Automatic Categorization of Whispers

Zack Winoker*

1 Note To The Reader

A "Whisper" is a piece of short text. In this project, I was given a data set of roughly 14,000 Whispers. Each was assigned to 1 of 17 different categories. The objective of the project was to create a classifier for Whispers.

2 Introduction

Given the task of automatically categorizing short text Whispers, I outlined the following course of action:

1. Conduct exploratory analysis of the data in order to better understand the problem.
2. Attempt to establish a new baseline using standard classifiers.
3. Review the relevant scientific literature to better understand state of the art approaches to this problem.
4. Design and implement attempted improvements on the state of the art.

My best model was a convolutional neural network that achieved a macro-F1 score of 0.60 and an accuracy of 0.66. This is at least on par with the state-of-the-art results available in the relevant literature.

3 The Dataset

The first thing I did was run the given baseline classifier. This resulted in a macro-F1 score of about 0.32. To improve this, I identified key aspects of the data and examined the baseline classifier's results in order to see what mistakes it was making.

The Whispers had an average word length of 11 words and were assigned to one of 17 different categories. I noted that these had significant semantic overlap. For example, Whispers in the 'family' and 'relationships' category will be more similar than those in 'tatoos' and 'faith'. Examination of the baseline's confusion matrix illustrates that this was one of its major sources of error (see Table 1). In particular, the catch-all categories of 'misc' and 'personal' were the source of many incorrect classifications.

-	**Personal**	**Relationships**
Classified as Personal	679	31
Classified as Relationships	133	154

Table 1: The Personal and Relationships confusion matrix for the given baseline model. Note that almost half of the relationship data were incorrectly classified as personal data.

*Email: zachary.b.winoker@gmail.com

The shortness of the text prevents each sentence from sampling enough words for the TF-IDF matrix alone to be a sufficient feature set. Furthermore, the lack of distinct categories must be addressed in some way.

I also discovered that Whispers are plagued with spelling errors and unconventional abbreviations, so any data cleaning and preparation should address this.

It was important to know if more data would improve this classifier. To see if this was the case, I randomly sampled some percentage of the training data, trained an SVM with a linear kernel (which had better results out-of-the-box than the baseline model), and ran it on the given test data. This was done 5 times on each data subset and the average macro-F1 score was taken. The results are plotted in figure 1.

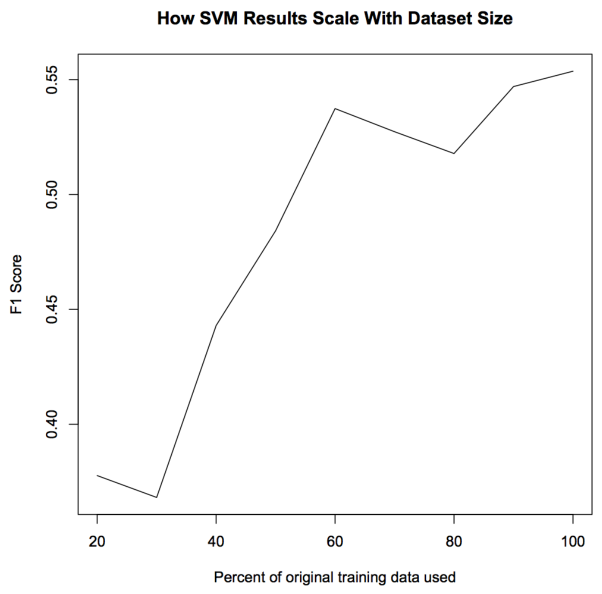

Figure 1: The benefit to increasing the size of the training set appears to begin leveling off at around 60 percent of the original size, or around 8500 training samples.

I saw that there was significant improvement until about 60 percent of the given data set is used. After that, the rate of increase drops sharply. I concluded that the amount of data available was not the chief obstacle to good classification for this dataset.

4 Literature Review

Given my unfamiliarity with short text classification, I decided to review the scientific literature on the subject. I found that the two central issues I identified, shortness and category overlap, significantly reduce the effectiveness of traditional text classifiers. More modern methods include using convolutional neural networks (CNNs) trained on matrices of word vectors. These achieve a peak performance of around 60 percent accuracy for datasets similar to the given Whispers [1,3]. It's notable that most literature does not focus on datasets with short documents and many categories, likely because the performance of published models is so poor for them.

A common approach for alleviating the shortness problem is to somehow increase the size of the text by adding relevant words. This is often done via some sort of internet query. For example, the word "Ireland" may be replaced with the first 4 paragraphs of the Wikipedia entry on Ireland. However, performance of this method on short texts with many categories is still relatively poor. Instead, I attempted to expand the text with words whose word vectors were similar to the word being expanded.

I found multiple approaches to fixing the semantic overlap problem, but again none were particularly effective for this type of dataset. I attempted two solutions to this problem, given below.

Note also that the macro-F1 scores for the most accurate CNN models I could find were not reported, so I will compare my results against their accuracies and the F1 score against the new linear SVM baseline I established.

5 Methods

To improve the baseline model, I first took the given features (TF-IDF matrix) and plugged them into a host of available classifiers in Sci-kit learn. I found that using the out-of-the-box linear SVM and a porter and stemmer from the NLTK package produced significantly improved results. I also employed a more modern approach using the convolutional neural network designed by Kim [1]. Using this method and tuning it according to [3], I was able to establish a new baseline with scores F1: 0.60, Accuracy: 0.66. Further attempts were judged against the CNN baseline.

Having established what appears to be a state-of-the-art baseline, I attempted to implement solutions for the shortness and overlap problems. For the shortness problem, I elected to try expanding the text using a word-vector based approach. For this and all other word-vectors, I used the word2vec embeddings provided by Google [4]. These embeddings were accessed and used via the gensim package [5]. Expansion for the training set proceeded as follows: For a given word w, the vector embedding v_w is found. Then the 100 word vectors closest to v_w are found, and their words are appended to the sentence after w. If v_w is not available, expansion is skipped, but the original word is kept. I then used the baseline features (bag of words TF-IDF) on this augmented data with a linear SVM classifier.

I tried two approaches to fixing the problem of overlapping categories. In the first approach, I attempted to find distinct subcategories within the catch-all categories of 'misc' and 'personal'. To do this, I took all of the training data labeled as 'misc', computed the average word vector for each sentence, and then applied k-means clustering to the resulting vectors. I then labeled each sentence according to its cluster, e.g. 'misc-4'. I did the same for the 'personal' category. I then used these finer-grain categories in place of the catch-all ones and applied the linear SVM with bag-of-words TF-IDF.

The second approach I employed for overlapping categories was to identify a hierarchy of categories. To do this, I first computed the word vectors for each category name. Certain category names were not available in the pre-computed word2vec vectors. These were 'lgbtq' and 'qna', which I replaced with 'sexuality' and 'questions'. I then repeatedly applied k-means clustering to the categories' word vectors with 2 clusters at each iteration. In doing so, I hoped to identify the best way to split up categories so that binary classifiers could be applied. Each binary classifier classified a sentence into one of two composite categories. Each of these categories had another binary classifier that the sentence was fed through. This was repeated until a non-composite category was assigned. For example, the top-level classifier categorized input as either 'animals-sexuality-relationships' or 'faith-family-fashion-food-meetup-military-misc-personal-popculture-questions-school-sports-tatoos-work'. Then if the input is categorized as 'animals-sexuality-relationships', a classifier was applied that outputted either 'animals' or 'sexuality-relationships'. If the result was 'animals', then that would be returned, as it was a non-composite category. Otherwise, the process continued with a binary classifier for 'sexuality' and 'relationships'. Training data had to be pre-processed for each of these classifiers. The full hierarchy is shown in figure 2. This method was an attempt to take advantage of the increased accuracy of binary classifiers vs non-binary ones and to use the fact that the distance between word vectors is related to their semantic similarity. I used the same linear SVM and features as before, and experimented with using a Multinomial Naive Bayes classifier for composite categories with few samples.

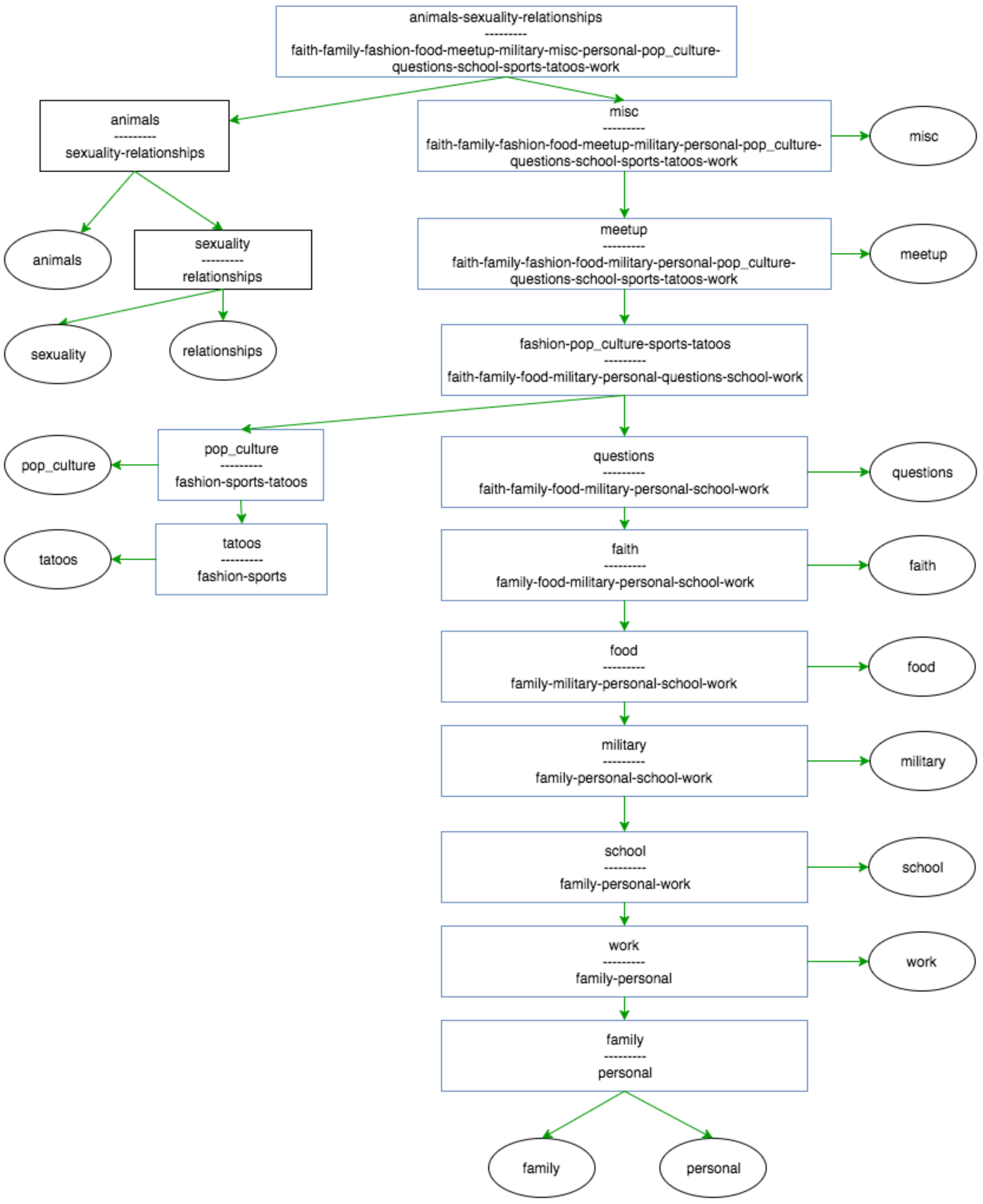
animals-sexuality-relationships
----
faith-family-fashion-food-meetup-military-misc-personal-pop_culture-questions-school-sports-tatoos-work
animals
----
sexuality-relationships
misc
----
faith-family-fashion-food-meetup-military-personal-pop_culture-questions-school-sports-tatoos-work
misc
animals
sexuality
----
relationships
meetup
----
faith-family-fashion-food-military-personal-pop_culture-questions-school-sports-tatoos-work
meetup
sexuality
relationships
fashion-pop_culture-sports-tatoos
----
faith-family-food-military-personal-questions-school-work
pop_culture
pop_culture
----
fashion-sports-tatoos
questions
----
faith-family-food-military-personal-school-work
questions
tatoos
tatoos
----
fashion-sports
faith
----
family-food-military-personal-school-work
faith
food
----
family-military-personal-school-work
food
military
----
family-personal-school-work
military
school
----
family-personal-work
school
work
----
family-personal
work
family
----
personal
family
personal

Figure 2: A tree representing the hierarchy of categories found using k-means clustering on word2vec word vectors of the categories. The rectangular boxes represent binary classifiers and the oval leaves represent final classifications.

6 Code

For the approach with convolutional neural networks, I used the Theano code written by Kim [2]. Performance of this system was measured using tools from Sci-kit learn. All other code is original and written in Python using Sci-kit learn. For more details, please see the provided README.

7 Results

My results are summarized in Table 2. The row 'CNN' indicates the performance of the CNN used without tuning any of its hyperparameters. Ie we use the model as it is provided. 'tree-linsvm' is the category-hierarchy model with all SVM classifiers. Note that including any MNB classifiers in the tree decreased performance.

Method	**F1**	**Accuracy**
Baseline	0.32	0.57
LinSVM	0.55	0.60
w2v expansion	0.50	0.57
w2v subcategory clustering	0.54	0.59
tree-linsvm	0.47	0.57
CNN	0.57	0.65
Tuned CNN	0.60	0.66

Table 2: F1 and accuracy scores for all of our methods.

All methods we tried performed better than the given baseline model. The success of the linear SVM in this regard indicates that certain regions of the data are easily linearly separable. These are likely just the categories with very little semantic overlap.

The expansion process may have suffered from two things in particular. First, many Whispers have 1-grams that are not words in the pre-trained word2vec model. This is because many online communications make extensive use of abbreviations, misspellings, and other irregular English. Therefore, a more thorough mapping of abbreviations, misspellings, etc. would help regularize the data. This would allow more words to be expanded, thereby potentially increasing the quality of the word-sample the comprises each document. Additionally, due to available computational power, we were limited to expanding each word by 100 additional words. If this number could be increased, the amount of specific, relevant content per Whisper could be increased, which may improve classification performance.

-	**Personal**	**Relationships**
Classified as Personal	593	72
Classified as Relationships	66	232

Table 3: The Personal and Relationships confusion matrix for the tuned CNN model. Note that we have significantly reduced the number of false positives for the Personal category and significantly increased the number of true positives for the Relationships category. This indicates that this model is more sensitive to the differences in overlapping categories.

The tree-linsvm model was not as successful as I hoped, but more variations on the hierarchy should be tested. Clustering may not have been the best method for establishing the exact tree to be used. Furthermore, the choice of 'sexuality' and 'questions' to replace 'lgbtq' and 'qna' may have affected the results. Other semantically similar replacements should be tested.

The sub-category model was slightly worse than the regular LinSVM model. This is likely due to the fact that I was unable to generate clusters with a sufficient number of samples once more than 3 centers were used. As such, I couldn't create too many subcategories for either 'misc' or 'personal', which generally diminishes this method's efficacy.

8 Conclusion and Future Work

In summary, the tuned CNN outperformed all of the other methods and produced an F1 score of 0.60. This is consistent with the literature, in that convolutional neural networks appear to be the most promising model for short-text classification, especially with many possible categories.

Given more time, I would try the following: First, try text expansion and/or w2v subcategory clustering with the same CNN architecture. Second, tune the hyperparameters of the CNN further. Third, modify the word expansion such that words in the training set are chosen to be similar to a sentence's given category. Fourth, explore the space of category hierarchies by either some kind of random search or different clustering methods. Note that this will require some refactoring of the TreeLinSVM code. Fifth, experiment with variable width filters for the CNN architecture. This may produce features based on subspaces of the word-vector space in which categories are more distinct. However, the training time for the CNN is very time-expensive. Without access to a cluster, these future experiments are less feasible.

9 Citations

1. Kim, Yoon. Convolutional Neural Networks for Sentence Classification. `http://arxiv.org/pdf/1408.5882v2.pdf`
2. `https://github.com/yoonkim/CNN_sentence`
3. Zhang, Ye and Wallace, Byron. A Sensitivity Analysis of (and Practitioners Guide to) Convolutional Neural Networks for Sentence Classification. `http://arxiv.org/pdf/1510.03820v4.pdf`
4. `https://code.google.com/archive/p/word2vec/`
5. `https://radimrehurek.com/gensim/models/word2vec.html`