Sentiment Analysis of Reviews using Opinion Mining on Web

Kimberley Faria
Department of Information Technology, Fr. Conceicao Rodrigues College Of Engineering Bandstand, Bandra(W), Mumbai-400 050

Karen Carvalho
Department of Information Technology, Fr. Conceicao Rodrigues College Of Engineering Bandstand, Bandra(W), Mumbai-400 050

Arshiya Thakur
Department of Information Technology, Fr. Conceicao Rodrigues College Of Engineering Bandstand, Bandra(W), Mumbai-400 050

Sarika Davare
Assistant Professor
Department of Information Technology,
Fr. Conceicao Rodrigues College of Engineering Bandstand,
Bandra(W), Mumbai-400 050

ABSTRACT

Sentiment analysis and opinion mining is the field of study that analyzes people's opinions, sentiments, evaluations, attitudes, and emotions from written language. It is the process of identifying opinions in large structured/ unstructured data and then analyzing polarity of these mined opinions. In this paper, the aim is to categorize polarity and identify the correct polarity for given sentiment. The intention is to use Recursive Neural Net-works(RNN) model. The goal is to gather product reviews and using this model to perform sentiment analysis on the collected data to make an informed and accurate suggestion to the user

General Terms

Sentiment analysis, Neural networks.

Keywords

Sentiment analysis, opinion mining, RNN

1. INTRODUCTION

Sentiment analysis (also known as opinion mining) refers to the use of natural language processing, text analysis and computational linguistics to identify and extract subjective information in source materials. It is used to analyze the reviews on various platforms such as social media, product websites etc. [1]

It helps to understand the attitude of the reviewer and accordingly determine the overall polarity of the review. The attitude can be described as judgment or evaluation.

Sentiment analysis has been a popular topic in the field of machine learning. It is largely applied to data that comes with self-labeled information such as movie reviews on imdb. A scalar score comes along with the review text a user writes, which provides a good and reliable labeling of the text polarity. This ability to identify the positive or negative sentiment behind a piece of text is even more interesting when it comes to social data. Twitter gets new user data literally every second. If the proposed system can predict sentiment labels for incoming live tweets, it is possible to understand the most recent user attitude towards a variety of topics from a commercial flight satisfaction to brand image. Reviews on e-commerce websites can be analyzed and summarized. Using complex structured neural networks, Recursive Neural Network(RNN) and Recursive Neural Tensor Network(RNTN), and existing data scraping tools, the proposed system will collect these reviews and determine the nature of these reviews using RNNs.

2. LITERATURE SURVEY

Turney and Pang used various methods to identify the polarity of product and movie review. This analysis is at the document level. Later Pang and Snyder classified document's polarity on a multi-way scale by expanding the basic task of classifying a movie re-view as either positive or negative. Pang and Lee predicted the star ratings on either a 3 or a 4 star scale. Snyder predicted ratings on a five-star scale for various aspects of a restaurant, such as the food and ambiance by performing an in-depth analysis of restaurant re-views. In most statistical classification methods, the neutral class is ignored under the assumption that neutral texts lie near the boundary of the binary classifier, but several researchers suggest that, as in every polarity problem, three categories must be identified. Also, it can be proven that specific classifiers such as the Max Entropy and the SVMs can benefit from the introduction of a neutral class and improve the overall accuracy of the classification. There are two principle ways for operating with a neutral class. One, the algorithm proceeds by first identifying the neutral language, filtering it out and then assessing the rest in terms of positive and negative sentiments, second, it builds a three way classification in one step. The second approach often involves estimating a probability distribution over all categories. The nature of the data: if the data is clearly clustered into neutral, negative and positive language, it makes sense to filter the neutral language out and focus on the polarity between positive and negative sentiments determines whether and how to use a neutral class. On the contrary, if the data is mostly neutral with small deviations towards positive and negative affect, this strategy would make clearly distinguish between the two poles harder. [1]

Another method for identifying sentiment is the use of a scaling system where words that are commonly associated with having a negative, neutral or positive sentiment with them are given an associated number on a -10 to +10 scale (most negative up to most positive). This makes it possible to adjust the sentiment of a given term relative to its environment. When a piece of text is analyzed using natural language processing, each concept in the specified environment is given a score based on the way sentiment words relate to the concept and its associated score. This allows a more sophisticated understanding of sentiment, as it is now possible to adjust the sentiment value of a concept relative to changes in the environment. [1]

Part-to-Speech Tagging(POST) is used for Identifying and categorizing words in a piece of text in a given language into defined parts of speech. Part-of-Speech Tagging (POST) or lexical set are used to find out the grammatical words in any document or user speech: like noun, verb, adjective, etc. This can be done either on the ba-sis of definition, e.g. all names are noun like India, or on the basis of context which depends upon the relationship with neighboring or similar words. Word classes is also known as POS tags. We use POS tags for specific sentence or task. In Classical grammar classifies eight words for user text and user speech: Like Conjunction, the preposition, pronoun, noun, adjective, verb, adverb and interjection. Part of speech tags are basically assigned to particular task and particular user speech.[2]

Earlier, the field of sentiment analysis was based on document level sentiment analysis and aimed at classifying the whole document as positive or negative. The assumption in this case was that each document expresses sentiment based on only one entity expressed by only one opinion holder. Sentiment can be classified based on supervised learning techniques and unsupervised learning techniques. Supervised learning techniques include text classification based on a classifier. Supervised learning technique takes into account features like ’terms and their frequency’, ’parts of speech’, ’sentiment words and phrases’, ’sentiment shifters’ and so on. Unsupervised learning techniques make use of fixed syntactic patterns that occur in an opinion. This technique uses POS tagging which identifies nouns, adverbs, adjectives etc in a sentence. Based on knowledge and arrangement of these words the system identifies the entity, aspect and the opinion. Another method under unsupervised learning technique is maintaining dictionary of sentiment words and their weights based on which opinion. This approach also takes into consideration effect of negation. Later sentence level sentiment analysis and aspect level sentiment analysis also emerged as field of research. In sentence level sentiment analysis the system per-forms analysis at sentence level. Here the basic aim is to identify subjective and objective sentences. A model, naive Bayes classifier is used for identifying subjectivity in sentences .Research has been done on aspect level sentiment analysis which aims to identify various product reviews available on the internet and analyzing them. Thus there are 2 basic tasks involved in aspect level sentiment analysis and they are, aspect extraction and aspect sentiment classification. The dictionary of sentiment words is used to analyze sentiments on aspect.

Aspect level sentiment analysis aims at identifying the aspects of the entity. For example, LOR (Letter Of Recommendation) system identifies these aspects in teachers’ remarks and evaluates an aspect value based on the aspect tree. The aspect tree stores a defined set of aspects. The students are analyzed over these aspects. In this case aspects can be student’s academics, sports, extra-curricular, co-curricular performance, personality traits etc. These are the ma-jor aspect categories having sub-categories under them. For ex-ample, a student can be good in DBMS. So DBMS would be a sub-aspect of subjects which is further a sub-aspect of academics. This is how a tree is maintained for aspects and weight is given to each branch of the tree. Lower the height of an aspect in the aspect tree lower is the specificity of the opinion or remark given by the teacher. This can be explained with an example. Considering two remarks,

—He actively participates in co-curricular activities.

—He actively participates in Anugoonj (University Fest) and Corona.(College Event)

Here the remark 2 is more specific than remark one. Co-curricular lies at lower level in the aspect tree whereas Anugoonj and Corona are at higher level hence having an elevated aspect value. Thus the remark 2 is more specific than the remark 1. Every aspect is represented by a node in the tree. Every branch is assigned a weight that depends on the node to which that branch points to. When a teacher gives remarks about a particular student, the remark is fetched sentence by sentence. Each sentence is stored as an array of words. From the tree of aspects, an aspect is searched in that sentence. When that aspect is found, the aspect value is calculated by traversing the tree. Traversal is done from the location of that aspect in the tree to the root of the tree. The weights of branches that are traversed are multiplied. This helps in identifying how much specific is that aspect which is mentioned in the remarks. It also increases or decreases the value of the remark. Thus the aspect value tells how much clear or specific the remark is. Once the aspect value is known the sentiment about the aspect is calculated from the array of words by means of sentiment dictionary. Here each word and negations in the sentence are analyzed to find out the polarity of the remark. Thus the sentiment and the aspect provide the two dimensions of the opinion. Now, summarizing the aspect and sentiment in the remark, overall opinion about the student is obtained.[3] Unlike post tagging and aspect oriented sentiment analysis Recursive Neural Network(RNN) does not just consider some bag of words and label them but also accurately captures the effects of negation and its scope at various tree levels for both positive and negative phrases. Sentiment of phrases following the conjunction ‘but’ dominate, is also learned by RNNs .

3. PROPOSED SYSTEM

We are dividing the task of sentiment analysis into two parts.

—Data Collection

—Analyzing the data

3.1 Data Collection

Data collection is done using crawlers on the web. Using the crawlers and scraping tools it is possible to scrap the data from the websites and store it as tuples in the database. Thus the reviews related to the product can be collect from web.

3.2 Analyzing the data

The collected reviews are to be analyzed. First, the words in a particular phrase will be represented as vectors. Recursive neural model will then compute parent vectors in a bottom up fashion using different types of compositionality functions. The parent vectors must be of the same dimensionality to be recursively compatible and be used as input to the next composition. Each parent vector, is given to the softmax classifier to compute the posterior probability over the labels given the word vector. This process will be continued for all the reviews. The system then takes a poll to determine the overall sentiment.[4]

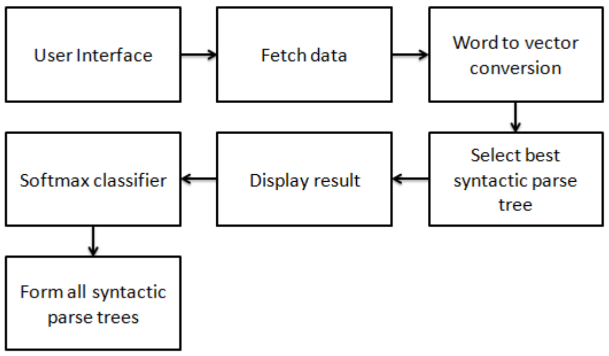
User Interface
Fetch data
Word to vector conversion
Select best syntactic parse tree
Display result
Softmax classifier
Form all syntactic parse trees

Fig. 1. Block Diagram

4. DESCRIPTION

4.1 Fetching Data

In this section we discuss about data scraping using scrapy- a free and open source web crawling framework, written in Python. Scrapy project architecture is built around spiders, which are self-contained crawlers which are given a set of instructions.

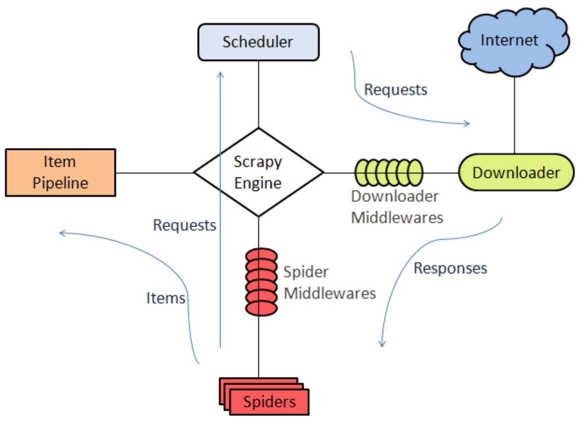

Fig. 2. Scrapy

The data flow in Scrapy is controlled by the execution engine, and goes like this:

—The Engine gets the initial Requests to crawl from the Spider.

—The Engine schedules the Requests in the Scheduler and asks for the next Requests to crawl.

—The Scheduler returns the next Requests to the Engine.

—The Engine sends the Requests to the Downloader, passing through the Downloader Middlewares.

—Once the page finishes downloading the Downloader generates a Response (with that page) and sends it to the Engine, passing through the Downloader Middlewares.

—The Engine receives the Response from the Downloader and sends it to the Spider for processing, passing through the Spider Middleware.

The Spider processes the Response and returns scraped items and new Requests (to follow) to the Engine, passing through the Spider Middleware.

—The Engine sends processed items to Item Pipelines, then send processed Requests to the Scheduler and asks for possible next Requests to crawl.

—The process repeats (from step 1) until there are no more requests from the Scheduler.

Components of scrapy are as follows:

—Scrapy Engine: The engine is responsible for controlling the data flow between all components of the system, and triggering events when certain actions occur. See the Data Flow section above for more details.

—Scheduler: The Scheduler receives requests from the engine and enqueues them for feeding them later (also to the engine) when the engine requests them.

—Downloader: The Downloader is responsible for fetching web pages and feeding them to the engine which, in turn, feeds them to the spiders.

—Spiders: Spiders are custom classes written by Scrapy users to parse responses and extract items (aka scraped items) from them or additional requests to follow. For more information see Spiders.

—Item Pipeline: The Item Pipeline is responsible for processing the items once they have been extracted (or scraped) by the spiders. Typical tasks include cleansing, validation and persistence (like storing the item in a database). For more information see Item Pipeline.

—Downloader middlewares: Downloader middlewares are specific hooks that sit between the Engine and the Downloader and process requests when they pass from the Engine to the Down-loader, and responses that pass from Downloader to the Engine. Use a Downloader middleware if you need to do one of the following:

—process a request just before it is sent to the Downloader (i.e. right before Scrapy sends the request to the website). —change received response before passing it to a spider. — send a new Request instead of passing received response to a spider.

—pass response to a spider without fetching a web page.

—silently drop some requests.

—Spider middlewares: Spider middlewares are specific hooks that sit between the Engine and the Spiders and are able to process spider input (responses) and output (items and requests).

4.2 Word Vector Representation

—Represent word by means of it's neighbors: The majority of statistical language processing and rule-based algorithms consider words as atomic symbols. This translates to a very sparse vector representation of the size of the vocabulary and with a single 1 at the index location of the current word. Problem with this so-called one-hot or one-on representations is that it does not capture any type of similarity between two words. For example, if a model sees hotel in a surrounding context it cannot use this information when it sees motel at the same location during test time. Instead of this simplistic approach, we represent each word by means of its neighbors. [5] This is one of the most successful ideas of modern statistical natural language processing (NLP). The word embedding matrix is given by,

$$l\ 2\ R^{n\ V} \quad (1)$$

where V is size of vocabulary. The embedding matrix L assumes that co-occurrence statistics is captured by its column vectors that have been learned by a separate unsupervised model. Alternately, initialize L to small uniformly random numbers and then train this matrix as part of the model. A sentence (or any n-gram) is represented as an ordered list of m words. Each word has an associated vocabulary index k into the embedding matrix that is used to retrieve the word's vector representation. Mathematically, this look-up operation can be viewed as a simple projection layer where a binary vector b 2 0; 1^V is used which is zero in all positions except at the kth index, $x_i = Lb_k\ 2\ R^{\ n}$. The result is a sentence representation as a list of vectors (x1; : : : ; xm).

—Window-based neural network: The main idea of the unsupervised window-based word vector learning model is that a word and its context is a positive training sample. A context with a random word in the center (or end position) constitutes a negative training example. One way to formalize this idea is to as-sign a score to each window of k words, then replace the center word and train the model to identify which of the two windows is the true one. For example, the model should correctly score:s(obtained great results on a)>s(obtained great Dresden on a), where the center word results was replaced with the ran-dom word Dresden. Any observed window is defined as x and a window corrupted with a random word as x_c. This score is computed by assigning each word a vector representation, giving these as inputs to a neural network and have that neural network output a score. Words are represented as vectors and retrieved from the L matrix. For each window, the system retrieves the word vectors of words in that window and concatenate them to a 5n-dimensional vector x. So, for the example score would be:
[2] $= [x_{obtained} x_{great} x_{results} x_{on} x_a]$. This vector is then given as input to a single neural network layer. The following equations compute a single hidden layer and a score:

$$z = W x + b \quad (2)$$

$$a = f(z) \quad (3)$$

$$s(x) = U^T a \quad (4)$$

Where size of the hidden layer is defined to be h and hence W = $R^{h\ 5n}$ and U 2 R^h. More compactly, this entire scoring function can be written as

$$s(x) = U^T f(W x + b)$$

The above equation described the so-called feedforward process of the neural network. The following function J is minimized for each window that can be sampled from a large text corpus:

$$J = \max(0; 1\ \ s(x) + s(x_c)): \quad (6)$$

4.3 Recursive Objective Function

The syntactic rules of natural language are known to be recursive, with noun phrases containing relative clauses that themselves con-tain noun phrases. The recursive structure in multiple modalities can be predicted using recursive neural networks (RNNs). Words are first mapped into a semantic space and then they are merged into phrases in a syntactically and semantically meaningful order. The RNN computes the three outputs

(i) a score that is higher when neighboring words(phrases) should be merged into a larger phrase,

(ii) a new semantic representation for this larger phrase,and

(iii) its class label. and attaches them to each node in the parse tree. The class labels are phrase types such as noun phrase (NP) or verb phrase (VP). The basic idea is to recursively merge pairs of word/phrase representations using following approach:

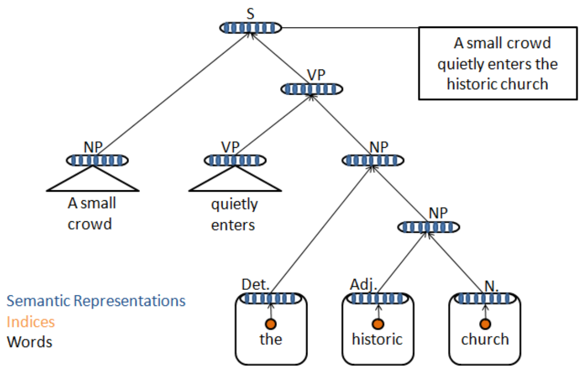

Fig. 3. Illustration of a recursive neural network architecture which parses natural language sentences

—Form syntactic binary tree.

—Compute parents by passing vector representations as inputs to the neural network.

—Repeat above step till you get the vector representation for the sentence.

We need two components:

—A model that merges pairs of representations.

—A model to determine tree structure.

4.4 Representations Are Merged?

The algorithm finds the pair of neighboring segments using the adjacency matrix A and adds their activations to a set of potential child node pairs:

$$C = f[a_i; a_j] : A(i; j) = 1g \quad (7)$$

Each pair of activations is concatenated and given as input to neural network.

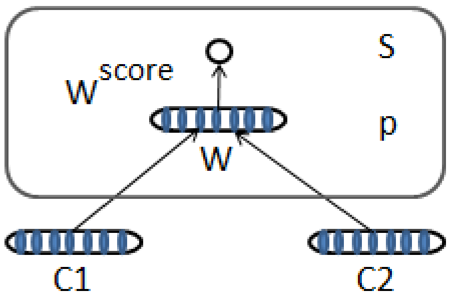

Fig. 4. One recursive neural network which is replicated for each pair of possible input vectors. This network is different to the original RNN formulation in that it predicts a score for being a correct merging decision

The network computes the potential parent representation for these possible child nodes:

$$p(i; j) = f(W [c_i; c_j] + b) \tag{8}$$

With this representation the system can compute a local score using a simple inner product with a row vector $W^{score} = R^{1\ n}$:

$$s(i; j) = W^{score} p(i; j) \tag{9}$$

Training will aim to decrease scores of pairs with different labels and increase scores of good segments pairs (with the same label). After computing the scores for all pairs of neighboring segments, the algorithm selects the pair which received the highest score. Let the score s_{ij} be the highest score, we then (i) Remove [ai; aj] from C, as well as all other pairs with either a_i or a_j in them. (ii) Update the adjacency matrix with a new row and column that reflects that the new segment has the neighbors of both child segments. (iii) Add potential child pairs to C:

$$C = C\ \ f[a_i; a_j]g\ \ f[a_j; a_i]g \tag{10}$$

$$C = C\ [\ f[p_{i;j}; a_k] : a_k \text{ has boundary with i or jg} \tag{11}$$

$$s(RNN(\ ; xi; y\hat{}\,)) =\ \ d2N(\hat{}y)sd: \tag{12}$$

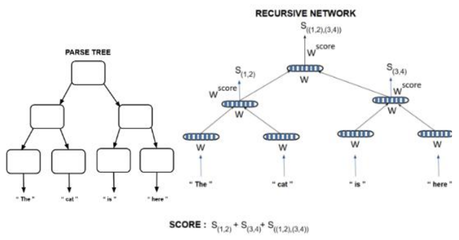

Fig. 5. The score of full tree is the sum of all merging scores.

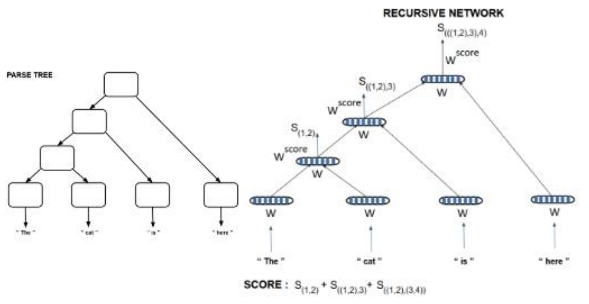

Fig. 6. Different parse tree for the above example.

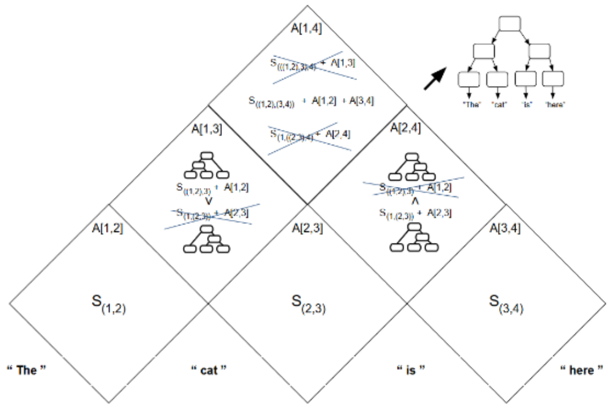

Fig. 7. Approximate best tree by locally maximizing each subtree

4.5 Labelling in RNN

One of the main advantage of this approach is that each node of the tree built by the RNN has associated with it a distributed representation (the parent vector p). This representation can be leveraged by adding to each RNN parent node (after removing the scoring layer) a simple softmax layer to predict class labels, such as syntactic categories:

$$label_p = sof\ tmax(W^{label} p:) \tag{13}$$

The error will backpropagate and influence both the RNN parameters and the word representations, when minimizing the cross-entropy error of this softmax layer .[5] Backpropagation through structure is a modification to the backpropagation algorithm due to the tree structure of recursive neural networks. There are two main differences resulting from the tree structure:

—Splitting derivatives at each node: During forward propagation, the parent is computed using the two children, hence, the error message that is sent down from the parent node needs to be split with respect to each of them. Hence the system will minimize the cross-entropy error.

—Summing derivatives of all nodes in the tree: Due to the recursion, the casual reader may assume that derivatives will be very complicated. However, a detailed look shows a simplifying equality: Assuming that the W matrices at each node are different and then summing up the derivatives of these different W's turns out to be the same as taking derivatives with respect to the same W inside the recursion.

4.6 Training RNN

Let y be the true parse tree and y^ be the predicted parse tree. We would like the score s(y) of y to be higher than the score s(^y) of y^ (unless y^ is actually y). To update the recursive network infer the predicted parse tree y^ and increase the score s(y) and decrease the score s(^y) by doing an update in the direction of the gradient

$$5\ s(y)\ \ 5\ s(\hat{}y) \qquad (14)$$

These gradient can be computed by backpropagating through the recursive network structured according to the parse trees y and y^.[5]

5. ACKNOWLEDGMENTS

We take this opportunity to express sincere thanks towards Prof. Jagruti Save, Head of Department of Information Technology, Fr. Conceicao Rodrigues College Of Engineering(FR.CRCE), for pro-viding the technical guidelines, and the suggestions regarding the line of this work and Srija Unnikrishnan, the Principal of FR.CRCE for the encouragement.

6. CONCLUSION

Most of the times, customers will not be aware of the product characteristics/features and may have to browse through multiple reviews on the web. The sentiment analysis model can provide a summarized view of the reviews, thus helping the customers to take better and informed decisions regarding the products. Also the proposed model analyzes dynamic reviews for a product using scrapy. This will ensure that the sentiment analysis is not done on stale data and provide a valid and concise view regarding the product. The results provided by using recursive neural network will be much better and thus provide most accurate rating for the product. In future, this algorithm can be enhanced to handle spam detection and fake reviews.

7. REFERENCES

[1] (2008) Sentiment analysis. http://www.egonomicslab.com/internet-of-things/sentimentanalysis/.

[2] N. K. S. Gaurav Dubey, Ajay Rana, “User reviews data analysis using opinion mining on web,” in 1st International Conference On Futuristic Trend In Computational Analysis And Knowledge Management, 2015.

[3] R. T. Deepali Virmani, Vikrant Malhotra, “Aspect based sentiment analysis to extract meticulous opinion value,” International Journal of Computer Science and Information Technologies, vol. 5(3), pp. 3262 – 3266, 2014.

[4] J. C. C. M. A. N. C. P. Alex Perelygin, Jean Wu, “Recursive deep models for semantic compositionality over a sentiment treebank,” august 2015.

[5] R. Socher, “Recursive deep learning for natural language processing and computer vision,” Ph.D. dissertation, Standford University, Stanford, CA 94305, USA, 2014.