Automated Categorization for Streamed Tweets

Nathan Vance

Goal

Before streamed data such as tweets can be efficiently processed, they often must first be categorized. Traditionally, this has been done by constraining the tweets crawled to a specific geolocation, time period, or set of keywords. This approach to obtaining a focused dataset is desirable because it is possible to stream the data in real time, and it often succeeds in achieving the required focus. Unfortunately, many tweets that may have been applicable but that didn't meet the search criteria are missed by this approach. This results in systematically omitting data from the dataset, which can inhibit accuracy or effectiveness when it comes to applications such as truth discovery.

To overcome this obstacle, we develop a Categorizing Automaton for Tweets that are Streamed (CATS), which is available at [1]. CATS is a text classifier that uses clustering to assign categories to a stream of tweets. It handles data one tweet at a time, processing and categorizing each tweet before proceeding to the next. This makes it ideal for use as a stage in a realtime data processing pipeline.

The input for CATS is the text body of each tweet in a stream of tweets (fed in one at a time), and the output for each tweet is a human readable word or phrase representing the category that CATS assigned to the tweet. For example, given the tweet (pseudorandomly selected from the Inauguration dataset):

> Defeated US oligarch Hillary Clinton displays customary warmth and gaiety at installation of Donald Trump as Chief Warlord

CATS may, correctly, select the category to be "trump". It may also select the category to be "clinton", which would be incorrect given the context of President Trump's inauguration.

Method

CATS operates by selecting a set of potential topics that each tweet addresses, then clusters the tweets based on the topics. Multiple strategies for both topic extraction and clustering were investigated. Topic extraction techniques include:

- **NLTK Nouns**: We tag each word in the sentence using NLTK in Python. The nouns (NN and NNP) are the potential topics. This method is used as the baseline.
- **Perceptron Named Entities**: We identify named entities using a perceptron model that had been trained on a twitter dataset. These named entities are the potential topics. If no named entities are identified for an input tweet, then the algorithm falls back to NLTK Nouns.
- **Everything**: We split the tweet on whitespace. The resulting words are the potential topics.

Clustering techniques include:

- **Naive Clustering**: We maintain a score for each topic. Every time a tweet is processed, we increment the score corresponding to each topic in that tweet by ten, and decrement every nonzero score for every topic by one. The topic from the tweet that has the highest score is the category assigned to the tweet. As far as we are aware, this project is the first time that this particular clustering method has been used, therefore we named it Naive Clustering, after its author's understanding of clustering techniques. This method was used as the baseline.
- **K-Means Clustering**: For each tweet processed, we cluster over the topics in that tweet and the previous 100 tweets that the algorithm has seen. Since the algorithm does not have access to the true number of clusters in the dataset, it varies the number of clusters that it forms over the interval:

$[2, \log_2(\text{numberOfTweets}))$

attempting a categorization with that many clusters. A score for each attempt is computed as follows:

score = sizeOfSmallestCluster / (numberOfTweets / numberOfClusters)

A higher score (closer to 1) corresponds to a more even distribution, which is indicative of a correct clustering. A skewed distribution (score is closer to 0) intuitively means that either the smaller clusters should be merged into the larger ones, or the larger clusters should be broken up. The clustering that achieves the highest score is returned, and the most frequent topic from the cluster that the processed tweet ended up in is returned as the plain-text category for that tweet.

Schemes under consideration included Open Set Recognition (OSR) [2], which is an image recognition algorithm that can detect anomalies. Such a tool would be useful in detecting when a new cluster of topics emerges in a tweet streaming application. While OSR is promising for use as a component of a clustering algorithm, it requires a consistent length vector representation of the data being categorized, which in our case is a list of topics. There are methods such as doc2vec [3] that condense a tweet down to a fixed length vector, however, we obtained poor results with this route and ultimately abandoned it.

Instead of OSR, we elected to use K-Means clustering with re-clustering the tweets every time a new one is added. This solves the issue of variable length input by allowing us to treat the topics as a bag of words, where each unique topic has its own dimension in vector space. If a tweet contains a topic that has not yet been seen by CATS, then the topic is simply added to the language and all vectors are grown by one dimension.

Experiments

We obtained the datasets used from [4] (must be on Notre Dame campus to access). In particular, we used the datasets on McAfee, Ebola, Inauguration, and Charlottesville. These sets were chosen in part because certain parings provide us with disparate data (topics in the McAfee set are quite different than the Inauguration set), while other pairings have significant overlap (McAfee and Ebola both use the word "virus", and Inauguration and Charlottesville both frequently mention President Trump).

The experimental setup was to randomly select a fixed number of tweets from each of the datasets, shuffle the tweets, and have CATS detect the categories for each tweet. CATS was evaluated based both on accuracy (the percentage of tweets correctly categorized, i.e., the category is a topic of the origin dataset) and precision (the number of detected categories divided by actual categories). Precision was defined this way because the point of CATS is to simplify the streamed data by dividing it into categories, so generating many one-tweet categories defeats the purpose.

This experiment was run for 10 trials of each combination of:

- Datasets: McAfee and Inauguration, or McAfee, Inauguration, Ebola, and Charlottesville
- Cluster Size: 100 or 1000 tweets randomly selected from each dataset
- Topic Extraction Method: NLTK Nouns (Base), Perceptron (Percep), or Everything (Every)
- Clustering Technique: Naive Clustering (Base), or K-Means Clustering (Cluster)

The results are below. Note that, in each case, higher values for accuracy and precision indicate a better performing algorithm.

As can be seen in Figure 1, K-Means Clustering achieves marginally better accuracy than Naive Clustering in every category except for the Everything topic extraction, where the results are ambiguous because of high levels of experimental error. Note that a random category assignment would achieve 50% accuracy only if it had prior knowledge of both the number and the identities of the categories (i.e., 100% precision), which CATS does not have access to. Therefore, even though some combinations of algorithms achieve accuracies at or below 50%, this should not be interpreted as being no better than random.

For the precision metric in Figure 1, K-Means clustering is the clear winner, being over 5 times as precise as in Naive Clustering; whereas Naive Clustering generated 66 clusters on average, K-Means clustering resulted in an average of only 12 clusters. Since the unnecessary proliferation of false-alarm categories has a negative impact on performance and/or effectiveness in later stages of the pipeline, using K-Means clustering is preferential.

Figure 1: Results for McAfee and Inauguration

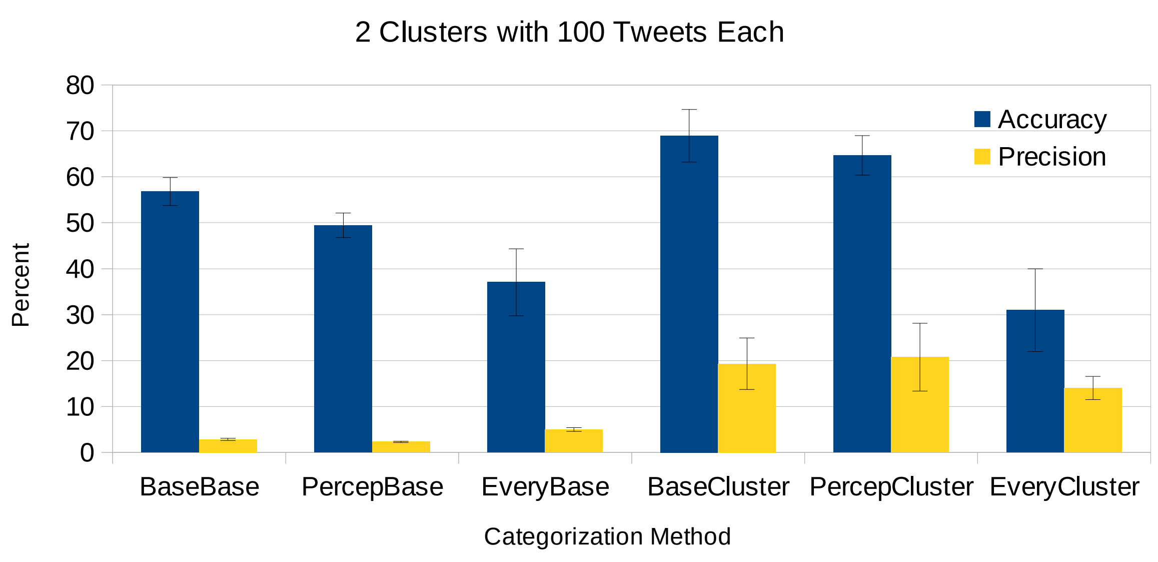
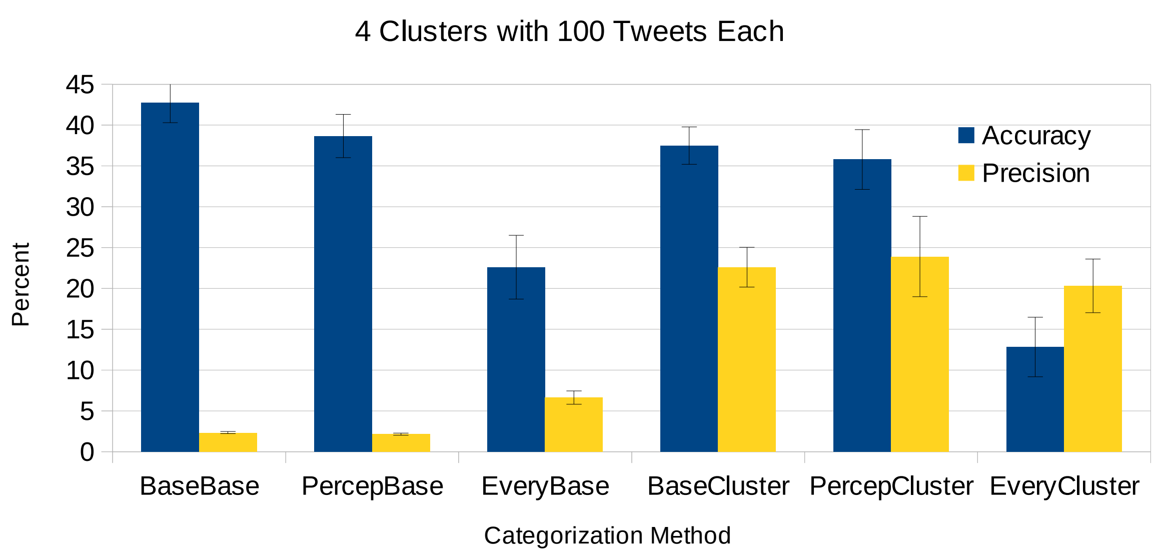


Figure 2: Results for McAfee, Inaugeration, Ebola, and Charlottesville



When the number of clusters was increased to four (Figure 2), we found that the trend for precision persisted, with K-Means continuing to achieve better results than the Naive Clustering algorithm. However, with four clusters the Naive Clustering algorithm achieves as accurate or more accurate results. This was caused by Naive Clustering being less specific in teasing apart data, so it lumped Inauguration and Charlottesville both into the category "trump", which the scoring method counted as correct. Meanwhile, the K-Means algorithm correctly identified that they were different, but made more errors in the process.

After running a small experiment with 100 tweets in each cluster, we investigated how the algorithms scale to sets with 1000 tweets each. Figures 3 and 4 show similar trends to what we found when datasets only had 100 tweets each, where K-Means Clustering achieves comparatively better accuracy with two clusters, while Naive Clustering is better with four. K-Means Clustering always achieves better precision than Naive Clustering. We also found statistically significant evidence to suggest that NLTK Nouns achieves the best accuracy, followed by Perceptron, then Everything topic extraction.

Figure 3: Results for McAfee and Inauguration

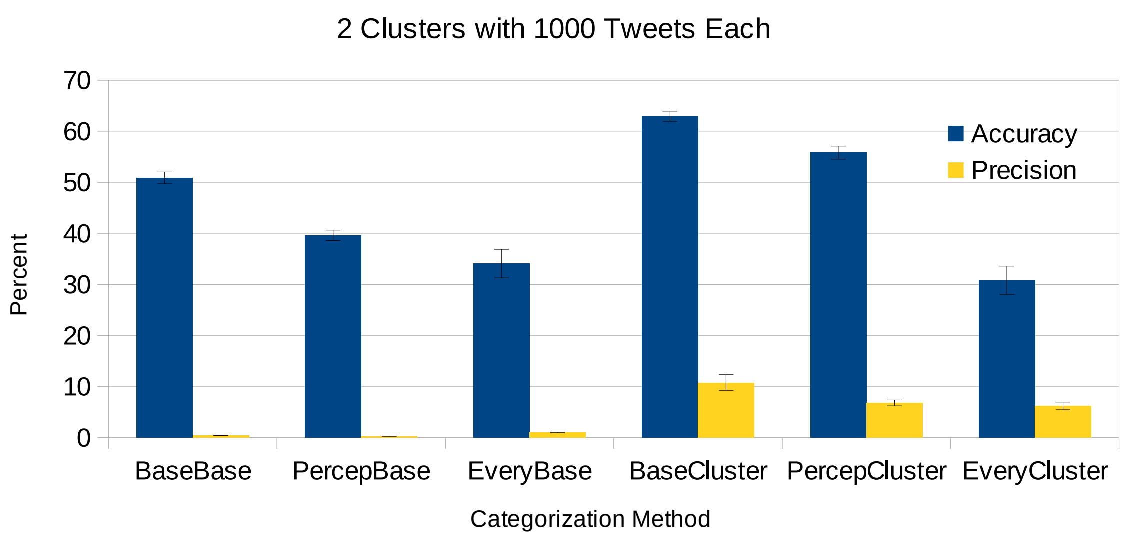
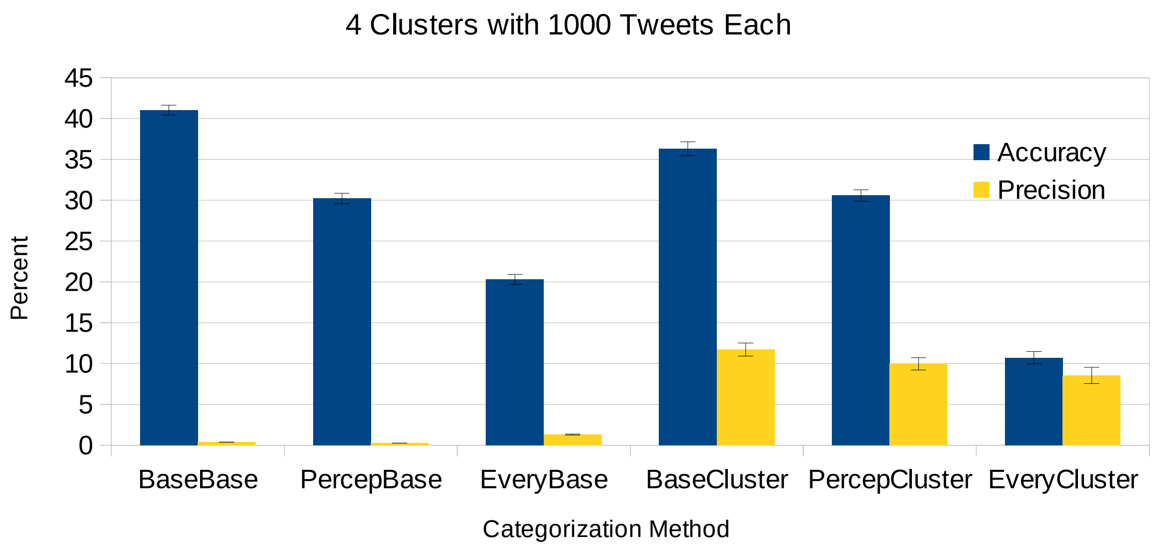


Figure 4: Results for McAfee, Inaugeration, Ebola, and Charlottesville



The increase in precision and decrease in accuracy for the Everything topic extraction when paired with Naive Clustering is caused by the inclusion of stop words in the set of topics. While the clustering algorithm is still capable of identifying the correct categories with a reasonable degree of accuracy, many of the tweets in the long tail and some of the tweets that would have otherwise been correctly categorized are placed in stop word categories such as "the", "a", and "and".

The K-Means Clustering ran much more slowly than Naive Clustering. The running time per tweet once K-Means gets going stays constant because the window of tweets the algorithm remembers is kept constant, so using K-Means Clustering is still theoretically viable for use in real time systems. However, we do not recommend its use in the mobile or embedded space yet. Other than running times, we did not find any indication that changing the size of the datasets changes the relative accuracy or precision of the compared methods.

In summary, NLTK Nouns for topic extraction achieves the highest accuracy. This is likely caused by NLTK using an advanced theory for tagging parts of speech, resulting in more realistic inputs to the clustering algorithm. K-Means Clustering achieves the highest precision across the board, and it is the most accurate when clustering two datasets, while Naive Clustering is more accurate with four datasets. This is likely caused by Naive Clustering being prone to merging similar categories while leaving a long tail of functionally uncategorized tweets.

Future Work

We designed CATS to be able to adapt to a change in categories in real time (i.e., topic A is tweeted about primarily in the first portion of the dataset but not in the second, where it may be replaced by topic B). The accuracy and precision of this feature across different schemes have yet to be evaluated. Additionally, the running time performance of the schemes should be quantified and, if possible, improved.

An important part of the evaluation of results that we did not adequately address is the merging of clusters that have overlapping topics, such as the Inauguration and Charlottesville sets both having the common topic "trump". Our evaluation scheme did not penalize a lack of differentiation between these, but perhaps should have. By calculating Jaccard distances, we could properly evaluate based on this metric.

Another direction that this research could go in is leveraging OSR, perhaps through the use of neural networks or a more sophisticated application of doc2vec.

Finally, we are interested in designing clustering algorithms can separate out the datasets even if they were interspersed among random twitter "noise". In real world examples, even during a catastrophe some people aren't interested in reporting on the emergency situation and instead, for example, tweet about their cats. CATS should be able to identify and isolate the relevant categories, even from among a substantial amount of noise. One way that this may possible is through the use of subcategories. For example, the top layer of categories may include tweets about cats, dogs, and emergency situations. Someone interested in emergency relief would then use CATS to explore the emergency situations category.

Conclusions

In conclusion, we found that K-Means Clustering performed substantially better than the baseline, Naive Clustering, in precision. For accuracy the results were largely inconclusive, with K-Means Clustering achieving better results with fewer source datasets and Naive Clustering excelling with more source datasets.

The best method for topic extraction was NLTK Nouns. This is a result of that tool being well tuned for accurately tagging parts of speech, so it accurately isolated the nouns from the input tweets.

References

[1] CATS:	https://github.com/NathanRVance/cats
[2] OSR:	https://www.wjscheirer.com/projects/openset-recognition/
[3] doc2vec:	https://radimrehurek.com/gensim/models/doc2vec.html
[4] Datasets:	http://apollo2.cse.nd.edu/now/index.html