Emotion classification from text (chat)
Course Project, CS-725
Spring 2015-16

Ravi Shankar Mondal, 133059003
Saurabh Jain, 143050048
Srikrishna Khedkar, 143050055
Sushant Mahajan, 133059007

March 21, 2016

Contents

List of Figures

List of Tables

Chapter 1

Project Initialization

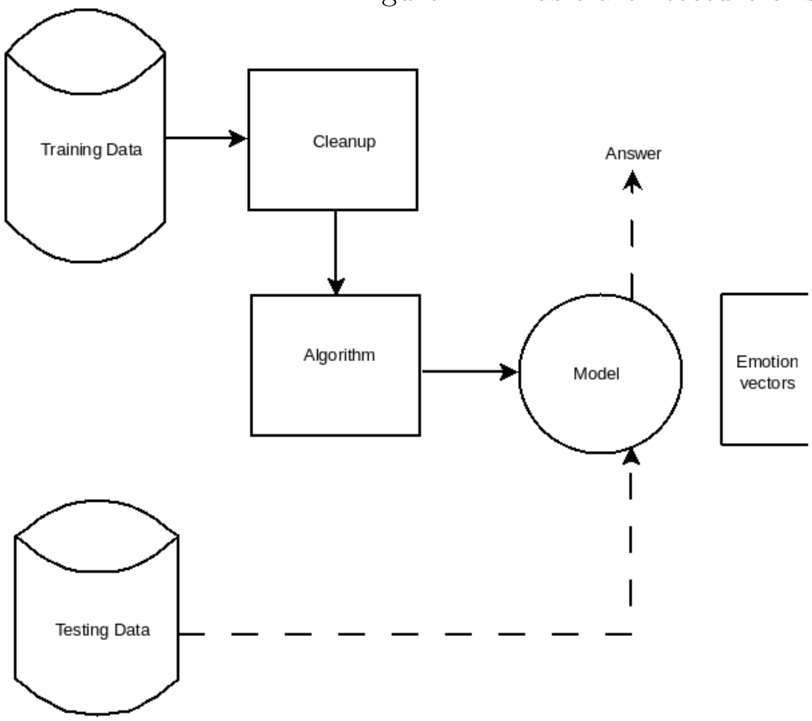
1.1 Feasibility

- We wish to create a system, which having been trained from sentences annotated with emotions portrayed, is capable of predicting the emotion for a new sentence.
- We will be closely following the paper Taner Danisman and Adil Alpkocak titled **Feeler: Emotion Classification of Text Using Vector Space Model**.
- We will need to build a lexicon, which is basically an ordered set of all *good* words in the training data.
- For each sentence we will build a weight vector of all terms in the lexicon.
- The weights are calculated using the **tf-idf** technique.
- For each emotion class we take the mean of all vectors within it.
- We will be utilizing python for developing the application and be using **nltk** for text clean up, stemming etc.
- Estimated time to develop the project would be a month.

1.2 Dataset

- The paper mentions the ISEAR dataset, which is composed of around 7500 annotated sentences.

Figure 1.1: Basic architecture of solution



1.3 Scope

- At its core, the idea is to find median vectors corresponding to each emotion (joy, sadness, anger, ...) and then find the cosine similarity of the new vector composed from test input with the former. The highest similarity vector will be the winner.
- Other than implementing their proposed solution, we will also try to incorporate methods like bayesian classification and neural networks and analyse the results.
- If time permits, we will build some application around our system.

Chapter 2

Submission 1

2.1 Project Description

2.1.1 Problem Statement

Text written by a person can help us determine what emotion he is portraying. Other such modalities can be facial cues, body language etc. In this project we will try to predict the emotion using information from a sentence input by a user. This can have applications in chatting programs, movie reviews etc. Consider the following scenarios:

Sentence	**Emotion**
•A close person lied to me	Sadness
•A colleague asked me for some advice and as he did not have enough confidence in me he asked a third person.	Sadness
•I got a present today.	Joy

Table 2.1: Some examples

2.1.2 Basic Details

- We wish to implement a system capable of predicting the emotion portrayed, given a sentence. The predicted classes will be limited to **anger, joy and sadness**.
- The project is inspired from the paper mentioned in section 1.1.
- The paper details the use of **Vector Space Models** (VSM) to implement the solution.
- We will be using our dataset, comprising sentences annotated with emotions to train the model. The trained model will then be presented with the test data in form of a vector and the most probable emotion will be displayed as output.

- Later, if time permits, we will implement some form of application to showcase the prowess of our project.

2.2 Papers

The paper implemented is mentioned in 1.1.

2.3 Dataset

The dataset used is mentioned in 1.2. The dataset has around 7500 sentences. We will use $2/3^{rd}$ of the dataset for training and cross validation and the rest for testing.

2.4 Method

- As mentioned in section 2.1.2, in line with the paper, we will be using VSM for out initial implementation.
- Put simply, we will have representative vectors for each of our emotion classes, **joy, anger and sadness**. Then we will transform our input sentence in the same way we found our representative vectors and find the cosine similarity with each of them.
- These similarities will be used to calculate probabilities of the input being in some emotion class and the most probable class will be output.
- The vectors will comprise of weights in terms of **tf-idf** frequencies. We could simply use word frequency instead but:
  - **tf-idf** was chosen to provide weights as some words occur sparsely in the document, but provide definitive information about the emotion being portrayed. For example, the word *devil* occurring in a sentence tips the scale towards negative emotion.
  - Had we chosen only frequencies, this could pollute the result.
  - Similarly, some words occur very frequently like *cat*, but do not provide much information. Frequentistically speaking these words will get higher weight.
  - **tf-idf** balances these disparities by multiplying the term frequency with inverse document frequency. The latter meaning the ratio of total number of documents and the number of documents containing that term.
  - For low frequency words, the **idf** will be high and vice-versa.

2.4.1 High level algorithm

Training

- The training set is cleaned, in that:
  - Stop words are removed, after comparing with standard stop word lists.
  - Apostrophes are expanded.
  - Words are reduced to their root using some stemming technique.
- We build a lexicon from $2/3^{rd}$ of the above cleaned data.
- Each term in the vector corresponds to a term in our lexicon. The lexicon is an ordered set of all words in the cleaned dataset.
- We build document vectors using weights assigned to each term. These weights are calculated using the **tf-idf** technique. These weights are further normalized.

$$w_{ki} = c(t_k, d_i) = \frac{tf_{ik} log(N/n_k)}{\sqrt{\sum_{k=1}^{n}(tf_{ik})^2[log(N/n_k)]^2}}$$

  where,
  $t_k = k^{th}$ term in document d_i
  tf_{ik} = frequency of the word t_k in document d_i
  $idf_k = \log(\frac{N}{n_k})$, inverse document frequency of word t_k in entire dataset.
  n_k = number of documents with word t_k.
  N = total number of documents in the dataset.

- Now each emotion class is a set of the corresponding vectors, calculated as above. We then find the mean of all these vectors to find a representative vector of the said class.
- Finally, we'll get s vectors corresponding to the distinct emotion classes.

Testing

- Each input sentence is cleaned, and transformed into a vector with weights calculated using **tf-idf** technique.
- The similarity is calculated as:

$$\text{sim}(Q, E_j) = \sum_{k=1}^{n} w_{kq} * E_{kj}$$

  where,
  Q is query vector,
  E_j is an emotion vector.

- Answer is calculated as:

$$\text{VSM}(Q) = \text{argmax}_j(\text{sim}(Q, E_j))$$

2.5 Submission on 21^{st}

We plan to complete the implementation of the paper along with relevant testing and output metrics, outlined above. We will complete the code for text cleaning and VSM.

2.6 Final submission

- Complete implementation of VSM.
- Implement some bayesian model using bigram, trigram or ngram approach and check the variation in results.
- If time permits, we'll develop some applications to showcase the model.

2.7 Team Members

Name	**Roll No**
Ravi Shankar Mondal **(leader)**	133059003
Srikrishna Khedkar	143050055
Saurabh Jain	143050048
Sushant Mahajan	133059007

Table 2.2: Team members

Chapter 3

Submission 2 (21^{st} March)

3.1 Progress so far

- We have throughly read the paper mentioned in 1.1, and implemented the algorithm.
- There were some problems in dividing the dataset as the ISEAR dataset mentioned in 1.2.
  - The dataset has a total of 7 emotion classes, but for the sake of simplicity we chose to model only 3 of them.
  - Out of the 7 classes, 6 are negative emotions (**sadness, guilt, shame, fear, anger**) and only 1 positive (**joy**).
  - We had 2 choices now, either to divide the leftover data for 4 classes into the 2 negative ones, we've chosen (**anger,sadness**).
  - Doing this caused some a lot of misclassification, so after some research, we've currently left out 2 emotions - **guilty,fear**.
- Python code to correctly implement the vector space model algorithm mentioned in 2.4.1. We are working collaboratively and have set up a github repo to house all the data, report and code. The link is `https://github.com/sm88/mlproject`.
- Due to the above mentioned reason for less data, we are looking into some more datasets, mentioned in the paper namely, the Wordnet-Affect dataset and the Semeval dataset.
- The wordnet database is not available easily as we need to fill out a form, which after review will enable us to download.
- We have throughly cleaned the dataset which had many problems (see figure 3.1. A few major ones that we solved are:
  - Many sentences we of form "No Response", which provides no information whatsoever.

  - There were some scattered braces and square brackets as well as some non-ascii characters, that took a while to catch.
  - Some sentences were repeated for multiple classes.
- We have done some preliminary tests on the training data itself and the results seem promising see table 3.1.

```
joy#On days when I feel close to my partner and other friends. When I feel at peace with myself and also experience a close contact with people wh
om I regard greatly.
fear#Every time I imagine that someone I love or I could contact a serious illness even death.
anger#When I had been obviously unjustly treated and had no possibility of elucidating this.
sadness#When I think about the short time that we live and relate it to the periods of my life when I think that I did not use this short time.
disgust#At a gathering I found myself involuntarily sitting next to two people who expressed opinions that I considered very low and discriminatin
g.
shame#When I realized that I was directing the feelings of discontent with myself at my partner and this way was trying to put the blame on him in
stead of sorting out my own feeliings.
guilt#I feel guilty when when I realize that I consider material things more important than caring for my relatives. I feel very selfcentered.
joy#After my girlfriend had taken her exam we went to her parents place.
```

Figure 3.1: Cleaned partial ISEAR dataset

	anger	**sadness**	**joy**
anger	1432	413	318
sadness	384	2221	620
joy	28	61	1001

Table 3.1: Confusion matrix for training set

Emotion	**Accuracy %**
anger	66
sadness	69
joy	92

Table 3.2: Per emotion class accuracy for training set

3.1.1 Other Info

- We believe that with bigger dataset, we'll be able to improve the accuracy of **anger, sadness** further.
- The training time is around a minute and query time is about a second, depending on the length of the sentence.
- Our document vectors mentioned in 2.4.1 are sparse in nature (total number of words compared to words in a sentence) as a result of which we have excellent opportunities to optimize our implementation.

3.2 Plan for the rest of the semester