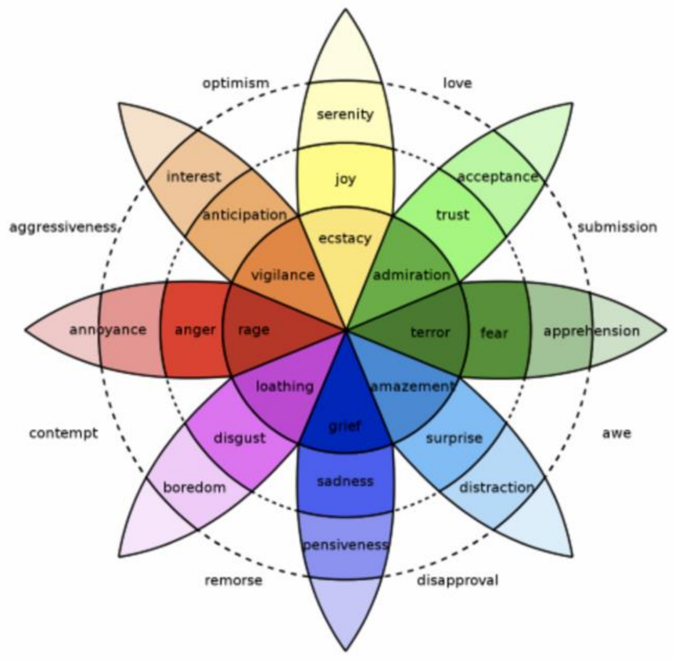
ITCS/DSBA 6162 Knowledge Discovery in Databases

Detecting Surprise from a text corpus

Final Project Report



Authors:

Varun Varia
Richmond Otchere
Preethika Adapa
Deekansha Tandon

https://www.precognox.com/blog/automatic-detection-emotions-text/

Introduction

An emotion is a particular feeling that characterizes a state of mind, such as joy, anger, love, fear, surprise and so on. Emotion detection from text has attracted growing attention due to its potentially useful applications. For examples, psychologists can better assist their patients by analyzing their session transcripts for any subtle emotions; reliable emotion detection can help develop powerful human-computer interaction devices; and deep emotional analysis of public data such as tweets and blogs could reveal interesting insights into human nature.

Large collections of documents can be hard for humans to sift through on their own. High-quality search can help find what you want, and if you have the resources to annotate documents with tags or taxonomic categories, you can bring some order to an unwieldy corpus. But when the number of potentially relevant documents is high and your time to individually examine them is low, what really matters is finding documents that tell you what you don't already know.

We were given two files: one contained 500 articles related to medical field. The second file was a csv format data file which contained article ID, its name, and contained reviews of each article by 3 human raters. These articles were rated from 1 - 5 based on likability, familiarity and how much the article surprised them with 1 being not surprising at all and 5 being highly surprising.

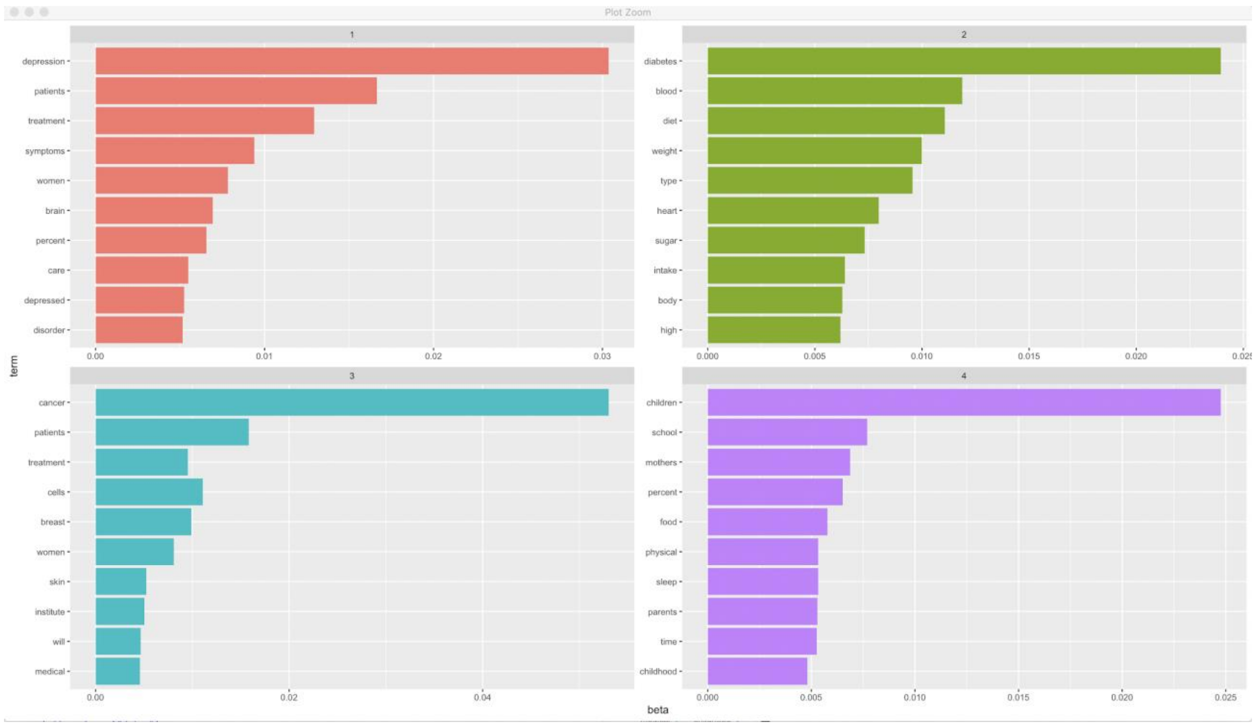
What did we do

Understanding the data

In order to get a better idea of the data, we tried to calculate Mutual information between the word 'surprise' and 'cancer'. According to our calculation, the mutual information between the word surprise and cancer is 0.00690., We already know that mutual information of two random variables is a measure of the commonality between the two variables. The mutual information value equals zero if and only if the two variables are statistically independent. Since 0.00690 runs pretty close to 0, we can say that these two words are not correlated a lot. In another word, when we see "surprise" in a document, we cannot deduct our uncertainty about the appearance of the word "cancer". Similarly, we tried a number of combinations with words: 'know', 'assume', 'believe',' think' with 'depression', 'diabetes' and 'cancer'.

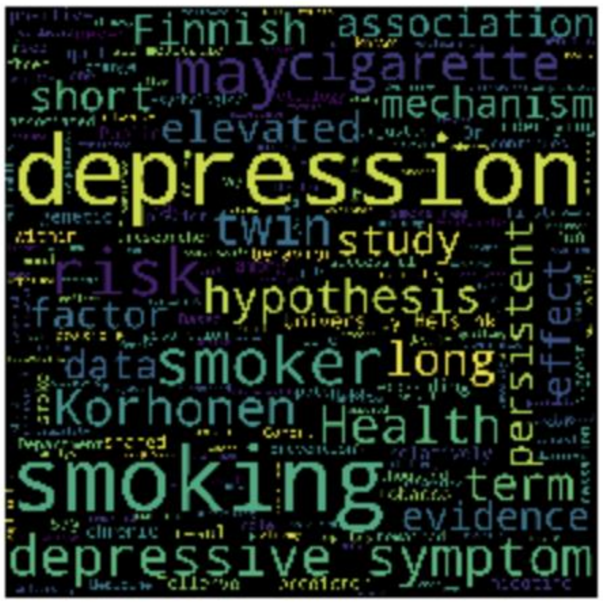
We try to understand the mutual information of the words that might be closely related to surprise. We try the words: 'know', 'think', 'believe'. Words with most mutual information with 'believe' were - patients, cancer, cells. Words with most mutual information with think were - children, believe, diabetes. Words with most mutual information with 'know' were - cancer, infants, mothers

Since any particular information is unclear from the data, we try to do topic modelling in order to understand what topics are about and we try to find top 10 words in each topic. We chose the number of topics as 4 thus using LDA function in the "topicmodels" package to build the topic model. Then, used the function in "tidy" package and "dplyr" package to help us format the data. In the end, we used the "ggplot2" package to visualize the data. We build LDA VEM model with 4 topics and find top 10 words in them. After plotting the graph using ggplot2 we get the following graph:



First topic is about depression with brain and women being the top ten words in it. The second topic is about depression and seeing the top ten words, we can say sugar, heart and weight are the most frequently occurring words. The third topic is about cancer. From top ten words, we can see that the articles could either be about the type of cancer as we see words like skin, breast, cells and also it could be about curing cancer since we find treatments, medical in it. The fourth topic is about children and we can assume it is related to food and sleep and time of these children.

We then used PorterStemmer and WordNetLemmatizer from the nltk package in order to perform lemmatization and stemming. We converted the text into tokens using words_tokenise function on the text corpora. After removing all the punctuations, we performed stemming and lemmatization of the data. Then in order to understand the content of the data, we implemented word cloud. We used frequency distribution function. A word cloud sample is shown below. As we can see in the word cloud screenshot, the word "depression" pops up as the most common in a particular topic which asserts our previous finding of word "depression" being one of the topics.



Cleaning the data

Before we performed any operations on the data, we needed to make sure the data is prepared in correct way. First, we loaded the required package in order to build the corpus. We used tm package in order to build the corpus. After we built the corpus, we needed to perform several procedures in order to clean the data. We began by removing white spaces and converting the text to lower case. This helps in removing redundancy and makes sure we can standardize our operations. Next step was to remove the punctuations. As punctuations do not add any significant value and we cannot get any valuable information from them, we need to remove it.

Next, we convert the corpus to a document-term matrix. Here in document term matrix, each row represents one document, each column represents one term, and each value indicates the number of appearances of that term in the whole corpus. We can go further and convert this document-term matrix to a data-frame. We want to get a basic idea of what the corpus is about which is why we try to find the most frequent terms in the corpus. We use “findFreqTerms” function to check how frequently do the terms appear in the corpus. After sorting the terms in descending order, we find that the term ‘cancer’ appears 1298 times, ‘diabetes’ appears 891 times, ‘patients’ appear 827 times, ‘depression’ appears 764 times, and so on.

We read the “ground_truth_one_row_per_article.csv” file in a variable called mydata. We will observe the structure of the data using str() function. We notice that there are 26 variables and number of observations are 500. Next in order of get familiar with the data, we use summary function. Variables surprise_r1, surprise_r2 and surprise_r3, like_r1, like_r2, like_r3 vary from 1- 5. In order to aggregate the surprise rating, we will calculate the mean of surprise rating of each article and store it in a new variable called Surprise.

The articles had a surprise rating from 1 to 5 where 1 meant not surprised at all and 5 represented highly surprised. We need to perform certain operation on the variable in order to make it binary. We treated 1 and 2 as not surprised and treated 4 and 5 as surprised. Representing 0 as not surprised and 1 as surprised, we found out that 289 articles were not surprising while 211 articles were surprising.

Problem I

Building first model

We removed any unnecessary columns which we do not need. For predicting surprise rating for each article, we will need to use attributes of the article and not the attributes of a human rater. Since we do not need to consider the traits of individual raters so for task one, we did not consider the factors such as likability of an article or familiarity of an article.

We tried to predict surprise ratings of the articles by using linguistic features of the articles. We used surprise related words to check the possibility of surprise in the corpus. We saw how many times these words appeared in each document so that we could use this for our model. The words that we chose were: believe, assume, know, surprise and think. After finding out the occurrence of these words in each document, we appended the frequency of the words to the dataframe. Our first feature set would consist of semantic features.

For our first model, we will use the semantic properties of the data as the feature set. We used the columns Believe_dis, Surprise_dis, know_dis, think_dis, assume_dis as the factors for our dataframe. When we view the summary of our dataset, we observed that surprise related words do not occur that frequently in the documents. The word 'know' occurred once 49 times and twice 10 times in 500 documents.

Next, we needed to divide our dataset as training data and test data. We chose to divide our data as 80% training data and 20% test data. Thus 400 rows or 400 articles became a part of training data and 100 test data.

Result: Using the above features, we got the accuracy of 42% and sensitivity of 55%.

Building Second model:

Again, for the second model, our focus was on the attributes of the article rather than that of individual raters. Since we used semantic features of the articles for the first model. We chose to select structural part of the data in the corpus. Our specific focus was on the length of words, length of sentences, average words per sentence and so on.

For figuring out the structural part of the article corpora, we chose nltk library and used jupyter notebook as our coding platform. Since NLTK is a leading platform for building Python programs to work with human language data, and it provides easy-to-use interfaces and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, we chose NLTK library.

We began by setting the correct path of the corpora and using the os.path.abspath function to load the correct data. After accessing correct data, we iterated to each file in the 500MNTNews folder. We calculated number of characters, number of words, number of sentences, and the vocabulary of each article and storing them in respective variables. We specifically calculated number of characters per words, number of words per sentence and vocabulary of the sentence. The number of characters per word were 5 - 6 while number of words per sentence were 20-25 and vocabulary of a sentence was 11-17. We then calculated positive and negative words in each article.

We plan on using the structural features along with positive and negative words as our feature set for the second model. We chose to use random forest for the second model as well.

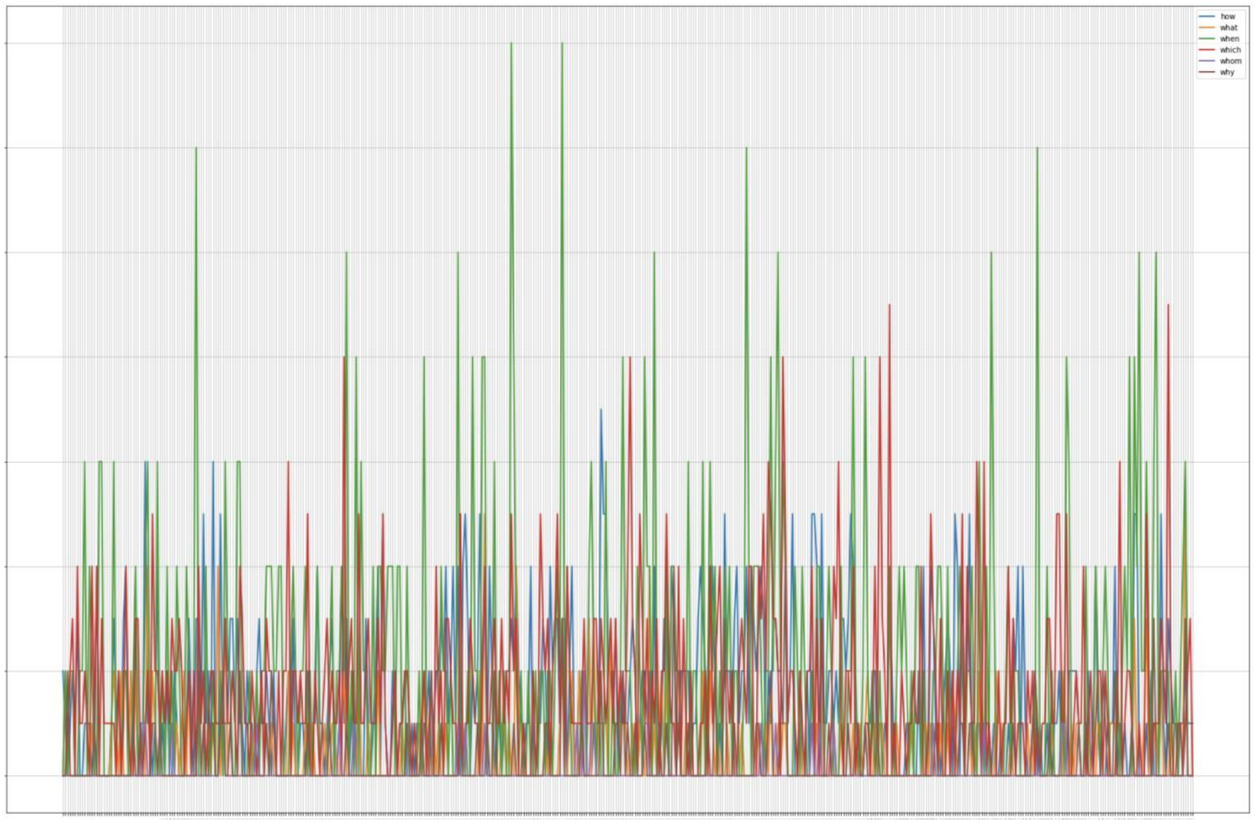
Result: Using the above features, we got the accuracy of 53% and sensitivity of 69%.

Building Third model:

For the third model, we chose to use feature sets from model one and model two as well we decided to explore our data for more feature sets. Our idea for the third feature set was that number of words or number of sentences would be a weak predictor of the surprise rating. After adding positive and negative words to the feature set, we hoped that we would get a good prediction. But since you cannot predict if a person can get surprised by either positive words or negative words, we did not get a very good accuracy of the model.

In order to increase the accuracy, we decided to add the ‘wh’ words to the feature sets. We included: What, Why, When, Which, How, and Whom. We iterated each article in the corpora and tried to find out the frequency of ‘wh’ words in each article. We again used NLTK library for this in jupyter notebook. Using FreqDist() function, we managed to collect the count of each ‘wh’ words in the text documents.

Since after displaying the frequency of these words, making sense out them was really difficult. We could not figure out which word occurred more frequently in the articles. Plotting the words frequency would provide a better understanding of these words which is why we used ConditionalFreqDist() function and plotted the graph using matplotlib library in python. We got the following output:



On the left axis of the graph is the count or frequency of the words and on x-axis are the articles from 1 to 500. Each of these words are color coded and represented on the graph. We can see that the word ‘when’ occurs more frequently than other words. We decided to add this as a feature set for our third model.

Result: Using the above features, we got the accuracy of 61% and sensitivity of 51%.

Problem II

In order to calculate individual surprise rating, we had to change our viewpoint from article to raters meaning we had to build our feature sets from the perspective of raters. We had to change the data from 500 rows to 1500 rows.

Building Fourth model:

After modifying the data, the dataset looked fairly different and it was from the point of view of a person who rated each article. For the fourth model, we used “like” rating given by each rater. We split the rating as liked or not liked and the basis of division was one and two represented that the article was not like whereas three, four, five represented that article was liked. Using this feature, we added the positive negative words of the document which was the attribute of the article and added it to the feature set.

We split the data in training set and test set and trained the model on the training data. Using random forest, we trained our dataset and used random forest model to predict the values for the train data. After training the model on 1200 rows, we used to predict function from the random forest library to predict surprise rating of the remaining 300 rows.

The reason we used likability of the article as a feature set was because we assumed the individual raters would be surprised by the articles they liked. It is very difficult to prove direct correlation between the feeling of liking and feeling of surprise. We added positive and negative words to feature set of the model as we predicted the more the

Result: Using the above features, we got the accuracy of 58%

Building Fifth model:

After cleaning and modifying the data as to what we need it like, the changes in the dataset are made in such a way that all the articles in it are reviewed and rated by each individual. Similar to the fourth model which is based on "liked/not-liked" this model is related to familiarity i.e., whether an article is familiar or not familiar to the reader. This is a binary output which is answered in either "yes" or "no".

The key word in this dataset is "surprise" because any reader who isn't familiar with the article is surprised and the reader who is familiar will not be surprised. Sometimes the reader who is familiar with the article is surprised if he/she isn't aware of some other fact related to the article. Thus, calculating the surprise factor and adding it to the feature set from this dataset gives the maximum accuracy and prediction. Similar to the above models we split the data as 80% training data and 20% test data and using the feature set we train the model on the training data.

We use random forest method because it gives out the maximum positive accuracy for any training dataset model and predict the values that are trained on the train data. Just like in the 4th model after training the model on 1200 rows we used the predict function from the same random forest library to predict the familiarity rating for the remaining 300 rows.

We take the familiarity as a criterion here just to compute and evaluate the values of how much a reader is familiar with the article, he/she is reading. This factor not only rates the readers but also rates the article based on how popular the article is. the feeling of surprise is directly proportional to the feeling of familiarity from the consents mentioned above. Additionally, we also evaluated the article's positive and negative words which will be further used when we compute the predictive models.

Result: Using the above features, we got the accuracy of 63%

Building Sixth model:

We combined the features of previous models in order to build our final model. We picked the linguistic features of the articles. The key word in this dataset is "familiarity" and "likability" because any reader who isn't familiar with the article will probably not like it and the reader who is familiar will like it . Sometimes the reader who is familiar with the article will not like the article as he/she isn't aware of some other fact related to the article. Thus, calculating the familiarity and likeability factor and adding it to the feature set from this dataset gives the maximum accuracy and prediction. Similar to the above models we split the data as 80% training data and 20% test data and using the feature set, we train the model on the training data.

We use random forest method because it gives out the maximum positive accuracy for any training dataset model and predict the values that are trained on the train data. Just like in the 5th model after training the model on 1200 rows we used the predict function from the same random forest library to predict the familiarity rating for the remaining 300 rows. We take the likability as a criterion here just to compute and evaluate the values of how much a reader is likes the article, he/she is reading.

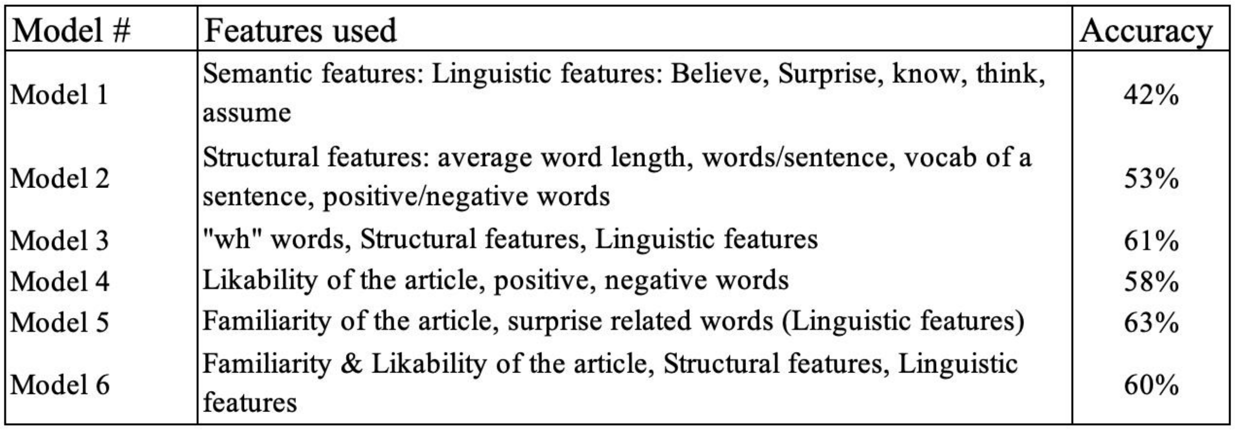
Additionally, we also evaluated the article's positive and negative words which will be further used when we compute the predictive models.

Result: Using the above features, we got the accuracy of 60%

What did we find out

We spent majority of our time understanding the data, figuring out what the data is about and building the correct feature sets for the model. Most famous topic that the articles were about are depression, cancer, children and diabetes. Building feature set was the most challenging part of the project. Since human nature is not very easy to predict and sentimental analysis is still an emerging field, we had a lot of difficult figuring out the correct method and approach for the problem. After implementing all the models, we concluded that familiarity was the best predictor of surprise and surprise rating which makes sense. As when a person is not familiar with a category of a topic, he tends to be more surprised with it. Whereas if the person is familiar with the topic, the chances of him being surprised by with the content reduces a lot. We mined the text and build a lot of feature sets which had attributes of the articles.

We have summarized the features used and the accuracy we achieved for our models in the table below:



Model #	Features used	Accuracy
Model 1	Semantic features: Linguistic features: Believe, Surprise, know, think, assume	42%
Model 2	Structural features: average word length, words/sentence, vocab of a sentence, positive/negative words	53%
Model 3	"wh" words, Structural features, Linguistic features	61%
Model 4	Likability of the article, positive, negative words	58%
Model 5	Familiarity of the article, surprise related words (Linguistic features)	63%
Model 6	Familiarity & Likability of the article, Structural features, Linguistic features	60%

References

Whitney, P., Engel, D., Cramer, N., & Whitney, P. (2009). Mining for Surprise Events Within Text Streams. Society for Industrial and Applied Mathematics. Proceedings of the SIAM International Conference on Data Mining, 617–617. Retrieved from http://search.proquest.com/docview/1671564969/