Emotion Detection from Textual Conversations

Namrata Bakre
University at Buffalo,
Buffalo, NY
UB Person ID: 50336849
Email: nbakre@buffalo.edu

Mrunal Inge
University at Buffalo,
Buffalo, NY
UB Person ID: 50337040
Email: mrunalna@buffalo.edu

Neel Dungarani
University at Buffalo,
Buffalo, NY
UB Person ID: 50338201
Email: neelvija@buffalo.edu

Abstract—**The idea of the project is from SemEval-2019 Task 3 - EmoContext: Contextual Emotion Detection in Text. Lack of facial expressions and voice modulations make detecting emotions in text a challenging problem. For instance, as humans, on reading "*Why don't you ever text me!*" we can either interpret it as a sad or angry emotion and the same ambiguity exists for machines. However, the context of dialogue can prove helpful in detection of the emotion. Based on given Textual Conversation (user utterance along with two turns of context), goal of our project is to construct a model that should be able to classify the emotion of user utterance as one of the emotion classes: Happy, Sad, Angry or Others. For implementation part of the project, we have used training data set of 30160 dialogues and two evaluation data sets containing 2755 and 5509 dialogues respectively. The data has been provided by Microsoft for the purpose of research only. In this project, we are implementing the idea of recognizing emotions through textual conversations using multiple different models and comparing the results of each of them.**

Keywords—contextual emotion detection, emotion classes, textual dialogue, inferences, comprehension, mood

I. INTRODUCTION

Emotions are basic human traits. They are naturally occurring response to a situation. Whether this response is the result of our own evaluation or an automatic one remains to be seen. most psychologists agree that a basic emotion like anger exists as an evolutionary trigger. We humans and most other animals appear to be equipped with a set of predictable responses to situations. We call these the **basic emotions**: anger, fear, surprise, disgust, joy and sadness, as described in the 1970s by anthropologist Paul Eckman [1].

Emotions may signal a change in our environment, a change within us or a change in both. These signals are generally fleeting in comparison to other states of mind. As a result, emotions are distinct from moods, which can last for hours, days or even weeks. They're also distinct from personalities, the lifelong set of traits that comprise our individual, predictable reactions to situations [2].

Detecting emotions in textual dialogues is a challenging problem in absence of facial expressions and voice modulations. It's easy when people say they are angry or sad or excited, or if they tack an emoji to the end of a text. Given that even face-to-face communication can be confusing, it should not surprise us that these truncated, dashed-off messages can result in disastrous misunderstandings.

In the age of technology, we not only need to decode in-person interactions, but textual transmissions as well. Table 1 represents few such examples where context of ongoing conversation can completely change the emotion for an utterance when compared to standalone evaluation of an utterance. Note that, in the first example, "What do you mean?" will perceived as annoyed by a majority, however considering it in the context, it turns out to be 'Happy' emotion. Similarly, in the second example, "It's a Friday, not Monday" is very likely to be perceived as 'Others', however again a majority will judge it as 'Sad' with the given context. Naturally, considering context to estimate emotion of a text utterance becomes even more important for aforementioned scenarios of digital assistants and conversational agents, because of their text-based conversational interface.

Task Definition: In a textual dialogue, given an utterance along with its two previous turns of context, classify the emotion of the utterance as one of the following classes: Happy, Sad, Angry or Others. Fig. 2 illustrates one such conversation between two people, where each utterance is labeled by the underlying emotion.

The rest of this paper is organized as follows. Section II discusses related work. Section III discusses data collection and pre-processing of data. Section IV discusses data analysis which highlights word count, Top Unigrams and Emoticons. Rest of the paper discusses, evaluation Metric used, Model Architecture and final comparative results between different models.

User 1	User 2	User 1	Actual Class
That was a joke btw	What do you mean?	It's okay, I like weird people	Happy
It's a Friday, not Monday	Isn't that the case on most Mondays!!	Yeah, that's why I am sad	Sad
Why??	Cause it's annoying	Okay, leave it.	Angry

Table 1: Examples showing influence of context in determining emotion of last utterance.

II. RELATED WORK

Image based emotion recognition as well as voice-based recognition are relatively old research areas and researchers have achieved good results on these areas. However, classifying textual dialogues based on emotions is relatively new research area. In more recent work, chatbots and chit-chat dialogue have become more prominent, in part due to the use of distributed (such as embedding) representations that do not readily support logical inference.

Conversations are broadly categorized into two categories: task oriented and chit-chat (also called as non-task oriented). Both kinds of conversation are governed by different factors or pragmatics, such as topic, interlocutors' personality, argumentation logic, viewpoint, intent, and so on. Emotion-detection algorithms for text can be largely bucketized into following two categories:

(a) Deep Learning based approaches:

Deep Neural networks have enjoyed considerable success in varied tasks in text, speech and image domains. Convolution Neural Networks (CNN) is one of the popular choices in the image domain. Recently, approaches which employ Deep Learning for emotion detection in text have been proposed. Zahiri and Choi [4] predicts emotion in a TV show transcript. It is worth noting that textual dialogues are informal and laden with misspellings which pose serious challenges for automatic emotion detection approaches. Prior to this task, to the best of our knowledge, the methods proposed by Chatterjee et al. [5] are some of the few methods that tackled the problem of emotion detection in English textual dialogues.

(b) Feature engineering-based approaches:

Early computational work on dialogue focused mostly on task-oriented cases, in which the overall conversational intent and step-by-step sub-goals played a large part. Cohen and Levesque [3] developed a model and logic to represent intentions and their connections to utterances, whose operators explicate the treatment of beliefs about the interlocutor's beliefs and vice versa, recursively. Emotion however played no role in this line of research.

Many methods exploit the usage of keywords in a sentence with explicit emotional/affect value. Methods which rely on extracting statistical features such as presence of frequent n-grams, negation, punctuation, emoticons, hashtags to form representations of sentences which are then used as input by classifiers such as Decision Trees, SVMs among others to predict the output. However, all of these methods require extensive feature engineering and they often do not achieve high recall due to diverse ways of representing emotions. For example, the following utterance, "Trust me! I am never gonna order again", contains no affective words despite conveying an emotion of anger or frustration perhaps.

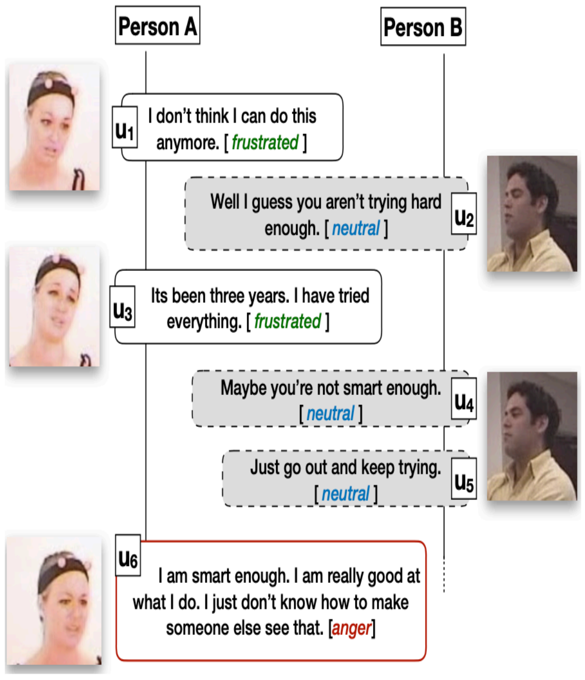

Person A
Person B
u1 I don't think I can do this anymore. [frustrated]
Well I guess you aren't trying hard enough. [neutral] u2
u3 Its been three years. I have tried everything. [frustrated]
Maybe you're not smart enough. [neutral] u4
Just go out and keep trying. [neutral] u5
u6 I am smart enough. I am really good at what I do. I just don't know how to make someone else see that. [anger]

Fig. 2: An abridged dialogue

III. IMPLEMENTATION

First and the foremost we took the data from SemEval-2019 and did the data preprocessing and cleaning. And after that we trained our model with cleaned data. [6] Implementation part consists of mainly three things.

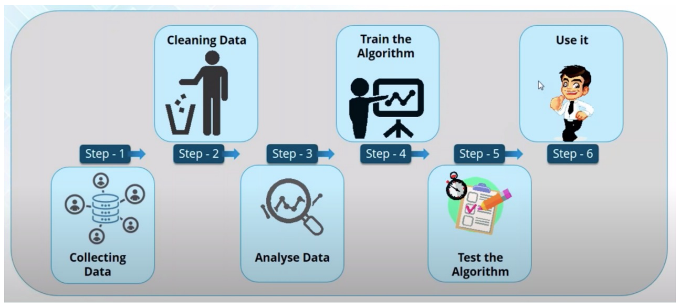


Fig. 3 Overall Process Flow

(A) DATA CLEANING: Data cleaning has been done in below four steps:

(i) Tokenization:

For tokenization first we are combining the three string chat data into single string and then we are using Treebank Word Tokenizer from nltk library which breaks the sentence in to list of meaningful words. It is different than splitting string with whitespace which consists of punctuations. In a string containing punctuation marks not every punctuation mark are useful and not every mark is useless so Treebank word tokenizer tokenizes string considering some useful punctuation marks and removing others.

(ii) Normalize Tokens:

Normalization of token means convert tokens to their root form which will help reducing the ambiguity of token.For example, if there is a sentence containing three different forms of same word then it will have three different tokens for same word so normalization of token converts all three forms to their root form and treats all three words as single token. For that generally used approaches are stemming and lemmatization. Lemmatization converts tokens to their dictionary format word (i.e. Lemma) and stemming uses different approaches to normalize different tokens it has set of rules which is applied to tokens to stem. We are currently using snowballStemmer and wordNetLemmatizer and we will eventually end up with either of them depending on their performance with different models.

(iii) Remove stop words:

Almost every string contains stop words and these words are meaning less (like, a, an, the, this, that, etc.) so frequency of such words are really high compare to other meaning full words and if we do not remove such stop words from sentence before training our model then it's frequencies can really dominate the results and increase errors. So to remove such keywords from consideration we are using list of stop-words from nltk library to compare tokens and remove token that are there in stop-words.

(iv) Normalize capital letters:

Capitals in the beginning of the sentence can also create ambiguity in list of tokens so that needs to be handled. And normalizing capital letter also helps in normalizing acronyms to some point (i.e. e.t.a,E.T.A., etc.)

(B) PREPROCESSING: We have preprocessed our data with two approaches:

(i) Feature Extraction (Bag of Words):

In feature extraction we are creating BOW out of whole data and vectorize tokens by giving them weights using TF-IDF approach and then establish each tokens as feature. We are using TF-IDF approach to determine weights of tokens because in BOF there are three types of frequency category possible for each tokens (i.e. high frequency tokens, low frequency tokens and medium frequency tokens). High frequency tokens can be stop-words which can increase error. Low frequency tokens can we either typos or rare n-grams. Which can be reason of overfitting and medium frequency tokens are the good n-grams for model so we are using TfidfVectorizer from sklearn library to avoid high frequency and low frequency n-grams. [7]

(ii) Feature selection:

We are using sklearn.feature_selection library and SelectKBest to select K number of best feature from feature vector. In this approach we are using Chi-squared stats of non-negative features for classification tasks. And K=10000 to select 10000 best features to train our model. [8]

(C) TRAINING MODEL:

We have split our data in to 70-30 ratio for training and testing respectively. And initially we have trained LinearSVM model using 'L1' penalty (Lasso) and C (number of mistakes) being 1 with maximum iteration of 3000. for training our model we have used sklearn LinearSVC library. [9]

IV. PLAN

We have trained our data on below models and captured their Accuracy.

(i) Logistic Regression:

Logistic Regression is a Machine Learning algorithm which is used for the classification problems, it is predictive analysis algorithm and based on the concept of probability. We can call a Logistic Regression a Linear Regression model but the Logistic Regression uses a more complex cost function, this cost function can be defined as the 'Sigmoid function' or also known as the 'logistic function' instead of a linear function.

Logistic Regression Assumptions:

Before diving into the implementation of logistic regression, we must be aware of the following assumptions about the same −

- In case of binary logistic regression, the target variables must be binary always and the desired outcome is represented by the factor level 1.
- There should not be any multi-collinearity in the model, which means the independent variables must be independent of each other.
- We must include meaningful variables in our model.

- We should choose a large sample size for logistic regression.

(ii) Decision Tree:

A decision tree is a flowchart-like structure in which each internal node represents a test on a feature (e.g. whether a coin flip comes up heads or tails), each leaf node represents a class label (decision taken after computing all features) and branches represent conjunctions of features that lead to those class labels. The paths from root to leaf represent classification rules.

Decision trees are constructed via an algorithmic approach that identifies ways to split a data set based on different conditions. It is one of the most widely used and practical methods for supervised learning. Decision Trees are a non-parametric supervised learning method used for both classification and regression tasks.

Steps for Creating a decision tree:

- Get list of rows (dataset) which are taken into consideration for making decision tree (recursively at each nodes).
- Calculate uncertainty of our dataset or Gini impurity or how much our data is mixed up etc.
- Generate list of all question which needs to be asked at that node.
- Partition rows into True rows and False rows based on each question asked.
- Calculate information gain based on gini impurity and partition of data from previous step.
- Update highest information gain based on each question asked.
- Update best question based on information gain (higher information gain).
- Divide the node on best question. Repeat again from step 1 again until we get pure node (leaf nodes).

(iii) Support Vector Machines

The objective of the support vector machine algorithm is to find a hyperplane in an N dimensional space that distinctly classifies the data points.

Support vectors are data points that are closer to the hyperplane and influence the position and orientation of the hyperplane. Using these support vectors, we maximize the margin of the classifier. Deleting the support vectors will change the position of the hyperplane. These are the points that help us build our SVM.

- Support Vectors − Datapoints that are closest to the hyperplane is called support vectors. Separating line will be defined with the help of these data points.

- Hyperplane − As we can see in the above diagram, it is a decision plane or space which is divided between a set of objects having different classes.

- Margin − It may be defined as the gap between two lines on the closet data points of different classes. It can be calculated as the perpendicular distance from the line to the support vectors. Large margin is considered as a good margin and small margin is considered as a bad margin.

- SVM Kernels - In practice, SVM algorithm is implemented with kernel that transforms an input data space into the required form. SVM uses a technique called the kernel trick in which kernel takes a low dimensional input space and transforms it into a higher dimensional space. In simple words, kernel converts non-separable problems into separable problems by adding more dimensions to it. It makes SVM more powerful, flexible and accurate.

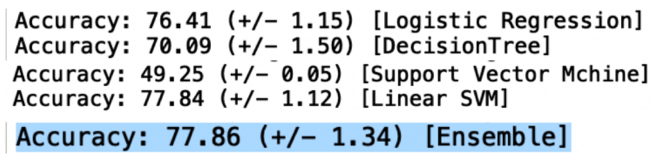
(iv) Ensemble Model:

Ensemble models is a machine learning technique that combines several base models in order to produce one optimal predictive model.

Types of Ensemble Model:

- Bagging, or Bootstrap Aggregating: Bagging gets its name because it combines Bootstrapping and Aggregation to form one ensemble model. Given a sample of data, multiple bootstrapped subsamples are pulled. A Decision Tree is formed on each of the bootstrapped subsamples. After each subsample Decision Tree has been formed, an algorithm is used to aggregate over the Decision Trees to form the most efficient predictor.
- Random Forest Models: Random Forest Models can be thought of as Bagging, with a slight tweak. When deciding where to split and how to make decisions, Bagged Decision Trees have the full disposal of features to choose from. Therefore, although the bootstrapped samples may be slightly different, the data is largely going to break off at the same features throughout each model. In contrary, Random Forest models decide where to split based on a random selection of features. Rather than splitting at similar features at each node throughout, Random Forest models implement a level of differentiation because each tree will split based on different features. This level of differentiation provides a greater ensemble to aggregate over, ergo producing a more accurate predictor. Refer to the image for a better understanding.

We have observed below Accuracy for the above-mentioned Classifiers and we have got quite comparative results.



```
Accuracy: 76.41 (+/- 1.15) [Logistic Regression]
Accuracy: 70.09 (+/- 1.50) [DecisionTree]
Accuracy: 49.25 (+/- 0.05) [Support Vector Mchine]
Accuracy: 77.84 (+/- 1.12) [Linear SVM]
Accuracy: 77.86 (+/- 1.34) [Ensemble]
```

Fig. 4 Accuracy Results

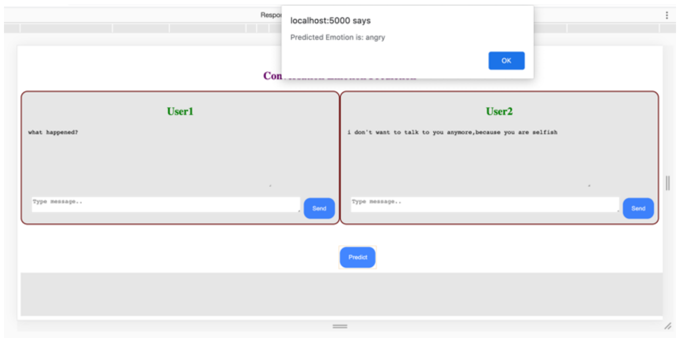
V. RESULTS

We have been successful in achieving following -

In a textual dialogue, given an utterance along with its two previous turns of context, classify the emotion of the utterance as one of the following classes:

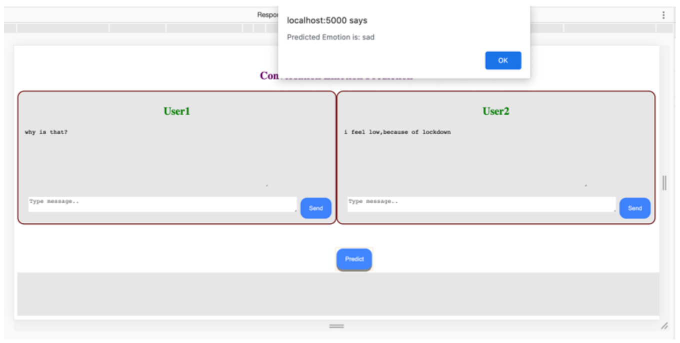
Happy, Sad, Angry or Others.

Below are some of the examples of 3 – turn conversations:



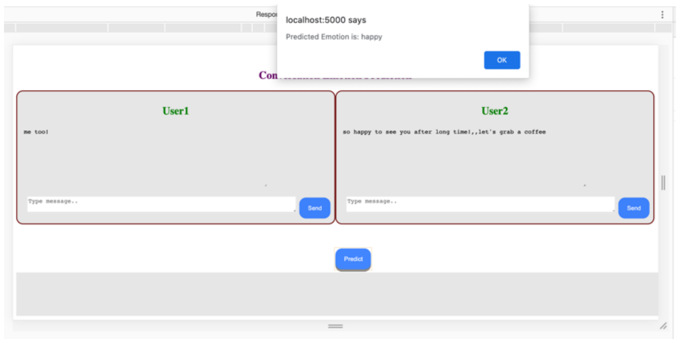
localhost:5000 says
Predicted Emotion is: angry
User1: what happened?
User2: i don't want to talk to you anymore,because you are selfish

Fig. 5 UI Implementation – Dialogue 1



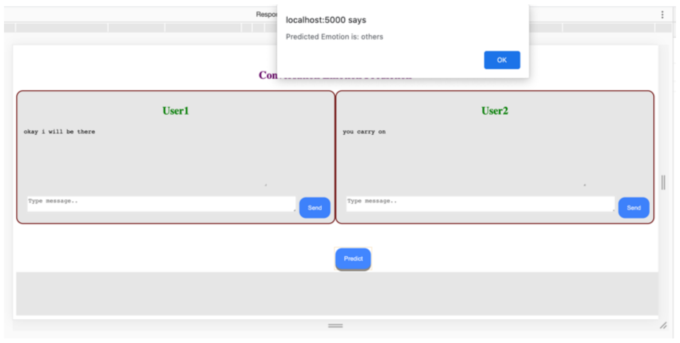
localhost:5000 says
Predicted Emotion is: sad
User1: why is that?
User2: i feel low,because of lockdown

Fig. 6 UI Implementation – Dialogue 2



localhost:5000 says
Predicted Emotion is: happy
User1: me too!
User2: so happy to see you after long time!,,let's grab a coffee

Fig. 7 UI Implementation – Dialogue 3



localhost:5000 says
Predicted Emotion is: others
User1: okay i will be there
User2: you carry on

Fig. 8 UI Implementation – Dialogue 4

VI. CONCLUSION

Emotion recognition in conversation has been gaining popularity among NLP researchers. In this work, we have summarized recent advances done in this field and highlighted several key research challenges associated with this research area. Overall, we have presented an effective emotion-detection model with the help of some basic machine learning algorithms like logistic regression, decision tree, SVM.

REFERENCES

[1]http://changingminds.org/explanations/emotions/basic%20emotions.htm

[2] https://www.unige.ch/cisa/emotion-details

[3] P. R. Cohen and H. J. Levesque, "Speech acts and rationality," in *Proceedings of the 23rd annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 1985, pp. 49–60.

[4] Sayyed M Zahiri and Jinho D Choi. 2017. Emotion detection on tv show transcripts with sequence- based convolutional neural networks. *arXiv preprint arXiv:1708.04299*.

[5] Ankush Chatterjee, Umang Gupta, Manoj Kumar Chinnakotla, Radhakrishnan Srikanth, Michel Galley, and Puneet Agrawal. 2019. Understanding emotions in text using deep learning and big data. Computers in Human Behavior, 93:309–317.

[6] https://www.humanizing-ai.com/emocontext.html

[7]https://scikitlearn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

[8] https://scikitlearn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html

[9]https://scikitlearn.org/stable/modules/generated/sklearn.svm.LinearSVC.html

Contributors to the Project:

1. **Namrata Bakre:**
   - Conceived the presented idea
   - Collected the required Train and Test data
   - Prepared the final report of the project
   - Prepared the final demo video
   - Contributed to final working code
   - Prepared the final demo presentation
2. **Mrunal Inge:**
   - Developed the UI of the project
   - Contributed in Integration of UI code with backend code
   - Contributed to final working code
   - Contributed in uploading the working code on GitHub Repository
3. **Neel Dungarani:**
   - Developed the final working code
   - Contributed to final report of the project
   - Contributed to demo video
   - Contributed to final demo presentation

All members provided critical feedback and helped shape the implementation and analysis.

GitHub Link of our Code:

https://github.com/CSE574Spring2020/project-submission/tree/TH_namrata_mrunal