Emotion Analysis using Word Embedding and Neural Network

Arora, Pragya*, Ghai, Piyush†, Gupta, Harsh‡ and Ramkrishnan, Navnith§
Department of Computer Science & Engineering, The Ohio State University
Columbus, OH 43202
Email: *arora.170@osu.edu, †ghai.8@osu.edu, ‡gupta.749@osu.edu, §ramkrishnan.1@osu.edu

***Abstract*—Human computer interaction and its applications such as chat bots and voice assistants will be more natural if computers are able to perceive and respond to human non-verbal communication such as emotions. Although a lot of work has been done in the field of sentiment analysis but relatively little work has been done to detect emotions such as joy, anger, surprise in a text. This paper tries to extract the embedding based on words from a text document and apply state of the art neural network techniques to predict what emotion the writer wants to convey from the text. We experimented on two approaches along with a baseline approach to extract emotion decision using word embeddings. We used ISEAR dataset to label seven emotions. The emotion classes are sadness, anger, joy, shame, disgust, guilt and fear. The results reveals that the system based on Long Short Term Memory (LSTM) gave better performance than the system based on Convolution Neural Networks (CNN) for the emotions considered. Results also show when our feature engineered vectors and Word2Vec were jointly learned, the performance and the robustness of the emotion recognition system improve measurably.**

I. Introduction

Emotion Analysis and Identification is the cornerstone of recent study in the fields of neuroscience, psychology, cognitive sciences and behavioral science as they serve as a good indicator of human nature. The major techniques to predict emotion are on the basis of supervised learning through hand annotated data to train and build predictive models. Despite these models achieving good results, the availability of large sets of tagged data and transferability of performance on different domains have been deterrents.

In this paper, we use the ISEAR (International Survey on Emotion Antecedents and Reactions) Dataset [1], which comprises of 7516 sentences and seven emotion categories including : *Anger*, *Disgust*, *Fear*, *Guilt*, *Joy*, *Sadness*, *Shame*.

II. Related Work

This section expands on lexical resources and research work supporting work on Emotion Analysis from a corpora and gives a brief about other such novel methodologies.

A. Lexical Resources

Lexical resources have long been used for emotion and opinion analysis. Strides in lexical resources began with a list of 1,336 hand annottated adjectives. Other advances include WordNet - Affect which comprises of hierarchial domain of affective labels. In the sphere of sentiment, SentiWordNet [2] accounts for opinion related properties of text. SentiFul database further assisted with automatic generated lexicons. Researchers came up with subjectivity lexicon which accounted for 8,000 words.

B. Emotion Detection Approaches

Classification of Emotion can be done based on the presence or absence of affective lexicons. Further distinction can be done based on keywords, linguistic rules and the application of machine learning approaches.

1) Keyword-based Techniques using Affective Lexicons: Keyword-based approach is the most primitive form implemented at the base word level and such models are incapable of handling affect expressed by interrelated words.

2) Linguistic Rules-based Approaches: Computational linguists define linguistic structure based on a set of rules.

- Rule-based approaches with Affect Lexicons: Classification of emotions in news headlines is the main purpose behind the ESNA System [3]. UPAR7 has hand annotated seed words in emotion lists aided by rules which outline the main context and use dependency graphs to boost emotion ratings. Recent strides in rule-based approaches recognize up to nine emotions. General Inquirer and Wordnet explore the effect of conjuncts using techniques of syntax and lexical resources. The definition and deciphering of complex structures are well handled in comparison to designing and modification by these approaches. One flaw in affect lexicons is the inflexibility to handle emotions not listed.
- Rule-based approaches without Affect Lexicons: In the absence of affect lexicons, semantic structure of language is the crux. Metaphorical data further aid in emotion recognition from text. These methods are flexible and relevant in real world implementation. The rules are structured and represent the extracted data in concurrence with the source.

C. Machine Learning Approaches

Machine learning techniques which can be broadly classified as supervised and unsupervised machine learning has been deployed to overcome the short falls of rule based methods which do not take into account all the possibly corner cases and is short at capturing the evolution of writing texts by human.

1) Supervised machine learning with affect lexicons: Under this approach a database is created for the affect words where each affect word is divided into smaller classes to make it a find grained resource and now this resource is used for multi class classification of emotions. Models like Support Vector Machine is used under this technique. For supervised learning the problem is that we need a lot of data to train the classifier and then the trained model works on a domain specific data set, which is its disadvantage.

2) Supervised machine learning without affect lexicons: Using a BOW approach or a tf-idf approach works well even if we don't consider the affect words while training the model. Even this kind of technique suffers from the bias of giving good results only for data coming from a specific origin.

3) Unsupervised machine learning with affect lexicons: An evaluation of two unsupervised techniques using WordNet-Affect exploited a vector space model and a number of dimensionality reduction methods. News headlines have been classified using simple heuristics and more refined algorithms (e.g., similarity in a latent semantic space).

4) Unsupervised machine learning without affect lexicons: Latent Semantic analysis single word approach which compares the similarity between text documents and all the emotion labels along with LSA synsets and WordNet synsets.

III. Proposed Approach

In our approach we have modeled our framework to run on state of the art machine learning algorithms and learn the context of the data using their vectors from Word2Vec [4] & GloveVec [5]. After the model is trained on several hyper parameters we tested our model on unseen data set and calculated the accuracy which is better than baseline given in related papers. Our emotion recognition algorithm includes three main components: preprocessing, semantic transformation of Word embedding and application of Machine Learning algorithm.

A. Preprocessing & Word Embedding

We used ISEAR data set to test our model. This dataset has sentences tagged with an emotion label. The preprocessing task consists of word level parsing along with tokenization and stemming which cleans the text document of unwanted stop words which don't relate to emotion. This enables us to extract the relevant affect-bearing words and the syntactic dependencies between words. The next step is to perform word-level analysis by computing a word embedding for the affect-bearing words by using the glove vector and also alternately learning the word embedding for this model by itself.

The transcript of the text documents were extracted, converted to all lower case characters. We did not remove punctuations here, because our intuition was that these punctuation marks can have a bearing on the emotion classification of a sentence. We then preprocessed the sentences using NLTK Tokenizer to give a list of tokens. This list of tokens was converted into a word embedding using Word2Vec model for (CNN) and Glove model (LSTM). For each word a 300 dimensional embedding was created. This could have been tweaked, however for simplicity, we stuck with a 300 dimensional vector. [7]

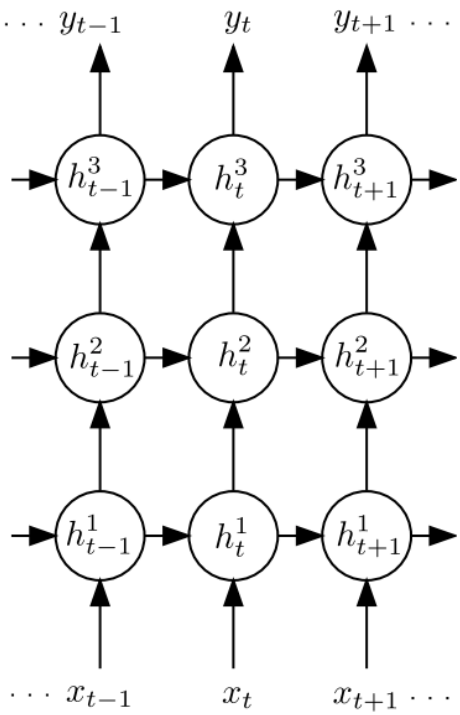

Figure III.1: LSTM model for word embedding

B. Application of Machine Learning Techniques

After the word embeddings are prepared, the word vectors are given as an input to our neural network models. The model is then trained on several hyper parameters like the number of epochs, number of layers, drop out probability, activation function, loss function, optimizer and learning rate.

1) Long Short Term Memory(LSTM)[6]: We have mapped each sentence into a real vector domain called GloVe embedding. This is a technique where words are encoded as real-valued vectors in a high dimensional space, where the similarity between words in terms of meaning translates to closeness in the vector space. We have mapped each word into 300 length vector and there are around 9000 unique words so a matrix of 9000x300 is created. This positive integer representation of words is fed into Embedding layer which converts it into word embedding. The first layer is the Embedded layer that uses 300 length vectors to represent each word. The next layer is the LSTM layer with 128 memory units and 0.6 dropout to input and recurrent connections of the memory units with LSTM. Finally, because this is a classification problem we use a Dense output layer with seven neurons, a softmax activation function with categorical crossentropy and ADAM optimizer. A large batch size of 128 reviews is used to space out weight updates. We observed that applying dropout as a parameter to the LSTM Model led to an increase in the accuracy.

In Figure III.1 xt's are word embedding which is 300 dimensional vector and y's are the output from the output layer. In our LSTM model we have three hidden layers which

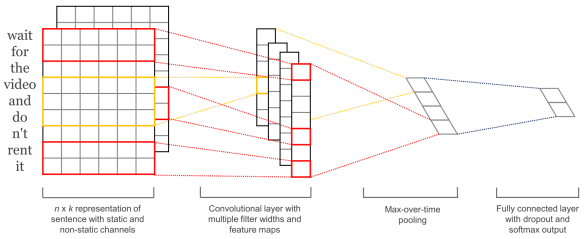

Figure III.2: CNN model

included embedding layer, lstm layer, dense output layer. The output is passed through a softmax function which gives us a 7 dimensional vector as the output.

2) Convolutional Neural Networks: For implementing CNN, we implemented a model similar to Kim Yoon's Convolutional Neural Networks for Sentence Classification [8] [9]. The CNN architecture used by us for the emotion analysis task is given in Figure III.2. The network used by us works roughly as follows :

- The first layer embeds the words into lower dimensional vectors.
- The next layer performs convolutions over the embeddings from the first layer using multiple filter sizes.
- Next up, we max-pool the result of the convolutional layer into a feature vector, over which we add dropout regularization and use softmax at the output layer to classify the result.

IV. Experimental Setup

We run our dataset on several models. All our models presented below were evaluated on a 70-30 train-test split and 90-10 train-dev set split.

A. Dumb Baseline

The dumb baseline model, predicts the emotions at random for a text from the given 7 emotion labels. This approach, not too surprisingly performs poorly.

B. Multinomial Naïve Bayes

We used TFIDF Vectorizer for Multinomial Naïve Bayes which gives us an accuracy of 0.567 on the test dataset. We also used Multinomial Naïve Bayes with a Bag of Words approach. This performed slightly lower than the TFIDF Vectorizer and we got an accuracy of 0.562. The results are summarized in Table IV.1.

C. Extra Tree Classifier

We also used Extra tree classifier (Random Forest Classification). With this classifier we changed the word representations using Word2Vec and GloveVectors representation. After changing the word representations, TFIDF Vectorizer was used. The results were not very encouraging with this though. We achieved a maximum accuracy of 0.4313 with Glove representation of the input text. The results are summarized in Table IV.1.

D. SVM

We next implemented a multi-class, one versus all classifier. The input to the SVM was a TFIDF vectorized representation of the words. The accuracy for One v/s All SVM was 0.56489.

Table IV.1: Accuracy on ISEAR Test Dataset for several algorithms

Model	Processing Approach	Accuracy
Dumb Baseline	–	0.194
Random Forest	Word2Vec GloveVec	0.567021 0.5625
Naïve Bayes	With TF-IDF BOW	0.567021 0.5625
SVM	One v/s All	0.5648

E. CNN

For Convolutional Neural Networks, we did not use deep networks, but a shallow network and experimented with other hyperparameter tuning as well as experimenting with the word embeddings. We fixed the sequence length to 75 words. For shorter words, we padded the matrix with all zeros. 75 words as per our intuition was long enough text to judge the emotion of a sentence. The vocab size was 9000. We also varied the filter sizes, which were convolved over the words. The filter sizes were taken as *3,4,5*. The number of filters were taken as 32. In the embedding layer, we used Word2Vec representation of our dataset and we also experimented by taking random initializing of the word vectors and trained the model to learn the representations on the given inputs. For the output layer, we used softmax as the activation function. The activation used in the intermediate layers was *Relu* Activation.

For hyperparameter tuning and generating accuracy, we split the training dataset into training and testing dataset present the accuracy on both dataset. The following parameters were experimented with : (The best values are indicated alongside)

- Number of Epochs : e {50}
- Batch Size : b {128}
- L2 Regularization : l2 {0.60}
- Dropout Probability : d {0.6}

For word embeddings, we used pre-trained *word2vec* vectors and also tried with learning embeddings from scratch. Not surprisingly, the model with word2Vec embeddings performed better in terms of overall accuracy. The results are presented in Table IV.4 & Table IV.3. The training accuracies are very high suggesting overfitting for the model. The training accuracy increases, while the test accuracy decreases indicating overfitting as given in Table IV.4. The same trend is followed when using with Word2Vec embeddings, however, we saw an improvement in the accuracy on the test dataset over the previous models used so far.

Table IV.2: Accuracy on ISEAR Test Dataset for CNN (With No pre-trained word embeddings) with hyperparameter tuning

Hyperparameters	Test Accuracy	Training Accuracy
e=50, b=30,l2=0.3,d=0.6	0.5877	0.85230
e=30, b=30,l2=0.5,d=0.6	0.5864	0.8872
e=50, b=128,l2=0.6,d=0.6	0.5890	0.9053721
e=40, b=128,l2=0.8,d=0.6	0.566489	0.955217

Table IV.3: Accuracy on ISEAR Test Dataset for CNN (Using Word2Vec trained word embeddings)

Hyperparameters	Test Accuracy	Training Accuracy
e=40, b=128,l2=0.5,d=0.6	0.59707	0.94812
e=30, b=128,l2=0.3,d=0.6	0.60106	0.94634
e=50, b=128,l2=0.6,d=0.6	0.63757	0.949
e=50, b=128,l2=0.8,d=0.6	0.56909	0.9846

F. LSTM

For LSTM, we used GloveVector as word embeddings. The arhictecture for the model used has been described above. For predicting the best accuracy, the model was trained using a 90:10 training and test split, 80:20 training and test split and finally 70:30 training and test split. The validation split was fixed at 90:10 for training and dev set. The dev set was used to tune the hyperparameters. We used Adam Optimizer as the gradient descent optimization. The following were the hyperparameters of the model that we played around with : (The best values are indicated alongside.) The activation function for output layer used was Softmax activation and the loss function used was categorical cross entropy loss. Also, compared to the CNN Model, the LSTM architecture model does not seem to overfit on the given dataset, indicating that LSTM performs better.

- Number of Epochs : e {40}
- Activation Function ac {softmax}
- Dropout Probability : d {0.6}
- Recurrent Dropout : ld {0.6}

Table IV.4: Accuracy on ISEAR Test Dataset for LSTM (Using GloveVec trained word embeddings)

Hyperparameters	Test Accuracy	Training Accuracy
e=30,b = 128, d=0.5, ac=sigmoid	0.5745	0.8527
e=50,b = 128, d=0.5, ac=sigmoid	0.5717	0.947
e=30,b = 100, d=0.6, ac=softmax	0.5971	0.8952
e=30,b = 256, d=0.6, ac=softmax	0.5984	0.8765
e=30, LSTMDropout : 0.5, recurrent dropout:0.5	0.637	0.7437
e=40, LSTMDropout : 0.6, recurrent dropout:0.6	0.6463	0.7288

V. Conclusion

This paper addresses an important and less examined area of sentiment research on emotions, that is, emotion detection from text. The major contribution of this project is to highlight the application of neural networks to learn semantics from a data in order to identify and distinguish various types of emotions in text. We presented Long-Short Term Memory (LSTM) model which is essentially a specific kind of Recurrent Neural Networks (RNNs) and Convolutional neural network (CNN) to detect emotions from text. Experimental results show that our models perform better than other existing Machine learning approaches. Among our approaches, LSTM outperformed CNN and achieved an accuracy of 64.63%. We tried to experiment by adding more convolutional layers to the CNN Model, but that did not boost the test accuracy a lot, instead the trend of overfitting was observed.

VI. Future Works

Emotion detection has been studied. Although not enough time has passed to have established standards in the field, there is some consistency between the approaches, and the algorithms are continuing to increase in accuracy by using more of machine learning approach in this domain. As our future work we would like to work on better text filtering techniques to understand the underlying semantic in documents. We would like to efficiently handle negation phrases in texts like : *The movie was surprisingly horrible*, *He is horribly good*. The former example is a classic example of a negative emotion, whereas the latter is a use of a misnomer. Learning with these bigram and other n-gram variations will help us increase the overall accuracy of the framework. We would also like to make our word embedding more robust by using a special kind of tagging approach called NAVA (Noun Adjective Verb Adverb). NAVA approach has been in use for emotion detection tasks to extract emotion bearing phrases, because some words express affect less apparent then others. Some words express affect more apparently than others. Lets consider the sentence, *Sheldon got lots of new toys for his birthday*. Here, the words **new**, **toys** and **birthday** together convey happiness although it may not seem so when these words are looked at individually. We will also like to use paragraph vector embedding instead of Word embedding for this task.

References

[1] The ISEAR Dataset is available from : http://emotion-research.net/toolbox/toolboxdatabase.2006-10-13.2581092615

[2] SentiWordNet is a lexical resource for opinion mining. More about it can be found at : http://sentiwordnet.isti.cnr.it/

[3] S. Mostafa Al Masum, H. Prendinger and M. Ishizuka, "Emotion Sensitive News Agent: An Approach Towards User Centric Emotion Sensing from the News," Web Intelligence, IEEE/WIC/ACM International Conference on, Fremont, CA, 2007, pp. 614-620. doi: 10.1109/WI.2007.124

[4] Word2Vec tool provides an efficient implementation of the continuous bag-of-words and skip-gram architectures for computing vector representations of words. These representations can be subsequently used in many natural language processing applications and for further research. https://code.google.com/archive/p/word2vec/

[5] GloVe is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space. https://nlp.stanford.edu/projects/glove/

[6] The following is a good link about understanding LSTMs : http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[7] A Keras implementation of using pre-trained word embeddings. https://blog.keras.io/using-pre-trained-word-embeddings-in-a-keras-model.html

[8] Yoon Kim, Convulational Neural Networks for Sentence Classification, http://arxiv.org/abs/1408.5882

[9] An implementation for CNNs for Sentence Classification can be found at : http://www.wildml.com/2015/12/implementing-a-cnn-for-text-classification-in-tensorflow/