SENTIMENT CLASSIFICATION WITH DEEP LEARNING MODEL

J.Vasanthaa[1], K.Sureka [2]
[1]Computer Science and Engineering, RVS College of Engineering,
vasanthaarangan1819@gmail.com
[2]Computer Science and Engineering, RVS College of Engineering, sindureka@gmail.com

Abstract -Sentiment classification is the concept that can be applied in various applications, such as in finding the sentiment of the product reviews, sentiment of the tweets etc. A Deep Learning based model for predicting the sentiment of the review will be designed. The sentiment classification can either be considered as a binary classification problem or multi class classification problem. A product review can be anywhere between either positive or negative. When most of the existing works consider it as a binary classification problem, the objective of the work is to create a multi class deep learning model. Before the implementation of the deep learning model conventional machine learning models will be implemented for comparison.

Keywords – Classification, deep learning, sentiment analysis, Machine Learning.

I. INTRODUCTION

1.1 Sentiment Analysis

Sentiment analysis is the process of using natural language processing, text analysis, and statistics to analyze customer sentiment. The best businesses understand the sentiment of their customers—what people are saying, how they're saying it, and what they mean. Customer sentiment can be found in tweets, comments, reviews, or other places where people mention your brand. Sentiment Analysis is the domain of understanding these emotions with software, and it's a must-understand for developers and business leaders in a modern workplace.

1.2 Classification

The classification is the process of assigning a label to the input and it is depicted in the following figure 1.1.

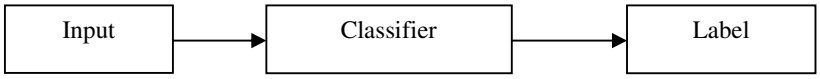

Figure 1.1 Classification Problem

1.3 Problem Statement

Given a set of tweets, the model should be capable of classifying it to positive tweet or negative tweet. It is considered as the classification problem. The problem statement is depicted in the following figure 1.2.

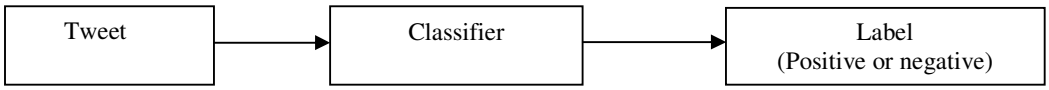

Figure 1.2 Problem Depiction

1.4 Models Employed

The models that are employed in this work are as follows

- Recurrent Neural Network
- Decision Tree
- Naive Bayes

All these models required pre-processing. The pre-processing required to be done for sentimental analysis includes functionalities from natural language processing.

1.5 Dataset

The dataset contains 4000 data out of which 2000 data belongs to positive class and 2000 data belong to negative class.

1.6 Objective

To design three different models for sentiment analysis and compare the results obtained from them.

II. SYSTEM DESIGN

2.1 Existing System

Most of the existing works consider this as the binary classification problem. This could be suitable for scenarios such as tweets but not for product reviews.

2.2 Proposed System

The following classifiers are implemented for sentiment analysis and the results are compared

- Multinomial Naive Bayes
- Decision tree
- Recurrent Neural Networks

The reasons for choosing these classifiers are as follows. From the Literature it has been observed that decision tree performs well than the other models so it is considered for comparison. Multinomial naive bayes classifier that has been specially designed for sentiment analysis and recurrent neural network is a kind of deep learning model with memory. All the three models are implemented and the results are compared.

2.3 System Architecture

The following figure 2.1 represents the system architecture of the recurrent neural network.

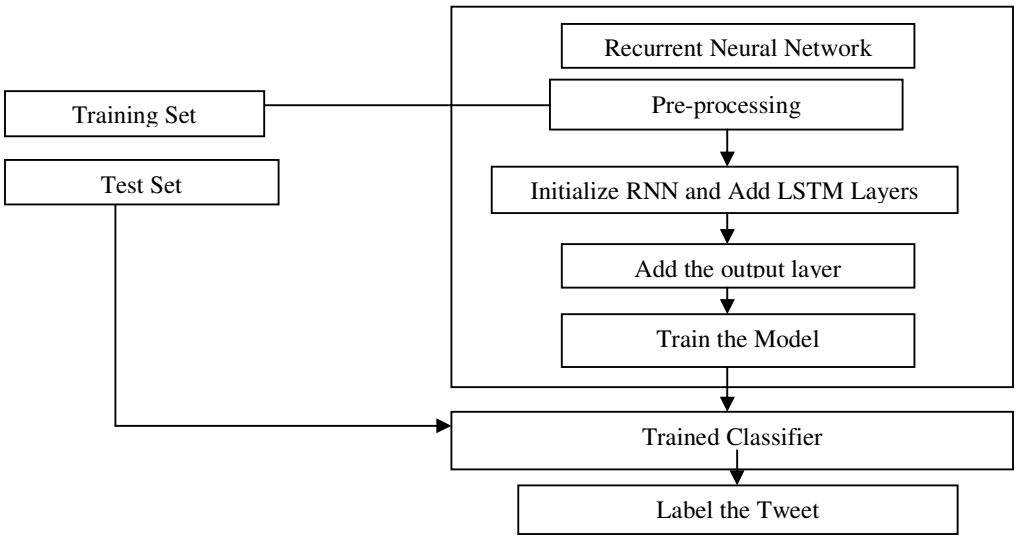

Figure 2.1 System Architecture-LSTM

The following figure 2.2 represents the architecture for the classifiers multinomial naive bayes and the decision tree models.

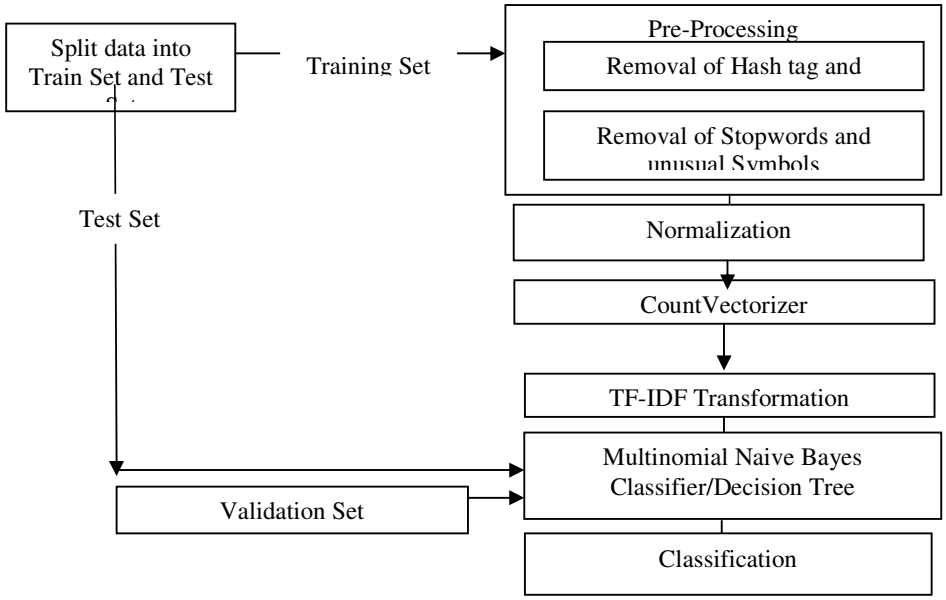

Figure 2.2 System Architecture – Naive Bayes and Decision Tree

The results obtained from the three models are compared.

2.4 Modules

The modules involved in the project are

- Pre-processing
- Building the recurrent neural network

2.4.1 Pre-Processing

The first process in pre-processing is tokenizing. 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.

With this, all the punctuations are removed and the texts are converted into space separated words. These sequences are then split into token of lists. Indexing or vectorization is applied then. The text are converted into numbers .the next process is to use padding process. Padding is done in order to ensure that zeroes are appended to all the sequences to ensure they are if same length.

2.4.2 Building The Recurrent Neural Network

The steps involved in building the recurrent neural network is

- Initializing the network
- Adding the Required LSTM Layer
- Adding the output layer

2.4.2.1 Initializing The Network

Initializing a network is nothing but deciding upon the model to be used with the keras. The model used in the system is the sequential model. It can be represented with a stack of layers. The layers can be LSTM layers or dropouts depending on the requirement. It is followed by embedding the various layers. Keras offers an Embedding layer that can be used for neural networks on text data. It requires that the input data be integer encoded, so that each word is represented by a unique integer. This data preparation step can be performed using the tokenizer API also provided with Keras. The Embedding layer is initialized with random weights and will learn an embedding for all of the words in the training dataset.

2.4.2.2 Adding the Required LSTM Layer

LSTM layer refers to the long short term memory layer which is used for avoiding the vanishing gradient problem that occurs in the recurrent neural network. Before explaining the LSTM, the problem of vanishing gradient descent is discussed.

Gradient descent algorithm is the one that is used for identifying the minimum cost function in the machine learning model. The difference between the value predicted and the actual value is calculated and the error is calculated and propagated back to the network for adjusting the weights.

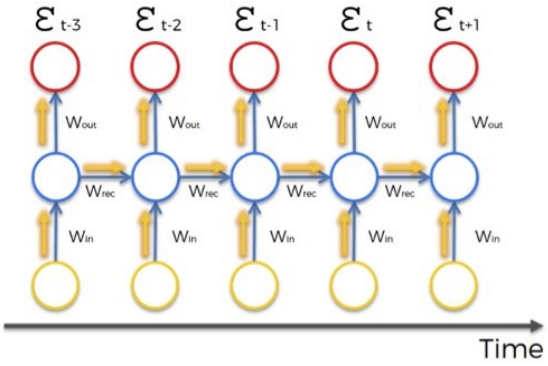

Figure 2.3 Vanishing Gradient Problem

The vanishing gradient problem can be solved by the following methods.

- Weights should be initialized in such a way that the probability of gradient vanishing is minimized.
- To use the network with echo state
- To introduce LSTM layers

The approach that has been followed in the proposed model is addition of LSTM layers. The operation of the LSTM layer is given below.

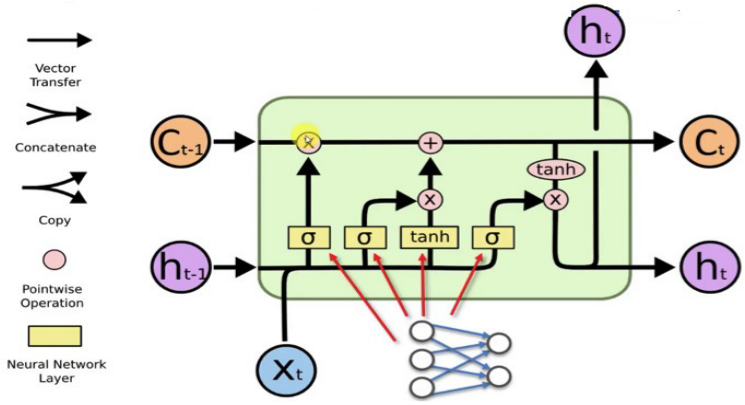

Figure 2.4 LSTM Layer

- The new value received and the value received from the previous node is obtained.
- The values are combined and given as input to the sigmoid function, with which the decision is made about the forget value. The decision can open; open to some extent or close.
- The same set of values is further passed through two functions, tanh and the sigmoid function. The former is used for deciding on the parameters which can be passed through the memory pipeline and the later decided whether the value will be send to the memory pipeline and if so, to what extent.
- When the memory is flowing through the pipeline, the memory will not change if we have forget valve opened and memory valve closed. The memory will be updated completely in the other case.
- This would help in deciding the output of the module.

2.4.2.3 Adding the Output Layer

The output layer is the dense layer with 2 dimensional outputs and the softmax layer. Once the model is designed it can be tested with the validation data and the accuracy is obtained.

2.5 Multi Nomial Naive Bayes Algorithm

Modules

- Pre-processing
- Normalization
- Count vectorization
- TF-IDF Transformation
- Classification with naive bayes

2.5.1 Pre-Processing

Pre-processing includes the following sub processes

- Removal of Hash tags and punctuations
- Removal of stopwords and Sentiment Classification

Twitter comments not only contain English words, it also contains punctuations and hash tags. Before sentiment analysis is made, all these should be removed because it does not contribute for sentiment analysis. This is done with a module called as TextBlob. With the help of this, the words are separated and these symbols are removed. Removal of stop words is also an integral part of the text pre-processing. Stopwords includes words like a, an, in, the etc.

2.5.2 Normalization

Text normalization is the process of transforming a text into a canonical (standard) form. For example, the word "gooood" and "gud" can be transformed to "good", its canonical form. Another example is mapping of near identical words such as "stopwords", "stop-words" and "stop words" to just "stopwords".

Text normalization is important for noisy texts such as social media comments, text messages and comments to blog posts where abbreviations, misspellings and use of out-of-vocabulary words (oov) are prevalent.

2.5.3 Count Vectorization

With the help of the CountVectorizer, a bag of words is created. Bag of words is created first by selecting the unique words from the set of words available and a table is formed as shown below.

There are n comments

Set of reviews=$\{r_0,r_2,.....r_n\}$

Set of unique words in all columns=$\{uw_1,uw_2,.....uw_n\}$

A table is formed, where the rows denote the set of comments and columns denote the set of unique words.

	Uw_1	**Uw_2**		**Uw_n**
comment 0				
comment 1				
.			.. . .	
.				
comment n				

Each cell of the Table is filled with either 0 or 1. For example, the cell {Comment 0, Uw_1} is filled with 1 if that particular word Uw_1 occurs in comment 0, else the cell is filled with 0.

2.5.4 TF-IDF Transformation

In order to re-weight the count features into floating point values suitable for usage by a classifier it is very common to use the tf–idf transform. Tf means **term frequency** while tf–idf means term-frequency time's **inverse document-frequency**:

tf-idf(t,d)=tf(t,d) X idf(t)
where

idf(t) = $\log\frac{1+n}{1+df(t)}$ +1

Where n is the total number of documents in the document set, and df(t) is the number of documents in the document set that contain term t. The resulting tf-idf vectors are then normalized by the Euclidean normalization.

2.5.5 Multi Naive Bayes Algorithm

Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes' theorem with the "naive" assumption of conditional independence between every pair of features given the value of the class variable. Bayes' theorem states the following relationship, given class variable y and dependent feature vector x_1 through x_n:

$$P(y \mid x_1, \dots, x_n) = \frac{P(y)P(x_1, \dots x_n \mid y)}{P(x_1, \dots, x_n)}$$

Using the naive conditional independence assumption that

$$P(x_i|y, x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n) = P(x_i|y),$$

for all i, this relationship is simplified to

$$P(y \mid x_1, \dots, x_n) = \frac{P(y)\prod_{i=1}^{n} P(x_i \mid y)}{P(x_1, \dots, x_n)}$$

Since $P(x_1, \dots, x_n)$is constant given the input, we can use the following classification rule:

$$P(y \mid x_1, \dots, x_n) \propto P(y)\prod_{i=1}^{n} P(x_i \mid y)$$

$$\Downarrow$$

$$\hat{y} = \arg\max_{y} P(y)\prod_{i=1}^{n} P(x_i \mid y),$$

and we can use Maximum A Posteriori (MAP) estimation to estimate $P(y)$and $P(x_i \mid y)$); the former is then the relative frequency of class y in the training set.

Multinomial naive bayes algorithm implements the naive Bayes algorithm for multinomially distributed data, and is one of the two classic naive Bayes variants used in text classification (where the data are typically represented as word vector counts, although tf-idf vectors are also known to work well in practice). The distribution is parameterized by vectors $\theta_y=(\theta_{y1},\dots,\theta_{yn})$ for each class y, where n is the number of features (in text classification, the size of the vocabulary) and θ_{yi} is the probability $P(x_i|y)$ of feature i appearing in a sample belonging to class y.

The parameters θ_y is estimated by a smoothed version of maximum likelihood, i.e. relative frequency counting:

$$\hat{\theta}_{yi} = \frac{N_{yi} + \alpha}{N_y + \alpha n}$$

where $N_{yi} = \sum_{x \in T} x_i$ is the number of times feature i appears in a sample of class y in the training set T, and $N_y = \sum_{i=1}^{n} N_{yi}$is the total count of all features for class y. The smoothing priors $\alpha \geq 0$ accounts for features not present in the learning samples and prevents zero probabilities in further computations. Setting $\alpha=1$ is called Laplace smoothing, while $\alpha<1$ is called Lidstone smoothing. The results obtained are discussed in the next chapter.

III. EXPERIMENTAL RESULTS

The parameters that are considered for evaluation are as follows.

- Precision(P)

- Recall(R)
- F1-score(F1)

Precision, recall and F1-Score are calculated with the following parameters

True Positive rate

$$\text{True Positive Rate (TPR)} = \frac{TP}{TP+FP} \quad (1)$$

Where TP =True Positive, FP =False Positive

False Positive rate

$$\text{False Positive Rate (FPR)} = \frac{FP}{FP+TN} \quad (2)$$

Where TN=True Negative

Precision

$$\text{Precision} = \frac{TP}{TP+FN} \quad (3)$$

Where FN is False Negative

Recall

$$\text{Recall} = \frac{TP}{TP+FP} \quad (4)$$

F-Measure

$$\text{F-Measure} = 2 \times \frac{Precision \times Recall}{Precision+Recall} \quad (5)$$

and**accuracy.**

Accuracy of the System is calculated with the following formula

Accuracy= (TP+TN)/ (TN+FP+FN+TP)

The False positive, False Negative, True Positive and True Negative are as follows

- True Negative refers to the number of negative reviews that has been identified as negative
- False Positive refers to the number of negative reviews that has been identified as positive
- False Negative refers to the number of positive reviews that has been identified as negative
- True Positive refers to the number of positive reviews that has been identified as positive

The results obtained with naive bayes classifier are as follows for training and testing phase.

Table 3.1 Classification Report – Naive Bayes – Training and Testing

	Precision	Recall	F1-score
0	0.89	0.88	0.88
1	0.88	0.89	0.89
Weighted Average	0.89	0.89	0.88

The accuracy obtained with naive bayes is 88.5% during training. The results obtained with naive bayes classifier are as follows for Validation set.

Table 3.2 Classification Report – Naive Bayes – Validation

	Precision	Recall	F1-score
0	0.80	0.99	0.88
1	0.89	0.31	0.46
Weighted Average	0.82	0.81	0.77

The accuracy obtained with naive bayes is 81% during validation. The results obtained with recurrent neural network model are as follows for Training and Testing Phase.

Table 3.3 Classification Report – RNN – Training and Testing

	Precision	Recall	F1-score
0	0.86	0.87	0.87
1	0.86	0.86	0.86
Weighted Average	0.86	0.86	0.86

The accuracy obtained with recurrent neural network is 86% during training and testing. The results obtained with recurrent neural network model are as follows for validation.

Table 3.4 Classification Report – RNN – Validation

	Precision	Recall	F1-score
0	0.54	0.92	0.68

1	0.56	0.11	0.18
Weighted Average	0.55	0.54	0.44

The accuracy obtained with RNN is 54% during validation. The results obtained with the Decision Tree model are as follows during training and testing.

Table 3.5 Classification Report – Decision Tree – Training and Testing

	Precision	Recall	F1-score
0	0.62	0.78	0.69
1	0.83	0.70	0.76
Weighted Average	0.75	0.73	0.73

The accuracy obtained is 72.75% during training and testing. The results obtained with the Decision Tree model are as follows during Validation.

Table 3.6 Classification Report – Decision Tree – Validation

	Precision	Recall	F1-score
0	0.75	0.96	0.84
1	0.67	0.21	0.32
Weighted Average	0.72	0.74	0.69

The accuracy obtained with Decision Tree is 71% during validation. The following figure represents the comparison of the accuracy obtained with the three models during training and testing phase.

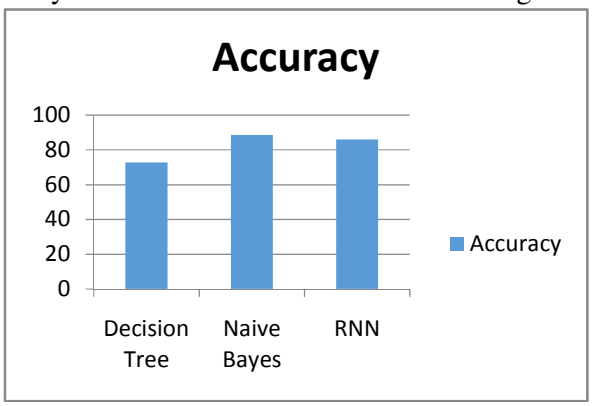

Figure 3.1 Comparison of accuracy of three models-Training and testing

The following figure represents the comparison of the accuracy obtained with the three models during validation phase.

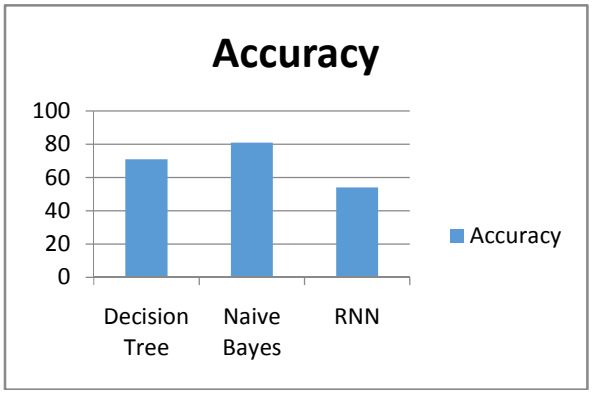

Figure 3.2 Comparison of accuracy of three models-Validations

IV. CONCLUSION AND FUTURE WORK

Three classifiers are designed for sentiment classification, two machine learning model and a deep learning model. The machine learning models implemented are multinomial naive bayes and decision tree. The deep learning model implemented is recurrent neural network with LSTM layer.

The dataset used in the work is a balanced model with 2000 positive reviews and 2000 negative reviews. Natural Language processing is used for pre-processing before the actual models are implemented. It has been observed that Naive Bayes model performs better than the other models. The future work would include designing a model that could produce better result in identifying the multiple polarities in a product review.

REFERENCES

[1] Feng Xu, Zhenchun Pan, Rui Xia, "E-commerce product review sentiment classification based on a naïve Bayes continuous learning framework", Information Processing & Management, Volume 57, Issue 5,2020.
[2] Hao Tang, et.al, "Dependency graph enhanced dual-transformer structure for aspect sentiment classification", Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, july2020.
[3] Jiangjiao Duan, Banghui Luo, Jianping Zeng, "Semi-supervised learning with generative model for sentiment classification of stock messages", Expert Systems with Applications, Volume 158, 2020.
[4] Md Shad Akhtar, Tarun Garg, Asif Ekbal, "Multi-task learning for aspect term extraction and aspect sentiment classification", Neurocomputing, Volume 398, 2020.
[5] Mohammad Ehsan Basiri, Moloud Abdar, Mehmet Akif Cifci, Shahla Nemati, U. Rajendra Acharya, "A novel method for sentiment classification of drug reviews using fusion of deep and machine learning techniques ", Knowledge-Based Systems, Volume 198, 2020.
[6] N. Jiang, F. Tian, J. Li, X. Yuan and J. Zheng, "MAN: Mutual Attention Neural Networks Model for Aspect-Level Sentiment Classification in SIoT," in IEEE Internet of Things Journal, vol. 7, no. 4, pp. 2901-2913, April 2020.
[7] Pinlong zhaoa, "Modeling Sentiment Dependencies with Graph Convolutional Networks for Aspect-level Sentiment Classification", 2019.
[8] R. Kiran, Pradeep Kumar, Bharat Bhasker, "Oslcfit (organic simultaneous LSTM and CNN Fit): A novel deep learning based solution for sentiment polarity classification of reviews", Expert Systems with Applications, Volume 157, 2020.
[9] Sridharan, K., Komarasamy, G. Sentiment classification using harmony random forest and harmony gradient boosting machine. *Soft Comput* 24, 7451–7458 (2020).
[10] T. K. Tran and T. T. Phan, "Capturing Contextual Factors in Sentiment Classification: An Ensemble Approach," in IEEE Access, vol. 8, pp. 116856, 2020.
[11] Wang, Z., Lin, Z. Optimal Feature Selection for Learning-Based Algorithms for Sentiment Classification. *Cogn Comput* 12, 238–248 (2020).
[12] Wei Li, Luyao Zhu, Yong Shi, Kun Guo, Erik Cambria, "User reviews: Sentiment analysis using lexicon integrated two-channel CNN–LSTM family models", Applied Soft Computing, Volume 94, 2020.
[13] Xu, L., Bing, L., Lu, W., and Huang, F., "Aspect Sentiment Classification with Aspect-Specific Opinion Spans".
[14] Zhang, S., Xu, X., Pang, Y. *et al.* Multi-layer Attention Based CNN for Target-Dependent Sentiment Classification. *Neural Process Lett* 51, 2089–2103 (2020).