (positive/neutral/negative), we find an average similar to how was done for the lexicon look-up method. Here, we calculate the average score $score_{avg}$ over all sentences in the review, not just those containing the actor name, and also the standard deviation $stdev$. If $V \geq score_{avg} + \frac{1}{2} stdev$ return Positive, if $V \leq score_{avg} - \frac{1}{2} stdev$ return Negative, else return Neutral.

3.3 Attention-based Long Short-Term Memory

We use the Attention-based LSTM model (Wang et al., 2016b) which was implemented by Yang Peng (Peng) and proposed by Yequan Wang, Minlie Huang, Li Zhao and Xiaoyan Zhu, all of Tsinghua University. The idea behind this model is to be able to effectively weight each word in the sentence so that the sentiment classification focuses on the parts of the sentence that are most relevant. The expectation is that paying more attention to the actor and the words surrounding its name will provide a higher accuracy than the other methods compared against. This model was trained using data obtained from the movie review dataset put together by Bo Pang and Lillian Lee of Cornell University (Pang and Lee, 2004). A total of 15 iterations were run. There were 300 hidden units, the learning rate was set to 0.1 and the batch size was set to 50. Additionally, the largest sentence length specified was set to 130 words. The number of possible classes was set to 3 since there are 3 possible sentiments for a given actor in a movie review. The L2 Regularization term is set to 0.001. The loss function used here is cross-entropy (Wang et al., 2016b). As discussed in the paper by Wang et. al, the cross-entropy loss value can be computed as:

$$loss = -\sum_i \sum_j y_i^j log\hat{y}_i^j + \lambda ||\theta||^2$$

In this equation, i corresponds to the index of the sentence that is being analyzed and j corresponds to the class. y and $\hat{y}$ are the actual sentiment and predicted sentiment of a particular actor. λ is the L2 regularization term. Lastly, θ is the set of parameters for the model. (Wang et al., 2016b) To effectively comply with the input of the program, the reviews were preprocessed in a manner where only the sentences which mentioned any actors were given as part of the training and testing dataset. From there, an attention weight vector α is constructed in addition to a weighted hidden representation r. This representation can be computed by using the matrix of hidden vectors H which are produced from the Long Short-Term Memory model. According to the paper, the equation for computing r is:

$$r = H\alpha^T$$

(Wang et al., 2016b)

Effectively, this weights each hidden vector by the computed attention for each word. This hidden representation will be used to compute the final sentence distribution, which is then utilized to calculate the conditional probability distribution that the sentiment of the actor is negative, neutral or positive (Wang et al., 2016b).

4 Results

The results calculated from these different models were all based upon the dataset obtained through the usage of the Amazon Mechanical Turk service. We requested each laborer to read a review, identify all of the actors mentioned, and then determine the overall sentiment of the actor from the review. The dataset consisted of 500 total reviews from the movie review dataset compiled by Pang and Lee (Pang and Lee, 2004). Of these reviews, 250 were positive and 250 were negative with regards to the entire movie. From these reviews, after the results from the Mechanical Turk were received, the data was then processed into 1479 data points. Each data point is a tuple in the form of (Actor, Review) and the output is the sentiment of the actor in the review $y \in \{-1, 0, 1\}$. -1 denotes a negative sentiment on the actor, 0 denotes a neutral sentiment on the actor and 1 denotes a positive sentiment

on the actor. After the data had been processed into this form, it was split into training and testing with 80% of the data going into training and 20% of the data used for testing and validation. From that split, there were 1179 data points that were part of the training set and 295 data points which were a part of the test set. From the training data points, 442 data points had a positive sentiment, 531 reviews had a neutral sentiment and 211 reviews had a negative sentiment. For the test set, 122 data points had a positive sentiment, 122 data points had a neutral sentiment and 51 data points had a negative sentiment.

The performances of Lexicon Sentiment Analysis, Recurrent Neural Networks, Convolutional Neural Networks and Attention-based Long Short-Term Memory were all evaluated against one another.

Method	Accuracy
Random Guess	0.333
Lexicon Sentiment Analysis	0.373
Convolutional Neural Network	0.349
Recurrent Neural Network	0.359
Attention-based LSTM	0.455

Table 1: Accuracy of Sentiment Analysis Methods

Overall, all of the models tested performed better than randomly guessing the sentiment. The model that gave the highest accuracy was the Attention-based LSTM model with 45.5% accuracy. The other models do perform better than random guess, but not by much. The convolutional neural network only performed a little over 1% better than randomly guessing and among the four tested models yielded the lowest accuracy.

As the Attention-based LSTM model was also trained and tested on the movie review dataset, the loss and accuracy on the testing dataset was recorded as well.

The test accuracy appears to only fluctuate a bit based on the current iteration. It reaches its highest accuracy during the third and eighth iterations but for some time in-between and after those intervals, it falls to below 40% [Figure 2].

The cross-entropy loss function was used by the Attention-based LSTM model (Wang et al., 2016b) The loss does drop over the number of iterations which is an interesting observation since the testing accuracy does not substantially improve over that same interval [Figure 3].

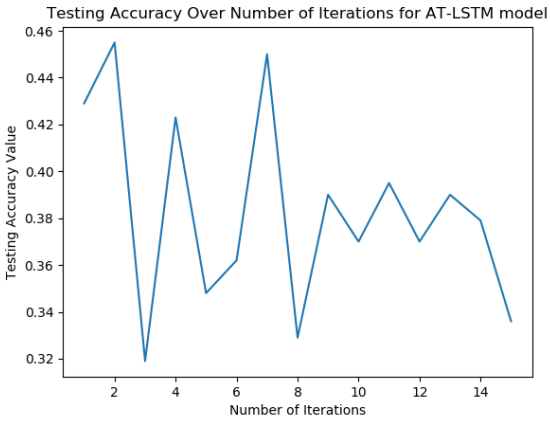

Figure 2: Test Error on AT-LSTM model

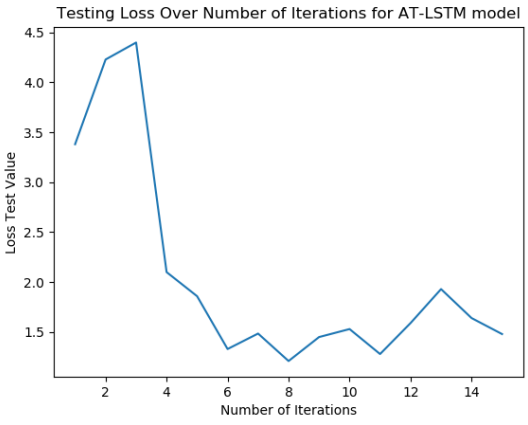

Figure 3: Test Loss on AT-LSTM model

5 Analysis

Upon evaluating the results, the Attention-based LSTM model AT-LTSM produced the highest accuracy at 45.5%. This supports our hypothesis prior to running the experiments. This makes sense because by focusing more on the important parts of a sentence that most relate to the actor, the model will be able to make a more accurate prediction on the sentiment. One thing which was a bit odd however was even when the loss value decreased over the number of iterations [Figure 3], the accuracy of the model did not subsequently increase [Figure 2]. Instead, it fluctuated a bit reaching a high accuracy of 45.5% and a low value of 31.9%. One possible explanation for this result is that even when the loss decreased, the model has already been relatively stabilized. Hence, even if the loss does drop, the accuracy does not substantially increase and even at times may decrease just by chance and possible overhead of TensorFlow. The training accuracy was also limited by the small amount of data we had available. Any further data would need to be manually retrieved since, to our knowledge, an actor polarity database for movie reviews does not already exist.

Between the other three models which were experimented with, the results only slightly outperformed random guessing. This shows the difficulty of the target-dependent problem: evaluating at a sentence-level rarely provides the focus needed on the actor entities. Of those three models, the lexicon sentiment analysis method gave the best score. This may be an indication that the training models for the CNN and RNN did not work properly or both are focusing on the wrong parts of the sentence, as opposed to the lexicon which weights all words of the sentence equally.

5.1 Error Analysis

There are a couple possible reasons for why the overall accuracy was relatively low from all of the different models. When using the Amazon Mechanical Turk service, there were some instances where the laborer would not do a proper job and misclassify data points. While we removed obvious cases of bad results, there likely exist more incorrect ground-truth results. Standard sentiment classification issues such as sarcasm, double negatives, ambiguity and excessive noise still impact results. In addition, since we used real user reviews, there were many punctuation and grammar errors which affect proper parsing.

For our specific problem, one common occurrence was sentences containing an actor name that described the character rather than the actor. For example, "James Earl Jones provides a terrifying, evil presence to the villain of Darth Vader." This sentence, evaluated at sentence-level, contains many largely negative words, but is actually a positive review of James Earl Jones' performance.

5.2 Case Study

One example review we used was: "I recently saw Infinity War. Great movie. Josh Brolin played the antagonist. Special effects were beyond amazing. Chris Evans was the star and was fantastic. Unfortunately, Benedict Cumberbatch was horrendous." The appropriate sentiment classification on this review is neutral for Josh Brolin, positive for Chris Evans and negative for Benedict Cumberbatch. Both the Attention-based LSTM and Lexicon look-up method correctly classified the sentiments on the three actors. For the CNN model, Chris Evans and Benedict Cumberbatch are classified correctly but Josh Brolin is classified as positive when he should be neutral. The CNN scores the sentence "Josh Brolin played the antagonist" as 11.5, not far behind the score of Chris Evans' sentence of 11.6. It seems to see the pattern "Josh Brolin played" as largely positive. Conversely, the sentence containing Benedict Cumberbatch is scored a very negative -60.7. For the RNN model, Chris Evans and Benedict Cumberbatch are still classified correctly but Josh Brolin is classified as negative instead of neutral. The results here score the sentence "Josh Brolin played the antagonist" as -0.97, seeming to retain memory and relate "Josh Brolin" with the typically negative word "antagonist." The lexicon is able to read the Josh Brolin sentence correctly as Neutral largely because the pos_score and neg_score of "antagonist" are both 0.

6 Future Work

There are many improvements that could be made to better the results. Validating the data for accurate ground-truth results and collecting more data would help training the models. More advanced tokenization that can deal with the many grammatical and punctuation errors found in real user reviews would increase the accuracy.

For the models themselves, Yoon Kim stated

the importance of training word embeddings when used in deep learning contexts (Kim, 2014), something we did not implement. The lexicon look-up model and the deep neural network implementations all were designed for binary classification (positive/negative), not the 3-class multinomial classification (positive/neutral/negative) we find more relevant. We used simple comparisons to average values to transform the two classes into three classes, but a machine learning model would improve results.

Implementing more NLP tools would also help. Coreference resolution could be implemented to find more references to actors, such as sentences using pronouns ”he” or ”she”. Dependency parsing would further focus the models on what words relate to the actors and which words do not. An actual entity linking ranking function would improve the accuracy when checking whether a named entity is actually an actor over just the IMDb API.

7 Conclusion

In the film industry today, movie reviews are an important component of rating how an actor did in a movie. However, review sentiment and the sentiment on a specific actor's performance in the film may not always be the same. Thus, our goal was to effectively identify a new model to carry out Actor Sentiment Analysis. We studied Lexicon-based analysis, Convolutional Neural Networks, Recurrent Neural Networks and Attention-based LSTM models. Upon testing the four different models, the one we found to have the highest accuracy was the Attention-based LSTM model [Table 1]. By focusing more on the actor in a sentence and the words which immediately relate it can more accurately classify the reviewers opinion towards the target actors performance in the film. Hence, for any future work that is done on trying to determine the sentiment of aspects within an entire review, we recommend to utilize an Attention-based LSTM model.

8 Acknowledgments

A major acknowledgment goes to Dr. Heng Ji of Rensselaer Polytechnic Institute under whom we learned a lot about Natural Language Processing and provided guidance while working on the project. In addition, another acknowledgment goes to Spencer Whitehead, a PhD candidate at RPI who helped us with specific knowledge for this project. Finally, thanks goes to Huang Lifu, another PhD candidate at RPI, for providing tools and references for us on this project.

References

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. Sentiwordnet.

Kristian Bannister. 2016. Sentiment analysis: How does it work? why should we use it?

Long Jiang, Mo Yu, Ming Zhou, Xiaohua Liu, and Tiejun Zhao. 2011. Target-dependent twitter sentiment classification. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 151–160. Association for Computational Linguistics.

Arthur Juliani. Dnn-sentiment.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Alberto Malagoli. Imdbpy.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*.

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. 2013. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*.

Annie Mueller. 2011. Why movies cost so much to make.

Tetsuya Nasukawa and Jeonghee Yi. 2003. Sentiment analysis: Capturing favorability using natural language processing. In *Proceedings of the 2nd international conference on Knowledge capture*, pages 70–77. ACM.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd annual meeting on Association for Computational Linguistics*, page 271. Association for Computational Linguistics.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of ACL*, pages 115–124.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of EMNLP*, pages 79–86.

Yang Peng. Td-lstm.

RPI. Elisa information extraction.

Benjamin Snyder and Regina Barzilay. 2007. Multiple aspect ranking using the good grief algorithm. In *In Proceedings of the Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics (HLT-NAACL*, pages 300–307.

Duyu Tang, Bing Qin, Xiaocheng Feng, and Ting Liu. 2015. Effective lstms for target-dependent sentiment classification. *arXiv preprint arXiv:1512.01100*.

Linlin Wang, Zhu Cao, Gerard de Melo, and Zhiyuan Liu. 2016a. Relation classification via multi-level attention cnns. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1298–1307.

Yequan Wang, Minlie Huang, Li Zhao, et al. 2016b. Attention-based lstm for aspect-level sentiment classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 606–615.