Team-18

IRE Major Project

Abstract

Most of the tasks in natural language processing involve looking at individual words and try to extract information from different factors, namely their distribution in the documents,syntactic contexts,their frequency weighting but always ignore the sentiment of the continuous form of these words. Here we want to come up with a method that learns both the semantic and the sentimental information in a conversation and use it for a better emotion detection of a three-way conversation.

Keywords : sentiment analysis, word embedding, multi-class classification, semantic analysis,

Problem Statement

Given a textual user utterance along with 2 turns of context in a conversation, classify the emotion of user utterance as Happy, Sad, Angry or Others. This task is part of the SemEval Contest, accessible here.

Background

The task, as introduced by Microsoft Research India is an extension of the paper *"A Sentiment and Semantics Based Approach for Emotion Detection in Textual Conversations", Gupta et. al. (2018)*. The paper describes an LSTM based architecture to solving the aforementioned task, and involves the usage of both semantic word vectors (in the form of GloVe embeddings) and sentiment word vectors. This approach shows good results with an average F1 score of 71.34, beating a number of previously set standards.

Theory

Long Short Term Memory Networks (LSTM)

*LSTM*s are a special kind of recurrent neural network (RNN) capable of learning long-term dependencies. They are explicitly designed to avoid the long-term dependency problem and the vanishing gradient. Given the highly contextual nature of our data wherein we get three turns of conversation, LSTMs were expected to perform better than other Deep Learning architecture involving SVMs , CNNs and the likes.

*Glo*bal *Ve*ctors for Word Representation (GloVe)

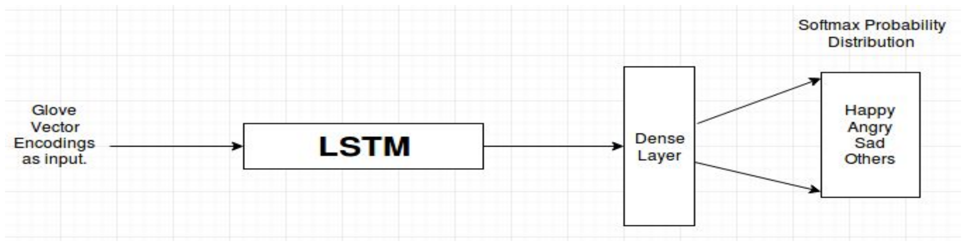
Distributional semantics refers to the theory that the meaning of words in languages can be derived from their distribution in language use, i.e. the occurrence of words with respect to their shared contexts.
GloVe is an unsupervised learning algorithm for obtaining vector representations for words based on their distribution. Pre-trained GloVe representations have been used throughout our project, as obtained from here.

Approaches

We now proceed to describe the various approaches we have currently employed, in chronological order, ending with the current best approach. The F1 scores, as reported on the contest portal, have also been mentioned.

Model 1 : F1 = 56.73%

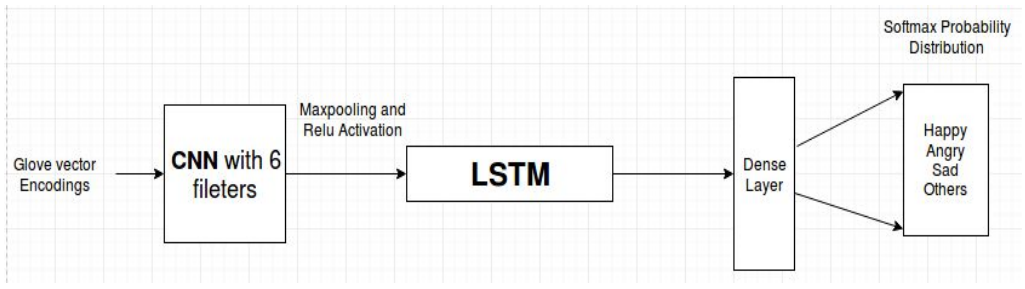
A simple LSTM followed by a dense layer, which utilizes a sigmoid activation function to give class-wise probabilities. The LSTM helps to retain some contextual information as explained earlier. We treat this is as our baseline.


Softmax Probability Distribution
Glove Vector Encodings as input.
LSTM
Dense Layer
Happy
Angry
Sad
Others

Model 1 Architecture

Model 2 : F1 = 55.xx%

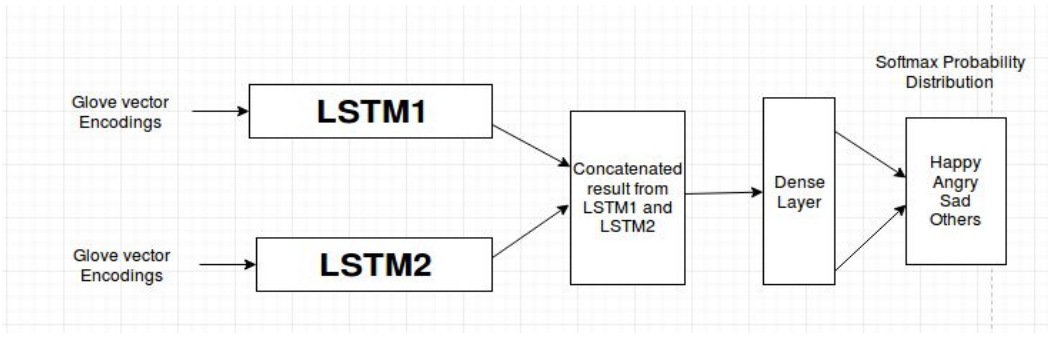
LSTMs, due to their recurrent behavior take time to train. Thus, we decided to experiment with CNNs. The second model comprises of a CNN having 6 filters, a max-pool, and then an LSTM followed by a dense layer and finally a four class softmax probability distribution layer. Although, we were expecting an increase in the F1 scores, the scores dipped a little bit, giving 55% as the results. The reason could have been that, the feature space was cut down too much in max-pooling that it even excluded some of the important contextual features which might have been useful for the LSTM.



Model 2 Architecture

Model 3 : F1 = 58.XX%

We followed with this architecture inspired by Siamese networks. Two LSTMs are applied in parallel, each fed with GLoVe vectors as inputs through the embedding layer. Now unlike Siamese, here, the outputs from the two parallel LSTMs are concatenated together and fed to a dense layer which is then followed by a four class softmax probability distribution layer.
This architecture will serve as a groundwork for the architecture of LSTMs as would be discussed further. In those models, we would be feeding different types of word embeddings so that both semantic and sentimental aspects of the text can be covered and passed on LSTM to work upon. However, for now, just for a trail run and see how much this model alone can improve the accuracy, we ran this model on test dataset and found it to be giving good increase than the one that used a CNN before an LSTM, Model 2.



Model 3 Architecture

Model 4 : F1 = 59.42%

Instead of trying to change the model we decided to try some feature engineering and make the model semi-supervised. First every word was represented by its GloVe vector. Then, a feature vector of nine binary (i.e. true/false type) features was generated for every word. These features have been listed below:

1. Is the word a *hedge*?
2. Is the word a *factive verb*?
3. Is the word an *assertive verb*?
4. Is the word an *implicative verb*?
5. Is the word a *report verb*?
6. Is the word an *entailment causing verb*?
7. Is the word a *subjective verb*?
8. If yes, is it *weakly subjective* or *strongly subjective*?
9. Is the *polarity* of the word *positive* or *negative*?

The reasoning employed behind the use of these features is that sentiment is highly subjective and these features all give insights into displays of subjectivity in language use (these features have seen usage for identification of bias in text as well).
Once the feature vector has been generated, it is concatenated to the regular GloVe vector. We believe our reasoning is justified as when the 100 dimensional GloVe embeddings in model 1 were replaced with these feature augmented embeddings, the F1 score immediately rose from ~56% to ~58%.

A further replacement of the 100 dimensional embeddings with 300 dimensional GloVe embeddings, in the same model, further improved our score to >59%.

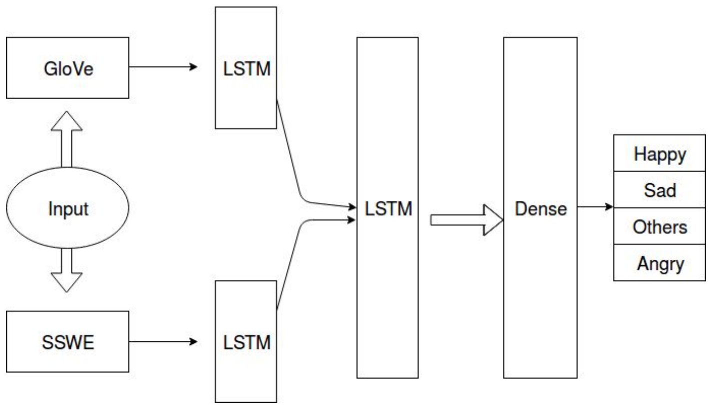
Model 5 : F1 = 61.15%

At the end of model 4, we redirected our efforts to increasing our model's focus on context. For this, we decided to replace the simple LSTM in model 4 with a Bidirectional LSTM (BiLSTM). The idea is that an LSTM only looks at past events as context, however just like language contains *anaphoric* references, it also contains *cataphoric* references, and so our model should, in some capacity, also focus on future events. Since the BiLSTM looks at inputs in both *directions*, it seemed like a good choice; unfortunately though the model failed catastrophically, giving extremely low results. While we haven't completely ascertained what went wrong, we believe part of the problem is that the model is overfitting.

The above failure also led to us trying a second idea; we now stack a second LSTM layer on the pre-existing layer as in model 4. The reasoning behind this is that we felt that our model did not have enough depth and complexity to actually make inferences from the temporality of the data it was being given. To avoid overfitting, we raised the dropout from the previous 20% to 30%. Rest everything remains the same; words are encoded as combinations of GloVe and the features (as detailed in model 4), and after passing through two hidden LSTM layers, they go through a dense layer with a sigmoid activation function, extrapolating our model's analysis into a probability distribution over the classes.

Model 6 : F1 = 66.09% (current best approach)

Up until model 5, we were using 300 dimensional GloVe embeddings for our model which was aimed at making our model predict the emotions based on the semantics. We wanted to exploit the advantages of semantics and sentiment of the conversation and hence introduced a Sentiment Specific Word Embedding (SSWE). SSWE aims at encoding sentiment information in the continuous representation of words. Pretrained SSWE were obtained from the authors of this paper.



The architecture of our proposed SS-LSTM model is shown in figure. The input utterances are now fed into two parallel hidden LSTM layers , one being the Semantic LSTM layer and the other being the Sentiment LSTM layer. These two layers learn semantic and sentiment feature representation and encode sequential patterns in the user utterance. These two representations are then concatenated and fed into another LSTM layer followed by a dense layer to model the interactions between semantic and the sentiment data to get the output probabilities for each of the four classes. GloVe is used our embedding for the Semantic LSTM layer and SSWE as our embedding for the Sentiment LSTM layer.

This has been the best approach so far, and gives us our current best score of **66.09%**.

Conclusion

Based on the current models, the best ways of improving performance seem to be directly correlated to three things - the amount of importance given to the context, semi-supervization via external feature generation, and hyperparameter adjustments. At the moment we have a number of other ideas that seem worthy of trial; future approaches will also involve more complex architectures including hierarchical models, and manipulation of training data.

Results of our models are up on the CodaLab results page here, assigned to the following usernames: *the0ne (Aditya)*, *KB09 (Ayush)* and *WinterRR (Hemanth)*.

Links

- **GitHub**
- **Project Website**
- **Video Demo**

References

- *A Sentiment and Semantic Based Approach for Emotion Detection in Textual Conversation* (Gupta et al.)
- *Learning Sentiment-Specific Word Embedding for Twitter Sentiment Classification*
- *Emotion Detection from Text*