Aspect-level sentiment polarity classification using LSTM

Anjali Chourdia
New York University

Abstract

Sentiment analysis is the task of predicting the polarity of a text. Although sentiment analysis has been very useful, it cannot pinpoint the root cause of the nature of polarity. It has been found that sentiment polarity is not only determined by content but also strongly interrelated to the input aspect. For instance, the polarity of "Food is pretty good, but the ambience is not so pleasant" is positive for aspect "food" but negative for aspect "ambience". Aspect-based sentiment analysis is thus a step up on the standard sentiment analysis techniques and provides a fine grained sentiment classification based on the context or the aspect with respect to which a written piece is being evaluated.

1 Introduction

Sentiment classification, also known as opinion mining (Pang and Lee, 2008) is a fundamental and crucial task in natural language processing. Sentiment analysis is important for understanding reviews, feedback and connotation of a written text and has become extremely crucial as more and more people continue to share their opinion and feedback on social media and internet. In the recent years, Aspect-based sentiment analysis has received a lot of attention in sentiment analysis owing to the need of a more fine grained classification based on the context.

Some earlier proposed models for aspect-based sentiment analysis utilized the Pointwise Mutual Information between the target words and words in text(Jiang et al., 2011). As a step up, some other models adapted RNN(Dong et al., 2014) and Target dependent LSTM (Tang et al., 2015) to deal with this task. Further, [Wang et al., 2016] suggested utilizing the attention mechanism with LSTM to attend to the important parts of a sentence with respect to a given aspect. Although these models work well on Pos/Neg classification, accurate three-way classification continues to remain a challenge due to the increased complexity of the task.

In this paper, I discuss my implementation of attention-based LSTM, an extension of ATAE-LSTM(proposed by Wang et al.), for two-way and three-way aspect-term based classification. The aspect-based sentiment analysis task dealt in this paper involves classifying a restaurant review as positive, negative or neutral, given the review and aspect category or aspect terms. For example. the expected polarity of a review that reads "Bagels and beverage selections were good, but the service was poor" is positive if the input aspect is "food" or "Bagels" or "Beverage selections", negative wrt the aspect "service" and neutral wrt the aspect "price". It should be noted that the input aspect can be a word or a phrase. For training, I initialize both word and aspect vector embeddings with the pretrained 50-dimensional GloVe vectors(Pennington et al. 2014) as opposed to 300-dimensional GloVe vectors used by Wang et al.. The word vectors used in this paper, were trained using the Wikipedia articles 2014 (Pennington et al. 2014), so I further added dropout (Srivastava et al., 2012) to reduce the generalization error for my dataset. I experiment on SemEval

2014 restaurant dataset and results show that my model performs slightly worst for pos/neg and pos/neg/neut classification as compared to the baseline due to the low dimensionality of vector space, as also observed by [Tang et al.] in their experiments.

2 Related Work

2.1 Long Short-Term Memory (LSTM)

Long Short-Term Memory (Hochreiter and Schmidhuber, 1997) was introduced as an improvement on the gradient vanishing or gradient exploding problem of RNNs. Each LSTM cell takes in input a word embedding concatenated with the hidden state vector generated generated at time t-1. It further generates a memory vector and a new hidden state vector utilizing three gates: input gate, forget gate and output gate. Following are the associated equations utilized by each LSTM cell:

$$X = \begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix}$$
$$f_t = \sigma(W_f X + b_f)$$
$$i_t = \sigma(W_i X + b_i)$$
$$o_t = \sigma(W_o X + b_o)$$
$$c_t = f_t.c_{t-1} + i_t.tanh(W_c X + b_c)$$
$$h_t = o_t.tanh(c_t)$$

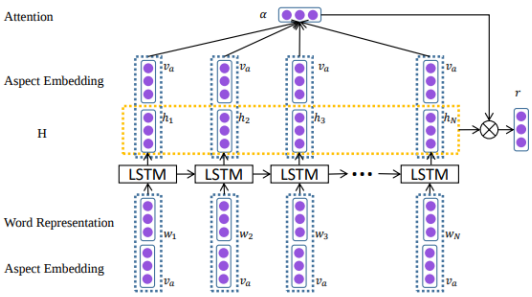
where W_f, W_i, W_o are weight matrices and b_f, b_i, b_o are bias vectors. Polarity prediction can be generated by linearizing the last hidden vector to a vector of required class labels and putting it through a softmax.

2.2 Attention-based LSTM (AT-LSTM)

Standard LSTMs can't detect which part of the sentence is important for the aspect-level sentiment classification. This problem is resolved with Attention-based LSTM[Wang et al., 2016], where the aspect embedding is concatenated with each hidden vector to generate an attention vector. The generated attention vector is then jointly used with the final hidden state vector to generate a sentence representation vector, which is further linearized to match the required output dimension and put into softmax to generate the final classification. An Attention mechanism has also previously shown to improve the model performance for other tasks including Image recognition (Mnih et al., 2014) and Machine Translation (Vaswani et al., 2017).

2.3 Attention-based LSTM with Aspect Embedding (ATAE-LSTM)

AT-LSTMs use the aspect information in computing the attention vector, but the hidden vectors contain no aspect information. To take better advantage of the aspect information, ATAE-LSTM(Wang et al., 2016) model is inputted aspect vector appended with word vector. In this way output hidden vectors have some information about the input aspect. Following are the equations associated with ATAE-LSTM cell, in addition to the basic LSTM equations:



Courtesy - (Wang et al., 2016)

$$M = tanh(\begin{bmatrix} W_h H \\ W_v v_a \otimes e_N \end{bmatrix})$$
$$\alpha = softmax(w^T M)$$
$$r = H\alpha^T$$

where, $M \in R^{d+d_a}$, $\alpha \in R^N$, $r \in R^d$. $W_h \in R^{dxd}$, $W_v \in R^{d_a x d_a}$ and $W \in R^{d+d_a}$ are projection parameters and d and d_a is the dimension of word and aspect embedding vectors used. α is a vector containing attention weights and r is weighted representation of the sentence writh respect to the input aspect. $W_v v_a \otimes e_N$ means repeating the linearly transformed vector v_a as many times as the number of words in the sentence. The final sentence representation is given by:

$h^* = tanh(W_p r + W_x h_N)$
where W_p and W_h are again projection parameters earned during training. This h* is further linearized and projected to c dimensional vector space(where c is the number of possible output class labels) and then finally put it into softmax to get the final output.
$y = softmax(W_s h^* + b_s)$

3 My work

I implemented the ATAE-LSTM cell network for classifying the restaurant review as one of the three labels negative, neutral, positive, given input as the review and the aspect which could be a phrase or a word. I computed the aspect vector by taking mean of the aspect level vectors of all the input aspect level terms of the phrase. The word and aspect vectors are initialized using the pre-trained 50 dimensional GloVe vectors. Additionally, I introduced a dropout layer right before the concatenated input and hidden vector is inputted into LSTM to better generalize the model and to have a better overall performance for the task at hand, as the GloVe vectors were trained using Wikipedia articles 2014.

3.1 Model Training

The model can be trained by using the technique of backpropagation. The goal of the training is to minimize the cross entropy error between the system generated classification(y) and the actual expected output(y_0) for each input. The parameter set to be trained consists of all the projection parameters and word and aspect vector embeddings. In addition, the word and aspect vectors are embedded into different vector space. Further, I utilized the AdaGrad(Duchi et al., 2011) method for optimization.

3.2 Dataset and Experiment

I experimented on the dataset of SemEval2014 Task 4 (Pontiki et al., 2014). The dataset consists of customer reviews for restaurants along with a list of aspect-level terms and corresponding polarities, the details of which can be found in Table 1 and Table 2. In my experiment, all word vectors are initialized by GloVe(Pennington et al., 2014). The word embedding are pre-trained on a corpus of size 6B tokens and 400k vocab. About 77% of the dataset words already existed in GloVe and rest of the words were randomly initialized from a uniform distribution of (-1,1). The dimension of word vectors, aspect embedding and hidden vectors is chosen to be 50 and the length of attention weights is taken to be the same as sentence length. PyTorch is used for implementing this ATAE-LSTM model. The model is trained with a batch-size of 16, L2 regularization weight decay of 0.001 and learning rate of 0.01 for AdaGrad.
I ran two two-way and one three-way classification experiment. Each two-way experiment training was done for 150 iterations and three-way training was done for 300 iterations due to the increased complexity of the classification.

Experiment	Pos	Neg	Neutral
pos/neg	1082	898	-
neg/neut	-	898	624
pos/neg/neut	1082	898	624

Table 1: Training dataset distribution across three classes

Experiment	Pos	Neg	Neutral
pos/neg	726	209	-
neg/neut	-	209	195
pos/neg/neut	726	209	195

Table 2: Test dataset distribution across three classes

4 Results and Analysis

The accuracy for my models for both pos/neg and pos/neg/neut classification is slightly worst than the baseline results(Table 3). This difference in the accuracy stems from the difference in vector space dimensionality between my models and the baseline. The baseline model(Wang et al., 2016) was trained using 300 dimensional vectors for both word

and aspect embeddings, whereas I utilized 50 dimensional vectors for initializing both word and aspect embeddings in my experiment. In the past experiments with LSTM, TD-LSTM and TC-LSTM(Tang et al.), it has been shown that models perform better with 100 and 200 dimensional vector as compared to 50 dimensional vector. Thus the difference in the accuracy of my implementation and baseline model is expected. The baseline models weren't run for neg/neut classification, hence there' no reference point for the accuracy results for neg/neut model, but I analyzed and compare the performance of two-way classification models(pos/neg, neg/neut) and three-way classification model(neg/neut/pos) by calculating and analyzing the recall, accuracy and f-scores for each class in each of the models.

The f-score for positive class prediction in both Experiment 1 and Experiment 3 is consistently higher than others. This can be attributed to two factors, first being the fact that it's inherently difficult for machine learning models to deal cases with negation, and secondly, also because the training dataset has more positive instances than negative instances. In Experiment 2(neg/neut classification), both neg and neut had roughly the same number of instances in training set, and it can be seen that the f-score for neg is 0.1 higher from Experiment 1(pos/neg classification), which justifies the low f-score of neg in Experiment 1 and 3.

Three way classification is a more complex than two way classification and it can be seen that the f-score for all the three classes reduce for Experiment 3(three way classification). Overall, the model performance for Experiment 1 and Experiment 3 are below baseline due to the use of 50 dimensional word vectors trained on 6B tokens as opposed to 300 dimensional vectors trained on a corpus of 420B tokens, used by the baseline model(Wang. et al., 2016).

Experiment	**Accuracy**	**Baseline Accuracy**
pos/neg	79.78%	90.9%
neg/neut	71.2%	-
pos/neg/neut	68.24%	77.2%

Table 3: Experiment results and baseline model performance(Wang et al., 2016)

4.1 Experiment 1: Pos/Neg

$$\begin{bmatrix} - & \mathbf{y = neg} & \mathbf{y = pos} \\ \mathbf{y_0 = neg} & 105 & 104 \\ \mathbf{y_0 = pos} & 83 & 633 \end{bmatrix}$$

pos accuracy - 633/(633+104) = 0.87
pos recall - 633/(633+83) = 0.88
pos f-measure - 2*r*a/(r+a) = 0.87
neg accuracy -0.55
neg recall - 0.50
neg f-measure - 0.52

4.2 Experiment 2: neg/neut

$$\begin{bmatrix} - & \mathbf{y = neg} & \mathbf{y = neut} \\ \mathbf{y_0 = neg} & 175 & 34 \\ \mathbf{y_0 = neut} & 82 & 113 \end{bmatrix}$$

neut accuracy - 113/(113+34) = 0.77
neut recall - 113/(113+82) = 0.58
neut f-measure - 0.66
neg accuracy - 175/(175+82) = 0.68
neg recall - 175/(175+34) = 0.83
neg f-measure - 0.75

4.3 Experiment 3: Pos/Neg/neut

$$\begin{bmatrix} - & \mathbf{y = neg} & \mathbf{y = neut} & \mathbf{y = pos} \\ \mathbf{y = neg} & 95 & 63 & 51 \\ \mathbf{y_0 = neut} & 47 & 45 & 103 \\ \mathbf{y_0 = pos} & 74 & 103 & 539 \end{bmatrix}$$

pos accuracy - 539/(539+103+51) = 0.78
pos recall - 539/(539+103+74) = 0.75
pos f-measure - 0.76
neut accuracy - 45/(63+45+103) = 0.21
neut recall - 45/(47+45+103) = 0.23
neut f-measure - 0.22
neg accuracy - 95/(95+47+74) = 0.44
neg recall - 95/(95+63+51) = 0.45
neg f-measure - 0.44

5 Future Work

Future work involves training model on datasets with roughly equal distribution of all three classes. In addition, attention-based LSTM models can be integrated with Recursive Neural Tensor Network models(Socher et al., 2013) which better take care of the semantic composition of the sentences and have shown better performance on predicting negative labels.

References

Yequan Wang, Minlie Huang, Xiaoyan Zhu, and Li Zhao. 2016. Attention-based LSTM for aspectlevel sentiment classification. In Proceedings of EMNLP. pages 606-615.

Li Dong, Furu Wei, Chuanqi Tan, Duyu Tang, Ming Zhou, and Ke Xu. 2014. Adaptive recursive neural network for target-dependent twitter sentiment classification. In ACL (2), pages 49-54

Duyu Tang, Bing Qin, Xiaocheng Feng, and Ting Liu. 2015a. Target-dependent sentiment classification with long short term memory. arXiv preprint arXiv:1512.01100

G.E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R.R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv:1207.0580, 2012.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In Neural Information Processing Systems (NIPS), 2017.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014),12:1532-1543

Sepp Hochreiter and Jurgen Schmidhuber. 1997. Long short-term memory. Neural computation, 9(8):1735 1780.

Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. 2014. Recurrent models of visual attention. In Advances in Neural Information Processing Systems, pages 2204-2212.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. The Journal of Machine Learning Research, 12:2121-2159.

Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. 2014. Semeval-2014 task 4: Aspect based sentiment analysis. In Proceedings of the 8th international workshop on semantic evaluation (SemEval 2014), pages 27-35.

Long Jiang, Mo Yu, Ming Zhou, Xiaohua Liu, and Tiejun Zhao. 2011. Target-dependent twitter sentiment classification. ACL, 1:151-160.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. Foundations and trends in information retrieval, 2(1-2):1-135.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In EMNLP 2013, pages 16311642.