Multi-Label Text Classification on the Whisper App Dataset

Team Members:
Toshal Phene (tphene)
Shubham Munot (samunot)
Jaydeep Rane (jrane)

Whisper is a mobile app that allows users to anonymously post their thoughts/feelings in any one of Whisper's 17 categories. **These thoughts/feelings are referred to as Whispers**. As part of our capstone project, we intend to use the dataset provided by Whisper as part of its data challenge, and categorise these whispers as accurately as possible into these 17 labels. One of the crucial advantages that this app provides its users is immunity from cyber-bullying.

How this app works:
The user posts his/her whisper on the app, and the app automatically categorises the whisper into one of the 17 categories using machine learning.

1. Description of the dataset

train.csv
This dataset consists of 14049 whispers as well as the category they are assigned to. There are two columns in this csv file namely ***category*** and ***text***.
E.g. would be as follows:
category: meetup
text: Just looking for someone to hold a conversation. Guy here

test.csv
Similar to train.csv, we have a test data that has around 3600 whisper along with the category that they are a part of. This dataset is used to evaluate the various models we make during the course of this project.

Following are few of the the important value additions from the business perspective of this dataset:

- Accurately classifying the data into the 17 categories so that users frequenting the
- **Identifying and predicting trends** based on whispers posted by people in categories like music, pop culture and sports etc.
- Getting a **general opinion of people** on sensitive current affair matters, this could help in political campaigns when the user base of the app grows.
- Since this app also has an inbuilt location detector, it can be used to identify **area specific sentiment** of the users.

2. Business Questions

Based on the value additions mentioned above, four business questions that can be answered using this dataset would be:

- Based on whispers in the "personal" category, which is the **most common suffering of people in a particular age group/location** ?
- What are the **most frequented and least frequented categories** of whispers in this app? Answering this question can help the company improve their accuracy in categorising whispers in the relevant categories.
- What is the **current hot topic** amongst users?
- Given the wide range of categories on this app, it is possible for the company to extract overall public sentiment on issues pertaining to that topic, assuming that the whispers have been accurately classified.

3. Target Question and Business Value behind answering it

The main assumption based on which the company can derive great value from the thousands of whispers they receive each day, is that they are able to successfully categorise the whispers into 17 categories.
Therefore the target question would be to identify which of the 17 categories does the whisper belong to. The model will be tested using the whispers from the test.csv dataset.

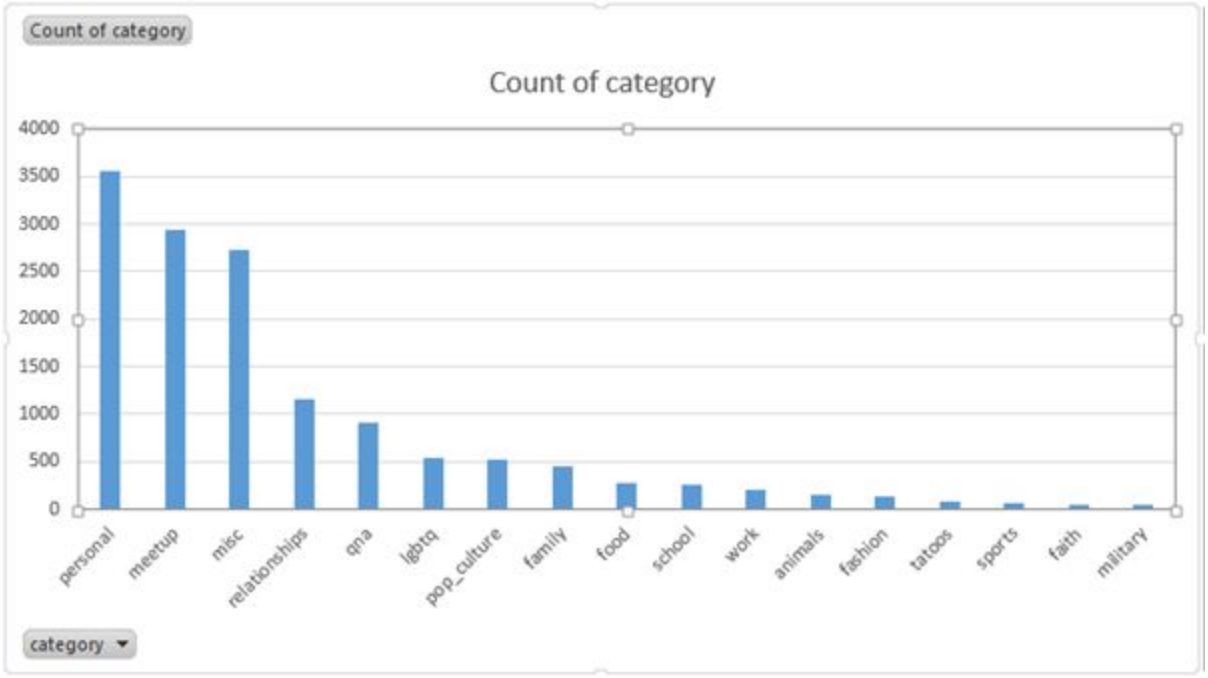
4. Flow of the project

This section of the reports intends to convey our approach to solve the target question, what we learned at each step of the process as well as an analysis of the results at each step.

Our main aim was to increase the accuracy of this multi-label text classification problem, and we used the **F1 Macro Score as the evaluation metric** to test our results.

Step 1

On briefly skimming through the data, it was clear that there needed to be pre-processing done on the training dataset in order to extract knowledge from it and feed it into various models to make predictions.
We also plotted a graph in order to get an understanding of the distribution of the training set data.



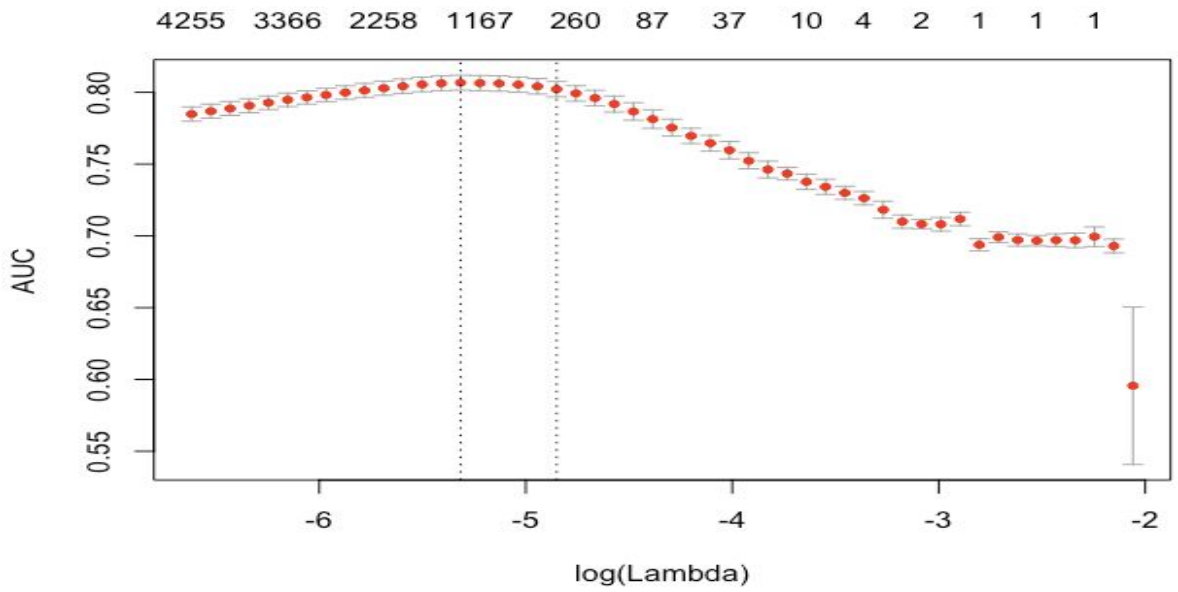
This was a clear case of multi-label classification having an imbalance of labels.

Step 2

Even though the end goal is to segregate the whispers into multiple categories but the primary aim was to remove the “personal” whispers because they are most important category of whispers. The “personal” category is important because the primary aim of the app is to help people with expressing those emotions and issues *anonymously.*

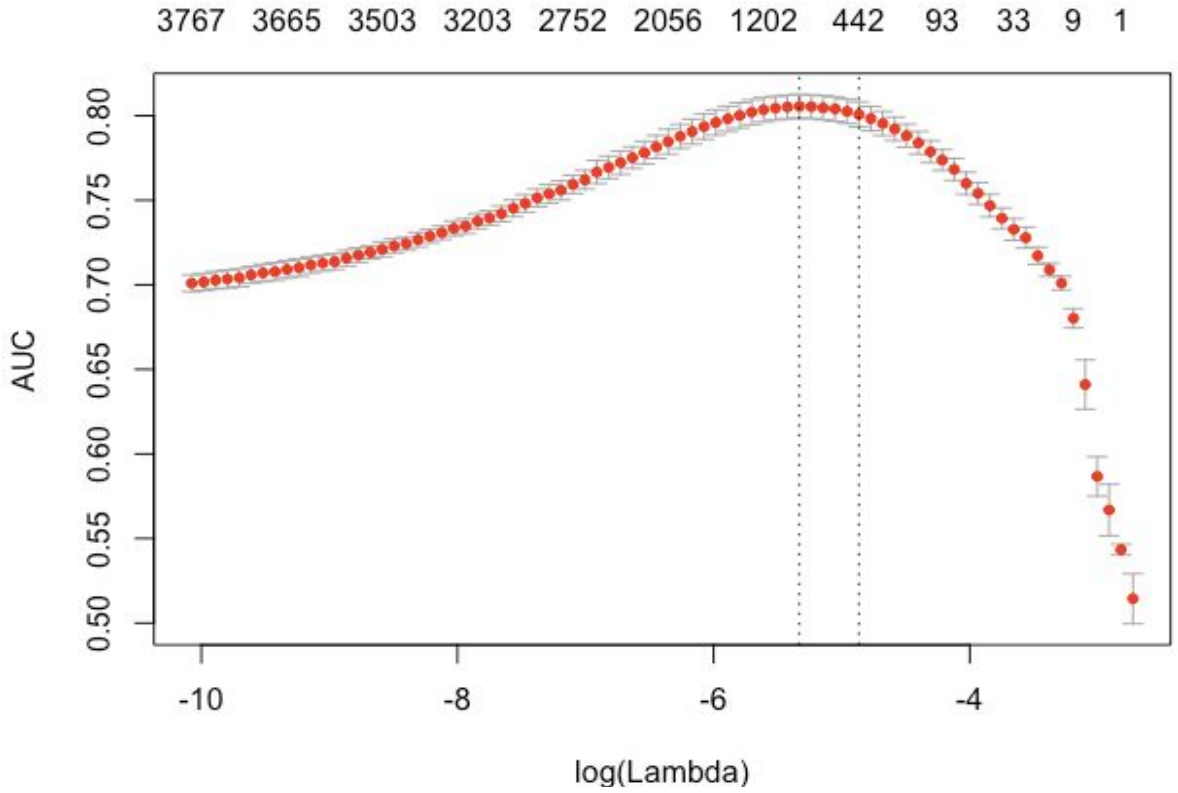
In order to achieve this we have implemented binary classifier on the data. We categorized the whispers in training data in two broad class of labels i.e personal and non-personal. The personal data was identified using the label 1 whereas all other classes were identified using the label 0. The Generalized linear model was implemented to obtain a binary classification of whispers and various other iterations of preprocessing were performed. The preprocessing included removing stopwords. But it is advisable not to remove stopwords while using doc2vec because the context is lost. Hence the stopword removal mechanism was implemented after the formulation of the doc2vec model.The R codes for all the sub-implementations have been included in one common R script named GLM.R

The plots for each iteration of the implementation have been included in the folder named plots
Iteration 1:
No preprocessing on whispers. Just identify the text and the category from training data to create a model. Here the personal data is labeled as 1 while all other categories are labeled as 0. The AUC graph has been shown below which clearly indicates an accuracy of 80%.

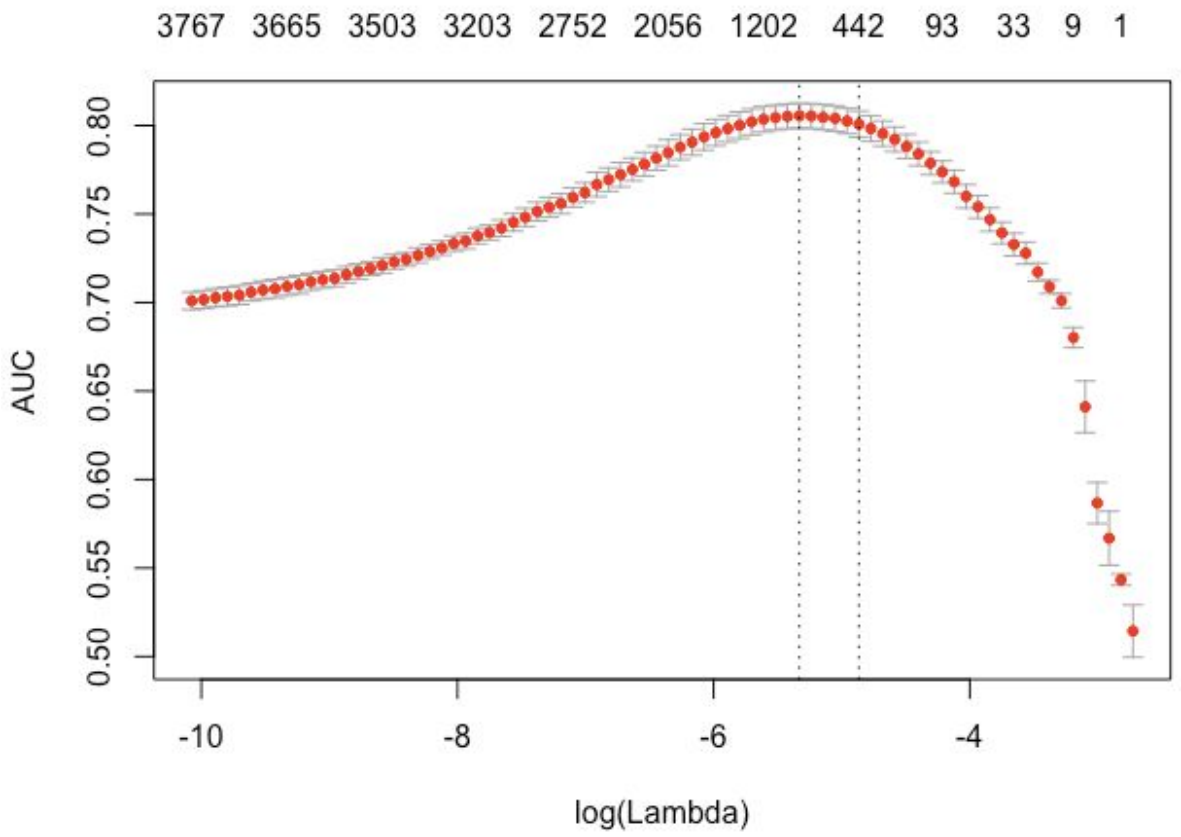


Iteration 2:

Stopwords removed from the data. Even though we did not receive many stop-words there was still a small rise of accuracy to 81%. This rise was accountable because frequency of irrelevant words increases in the bag of word model.



Iteration 3:
The whispers from which stopwords were removed were passed through a tfidf pipeline to obtain features which were used as input for the GLM classifier. There was not much increase in the accuracy as the contextual information of words was lost while using tfidf. The accuracy was around 79% which was slightly lower than the previously used methods



Step 3

We realised that we needed a benchmark or a baseline model for us to compare the accuracy of any models that we planned to build. On researching online and going through various papers, we thought of two alternatives for having a **baseline model:**

1. Accuracy of a model that classifies all the whispers as “personal”
2. **Using tfidfvectorizer to make feature vectors for each whisper in the training set and then building a model using Logistic Regression.**

We decided to go ahead with the second alternative. The F1 Macro score for this baseline model was 0.3236

Step 4

Observations of the baseline model:
On further probing the baseline model and interpreting the results, it was seen that some categories did not have any whispers assigned to it at all, which pulled down the F1 Macro. Based on the performance of this model, we adopted the approaches that follow.

Step 5
Attempt with Doc2Vec using Naïve Bayes, Linear Models

In this attempt we tried predicting the labels for the data in the test set using the Doc2Vec model. This included making feature vector representation for each whisper which would then be fed to the predicting models

Naïve Bayes Implementation:

- **Below Baseline Performance:**
    - The reason for this is probably hidden in the imbalance of the labels. As seen in the excel graph above, there are approximately 3-4 dominant labels in the list of 17 labels. Naïve Bayes takes prior probability of the label into account (i.e. probability of label occurrence in training set) while predicting probability of labels on test set, thus leading to bias.
    - **F1 Macro Score: 0.1392**

Linear Model Implementation:

- **Below Baseline Performance:**
    - This result was slightly perplexing to me because we had expected it to do well on this model. However the most plausible explanation for this poor performance could be the fact that the feature vectors generated using the data was a poor representation of the whispers. The reason being not enough vocabulary for per-category whispers to make and train the doc2vec vectors.
    - **F1 Macro Score: 0.1150**

Interesting Insight: In order to find optimal value for the "window" parameter of the Doc2Vec function, a parameter that determines the range of the words considered while predicting a word in the whisper, we took the average length of the whispers after some initial pre-processing which included deleting stopwords and words of length less than 3. The value was roughly 8 probable useful words per whisper.

Step 6
Attempt to find accuracy of categorising top 4 labels using baseline model

- This was an attempt to check whether it was possible to split the data into five labels. The top 4 labels in terms of weight would be represented as it is, whereas the others would be labelled as "misc2".
- The logic behind this was to achieve higher accuracy for the most frequented whispers categories. However, there was no significant change in the predictions in this attempt and those for the top 4 categories in the baseline model.
- **F1 Score for predicting top 4 labels: 0.6337**

Interesting Insight 1: While attempting to increase the accuracy of the model for the top 4 labels, we experimented with various parameters of the TfidfVectorizer to further investigate any scope for pre-processing. This led to finding out near optimal value for maximum features in the data set based on term frequency. Sublinear term frequency scaling also was a helpful revelation. These setting were used in the models that follow.

Interesting Insight 2: Naïve Bayes performed as well or slightly better than Logistic Regression in this model, thus validating the uneven label distribution.

Step 7
Attempt using Ensemble of Models (Random Forests) and Manually Engineered Features

This was an effort to use an ensemble of Decision Tree models in order to come up with the best possible prediction.

- **Above Baseline Model Performance**
    - This attempt turned out to be a successful one. The model was built in such a way that random subset of features in the vocabulary were considered while considering the best split.
    - In addition to that, a few keywords were found for those labels having no whispers assigned to them. Due to the excessively small number of whispers listed in these categories, coming up with the keywords required only a brief glance through the data. However the same process can be automated by counting frequency of words in each category to determine top two or three

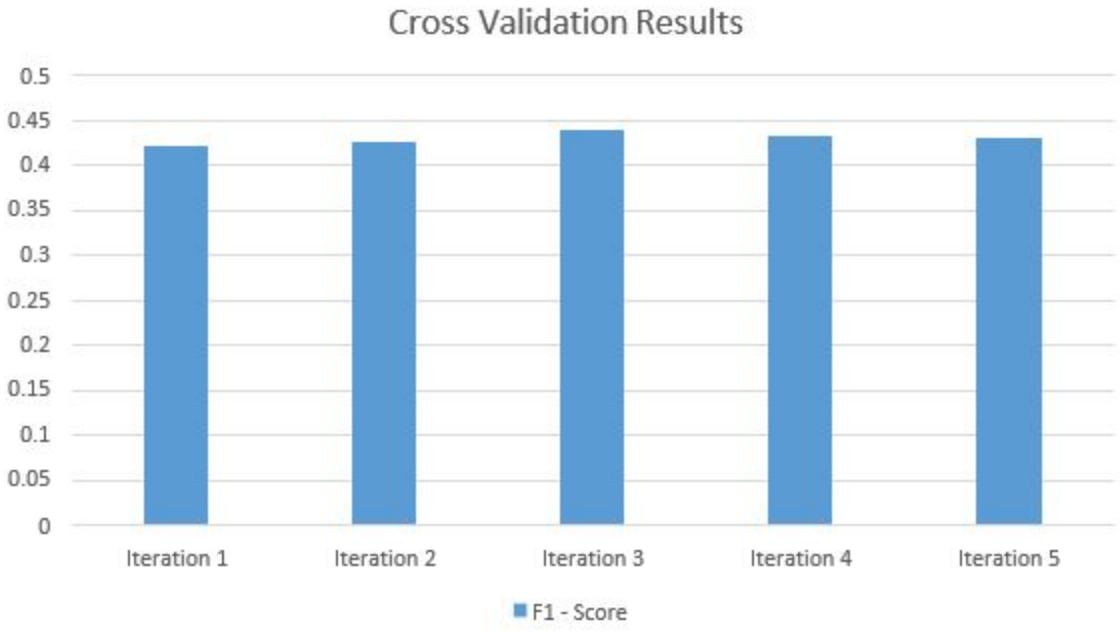
keywords. After getting a prediction from the model, based on these keywords the predicted labels were reassigned.

- **F1 Macro Score: 0.40308 (slight variations observed as decision trees are random)**

Step 8

Attempt to build on Baseline Model using learnings from above attempts and Cross Validation

- **Best Performance**
  - This attempt included gathering all the learnings from the previous attempts and trying to replicate them in the baseline model such as the parameter tuning of TfidfVectorizer, usage of manually engineered features etc.
  - In order to verify our results, we also **implemented the concept of cross validation** in this. This would help us understand whether we did any sort of overfitting, and estimate the **out of sample accuracy** for the same.
  - Making a non-linear model on this dataset could lead to overfitting given the dearth of data, hence leading to incorrect predictions or susceptibility to noise. In contrast the linear model seems to have done a good job at avoiding any sort of overfitting.
  - The F1 macro scores recorded during cross validation are very similar to final F1 macro score for this model which reconfirms no overfitting.



F1 Macro Scores for cross-validation:

0.4218
0.3978
0.4260
0.3814
0.4305

- **Final Model F1 Score: 0.4305**

Interesting Insight: The manually engineered features led all categories having whispers assigned to them.

As seen above, we were able to increase the F1 score of our final model by around 11% when compared to the score of the baseline model.

There were several other approaches that we thought of implementing for bettering this F1 score, however were not able to implement it due to shortage of time:

- We had thought of trying a method of upsampling and downsampling, i.e. manually doubling the whispers of under-represented categories in the training set till they reached at least the average value of the distribution of the categories in the set. We intended to make up for the low amount of training data for the under-represented categories this way. This is similar to the implementation of the concept of “Bagging”
- We also tried investigating the composition of the list of stopwords used in Python, so as to see if there were any words having lexical values that were being omitted from the dataset. This led to the thought process of developing our own list of stopwords based on the tf-idf of the words in the data set.

Python packages used:

1. sklearn
2. gensim
3. csv
4. nltk
5. random

R packages used:

1. text2vec
2. readr
3. glmnet
4. caret
5. ROCR

5. To recommend 3 research papers that could be relevant to solving your target question and write a paragraph summary for each paper: When making your recommendations, pay attention to how many people cited such papers (Cited by field in Google Scholar): highly cited papers may indicate more impactful technology and/or earlier technology, and Whether these papers have been published in high impact conferences such as SIGMOD KDD, ICDM, ICDE, NIPS, etc. (see computer science conference rankings: https://en.wikipedia.org/wiki/List_of_computer_science_conferences).

All the papers that have been cited or used have been included in the folder named research papers.

Following are the research papers we referred which helped us gain insight:

- Quoc Le, Tomas Mikolov. **Distributed Representations of Sentences and Documents:**

Bag of words features has two main shortcomings:

1. They lose the ordering of the words.
2. The also lose ignore semantics of the words.

But paragraph vectors overcomes the above shortcomings. The text can be variable-length from as short as sentences to as long as documents. Basically, it adds the paragraph vectors with many word vectors from a paragraph and then it starts predicting the following word in the given context. Both word and paragraph vectors are trained by stochastic gradient descent and backpropagation. Paragraph vectors are predicted by fixing the word vector and training the new paragraph vector until convergence. By the end of the paper, for text classification problem, this method beats bag-of-words model by giving a relative improvement of 30%.

- Mingyong Liu and Jiangang Yang. **An improvement of TFIDF weighting in text categorization**

Due to the rapid growth in online information and wide usage of text categorization in fields like email classification, junk email filtering, topic spotting etc there is a need to find new document representation models. The paper proposes an improvement in TFIDF where it considers the term frequency related to a class which is overlooked in the standard implementation. The initial part of the paper focuses on preprocessing and standard approaches used for text categorization.
Regarding the shortcomings TF-IDF has, the paper introduces a new parameter to represent the in-class characteristics, and we call this class frequency, which calculates the term frequency in documents within one class.

$$a_{ij} = \log(tf_{ij}+1.0) * \log(\frac{N+1.0}{n_j}) * \frac{n_{cij}}{N_{ci}}$$

n_{cij} represents the number of documents where term j appears within the same class c document i belongs to, N_{ci} represents the number of documents within the same class c document i belongs to.
This improvement increases the accuracy significantly, therefore we think this improvement is promising.

- Xiaojin Zhu **CS838-1 Advanced NLP: Text Categorization with Logistic Regression**

Naive Bayes is a generative model. It models the joint p(x, y). However, if our ultimate goal is classification, the relevant part is p(y|x). In Naive Bayes this is computed via Bayes rule. One might wonder whether it is possible to estimate p(y|x) directly. A model that estimates p(y|x) directly is known as a discriminative model. Logistic regression is one such model. The paper discusses 3 parts namely the training method for logistic regression, graphical model and logistic regression as a linear classifier. Recall that Naive Bayes is a linear classifier too. They both divide the feature space X with a hyperplane. Their essential difference is how they find their hyperplane: Naive Bayes optimizes a generative objective function, while logistic regression optimizes a discriminative objective function. It has been shown that logistic regression tends to have higher accuracy when training data is plenty. However, Naive Bayes can have an advantage when the training data size is small.

Survey Papers:

- **Text Classification by Augmenting the Bag-of-Words Representation with Redundancy-Compensated Bigrams**
  Constantinos Boulis, Mari Ostendorf

- **Credibility Adjusted Term Frequency: A Supervised Term Weighting Scheme for Sentiment Analysis and Text Classification**
  Yoon Kim, Owen Zhang