Emotions from Text

Course: SMAI

Submitted to:

Prof. Ravi Kiran Sarvdevbhatla

TA Mentor:

Himanshi Sharma

Submitted By:

Padma Dhar(2018201011)

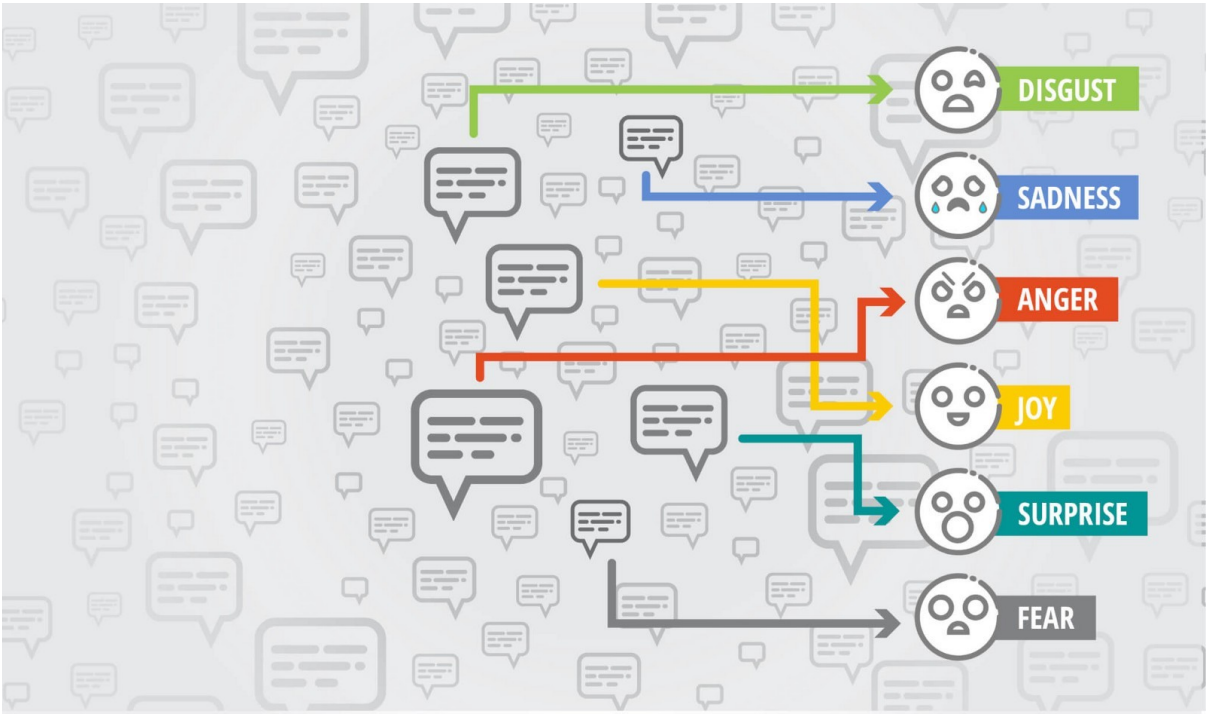
Divyanshi Kushwaha(2018201046)

Priya Upadhyay(2018202012)

Himani Gupta(2018202014)

Problem Statement

Text not only communicates information and data and content, but also the moods and the emotions of the person involved. Emotion classification from text consists of taking an input phrase from the user and the system is a built such that it can predict the emotion of the phrase.



Dataset

Two datasets were used, one for single emotion classfication, second dataset is for multi-label classification.

Single emotion classifer dataset

Dataset reference: https://github.com/yanghoonkim/attnconvnet

Dataset Unique emotions-

- Joy

- Anger
- Disgust
- Fear
- Shame
- Guilt
- Sadness

Dataset Size: 7652 samples

Multiple emotions classifer dataset

Dataset reference(for multilabel): https://github.com/tpsatish95/emotion-detection-from-text/blob/master/datasets/Emotion%20Phrases.csv

Dataset Unique emotions-

- Anger
- Anticipation
- Sadness
- Love
- Fear
- Joy
- Disgust
- Optimisim
- Pessimism
- Surprise
- Trust

Dataset Size: 6839 samples

Data Pre-processing

Tokenization—convert sentences to words

Removing unnecessary punctuation, tags

Removing stop words—frequent words such as ”the”, ”is”, etc. that do not have specific semantic

Stemming—words are reduced to a root by removing inflection through dropping unnecessary characters, usually a suffix.

Lemmatization—Another approach to remove inflection by determining the part of speech and utilizing detailed database of the language.

Remove numbers- Remove numbers if they are not relevant to your analyses. Usually, regular expressions are used to remove numbers.

Convert to lowecase.

Word Embedding

Word embeddings are of two types-

1. Sparse

- Bag of Words(TF-IDF)

2. Dense

- Tokenizer API by Keras

- GloVe Embbeding

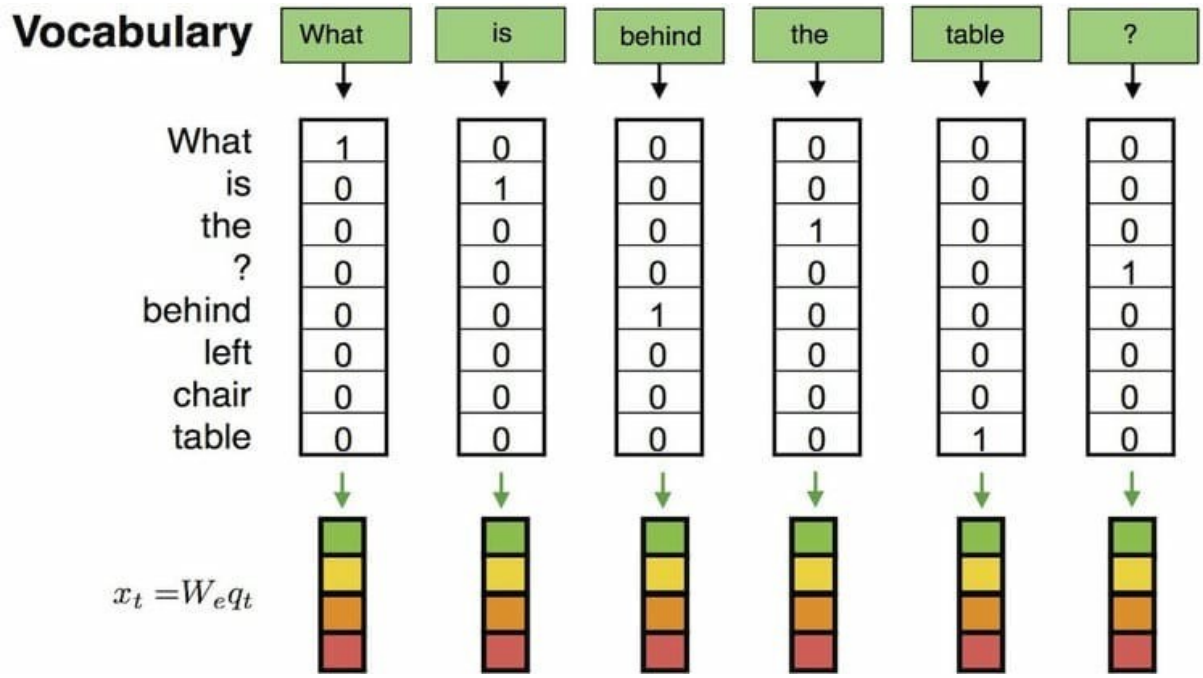
TF-IDF(Sparse Embedding)

One of the simplest techniques to numerically represent text is Bag of Words.

Bag of Words (BOW): We make the list of unique words in the text corpus called vocabulary. Then we can represent each sentence or document as a vector with each word represented as 1 for present and 0 for absent from the vocabulary. Another representation can be count the number of times each word appears in a document. The most popular approach is using the **Term Frequency-Inverse Document Frequency (TF-IDF)** technique.

- **Term Frequency (TF)** = (Number of times term t appears in a document)/(Number of terms in the document)
- **Inverse Document Frequency (IDF) =** log(N/n), where, N is the number of documents and n is the number of documents a term t has appeared in. The IDF of a rare word is high, whereas the IDF of a

frequent word is likely to be low. Thus having the effect of highlighting words that are distinct.



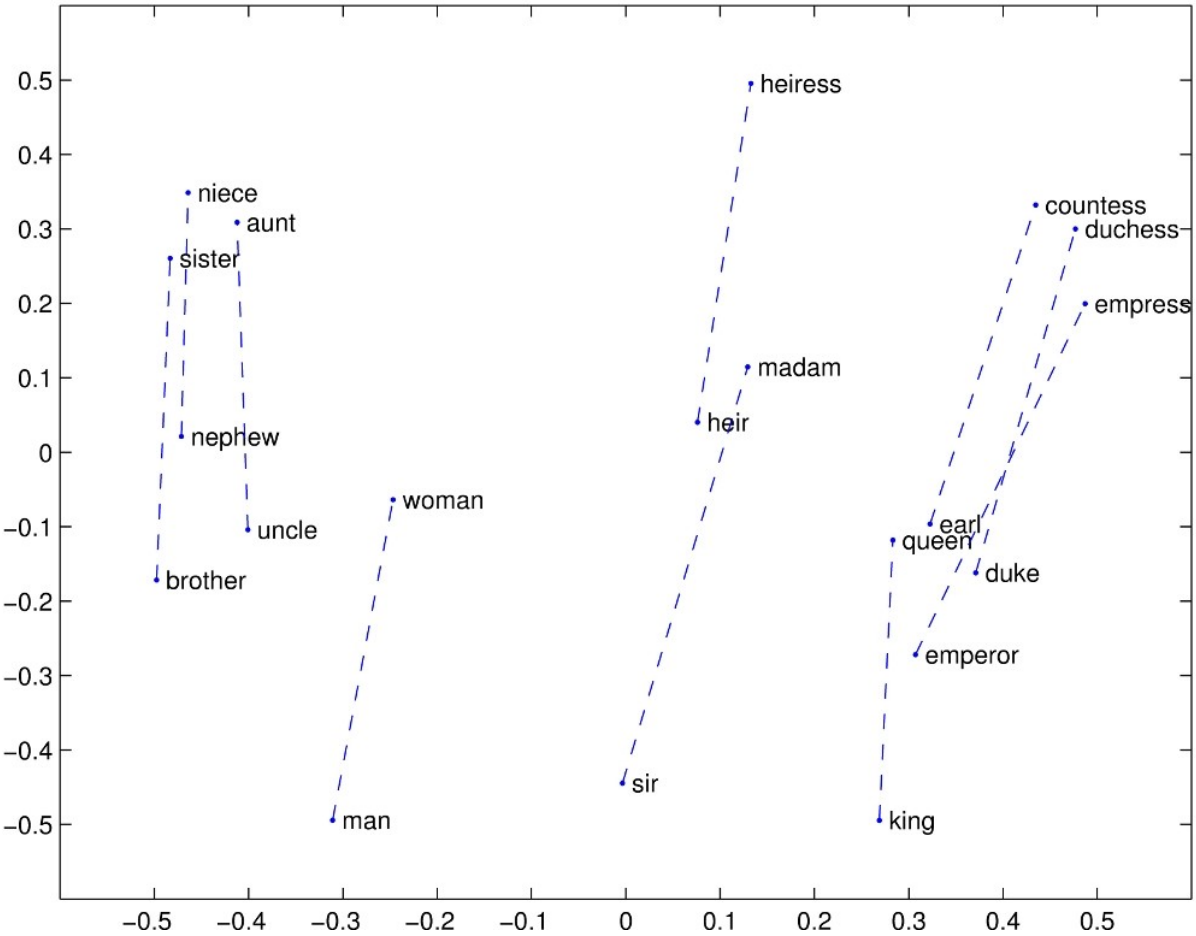
Tokenizer API by Keras

Keras provides a more sophisticated API for preparing text that can be fit and reused to prepare multiple text documents. This may be the preferred approach for large projects.

Keras provides the Tokenizer class for preparing text documents for deep learning. The Tokenizer must be constructed and then fit on either raw text documents or integer encoded text documents.

GloVe Embedding

Goblal Vectors for Word Representation. 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.



Projection for similar words are collinear.

Machine Learning Models for text classification

We experimented with various machine learning models, to find out which model gives the best performance with respect to our dataset.

We tested with following models-

Single Emotion Classification

1. Naive Bayes
2. K-nearest Neighbours
3. LSTM
4. CNN-LSTM
5. LSTM-CNN
6. CNN-LSTM (with GloVe embedding)
7. LSTM-CNN (with GloVe embedding)

Multi-label Emotion Classification

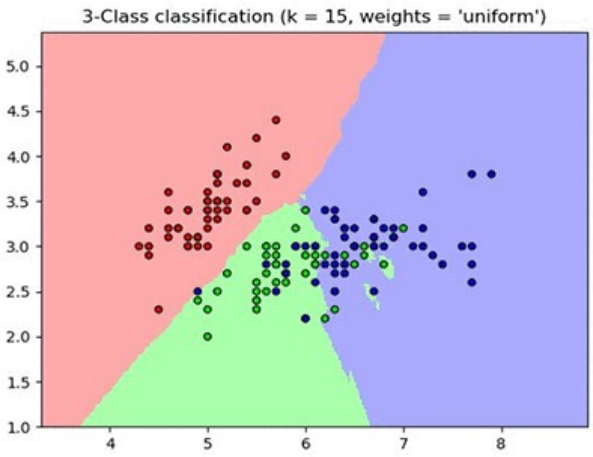
8. Naive Bayes for multilabel classification
9. K-nearest Neighbours for multilabel classification

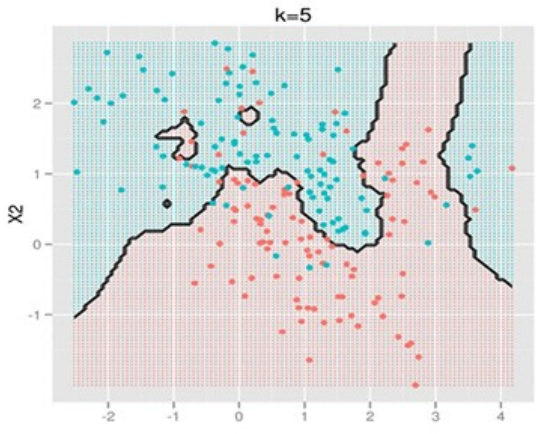
K- Nearest Neighbour

In pattern recognition, the *k*-nearest neighbors algorithm (*k*-NN) is a non-parametric method used for classification and regression. In both cases, the input consists of the *k* closest training examples in the feature space.

KNN algorithm is one of the simplest classification algorithm. Even with such simplicity, it can give highly competitive results.

Thus, KNN can be used to cluster text with similar emotions. Before applying th eknn, we have done the preprocessing techniques of tokenisation, lemmatisation and Term frequency Inverse document frequency. Using TFIDF, a text can be represented as a vector in vector space. The words that are closely related are closer together in the vector space while unrelated words are represented further away. Simmilarity metric used in the knn is the distance between the two points in the vector space which is Minkowski.





Naive Bayes

The Naive Bayes algorithm is an intuitive method that uses the probabilities of each attribute belonging to each class to make a prediction. It is the supervised learning approach you would come up with if you wanted to model a predictive modeling problem probabilistically.

Naive bayes simplifies the calculation of probabilities by assuming that the probability of each attribute belonging to a given class value is independent of all other attributes. This is a strong assumption but results in a fast and effective method.

The probability of a class value given a value of an attribute is called the conditional probability. By multiplying the conditional probabilities together for each attribute for a given class value, we have a probability of a data instance belonging to that class.

To make a prediction we can calculate probabilities of the instance belonging to each class and select the class value with the highest probability.

The steps to do the Naive Bayes classifier for our system is as follows:

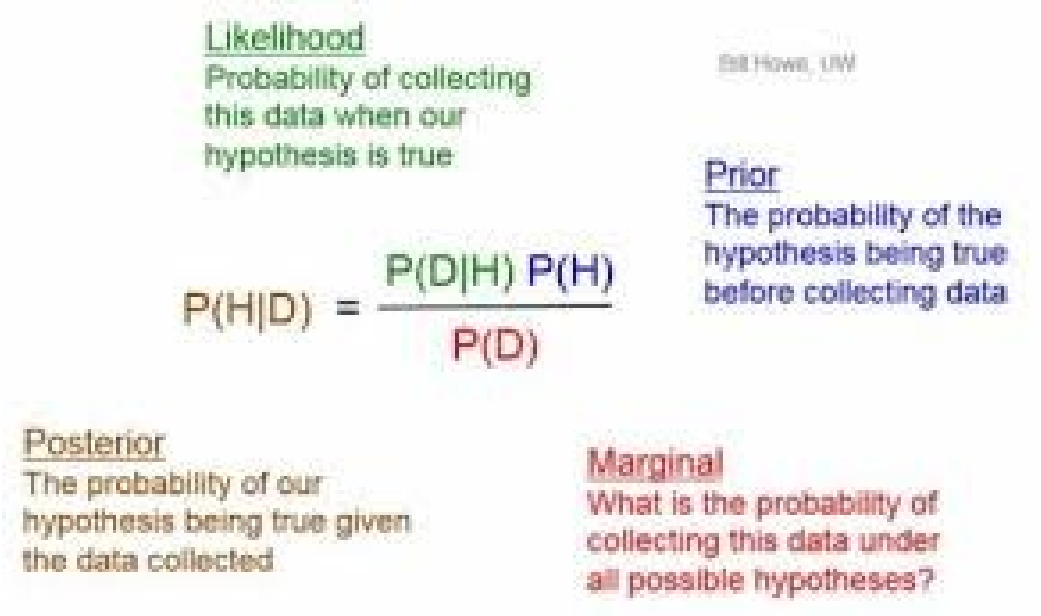
1. For every emotion, we calculate the prior probability.

2. Find the likelihood probability with each phrase for each emotion.

3. Put this in Bayes formula and calculate posterior probability

ADVANTAGES:

Fast, reliable and accurate in a number of applications in NLP

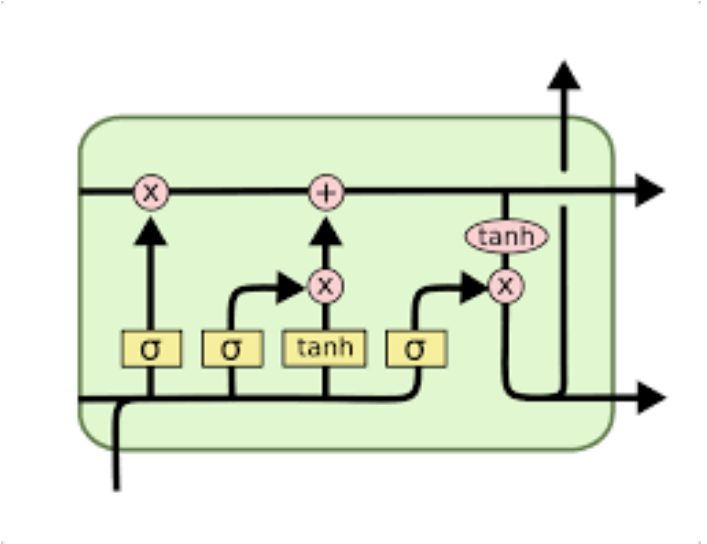


LSTM

Long short-term memory (LSTM) is an artificial recurrent neural network, (RNN) architecture used in the field of deep learning capable of learning long-term dependencies. . Unlike standard feedforward neural networks, LSTM has feedback connections that make it a "general purpose computer" (that is, it can compute anything that a Turing machine can). It can not only process single data points (such as images), but also entire sequences of data (such as speech or video).

A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell remembers values over arbitrary time intervals and the three *gates* regulate the flow of information into and out of the cell.

LSTM networks are well-suited to classifying, processing and making predictions based on time series data, since there can be lags of unknown duration between important events in a time series.

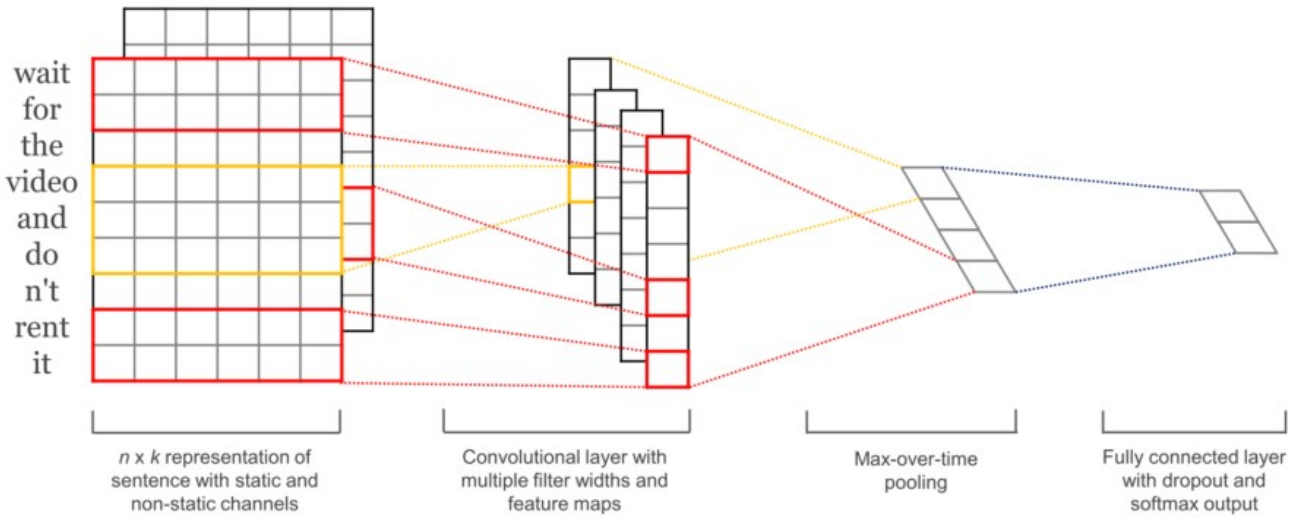


CNN-LSTM and LSTM-CNN

CNNs

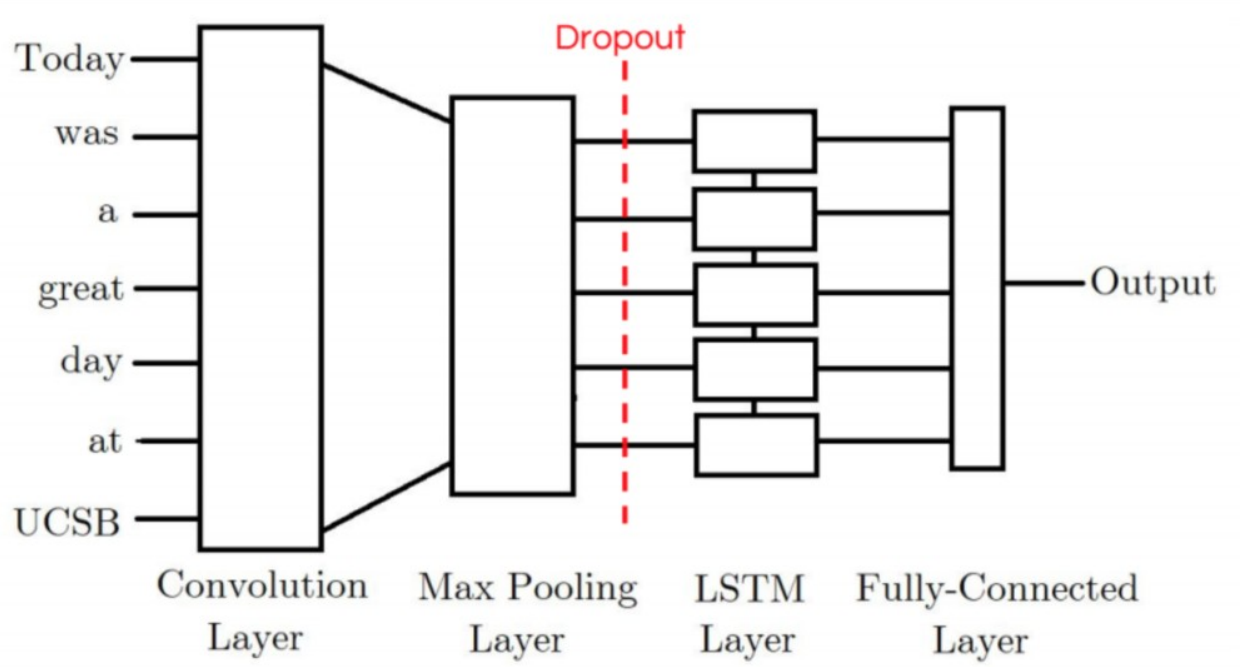
Convolutional Neural Networks (CNNs) are networks initially created for image-related tasks that can learn to capture specific features regardless of locality.

For a more concrete example of that, imagine we use CNNs to distinguish pictures of Cars vs. pictures of Dogs. Since CNNs learn to capture features regardless of where these might be, the CNN will learn that cars have wheels, and every time it sees a wheel, regardless of where it is on the picture, that feature will activate.



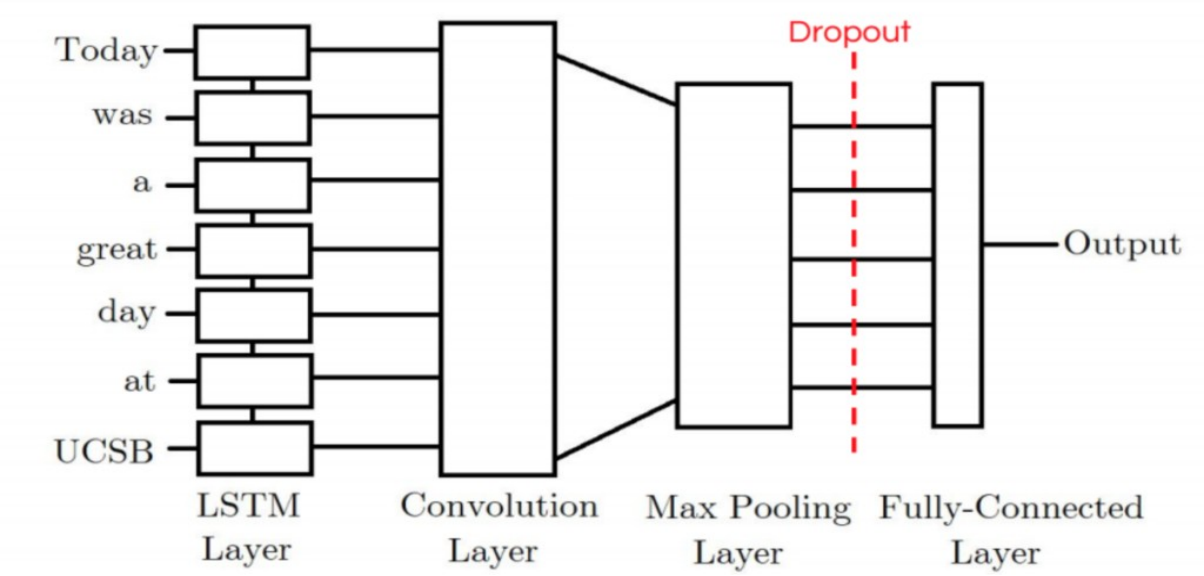
CNN-LSTM

Our CNN-LSTM model combination consists of an initial convolution layer which will receive word embeddings as input. Its output will then be pooled to a smaller dimension which is then fed into an LSTM layer. The intuition behind this model is that the convolution layer will extract local features and the LSTM layer will then be able to use the ordering of said features to learn about the input's text ordering. In practice, this model is not as powerful as our other LSTM-CNN model proposed.

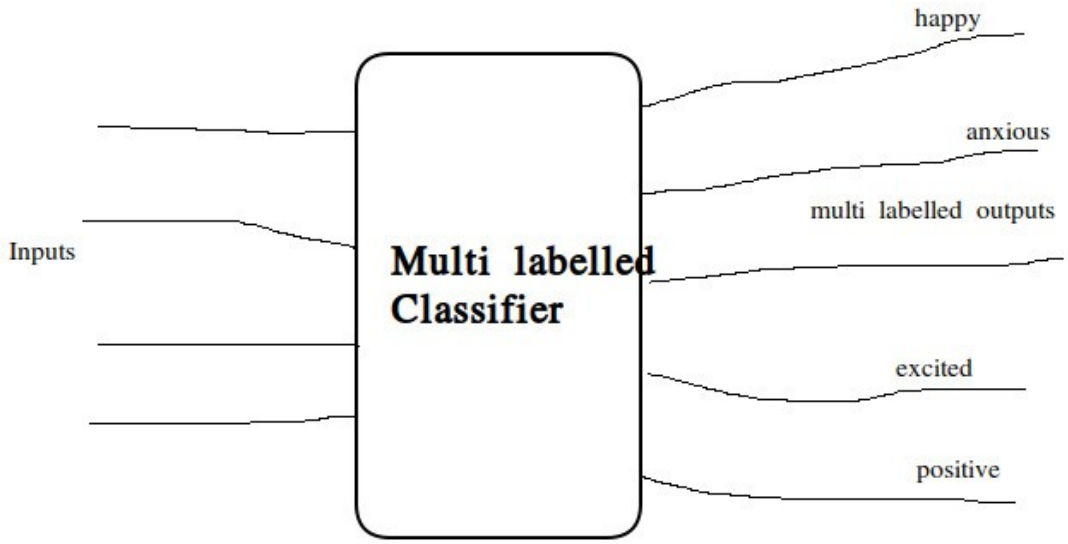


LSTM-CNN

Initial LSTM layer which will receive word embeddings for each token in the text as inputs. The intuition is that its output tokens will store information not only of the initial token, but also any previous tokens; In other words, the LSTM layer is generating a new encoding for the original input. The output of the LSTM layer is then fed into a convolution layer which we expect will extract local features. Finally the convolution layer's output will be pooled to a smaller dimension and ultimately outputted as either a positive or negative label.



Multi-label Classification



Naive Bayes

Combined Naive Bayes with One vs Rest approach to classify text into multiple emotion classes.

K- nearest neighbours

Predicts multiple emotions based on its neighbours.

Performance of various models

Model Name	Training Accuracy	Testing Accuracy
LSTM	82%	52%
Naive Bayes	80%	53.1%
CNN-LSTM	82%	49.11%
LSTM-CNN	68.9%	48.0%
K-nearest neighbour	65.8%	38.6%

Distribution of Work

LSTM

Himani,Priya

CNN-LSTM

Padma, Divyanshi

CNN-LSTM(glove)

Priya, Padma

LSTM—CNN

Divyanshi, Himani

LSTM-CNN(glove)

Priya,Divyanshi

Naive Bayes

Himani, Padma

K-Nearest Neighbours

Divyanshi, Himani

Naive Bayes(multilabel)

Padma, Priya

K-Nearest Neighbours(multilabel)

Himani, Priya

Bibliography

1. http://konukoii.com/blog/2018/02/19/twitter-sentiment-analysis-using-combined-lstm-cnn-models/

2. https://towardsdatascience.com/machine-learning-text-processing-1d5a2d638958

3. https://towardsdatascience.com/multi-label-text-classification-with-scikit-learn-30714b7819c5

4. https://nlp.stanford.edu/projects/glove/

5. https://realpython.com/python-keras-text-classification/

6. https://sebastianraschka.com/Articles/2014_naive_bayes_1.html

7. https://towardsdatascience.com/text-classification-using-k-nearest-neighbors-46fa8a77acc5