Exploring Linguistic Signal for Schizophrenia

Hang Ma, Hong Wei, Zehua Zeng, Heng Zhang

May 16, 2016

1 Introduction

1.1 Team Members and Roles

Our team includes Hang Ma, Hong Wei, Zehua Zeng and Heng Zhang. Roughly we are breaking down the project (excluding writing the report) into 3 parts: propose and extract features, apply supervised classifications and evaluations. More specifically, the individual task is listed as follows:

- **Hang Ma:**
  - Extract features: neologism; sentence incoherence
- **Hong Wei:**
  - Extract features: word classes, spelling errors, n-grams; topic incoherence; readability; social activities
  - Adopt vowpal wabbit software to conduct supervised classification
  - Evaluate and analyze the results of vowpal wabbit
- **Zehua Zeng:**
  - Extract features: occurrence of clausal junction words, occurrence of proposition words; association with schizophrenic topic
  - Write result analysis and plot all figures in the report
- **Heng Zhang:**
  - Extract features: all sentence-based features
  - Apply Python scikit-learn package to conduct supervised classification, including Nearest Neighbor, Support Vector Machine, Decision Tree, Adaboost and Logistic regression classifiers.
  - Compare the classification accuracy of each type of features (word-based features, sentence-based feature, topic-based features etc) as well as the overall fusion of all the types of features

For writing the report, each person is in charge of describing the part of work she/he has done. In the end we discuss and review the report as a whole.

1.2 Proposal Review

In the project plan, we proposed to extract a list of relevant properties of languages we are trying to exploit as potential features distinguishing “schizophrenic users” from controls. The features are categorized into word-based, sentence-based, topic-based, readability-based, and non-language-based. Choosing these features is guided through assumptions that potential association exists between them with schizophrenia.

Eventually, for each anonymized user, we extracted 313 features in total. By taking all 313 features together, we conducted supervised classifications by using scikit-learn and vowpal wabbit, and expected
Furthermore, in order to validate the effectiveness of the extracted features, we did a raw category-grained feature selection procedure to determine which category of features are the prominent ones in distinguishing schizophrenic users from control users.

2 Data and Methods

2.1 Qntfy Dataset

The Qntfy dataset is provided by Glen Coppersmith for educational/research use in Prof. Philip Resnik’s class Computational Linguistics. The dataset contains Twitter data from 137 self-reported schizophrenic users, along with another 137 users in control. The Qntfy dataset also has a manifest CSV file identifying users in the schizophrenia and control groups by anonymous ID, along with gender, age estimate, number of tweets and a cross-fold validation group id.

In each user, the history of his available Twitter statuses is provided in json format. Note that, the amounts of statues of users are not the same. An overview of tweets count is provided in Figure 1. The x-axis is referring to the 137 pairs of schizophrenia/control users (every such pair is represented by its order ranging from 0 to 137 in the manifest list, the y-axis denotes the amount of tweets collected. The value of tweets count in Figure 1 is calculated through counting how many lines in each user’s tweets file. In each pair of histograms, the red one represents the schizophrenia user, and the green one represents his/her counterpart user in control. Although it might not be a consistent observation, the users usually having the most tweets fall into schizophrenia, for example, the top 6.

The Qntfy CSV file also provides a tweets count for each user. However, a classmate found this value might be inconsistent with the one we calculated in Figure 1. Therefore, we plotted Figure 2 to show the above inconsistency. The x-axis is referring to each user with his order in the manifest list. And the y-axis demonstrates the differences by subtracting the tweets number calculated through tweets files from the tweets number provided in manifest CSV file. The results are quiet surprisingly i) more than half of the tweets numbers (153 out of 274) are inconsistent, ii) the deviations can be huge with the maximum being -10384, iii) all values in manifest CSV file are smaller than the ones calculated through
2.2 Data Preprocessing

As the features every team member proposed to extract are different, the preprocessing procedures each one used are different. But generally, a space-delimiter based tokenizer or the default English tokenizer in Python’s NLTK is used for tokenization. For more details of the procedures used in extracting features, see section 2.3.

2.3 Features

The features our team extracted are categorized into word-based, sentence-based, topic-based, readability-based, and non-language-based. In the following, we will describe these characteristics we looked at, along with the methods we used to extract them.

1. Word-Based

- **Word Classes:** The data of word class for each English word is obtained in WordNet, or more specifically, its Synonym Set and Hypernyms. In preprocessing, a default tokenizer in Python’s NLTK (http://www.nltk.org/) is used and then the tokens that are not made up of only alphabets are discarded. For the rest of the tokens, in order to retrieve each word’s word class, we first look-up all its synonyms. A synonym consists of three parts: word, tag-of-speech, and priority. Here, we only keep the synonyms of a input word, in which the word matches the input word and priority is the highest, i.e. 01. At last, we look-up the hypernyms of each input word’s synonyms and treat them as the word classes of the input word. Note that each input word might have more than one word classes in the end. In the end, for each user, we take his top 8 word classes by frequency as a feature. The reason by choosing top 8 is because one of the schizophrenia user ioY8SXeZ40 has only 8 available word classes. In order to align the features format among all the users, word-class feature is truncated to only have 8 word classes. The reason ioY8SXeZ40 has so few word classes might be his enormous usage of non-English words, e.g., his last tweet looks like:

  
  \u0412\u043e\u0440\u043e \u043f\u043e\u0442\u0430\u043a\u0430\u0442\u044c,  
  \u0447\u0442\u043e \u0441\u0430\u043c\u0443  
  \u0432\u043e\u0440\u043e\u0432\u0430\u0442\u043e\u043c\u0443

In the end, as classifiers usually only take numbers as input features (vowpal wabbit is an exception), each of the above word classes is eventually represented by its index in a dictionary composed of all WordNet’s synonyms.

- **Spelling Errors:** For each user, we extract his statistical characteristics regarding to his spelling error ratios, i.e., the ratio of the number of spelling error in a tweet over the length of this tweet. We then calculate min, max, mean, median, standard-deviation, variance for each of the user and use these statistical values as his feature of spelling error. To check if a token is a spelling error, we used Python’s enchant library by checking if the set of words suggest for this token contains it, otherwise, we treat this token as a spelling error.

- **N-Grams:** For each user, we extract top 20 unigrams and bigrams. The choose of value 20 is based on the experience in class assignments. For unigram, the top 20 tokens are chosen by frequency; for unigram, we used three different measures to calculate the top 20: χ-square, log-likelihood and frequency. Like what we did to transform word-classes to integers, we use words list in Python’s NLTK as the dictionary and represent each of unigram to its index in words list, each of bigram to a pair of indices.

- **Occurence of Clausal Junction Words:** We select a set of common clausal junction words. For each tweet, we count the occurrence of these clausal junction words. We then calculate min, max, mean, median, standard-deviation, variance for each of the user and use these statistical values as his feature of occurrence of clausal junction words. Note that, in this step, we don’t have a parsing procedure, which will be done to extract clausal embeddings. Thus, we might get false positive clausal junction words here.
• Occurrence of Preposition Words: Like the above feature, we select a set of common proposition words. Then, for each tweet, we count the occurrence of these proposition words. We then calculate \( \text{min, max, mean, median, standard-deviation, variance} \) for each of the user and use these statistical values as his feature of occurrence of proposition words. Note that, in this step, we don’t have a parsing procedure, which will be done to extract prepositional phrases. Thus, we might get false positive preposition words here.

• Neologism: In the literature, higher-than-normal use of neologisms is a main feature in the language use of schizophrenia. In this project, we explore this feature in the provided dataset. To identify a word as neologism, we define it as the word which had been rarely used by other people, i.e., the self-made or the newly made-up words. In this project, we only focus on the unigram. The neologism is identified though comparing with another corpus on the Twitter data. We use the unigram model obtained from [http://clic.cimec.unitn.it/amac/twitter_ngram/](http://clic.cimec.unitn.it/amac/twitter_ngram/). If a specific word in the current dataset is not in the previous corpus, this word is identified as a neologism.

By looking through the results, we find some of the words are just normal use today, but are not in the unigram dataset. For example, the word “snapchat” is identified as a neologism while we know using this word is quite normal. The reason is that the corpus used for comparison is based on 75 million English tweets extracted from a larger sample of 240 million tweets collected from the public stream of Twitter, between December 2010 and July 2011. At that time, Snapchat had not been founded yet. In the lack of proper recent Twitter corpus, we add one more filter to reduce the false alarm in the neologism detection.

If a word is identified as a neologism in the previous step, we look at the use of this word in the current dataset. We calculate the exclusive usage score of this word by a user, which is the total use of this word by this specific user divided by the total use of this word by all the users. If a word is self-made, it should be exclusively used by one specific user while not used by others. As a result, the exclusive usage score is high, and the word is identified as a neologism. On the other hand, if a word is not self-made, it is widely used. Consequently, the exclusive usage score is low, and this word is not identified as a neologism.

2. Sentence-Based To analyze the syntactic structure of the tweets, we first tokenize the tweets using the TweetsTokenizer in NLTK, and specifically we set the `reduce_len` option and `strip_handles` option to be true and this helps to better tokenizer the tweets. Then, we apply Stanford Parser ([http://nlp.stanford.edu/software/lex-parser.shtml](http://nlp.stanford.edu/software/lex-parser.shtml)) to parse the tokenized tweets and obtain the parse tree. From the parse tree, we can get the height of the tree, width of the tree (length of the tweets) and explore the Subordinate Clauses which are sub-tree rooted at the node with label `SBAR`. As a consequence, for each tweet, we use the length of the tweet, height of the parse tree, number of Subordinate Clauses and maximum length of all the Subordinates Clauses. Note that the last two numbers could be zero as many tweets do not contain any Subordinate Clause. For each user with various number of tweets, we compute the minimum, maximum and mean value for all the four numbers above using around 300 sampled tweets. Therefore, we have a 12 dimension feature vectors capturing the syntactic structure of each user.

3. Topic-Based

• Topic Incoherence: To check a user’s topic incoherence, i.e., his variance on the distribution of topics extracted from his tweets, we used TwitterLDA [7], which is a unsupervised LDA developed specifically for tweets. One of the TwitterLDA’s implementation can be found at [https://github.com/minghui/Twitter-LDA](https://github.com/minghui/Twitter-LDA) For each user, we generate a document for him in which each line composes of one of his tweet text. We chose \( k = 100 \) as the number of topics and iterate 50 times. In the end, we obtained each user’s topic distribution (i.e., a 100 dimension vector of float numbers) on these 100 topics. Then the feature of topic incoherence
is represented, like spelling errors, by this distribution’s statistical characteristics: min, max, mean, median, standard-deviation, variance.

• Association with Schizophrenic Topic: Different schizophrenic patients might show some similar symptoms. For each user, we first get the “clean text” of all tweets, and then use supervised topic model (sLDA) to a model on the “clean text”. We get rid of all website urls, hashtags, user mentions and stop words to get the clean text. For tokenization, we also lowercase all alphabet characters, use Twitter Tokenizer (http://www.cs.cmu.edu/ark/TweetNLP/) to tokenize all tweets. We also use WordNet Lemmatizer in NLTK to lemmatize words in the “clean text”. With the sLDA model, we calculate the regression value for each user. The regression value can not only be the feature of each user, but also be the prediction of each user.

• Semantic Incoherence: As one of the schizophrenic symptom, the patient might have difficulty beginning or sustaining activities and trouble focusing. Therefore, in this work, we would like to used the topic based model to evaluate the topic consistency of their tweets. To evaluate the topic consistency, in this work, we use the latent semantic analysis (LSA). By reducing the dimension in the SVD, the tweets of a specific user are grouped into these dimensions, and each dimension is corresponding to one topic. We evaluate the topic consistency between consecutive tweets by measuring the cosine distance between consecutive tweets. It is assumed that the topics of a schizophrenia patient is more divergent, and the cosine distance between the consecutive tweets are larger. If a user has N tweets, then N-1 distances are calculated. The max, min, mean, median and standard deviation of the N-1 distances are recorded. Moreover, since there is no prior knowledge on the number of topics, we vary the dimension from 1 to 7. The LSA is performed using the tools available at github.com/josephwilk/semanticpy.git.

4. Readability-Based

• To evaluate how readable a user’s tweet texts, we use the readability library in Python. This library takes a sentence and outputs measures of readability. There are 8 different readability measures in total: Kincaid, ARI, Coleman-Liau, FleschReadingEase, GunningFogIndex, LIX, SMOGIndex, RIX. Each of these measures gives a float number. As we don’t know which of them would be better, all of them are taken into consideration. For each tweet text, we obtain these 8 measures, then for a user of many tweets, we use 8 set (each set corresponds to a measurement) of statistical characteristics, i.e., min, max, mean, median, standard-deviation, variance, to represent the user’s readability.

5. Non-Language-Based/Social Activity Based

• For non-language-based feature, we extract a Twitter users’ profile-related features and tweet-related features. Profile-related features are listed as: favourites count, tweet rate (number of posts per day), friends count, follower count, tweet count, statuses count and listed count. Tweet-related are listed as: activities, in reply count, retweeted count, retweetings count, favorite count. For profile-related features, it is represented by a single numerical value; for tweet-related features, we as usually used their respective statistical characteristics. Note that the activities count refers to the count of user mentions.

3 Classification Algorithm/Software

In this Section, we present the classification results based on two machine learning packages. The experiment we conduct is binary classification as there are only two classes, i.e control and schizophrenia.

3.1 Classification using Python scikit-learn Package

Python scikit-learn package contains a bunch of supervised and unsupervised machine learning models which make classification experiment easy. First, we apply min-max normalization to feature vectors and this helps the learning algorithm converge fast. Then, we apply nearest neighbor classifier, Support Vector Machine, Decision Tree, Adaboost classifier and Logistic Regression Classifier. To explore the contribution of each type of feature vectors, we also conduct classification using single type of features.
3.2 Vowpal Wabbit Fast Learning

Instead of the above classification method, we also tried our features on Vowpal Wabbit at [http://hunch.net/~vw/](http://hunch.net/~vw/). By default, Vowpal Wabbit uses a gradient descent procedure to optimize a weight function with a given loss function, which by default is “squared”, more details can be found at [http://www.zinkov.com/posts/2013-08-13-vowpal-tutorial/](http://www.zinkov.com/posts/2013-08-13-vowpal-tutorial/). A command to train a model usually looks like:

```
vw anonymized_cs.10-0.tr -c --passes 77 -l 0.0625 --ngram 2 -f anonymized_cs.10-0.tr.model
```

The relevant parameters are explained as in the following:

- `anonymized_cs.10-0.tr`: the input training data, number 10 means we divide the data into 10 parts (0 ∼ 9) as specified in the manifest file, −0 means to remove the first part of data and use the rest 9 as the training dataset, thus making the first part of data as the testing dataset.
- `-c`: this option tells Vowal Wabbit program `vw` to use cache in multiple pass learning, and it is usually accompanied by the following `--passes` option.
- `--passes`: this indicates the number of times the algorithm will cycle over the training data.
- `--l`: this option denotes the value of learning rate or the step length in gradient descent.
- `--ngram`: ngram option is the most important parameter in our experiment settings. As Vowpal Wabbit also handles features formed as plain text, it has an option “–ngram” to decide if ngrams in the given text are extracted as new features. As we talked in section 2.3, top 20 bigrams for each of the user are also extracted as features, we turned on the “–ngram” option with value of 2. Note that in order for this bigram to work, the sequence of words in bigrams are kept in the feature list fed to Vowpal Wabbit. Our experiments show that “–ngram” option plays a deciding role in improving accuracy. With this option on, we are usually able to achieve a 100% accuracy across different cross-validation settings. The x-axis is referring to which set of users are treated as the testing data. For the data size in each testing settings, please refer to Figure 6. As we can see, here the accuracy is achieved at 100% across all the testing settings in 10-fold cross-validation. the 137 pairs of schizophrenia/control users (every such pair is represented by its order ranging from 0 to 137 in the manifest list, the y-axis denotes the amount of tweets collected.
- `-f`: this specifies the file name of the trained model going to be saved.

In the above command, the values 77 of “--passes” and 0.0625 of “-l” are found in a grid searching method with the former ranging from 1 to 100, and the latter $2^{-10}$ to $2^{10}$. Note that there are more than one optimal settings of “--passes” and “-l” for the model to get highest accuracy.

Please also note that, Vowpal Wabbit by default turns on “holdout” validation option, which is to hold out a 1/10 subset of examples and use it to calculate test losses in multiple pass learning whenever “passes” > 1.

4 Evaluation and Analysis

4.1 Exploratory Data Analysis

As mentioned in section 1.2 we have extracted 313 features in total. Moreover, in section 1.2 we show that models trained on sentence-based features can get a high accuracy in predicting schizophrenia users. In this section, we choose “Mean Width of SBAR” feature (shown in Figure 3) and “Mean Numbers of SBAR” (shown in Figure 4) to analyze the difference between schizophrenia users and control users.

For each user, the mean width of SBAR in Figure 3 is calculated through adding the width of each subordinate clause and dividing the overall number by the number of subordinate clauses. The mean numbers of SBAR in Figure 4 is calculated through adding the number of subordinate clauses in each tweet, and then dividing the overall number by the number of tweets. In each pair of histograms, the red one represents the schizophrenia user, and the green one represents his/her counterpart user in control.

From Figure 3 we can see that the majority (97 out of 137) of schizophrenia users have a higher mean width of SBAR than corresponding control users. The result of “mean numbers of SBAR” is
similar with the result of “mean width of SBAR”. Figure 4 also shows that the majority (97 out of 137) of schizophrenia users have a higher mean numbers of SBAR than corresponding control users. These results are opposite to our expectation. In proposal, we supposed that less phrasal complexity would be one of the schizophrenic symptoms. However, from Figure 3 and Figure 4, we can see that schizophrenia users not only have higher mean width of SBAR and higher mean numbers of SBAR, but also have some especially high values of mean width of SBAR and mean numbers of SBAR. We think perhaps the majority of schizophrenic users would unconsciously write sentences with more subordinate clauses and with more phrasal complexity than control users.

Although the results from sentence-based features are opposite to what we expected, those features capture the schizophrenic symptoms with a high quality, which also well proves our following classification validation results.

4.2 Classification Validation and Analysis

4.2.1 scikit-learn

We split the data (137 control users and 137 schizophrenia users) into 5 folds randomly, and train on any 4 folds and test on the remaining one fold. For SVM classifier and Logistic Regression Classifier, we tune the parameters and present the best results. We report the mean classification accuracy. The results are shown in Table [1]. From this table, we can have the following observations: (1) Word-level features (dimension 175), topic-level features (dimension 41), text-readability features (dimension 48) and social features (dimension 37) give pretty much the same results with accuracy below 70%; (2) sentence-level features (dimension 12) works significantly better under all the classifiers compared to other type of features. (3) Fusion of all the type of features does not yield better results, and the performance varies using different classifiers. Therefore, from these results, we can conclude that the sentence-level features which capturing the syntactic structure of users’ tweets can be a strong indicator of schizophrenia.

4.2.2 Vowpal Wabbit

- Metric: The metric we used is accuracy, i.e., how many of the users in a given testing dataset we can rightly classify to schizophrenia/control. In experiment settings, we label “schizophrenia” by 1 and “control” by 0. As the default learning algorithm in Vowpal Wabbit is a regression based, we discrete the returned label value to 0 if smaller than 0.5, otherwise to 1.
<table>
<thead>
<tr>
<th>Methods</th>
<th>Word</th>
<th>Sentence</th>
<th>Topic</th>
<th>Readability</th>
<th>Social</th>
<th>Fusion</th>
</tr>
</thead>
<tbody>
<tr>
<td>1NN</td>
<td>57.23</td>
<td>98.90</td>
<td>52.95</td>
<td>59.93</td>
<td>53.29</td>
<td>61.24</td>
</tr>
<tr>
<td>3NN</td>
<td>54.76</td>
<td>97.84</td>
<td>55.09</td>
<td>60.25</td>
<td>57.01</td>
<td>63.44</td>
</tr>
<tr>
<td>5NN</td>
<td>60.26</td>
<td>97.47</td>
<td>57.76</td>
<td>62.43</td>
<td>55.90</td>
<td>63.10</td>
</tr>
<tr>
<td>7NN</td>
<td>58.72</td>
<td>97.84</td>
<td>59.43</td>
<td>60.54</td>
<td>56.64</td>
<td>62.04</td>
</tr>
<tr>
<td>SVM</td>
<td>59.50</td>
<td>97.43</td>
<td>65.01</td>
<td>65.68</td>
<td>68.99</td>
<td>69.00</td>
</tr>
<tr>
<td>Decision Tree</td>
<td>59.17</td>
<td>98.54</td>
<td>60.20</td>
<td>64.17</td>
<td>54.70</td>
<td>95.57</td>
</tr>
<tr>
<td>Adaboost</td>
<td>59.19</td>
<td>98.90</td>
<td>63.13</td>
<td>62.06</td>
<td>63.16</td>
<td>98.54</td>
</tr>
<tr>
<td>Logistic Regression(L2)</td>
<td>57.34</td>
<td>97.42</td>
<td>64.26</td>
<td>67.14</td>
<td>66.44</td>
<td>67.50</td>
</tr>
<tr>
<td>Logistic Regression(L1)</td>
<td>55.84</td>
<td>97.79</td>
<td>67.48</td>
<td>67.55</td>
<td>66.81</td>
<td>82.44</td>
</tr>
</tbody>
</table>

Table 1: Classification Accuracy (in %).

Cross Validation: We used a 10-fold cross-validation as suggested in the manifest CSV file. In this manifest file, the dataset are participated into 10 parts denoted by 0 ~ 9, therefore, we have 10 different validation settings if we take one part as the testing data and the rest as the training data. Figure 5 shows the accuracy under 10 different testing dataset in 10-fold cross-validation as specified in the main manifest CSV file. The x-axis is referring to which parts of users are treated as the testing data. For the size of each testing data, please refer to Figure 6. As we can see from Figure 5, an accuracy of 100% is achieved across all testing data settings in 10-fold cross-validation. This is really shocking and far beyond our expectations. Therefore, in order to verify the results furthermore, we did another 5-fold cross-validation. We divide the users into 5 parts by firstly ordering their anonymized name alphabetically and then splitting them into 5 segments with length as equal as possible (i.e. the lengths are [55, 55, 55, 55, 54]). The results of 5-fold cross-validation are showed in Figure 7 and Figure 8. Still the accuracy of 100% is always achieved.

Figure 5: Accuracy in 10-fold cross-validation
Figure 6: Train/test sizes in 10-fold cross-validation

The extremely perfect results make our classification suspectable. Like we mentioned before in section 3.2, option “-ngram” plays a decisive role in achieving such perfect 100% accuracy. We therefore compare “-ngram” to two different options: online SVM learning model “-ksvm” and default learning model (i.e., neither “-ngram” nor “-ksvm” are specified) using 10-fold cross-validation dataset. In each of these three model, parameters “-passes” and learning rate “-l” are pre-tuned to achieve highest accuracy on testing group 0 (The details of tuning these two parameter can be found in the following paragraph Parameter Tuning). For default learning with “-ngram” set to 2, the values of “-passes” and “-l” are 77 and $2^{-4}$; for default learning without “-ngram” option, the values of “-passes” and “-l” are 2 and $2^{-8}$; for “-ksvm” learning method, the values of “-passes” and “-l” are 77 and $2^{-4}$ (as it looks like these two parameters don’t affect the accuracy in this method). Figure 9 shows the comparison among these three learning methods. As we can see, the default learning method coupled with an option “-ngram=2” makes the accuracy perfectly high and stable. The reason “-ksvm” stays at accuracy 0.5 is that the testing results given has a lot of “nan” instead of −1 or 1, even though we did a lot of different tries regarding to the related parameters.
Figure 7: Accuracy in 5-fold cross-validation

Figure 8: Train/test sizes in 5-fold cross-validation

Figure 9: Accuracy comparison between three different learning methods in Vowpal Wabbit.
• Parameter Tuning: Before running cross-validation, we always tune parameters “–passes” and learning rate “-l” on the first testing dataset (i.e., the first subset of the dataset) to achieve the highest accuracy. Then we fix these two parameters when running cross-validation. As there are only two parameters, we did a simple grid parameter search procedure: iterate through all possible values of “–passes” and “-l”. Here, “–passes” ranges from 1 to 100 and “-l” ranges from $2^{-10}$ to $2^{10}$. Figure 10 shows values of accuracy under different “–passes” and “-l” by using default learning model with option “–ngram” set to 2. As we can see, the parameter “–passes” barely effect the accuracy even though there exists few points that a different passes value would lead to a different accuracy, on the other hand, parameter learning rate “-l” only effects the accuracy in the very beginning (roughly $2^{-10}$ to $2^{-7}$), and then the accuracy stays at 100% almost all the time. Therefore, in the 10-fold cross-validation experiments, we set “–passes” to 77 and “-l” to $2^{-4}$.

• Feature Prominence: In this paragraph, we talk about which features contribute to our classification prominently through accuracy by only taking one category of features, i.e. one of word, sentence, topic, readability, non-language. All the experiments regarding feature prominence is done by using the default learning model with “–ngram” set to 2, and “–passes” to 77 and “-l” to $2^{-4}$.

– Contributions of different category of features: We choose the group 0 (the first part) as the testing dataset in 10-fold cross-validation to evaluate the contributions of different categories of features. The accuracy achieved by using each of the category of features is shown in Figure 11. As we can see, “word”, “sentence”, “topic” contribute most. This is probably because “word” might capture what vocabulary schizophrenia user choose to use, “sentence” reflects phrasal complexity/embedding in schizophrenia’s sentence and “topic” demonstrates schizophrenia users’ trouble in focusing or loose association between topics.

5 Discussion and Future Work

In this project report, we present our work in predicting users’ schizophrenia given their Twitter data. Our accomplishments are as follows:

• Feature Extraction: We accomplish our first goal by extracting word-based, sentence-based, topic-based, readability-based and non-language-based features. For the features extracted from every tweet, we also do a simple feature reduction by calculating min, max, mean, median, standard-deviation, variance for each user.
Figure 11: The accuracy by using each category of feature alone.

- Classification: We successfully train predictive models by using two toolkits. With Python scikit-learn package, we use five-fold cross-validation and nearest neighbor classifier, SVM, decision tree, Adaboost classifier and logistic regression classifier respectively to train our models. We also use an additional 10-fold cross-validation and Vowpal Wabbit to predict schizophrenia users.

Generally, the results either from Python scikit-learn package or Vowpal Wabbit are surprisingly better than what we expected. This might be caused by the relative smallness of the data but does not exclude the probability that the features we extract might indeed be capable of reflecting the characteristics of a schizophrenia user. Our experiments furthermore show that among the various features, language related features such as the ones “word”, “sentence” and “topic” categories contribute most to distinguishing schizophrenia users from control users.

For future research, there are a couple of directions to improve our model. One is we can try Recurrent Neural Networks to analyze the tweets data. Another is to explore more discriminative features to help predicting schizophrenia users, such as how schizophrenia level changes from time to time or before and after clinical treatment.

References


