# Tweet Classification: Filtering of Twitter for Company-relevant Tweets

**Curtis Ullerich, Daniel Stiner, Brandon Maxwell**
Department of Computer Science and Engineering
Iowa State University Ames, Iowa, USA
{curtisu,stiner,bmaxwell}@iastate.edu

### Abstract

Twitter is a popular source for data mining due to its massive scale and inclusion of current trends. It is also a veritable goldmine of useful data for businesses seeking to discover public opinion about themselves or their products. Using machine learning techniques, we analyze the accuracy of classifying whether tweets directly mention a particular company or merely contain keywords related to the company. Tweets present interesting classification challenges due to their irregular formatting and relatively small number of features. We aim to survey a variety of text processing steps combined with a Naive Bayes classifier, focusing on the effects of different tokenizations. Our dataset consists of approximately 2,000 tweets of which 64% directly mention the company Apple, Inc or one of its products. The other 36% do not, but still contain keywords related to the company. Preliminary results have shown increases of greater than 10% in accuracy by using Twitter-aware tokenization during preprocessing.

## Introduction

Tweets are used as a source of data mining for applications ranging from predicting future stock price changes based on user sentiment (Ruiz et al.) and identifying characteristics of Twitter users (Pennacchiotti and Popescu) to tracking political protests and uncovering scams (Brendan O'Connor and Ahn). In many cases, the problem of determining which tweets are relevant for a particular application is prerequisite to obtaining meaningful results. Use of hashtags and username references on Twitter through syntax conventions (prepending with # and @, respectively) can be used to select a subset of the tweets about a particular topic.

| |
|---|
| **@Apple** please make a squid **#emoji** so I can put one next to **@squidneyy22**'s name |
| Maybe **#Apple** will release **#iTunes11** tomorrow. Maybe? |

Table 1: Hashtags and username references (note that Apple Inc. does not own the handle @Apple, though it is commonly used this way)

In some domains, filtering tweets based solely on keywords may be appropriate, but disambiguation remains an interest natural language processing problem in many cases, particularly when the topic in question is a commonplace (and thus ambiguous) word, such as 'apple,' 'blackberry,' 'android,' or 'adobe' (Yerva et al.).

Additionally, tweets contain a high rate of errors in grammar, spelling, and punctuation. These features are present alongside myriad augmentations of the English language due to internet trends and culture, as well as the desire of packing more information into Twitter's 140-character message limit (Laboreiro et al.). Preprocessing is necessary to avoid discarding or misinterpreting these features as noise or part of other tokens during tokenization.

| |
|---|
| **Nvm** i'll buy you an ipad **^-^** |
| #Apple fires 'maps' project manager **looooool** big news story here in San Francisco! Go figure. **:-)** |
| Think I might have a **gf**! **O_o** oh lord**..** Lets see how this goes. |

Table 2: Examples of emoticons and common abbreviations found on Twitter

We present a system to disambiguate tweets about the company Apple using a Naive Bayes model, increasing its baseline accuracy by over 10% through use of specialized, Twitter-aware tokenization.

## Related Work

DUALIST is a system that extends Mallet, our machine learning API of choice, by user-interactive feedback during training (Settles). Included in his implementation was a preprocessing step that performed some simple Twitter-specific feature extraction and tokenization. Part of the Sentiment Analysis Symposium focuses on Twitter-aware tokenization of tweets for sentiment analysis, demonstrating that it provides consistent improvement over whitespace- and Treebank-based tokenization Potts (2011). A good deal of research has been done into sentiment classification of tweets, which requires very similar text processing and tokenization Pak and Paroubek (2010b). Laboreiro et al. focus on the problem of tokenizing, using support vector machines (SVM) to improve upon the accuracy of rule-based tokenization, succeeded in improving from 85% to 96% token accu-

racy (Laboreiro et al.). Yerva et al. construct classifiers to disambiguate tweets containing company-related keywords, primarily using profiles containing company background knowledge (Yerva et al.). Sriram et al. presented a system that uses features extracted from the tweet author's profile and the tweet text to classify tweets into the categories of News, Events, Opinions, Deals, and Private Messages (Sriram et al.).

# Method

## Corpus creation

By accessing the Twitter 'Firehose' stream through their public API, we harvested 100,000 tweets using broad keyword filters with the goal of collecting a superset of relevant tweets. This provides us a smaller set of features in the overall corpus, with a much higher prior confidence that each individual tweet is of interest. We collected the keywords we used manually from information on the company's website and the Wikipedia page about Apple:

| | | |
|---|---|---|
| apple | itunes | ipad |
| ipod | mac | ios |
| iphone | AAPL | cupertino |
| safari | ilife | iwork |
| garageband | ibook | powerbook |
| itouch | app store | macworld |
| facetime | icloud | mobileme |
| siri | imovie | iphoto |
| quicktime | logic pro | think different |

This set of keywords clearly allows a large number of false positives:

| |
|---|
| nice fish and chip.. good pie apple :-) http://t.co/KVDN5r5l |
| 20 chicken nuggets, chips, big mac, and a vanilla milkshake LIFE IS GOOD |
| 1 more day whoop plus safari ride yay |
| Ashley staples an apple. 'AHHHH!!!! Apple juice in my eye!' |
| I ain't seen uncle Mac in a while |

Table 3: Examples of false positive tweets

We define a tweet to be 'about' Apple if the tweet mentions the company (via direct reference or stock symbol), one of its products, or a service it offers. We make the assumption that spam will be filtered from the testing set. This could be done via an existing Naive Bayes classifier, but we simply did not include instances of spam in our corpus during the manual labeling process. We collected our tweets over the course of twelve hours. On Twitter, a single message posted is often reposted, or 're-tweeted,' many more times in a very short timespan. Because of the small timespan of our data collection, a large number of retweets (effectively duplicates) appeared in the dataset. We filtered these duplicates out of the dataset as well to prevent over-fitting in our model. After filtering, our original set of 100,000 tweets was reduced to 50,000. A representative sample of non-spam, non-duplicate tweets has a composition of 64% topical instances and 36% false positives.

| |
|---|
| I've collected 10,650 gold coins! http://t.co/H0V0O6yP #ipad, #ipadgames, #gameinsight |

Table 4: A computer-posted spam tweet

| |
|---|
| RT @PHILerNotebook: Trying to fix my Mac. This gray loading bar takes 10 minutes to start up. Grr. Shall take it to an _Apple_ _store_ soon! |

Table 5: A retweet, signified by 'RT'

We hand labeled 2000 tweets for use in training and testing, including only English-language tweets in our training set. Data sets and scripts used for processing text data are available on the project website.
(http://tweets.curtisullerich.com/)

## Preprocessing Pipeline

We do all machine learning with the Mallet library developed by the University of Massachusetts McCallum (2002). Mallet includes abstractions for a data processing pipeline, during which we use both existing and custom processing 'Pipes' to transform the data, beginning in our case with raw tweet text and ending in Mallet's internal FeatureVector format. We built and tested several Pipes to analyze the effect of different filters and processes on the accuracy of classification. Not every Pipe proved to increase accuracy–some decreased it, as presented in the results. We present the descriptions of and motivations for these Pipes here.

**Feature-space Enrichment**
*Stopword removal*: We used Mallet's TokenSequenceRemoveStopwords Pipe to remove 500 of the most common words in the English language from each tweet. We remove these words because of the high likelihood of many tweets containing these words, without contributing significant discriminatory features to the tweets.

*Bigrams*: Using n-grams is a common method of increasing the number of features present in short text. Often, this is enough to increase accuracy significantly. As discussed in the tokenization section below, we used selective bigramming, which yielded increases in accuracy in almost all test cases.

*HTML corrections*: When accessing tweets through Twitter's public API, the native JSON format includes HTML-encoded entities that can add confusing features to the tweets if not un-escaped. For instance, angle brackets are replaced with < and > (less-than and greater-than), causing emoticons, for instance, to no longer be parseable. The emoticon '>.<', for instance, would be '>.<'. Necessarily, this Pipe un-escapes all such entities, making

them easier to parse.

*URL replacement*: In our dataset, 60% of the 100,000 tweets we collected contained a link. The vast majority of these are unique, and shortened using a URL shortener, often Twitter's own t.co. Links, then, lose any features that may have been parsed from the URL, which often contains a domain name or title that may be useful in discriminating especially ambiguous links. Consider, for instance, the tweet 'hmmm http://t.co/39D8I3qM' which leaves a parser completely unaware of the true content of the tweet. Following this URL leads to 'http://www.redmondpie.com/iphone-5-vs-iphone-4s-should-you-upgrade/', which is clearly related to Apple. This pipe replaces the link itself in the tweet with the title of the destination web page: 'iPhone 5 vs iPhone 4S: Should You Upgrade? — Redmond Pie.' In cases where the title is empty, we instead replace the shortened link with the resolved URL, which is 'http://www.redmondpie.com/iphone-5-vs-iphone-4s-should-you-upgrade' in this case. In many cases like this, there is no other way to distinguish the content of a tweet. From our dataset of 100,000 tweets, 102 tweets contained *only* a link, while 608 tweets contained only a link and fewer than 20 other characters. While this is a fairly small portion of the overall dataset for training purposes, it can be crucial in making decision on novel test instances. Pak and Paroubek also noted the need to handle URLs in tweets, however, rather than adding any new text, they chose to remove the link entirely (Pak and Paroubek).

*Spell checking*: Twitter's user-generated content contains a large volume of misspelled words (Laboreiro et al.). Using a spell-checking library, we wrote a Pipe to replace unrecognized words (misspelled words) with their corrections if the confidence for correction was above a certain threshold. Our intention was to reduce the noise in the feature set by eliminating common misspellings.

**Disambiguation**

*Stemming*: Stemming is the process of reducing words to their 'stem,' in order to group different forms of the same word into a single feature. While this stem does not necessarily have to be the root of the word, it should satisfy the property that related words have the same stem. For instance, the stem of the related words 'argue,' 'argued,' and 'arguing' all have the stem of 'argu,' even though argu is not the true linguistic stem, nor a real word. We used the standard Porter stemmer to implement this (Porter). In some cases, stemming proved to increase our accuracy.

**Tokenization**

Tokens are atomic symbols in text that we use as features in training a classifier (Laboreiro et al.). Twitter data, and user-generated web content in general, provides unique challenges to an automatic tokenization system due to the high level of noise in the feature set. Tweets are limited to 140 characters, which limits the total number of features possible. Some of the many natural language processing dilemmas present in tweets include excessive and irregular punctuation, frequent misspellings, emoticons, URLs, and hashtags, as discussed in the introduction.

| |
|---|
| Cool ranch , 4 berry sundae , apple juice **#yessuh #latenight #snack** http://t.co/ttc3vujp |
| I just poured apple juice on my cereal**..** Gah. **#SoTired** |
| I hate Siri**.. -**___**-** |

Table 6: Examples of difficult tokenizations

Searching for emoticons in the tweet 'Now on my iphone ?? http://t.co/Rqr1RL2R' will extract ':/' as an emoticon, effectively breaking up the URL token. Additionally, users often include emoticons or links after text with no delimiting whitespace, such as 'I listen to my iPod every day and it's just broke **omg:(!!**.' We approach this problem using a layered token extraction system. Our goal is that any token pattern which may contain false positives of other tokens (e.g. emoticons inside URLS) is extracted before any potential interior false positive tokens. Extracting tokens using our layered approach eliminates many problems introduced by Twitter's unique feature space. Note that compiling an exhaustive list of all emoticon combinations is difficult, if not impossible, because the landscape of emoticons is very diverse and changes over time (Laboreiro et al.). Instead, we focus on capturing a 'large enough' set of emoticons by extracting a large number of the most common ones.

After tokenizing these special features in order (URLs, emoticons, usernames, and hashtags) we are left with a fragmented tweet containing myriad punctuation, capitalization, and creative spellings. As done by Potts (2011) we normalize the length of all letter sequences greater than two, as in English these are invariably due to user-added emphasis, and condensing the larger set of tokens focuses the feature set. In the following tweet, for instance, 'loooool' would be normalized to 'loool'.

| |
|---|
| #news can't wait for my signed #OoRITE2OutNow it's emotional here right now **loooool** #OoRITE2OutNow https://t.co/r4n4cn6C |

Table 7: Example of normalizing repeated characters

Note that such repetitions are often valid in URLs and some emoticons. Normalizing them prior to this step would make URLs invalid if using our link replacement Pipe.

As presented in our results, because of irregular capitalization, two tokens likely to be semantically identical with respect to company classification may be represented in multiple cases. {Nooo, NOOO, nooo, NoOo} is a prime example. To alleviate this, we attempt three different approaches. In the first, we simply lowercase all remaining tokens (note that many URLs become invalid after modifying case). In the second, we lowercase any text that is not in all caps. This serves to retain acronyms and any text specifically placed in all caps, while combining

words capitalized because of sentence boundaries with their lowercase appearances, for instance. In the third approach, we change all remaining tokens to lowercase, except for leaving the case of all strings containing 'apple' (case insensitive) alone. We made this decision based on the semantic difference between the capitalization of Apple the company and apple the fruit, which is a very significant feature in our dataset. While correct capitalization is naturally not always present for this token, this is one case where capitalization serves to disambiguate. In our dataset, 'Apple' appeared 11784 times in 10500 tweets, 'apple' appeared 4765 times in 4508 tweets, and other case-variations appeared 292 times in 237 tweets (all but one of which was 'APPLE'). This same principle can be generalized to a larger set of keywords and applied to other companies in a similar fashion.

Optionally, our tokenizer will remove the token 'RT' from all tweets. 3986 tweets in our dataset contain this token, signifying a retweet. Not knowing whether this feature added discriminatory power to our classifier, we toggled its inclusion during testing and found that removing it did lead to a slight increase in accuracy on our tests.

Any tokens remaining at this point during tokenization are very likely to be words in phrases. In order to focus the feature set, we split by whitespace and trim any non-word characters other than percent and dollar signs from token boundaries, stripping punctuation, quotation marks, parentheses, etc.

We chose to apply bigramming to only the tokens parsed in this final step. Bigramming led to a statistically significant increase in our results over the same tokenization without bigramming, and even moreso over whitespace-split tokenization with or without bigramming applied.

We constructed this tokenizer such that any step in tokenizing can be toggled upon instantiation in order to test the impact of each feature.

## Experimental Setup

In order to determine which combination of preprocessing steps would result in the highest performance, we built a custom evaluation utility that iterates through all possible valid combinations of pipes and evaluates their performance. This led to nearly one hundred combinations, as each of six tokenizer variations was run with and without bigramming, with and without stemming, etc. For consistency, the disjoint training and test sets used to evaluate each combination were identical for each iteration. To best observe the effects of training set size, training sizes of 50 to 1500 were tested in increments of 50, where each set is a superset of the previous set with 50 new randomly chosen instances, as opposed to using n-fold cross validation, which would not display this characteristic evenly across varied training and testing sizes.

To evaluate this large set of results, we chose three relevant evaluation metrics. The first is precision, or the ratio of correctly labeled tweet instances to the number of instances assigned a particular label ("apple" or "none" in our case). Higher values for this metric mean fewer false positives for the desired company, which is the overall goal in this classification task. The second metric was recall, or the ratio of correctly classified instances out of the full set of instances which should have the label in question. This metric is less relevant to our purpose, but still interesting. The sheer scale of Twitter means missing even half of the tweets mentioning a company and mislabeling them as about 'none' is of little consequence. Finally we use the F1 metric, a standard combination of precision and recall that is useful in determining overall if some pipeline combination is better than another in both precision and recall.

## Experimental Findings

Our desired result was a high precision classifier that could be useful for gathering a large number of tweets mentioning a company or its products. Looking at an overview of the results in Figure 1, we can see all pipeline combinations had over 70% of "apple" labelings correct with only 50 training tweet instances. However, given that a simple strategy of labeling every tweet as "apple" would be correct 64% of the time, this is only an improvement in precision of approximately 9%. The more relevant and interesting result is the 43% improvement in precision from the simple strategy seen with our custom Twitter-aware tokenizer and all other Pipes like lowercasing, Stemming and Bigrams trained on the maximum of 1500 instances. Futhermore, we observed 12% and 13% increases in precision versus plain Whitespace and Alphanumeric based tokenizers respectively. This relatively large improvement indicates not only if the tokenization strategy used important, but also that we were relatively successful in creating an effective tokenizer that can be further augmented by standard natural language processing techniques like stemming, removing stop words, applying n-grams, or simply lowercasing the text. We can see in Table 8 below that all of top five pipeline combinations include our Twitter-aware tokenizer. In fact, one must go down to the 24th and 33rd ranked combinations to find ones that utilize the Alphanumeric and Whitespace based tokenizers, respectively.

### Tokenizers

We found that our Twitter-aware tokenizer with the full suite of stemming, stopword removal and bigramming outperformed the whitespace tokenization with comparable stemming, stopword removal and bigramming by 5.2% in the precision metric when trained on 1500 instances. A similar comparison applies to the alphanumeric tokenizer, our Twitter-aware tokenizer was 4.5% more percise labeling tweets as mentioning Apple. Applying stemming and stop word removal to both pipelines increased each metric slightly, giving rise to the well distributed set of results seen in Figure 1. Bigramming and other pipes had some effect, but compared to the multiple percentage point increases

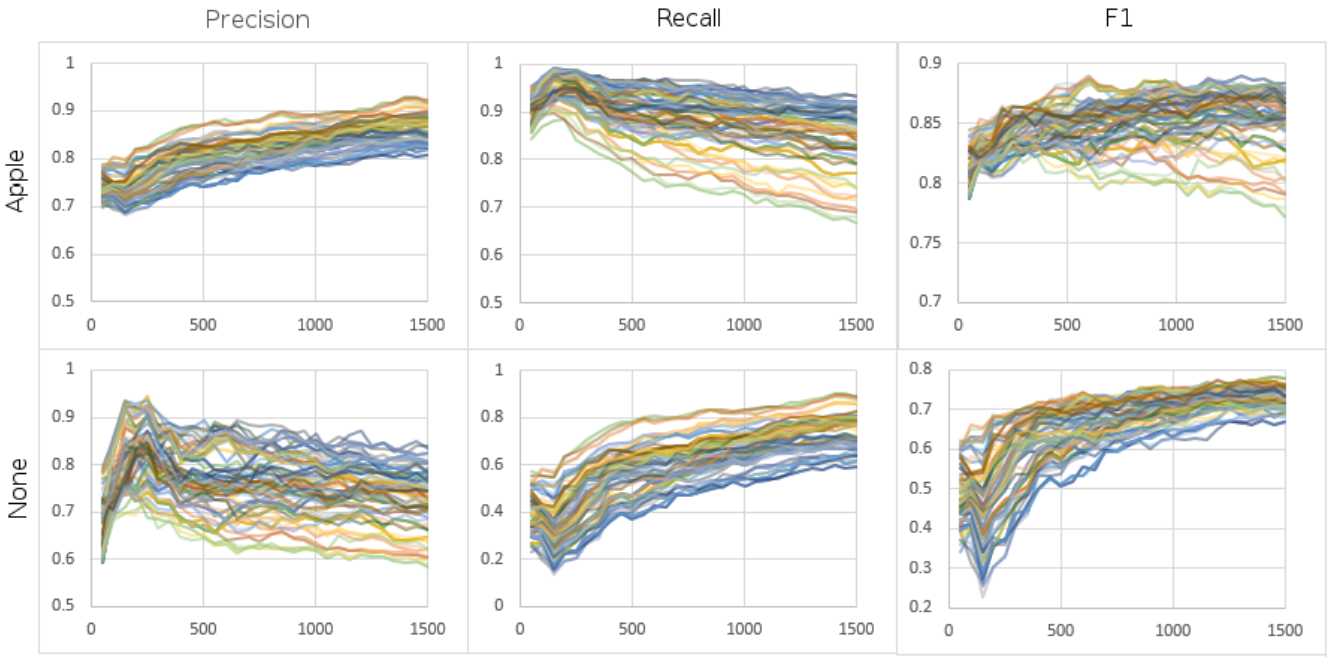


Figure 1: Metrics for all Pipeline combinations, training set sizes 50 through 1500 instances

| *Rank* | *Pipeline Description* | *Precision for Apple* |
|---|---|---|
| 1 | Twitter Tokenizer, Lowercased, Stemmed, Stopwords Removed, Bigrams | 92.2% |
| 2 | Twitter Tokenizer, Lowercased, Stopwords Removed, Bigrams | 91.6% |
| 3 | Twitter Tokenizer, Stopwords Removed, Stemmed, Bigrams | 91.6% |
| 4 | Twitter Tokenizer, Stemmed, Bigrams | 91.4% |
| 5 | Twitter Tokenizer, Stopwords Removed, Bigrams | 91.3% |
| 24 | Alphanumeric Tokenizer, Remove Stopwords, Bigrams | 88.2% |
| 33 | Whitespace Tokenizer, Stemmed, Remove Stopwords, Bigrams | 87.7% |

Table 8: Top performing pipeline combinations for 1500 training instances

from changing the tokenization method, they had relatively minor effects individually.

Table 9, the most common features for various pipelines, shows the large effect that preprocessing steps like stopword removal, capitalization normalization, bigramming, and tokenization have on the feature distribution. The bigrams iphone_5 and my_iphone prove to be useful when finding tweets about Apple and the bigrams new_post and apple_juice are useful when finding tweets not about Apple. Note the case of the word 'apple' in its various forms. It appears most commonly as 'Apple' in tweets about apple and lowercased in other tweets. This is expected, but there is also a high incidence of the word 'Apple' in tweets not about the company. It is interesting that '...' appears as a very common feature using our Twitter-aware tokenizer, as well as the token 'rt'. These appear with equal weight in tweets both about and not about Apple, so we may benefit from removal them during tokenization.

## Contributions and Future Work

We have presented a pipeline for processing tweets that increases accuracy and precision of the resulting models. We introduce a layered approach to tokenization that sees improvements over other rule-based methods. We have built several reusable Java components that extend the Mallet API for processing of data from Twitter. We created a TweetJsonIterator that accepts a file of Twitter's JSON-formatted tweets and creates training Instances containing their values for easy processing during model training. We have implemented several Pipes that can be reused or easily modified to suit a similar Twitter processing purpose. These include Link2Title, Stemmer, SpellCheck, and Tokenize, the latter of which serves as a more extensible tokenizer than Mallet's default. We release our testing utility as a new way of comparing multiple similar pipelines. All code is available through our project website. We also provide a corpus of 100,000 tweets selected by Apple-related keyword and our labeled data set of 2000 tweets. The bash scripts used to filter the data for near-duplicates and spam are also available for download.

We could test the robustness of this system by applying this process to other companies for a similar-length time slice of training data. We could also test this model against

| Twitter-aware tokenizer | | | | alphanumeric_ tokenizer | | | | whitespace tokenizer | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| no removal | | stopword removal | | no removal | | stopword removal | | no removal | | stopword removal | |
| **apple** | **none** | **apple** | **none** | **apple** | **none** | **apple** | **none** | **apple** | **none** | **apple** | **none** |
| Apple | i | Apple | apple | t | t | Apple | http | Apple | apple | Apple | apple |
| iphone | apple | iphone | Apple | co | co | http | apple | to | I | iPhone | Apple |
| i | the | ... | ... | Apple | http | iPhone | Apple | I | the | - | - |
| the | Apple | ipad | mac | http | I | iPad | Mac | the | Apple | RT | Mac |
| to | and | ipod | juice | I | apple | RT | ED | my | to | iPad | RT |
| my | to | rt | rt | to | Apple | apple | juice | a | and | iPod | juice |
| a | a | #apple | post | the | the | iPod | RT | iPhone | a | 5 | & |
| ... | ... | 5 | new_post | iPhone | to | 5 | iPhone | for | for | iphone | iPhone |
| on | for | iphone_5 | iphone | my | and | iphone | post | on | - | apple | post: |
| for | new | my_iphone | apple_juice | a | a | ED | A0 | and | my | #apple | I'm |

Table 9: Ten most common features for various pipelines.

tweets from a different time span to determine whether the data is subject to temporal overfitting, and to what degree.

Yerva et al. demonstrate a system of using background knowledge and relatedness, in which current information in the Twitter stream is used to more accurately estimate the priors for a Naive Bayes classification (Yerva et al.). We believe that inclusion of this in our method could allow us to perform more accurate disambiguation.

Wang et al. have shown improvements over the bag-of-words approach to text classification using Wikipedia as a thesaurus for term disambiguation. We could use this to enrich the feature set found in tweets and relate similar or co-occurring words in tweets (Wang et al., Wang et al.).

Our tokenization process makes several simplifying assumptions that we could address. We assume certain classes of emoticons are not used often enough to be significant, and leave punctuation out of our feature set. We could measure the presence of these (and more) features on Twitter data and adjust our tokenization process to reflect that.

Sriram et al. found that utilizing author profiles provides a good method for further disambiguation of tweets into categories, as authors tend to compose tweets within a limited set of categories. They also delineate a set of eight features ('8F') used for training, instead of using the bag-of-words approach as we do. Through further analysis of the most useful features found in our approach, we could also choose a discrete set of binary or n-ary features most relevant to this classification task and explore its accuracy versus bag-of-words (Sriram et al.).

## Individual Contributions

Specific team member contributions follow from what we originally stated in our project proposal. Brandon implemented utilities to push and pull tweets from our database in Java. He also researched about and wrote different Pipes that could be used by Mallet to transform the tweets during training. Brandon architected the project structure, and helped edit the project's paper and website. Curtis integrated Mallet into the project, prototyped the testing utilities, and wrote the JsonTweetIterator. He also wrote the majority of the paper, implemented the Twitter-aware tokenizer, and wrote bash scripts to process corpora for testing and analysis. Dan harvested tweets from Twitter and managed a database to store all our content. He designed and implemented the testing utility to compare all combinations of Pipes. Dan performed the statistical analysis and generated the graphs for the paper.

We, Brandon Maxwell, Curtis Ullerich, and Daniel Stiner, hereby concur with the contents of this report, in the knowledge that it is an accurate representation of the work we did and the results we found, for our term project.

## Acknowledgments

Thank you to Dr. Jin Tian and our class teaching assistant, Ru He, for their helpful guidance during this research project. We also extend our thanks to our friends for their help in manual labeling of tweets.

## References

Michel Krieger Brendan O'Connor and David Ahn. Tweetmotif: Exploratory search and topic summarization for twitter, 2010. URL http://aaai.org/ocs/index.php/ICWSM/ICWSM10/paper/view/1540/1907.

Gustavo Laboreiro, Luís Sarmento, Jorge Teixeira, and Eugénio Oliveira. Tokenizing micro-blogging messages using a text classification approach. In *Proceedings of the fourth workshop on Analytics for noisy unstructured text data*, AND '10, pages 81–88, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0376-7. doi: 10.1145/1871840.1871853. URL http://doi.acm.org/10.1145/1871840.1871853.

Andrew Kachites McCallum. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu, 2002.

Alexander Pak and Patrick Paroubek. Twitter as a corpus for sentiment analysis and opinion mining. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Bente

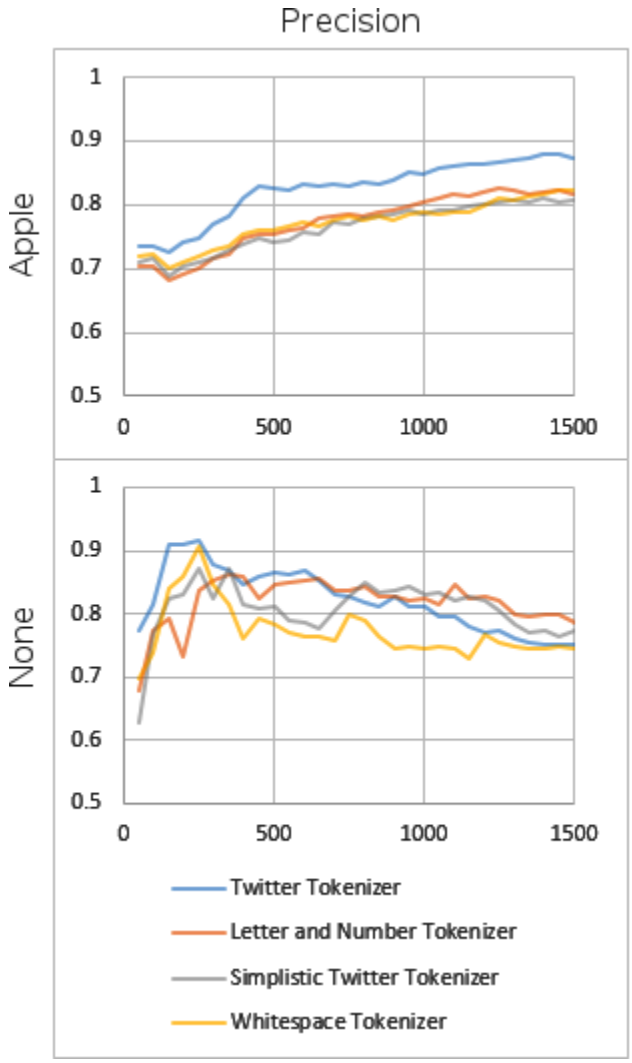


Figure 2: Precision of main tokenizers


Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, Mike Rosner, and Daniel Tapias, editors, *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta, may 2010a. European Language Resources Association (ELRA). ISBN 2-9517408-6-7.

Alexander Pak and Patrick Paroubek. Twitter as a corpus for sentiment analysis and opinion mining. In *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*, Valletta, Malta, May 2010b. European Language Resources Association (ELRA). ISBN 2-9517408-6-7. URL http://www.lrec-conf.org/proceedings/lrec2010/pdf/385_Paper.pdf.

Marco Pennacchiotti and Ana-Maria Popescu. A machine learning approach to twitter user classification. In Lada A. Adamic, Ricardo A. Baeza-Yates, and Scott Counts, editors, *ICWSM*. The AAAI Press, 2011. URL http://dblp.uni-trier.de/db/conf/icwsm/icwsm2011.html#PennacchiottiP11.

M. Porter. An algorithm for suffix stripping. *Program*, 14 (3):130–137, 1980.

Christopher Potts. Sentiment symposium tutorial: Tokenizing, 2011. http://sentiment.christopherpotts.net/tokenizing.html.

Eduardo J. Ruiz, Vagelis Hristidis, Carlos Castillo, Aristides Gionis, and Alejandro Jaimes. Correlating financial time series with micro-blogging activity. In *Proceedings of the fifth ACM international conference on Web search and data mining*, WSDM '12, pages 513–522, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-0747-5. doi: 10.1145/2124295.2124358. URL http://doi.acm.org/10.1145/2124295.2124358.

Burr Settles. Closing the loop: fast, interactive semi-supervised annotation with queries on features and instances. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1467–1478, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics. ISBN 978-1-937284-11-4. URL http://dl.acm.org/citation.cfm?id=2145432.2145588.

Bharath Sriram, Dave Fuhry, Engin Demir, Hakan Ferhatosmanoglu, and Murat Demirbas. Short text classification in twitter to improve information filtering. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '10, pages 841–842, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0153-4. doi: 10.1145/1835449.1835643. URL http://doi.acm.org/10.1145/1835449.1835643.

Pu Wang, Carlotta Domeniconi, and Jian Hu. Using wikipedia for co-clustering based cross-domain text classification. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, ICDM '08, pages 1085–1090, Washington, DC, USA, 2008. IEEE Computer Society. ISBN 978-0-7695-3502-9. doi: 10.1109/ICDM.2008.136. URL http://dx.doi.org/10.1109/ICDM.2008.136.

Pu Wang, Jian Hu, Hua-Jun Zeng, and Zheng Chen. Using wikipedia knowledge to improve text classification. *Knowl. Inf. Syst.*, 19(3):265–281, May 2009. ISSN 0219-1377. doi: 10.1007/s10115-008-0152-4. URL http://dx.doi.org/10.1007/s10115-008-0152-4.

Surender Reddy Yerva, Zoltn Mikls, and Karl Aberer. Entity-based classification of twitter messages. *IJCSA*, 9(1):88–115, 2012. URL http://dblp.uni-trier.de/db/journals/ijcsa/ijcsa9.html#YervaMA12.