Studying and Analysing the Evolution of a User's Answering Style Over a Period of Time on Quora and Developing Models to Predict it

Group No.4, members :

- Manav Kedia, 12CS10057 (Team Leader)
- Pramesh Gupta, 12CS10036
- Sudhanshu Bahety, 12CS10063
- Sunny Dhamnani, 12CS30037
- Sanket Kedia, 12CS30044
- Vatsalya Chauhan, 12CS10053
- Swapnil Agarwal 12CS10052

Mentor:

- Suman Kalyan Maity

Objectives and Motivation:

- Quantify different characteristics of a user's answering style namely:
    - Humour
    - Satire
    - Abusiveness
    - Readability
    - Formalness
    - Detailing of answer
    - Sarcasm
    - Argumentation

  by assigning a probability to each feature

- Acquire temporal answer data for different users and analyze and quantify how a user's writing style changes over the course of his answers

- Develop and evaluate models to predict these changes in the future for a particular reader. The model would take several inputs which could range from upvotes, likes, views, his previous answering preferences and past answering temporal changes

Quora dataset description:

Obtained a list of 1690 users that have cumulatively answered 104697 questions. From these list of users filtered the list of users that have answers over a period of 1 year and have answered at least 50 questions so that we have enough temporal data for our analysis

Feature Modules

Name: Humor

Done by: Manav Kedia

Datasets: 16000 one liners classified as humor and 16000 one-liners classified as serious text

Brief method: In order to convert each training example into a feature vector, I extracted the counts of each bigram and normalized them. This would become a very sparse vector. Then I used text classification algorithms. I ran both Multinomial Naive Bayes and SGD for classification. SGD classification is basically a linear classifier with SGD training(Stochastic Gradient Descent), it was very relevant for this case, since the feature vector was very sparse. SGD classifier outperforms naive bayes as discussed in the results section.

Implementation: Done in python using sklearn library

Results: I ran a 5 fold cross validation on the model and the accuracy for SGD was around 90% and the accuracy for Naive Bayes was around 82%

References: https://www.cs.ox.ac.uk/files/244/mihalcea.cicling07.pdf

Name: Readability

Done by: Sanket Kedia

Datasets: Dale-Chall word list of easy words, Spache word list, list of exceptional words and their syllables

Brief method:
Flesch-Kincaid readability test: Uses word length and sentence length as core measures of readability.

$$206.835 - 1.015\left(\frac{\text{total words}}{\text{total sentences}}\right) - 84.6\left(\frac{\text{total syllables}}{\text{total words}}\right).$$

Gunning-fog index: Works best for media data.

$$0.4\left[\left(\frac{\text{words}}{\text{sentences}}\right)+100\left(\frac{\text{complex words}}{\text{words}}\right)\right]$$

Coleman-Liau index:

$$CLI = 0.0588L - 0.296S - 15.8$$

L is the average number of letters per 100 words and S is the average number of words per 100 Sentences.

SMOG Index:

$$\text{grade} = 1.0430\sqrt{\text{number of polysyllables} \times \frac{30}{\text{number of sentences}}} + 3.1291$$

Automated Readability Index:

$$4.71\left(\frac{\text{characters}}{\text{words}}\right)+0.5\left(\frac{\text{words}}{\text{sentences}}\right)-21.43$$

Based on these indices we output a score out of 100. More the score, better the readability of the text.

Implementation: php

Results: Accuracy: around 90%

References: http://en.copian.ca/library/research/readab/readab.pdf
wikipedia of various readability indices

Name: Satire

Done by: Pramesh Gupta, Sudhanshu Bahety

Datasets: News article corpus by Burfoot et al with 4000 real news and 233 satire news. Slang and exaggeration vocabulary consisting of 6000 and 150 words respectively from Wiktionary.

Brief method: Many characteristics particular to satire can be effectively evaluated by bag-of-words model. First we parsed and grammatically tagged words in the corpus using the Stanford Parser. The features are:

1. Colloquialism - determine the fraction of words belonging to the slang vocabulary in the document using the bag of words model
2. Exaggeration - determine the fraction of words belonging to the exaggeration vocabulary in the document using the bag of words model
3. Topicality - determine the irrelevance of the document using Naive Bayes model.

We then select the optimal subset of vocabulary and use binomial separator (BNS) for feature weighting.

Implementation: Done in python

Results: Awaiting response from the author on a few queries to complete the work.

References: http://rfol.io/satire.pdf

Name: Abuse

Done by: Sudhanshu Bahety

Brief method: Used bag of words technique to measure the abuse. The abusive words were taken from www.noswearing.com/dictionary. The method involved:

1. Removing stopwords from nltk library.
2. Count the total number of swear words.
3. Count the total number of words (excluding stopwords)
4. Output the fraction of swear words in the document

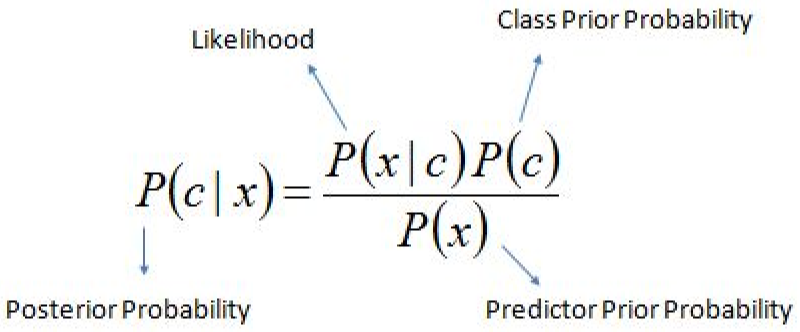
Implementation: Done in python
Output: A number in range [0,1].

Name: Formal/Informal

Done by: Sunny Dhamnani

Datasets: Used public classifier at uclassify.com

Brief method: The implementation uses naive bayes classifier which gives the probability of the input belonging to a particular class. In our case the classes are formal or informal, so the result is the probabilities belonging to this particular class.



$$P(c \mid x) = \frac{P(x \mid c)P(c)}{P(x)}$$

Features used in the model include, formal word list, informal word list, active voice, passive voice, type token ratio, abbreviations and phrasal verbs.

Implementation: Implemented in python
Output: A number in range [0,1].

$$P(c \mid X) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

References: https://www.site.uottawa.ca/~diana/publications/Fadi_IEEE_NLPK2010_camera_ready.pdf
http://journals.linguisticsociety.org/elanguage/lilt/article/download/2844/2844-5756-1-PB.pdf

Name: Sarcasm

Done by: Sunny Dhamnani

Discussion: Sarcasm detection is hard problem because it requires the knowledge how the sentence was spoken and also to detect the sarcasm we need the knowledge of the context.
As a result the current work uses some proxy to detect sarcasm

1. In twitter data the hashtags used by the users are used for sarcasm detection.
2. In amazon.com data the sarcasm is detected on reviews and the methodoly uses how positive the review is and rating given by user, so if rating is low and customer is giving positive review then there is a high chance that customer has posted sarcastic review.

Features used by these classifiers are generally sentiments, opinion score of sentence, mood etc.
We are still in a phase to find the proxy for tagging our data, we have come up to the conclusion that manually tagging data would be the best choice in this scenario.

Name: Argumentative

Done by: Vatsalya Chauhan

Discussion: In wikipedia, users are allowed to provide their own arguments based on some facts or knowledge. Which sometimes go in favor or against the arguments proposed by other users. And as a result, the information available on wikipedia varies with time. This paper propose a framework for the study of relationship among different arguments, which is basically composed of a natural language module. This framework can be used to answer following questions: (i) how to automatically discover the arguments and the relations among them?, (ii) how to have the overall view of the ongoing discussion to detect the winning arguments?, (iii) how to detect repeated arguments and avoid loops of changes ?, (iv) how to discover further information on the discussion history?

Datasets: Five most revised pages (i.e. George W. Bush, United States, Michael Jackson, Britney Spears, and World War II) of the English Wikipedia (Wiki 09, and Wiki 10). And then follow their yearly evolution, considering how they have been revised in the next Wikipedia versions (Wiki 11, and Wiki 12).

Brief method:

1) Data Set Generation method :- Position Independent Word Error Rate (PER) (0.2<PER<0.6) is used to measure the similarity between the sentences in a pair corresponding to different Wiki-versions. For each pair of extracted sentences, Textual Entailment (TE) pairs are created, setting the revised sentence (from Wiki 10) as T and the original sentence (from Wiki 09) as H. Pairs have been annotated with respect to the TE relation (i.e.YES/NO entailment).

2) Textual Entailment (TE system) :- EDITS system (Edit Distance Textual Entailment Suite) version 3.0, an open-source software package for RTE are used for determining the relationship between of arguments. EDITS implements a distance-based framework which assumes that the probability of an entailment relation between a given T-H pair is inversely proportional to the distance between T and H (i.e. the higher the distance, the lower is the probability of entailment).

Reasons for leaving the implementation in between:- As par our later discussion, we came to the conclusion that, we will not find Test data for Quora answers with reasonable PER. But, might be possible with manually annotated dataset.

Name: Detailed answer

Done by: Swapnil Agrawal, Vatsalya Chauhan

Dataset: Public Yahoo dataset for Questions and Answers (including best answer)

Brief Method:
This feature determines the Quality of Answer. In order to produce training dataset from Question-Answer pair, we extracted following features from them:-

1. Overlap-QA - This feature gives the proportion of the number of overlapping 1-grams between the answer and the question
2. Containment-QA - The containment gives the proportion of 1-grams from the answer that also appear in the question
3. Ratio A and Q's Raw Length - This feature gives the ratio between the raw lengths of a question and its answer.
4. Answer length - We expect that a good answer has a considerable word length.
5. Entropy of Answer - We measured the complexity of answers by the entropy of the words in the answers
6. Number of URLs used - For references
7. Punctuator density
8. Total Count of Nouns used
9. Total Count of Verb used
10. Number of Non-stop words

The importance of every feature is provided in the paper given below. Currently we ran Logistic Regression (it outperforms SGDClassifier) over training data for two classes { detailed, not-detailed } and results were promising but it's important to have more classes. Hence for our manually tagged dataset we are going to use four classes i.e. { 25%, 50%, 75%, 100% } detailed.

References: http://www.sciencedirect.com/science/article/pii/S0020025513007573#

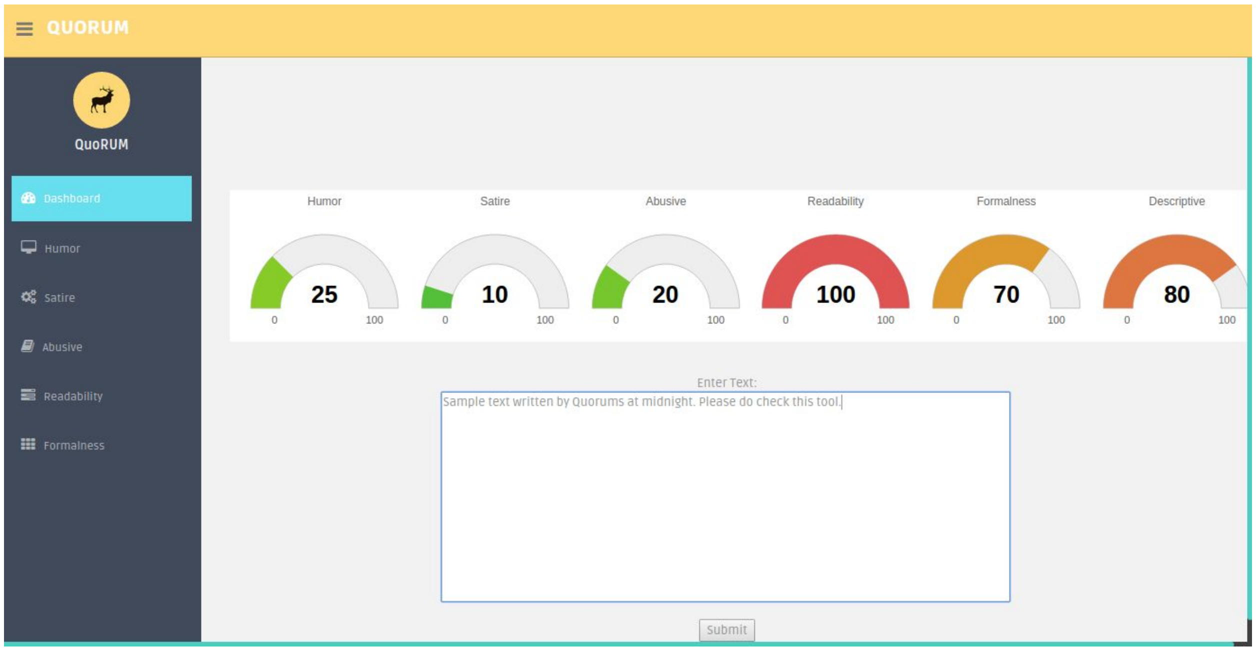
System design:

Done by: Vatsalya Chauhan, Sunny Dhamnani, Swapnil Agrawal

- User friendly UI - Runs all the codes on single click, great visualization tool
- Used D3 javascript to render real time visualizations
- Used Python flask for rendering html content
- Used php to communicate between javascript and feature implementations

System will be up by tomorrow and link will be mailed

Screenshot of the tool



Sample text written by Quorums at midnight. Please do check this tool.

Timeline for the remaining semester:

- Some of the features like humor currently only give a binary output, we plan to convert it into a percentage score
- Manually annotate the dataset for each of the features for 1200 answers to obtain the oracle. Compare this with the current datasets of different modules and see how it applies on our dataset of quora answers
- Evaluate our model for each feature from this manually annotated dataset
- Combine the individual modules to a final output vector that would be representative of all the above feature modules
- Observe variations in writing style from this combined feature module
- Fit models to predict the how the writing style of a user would change over time

Appendix:

Name	Roll No.	Contribution
Manav Kedia	12CS10057	Implemented the humor feature module, temporal data extraction from the set of answers and managing the team
Sanket Kedia	12CS30044	Readability feature module, parsing the dataset to obtain question, answer, upvotes, links etc.
Pramesh Gupta	12CS10036	Satire feature module and parsing the initial dataset of quora user logs to extract answer links.
Sudhanshu Bahety	12CS10063	Satire feature module and abusive measure in the document
Vatsalya Chauhan	12CS10053	1. Argumentation feature study and partial implementation. 2. Detailed answer feature module. 3. System design frontend and back-end (back-end code runs all the feature modules and shows individual and weighted score of features).
Sunny Dhamnani	12CS30037	Implemented formal/informal writing style feature, working on sarcasm feature. Built the web interface for our team to visualize our results better.
Swapnil Agrawal	12CS10052	Implemented detailed answer feature. Contributed in building back-end of UI system.