Auto-Generating Relevant Twitter #Hashtags

Matthew Hauser
Johns Hopkins University
mhauser5@jhu.edu

Jade Huang
Johns Hopkins University
jhuang61@jhu.edu

Abstract

We introduce an approach of using a Naive Bayes classifier to “auto-generate” relevant Twitter hashtags. Relevancy is determined based on frequency of a word and hashtag occuring together as well as frequency of a hashtag. This could potentially be useful to help generate descriptive tags for users to increase their visibility and searchability. Our training data consists of a mix of precollected corpuses of tweets as well as randomly collected tweets using two libraries of Python scripts and the Twitter API.

1 Introduction

Hashtags, or #hashtags, are the new Dewey-Decimal system for organizing data on the web. On social media outlets such as Twitter, Facebook, Instagram, and YouTube, hashtags allow users to to“label” statuses and pictures, enabling their content to be searchable by other users. One only has to click on the hashtag #MillionsMarchNYC on Twitter to see a conglomeration of different tweets and pitures about the Millions March in NYC. Being searchable has the potential to increase hits, popularity, followers, and influence as well as raise awareness about a particular topic.

There are no set boundaries on what can be hashtagged–no predefined set of hashtags that all users are confined too. A user can hashtag virtually anything, though he or she is restrained to 140 characters on Twitter. While users typically use hashtags to identify their status or picture with a particular topic (or many topics), hashtags are often abused or misused out of selfish intent, sarcasm, and ignorance. Spammers often jam tweets with trending hashtags or topics in order to field traffic to their account. Other users use hashtags as a continuation of their statuses #CreatingUnSearchableTags but recreating ”speaking under his or her breath” in social media interaction. Others are simply unaware of tag conventions and often nullify tags by following a hashtag with punctuation, rendering their hashtag unsearchable. Auto-generated tags based on content can help users avoid mistakes in hashtagging as well as ensure users are using hashtags for their designed purpose: for enabling searchable objects.

Our proposal is to use a Naive Bayes Classifier to “auto-generate” relevant hashtags for Tweets. There is an enormous mass of hand-labeled data: tweets hashtagged by their own users. We use various corpuses retrieved from the Internet, including the Sentiment 140 corpus data[1], the Twitter Political Corpus[2], and randomly-retrieved Tweets using a Python script utilizing the Twitter Stream API[3].

2 Related Work

Various tag generators already exist online, such as Hashtag Generator[4] and Better Way to Web’s Tag Generator for YouTube[5]. HashTag Generator generates three tags based on user input: a tag of the entire user input without spaces, a tag of the entire

[1] http://help.sentiment140.com/for-students/
[2] http://www.usna.edu/Users/cs/nchamber/data/twitter/
[3] https://github.com/mdredze/twitter_stream_downloader
[4] http://www.hashtaggenerator.com/
[5] http://www.betterwaytoweb.com/tag-generator-for-youtube/

user input without vowels, and a tag of the first letter of each word in the user input. Such options may be relevant in simplified cases, such as reducing a long string such as “Music Teacher’s Association of America” down to “MTAC”, or “Tracking The Virus On Twitter” down to “TrckngThVrsNTwttr”. However, this does not provide any categorical information about the tweet.

Better Way to Web’s Tag Generator for YouTube generates a set of related keywords based on user input. For example, keywords related to “cookie” include the following: chocolate, how to, food, sweet, easy, simple, kids, shop, candy, DIY, toy, bakery, video game (industry), sugar, shopping, milk. Generating a set of related words to serve as descriptive keywords, be it through synonyms or words commonly seen with the word in question, can be helpful when it comes to to driving traffic to a video.

Schlomi Babluki, Chief Scientist at Swayy[6], created a tag generator based on Natural Language Processing tools: word frequency, stemming, and stopwords in an effort to tag news article titles in tweets based on the content of the article. Words seen the most in the body of the article and also included in the title are hashtagged.

3 Naive Bayes

Naive Bayes Classifiers are a family of probabilistic models based on Bayes Theorem with naive assumptions about the data. Bayes Theorem states

$$P(A|B) = \frac{P(A)P(B|A)}{P(B)} \tag{1}$$

Where

1. $P(A)$ represents the prior probability, or the initial belief in A.
2. $P(A|B)$ represents the conditional probability of A given B.
3. $\frac{P(B|A)}{P(B)}$ represents the support B provides for A.

In our case, we are examining the following conditional probability:

$$P(t|\mathbf{W}) = P(t|w_1, w_2, ..., w_N) \tag{2}$$

[6]http://blog.swayy.co/post/61672584784/an-algorithm-for-generating-automatic-hashtags

where t is a hashtag and $\mathbf{W}$ is a sequence of N words $w_1, w_2, ..., w_N$

We can expand this using Bayes Theorem:

$$\begin{aligned} P(t|w_1, w_2, ..., w_N) &= \frac{P(t)P(w_1, w_2, ..., w_N|t)}{P(w_1, w_2, ..., w_N)} \\ &= \frac{P(t, w_1, w_2, ..., w_N)}{P(w_1, w_2, ..., w_N)} \end{aligned} \tag{3}$$

The problem is how to calculate $P(t, w_1, w_2, ..., w_N)$. We *can* approximate $P(w_i|t)$ for any given word w_i with maximum likelihood estimates and counts, namely:

$$P(w_i|t) = \frac{P(w_i, t)}{P(t)} = \frac{count(w_i, t)}{count(t)} \tag{4}$$

Where

1. $count(w_i, t)$ counts the number of times word w_i and tag t are seen together in training data
2. $count(t)$ counts the number of times tag t is seen in training data

We assume that each word, w_i is conditionally independent of all other words. At first, this may appear as a poor assumption because the words of a sentence are generally related to one another, such as in the short Tweet: “I like tasty cookies #yum”. Clearly “tasty” and “cookies” are related and help to create “#yum”. This simple case is complicated with real-world Twitter data, where you find sentences constructed where the tag is incorporated into the tweet, such as in “I like #yummy cookies”. Now, constructing a metric for describing dependence, using methods such as bigram or trigram models, or parsing the sentence, appear much less feasible. Further, it creates a sentence where each word does become conditionally independent (or at least more closely approproximates conditional independence) with respect to the tag.

Under these independence assumptions, we are able to calculate $P(t, w_1, w_2, ..., w_N)$. First we expand using the chain rule:

$$\begin{aligned}P(t, w_1, w_2, ..., w_N) &= P(t)P(w_1, w_2, ..., w_N|t)\\ &= P(t)P(w_1|t)P(w_2, ..., w_N|t, w_1)\\ &= P(t)P(w_1|t)P(w_2|t, w_1)P(w_3, ..., w_N|t, w_1, w_2)\\ &= P(t)P(w_1|t)P(w_2|t, w_1)P(w_3|t, w_1, w_2)\\ &\quad *P(w_4, ..., w_N|t, w_1, w_2, w_3)\end{aligned} \tag{5}$$

Under naive assumptions that each word w_i is conditionally independent, we can reduce the following:
$P(w_2|t, w_1) = P(w_2|t)$
$P(w_3|t, w_2, w_3) = P(w_3|t)$
$P(w_i|t, w_{i+1}...w_N) = P(w_i|t)$
$P(w_i|t)$ is a value we can calculate through maximum likelihood estimates. This means $P(t, w_1, w_2, ..., w_N)$ can be calculated as:

$$P(t, w_1, w_2, ..., w_N) = P(t)\prod_{i=1}^{N} P(w_i|t) \tag{6}$$

This means $P(t|\mathbf{W})$ can be calculated as:

$$P(t|\mathbf{W}) = \frac{P(t)}{P(w_1, w_2, ..., w_N)}\prod_{i=1}^{N} P(w_i|t) \tag{7}$$

4 Data

Our data consisted of precompiled corpuses of Tweets as well as automatically generated corpuses using

4.1 Precompiled sources of data

- Sentiment140 Corpus: intended for discovering sentiment about a subject through Tweets, consisting of 1,048,576 tweets, one per line. These tweets were automatically collected through the Twitter Search API using keyword search and were stripped of emoticons during processing.
- Twitter Political Corpus: intended for classifying whether a given Tweet was political or not, consisting of about 4,000 tweets. Half of the tweets are a randomly selected set of tweets from Twitter's "spritzer" feed collected between June 1, 2009 and Dec 31, 2009. The other half randomly selected from a subset of tweets that contained at least one political keyword in each tweet.

4.2 Generated data

Data was generated using the following libraries:

- Python Twitter: a Python wrapper around the Twitter API, used to fetch about 3,000 tweets from 20 randomly-selected users.
- twitter_stream_downloader: a simple Python script used to download a random sample of 570,112 tweets from the Twitter streaming API.

4.3 Processing data

We processed our data into the following format using hand-written Python scripts:

- One tweet per line, for reading purposes
- Only tweets with hashtags included, in order to train and evaluate our classifier

Modifications made to corpuses:

- Sentiment140: Eliminated tweets without hashtags
- Twitter Political Corpus: Eliminated labels (political or not) and tweets without hashtags
- Data collected using Python Twitter: Eliminated tweets without hashtags
- Data collected twitter_stream_downloader: Parsed resulting JSON file to include only the text of tweets and eliminated tweets without hashtags.

5 Model

5.1 Training

During training, we accumulate two sets of counts while reading the training corpus:

- Words and tags: how often a word and a tag occur together in a Tweet
- Tags: how often a tag occurs in a Tweet

5.2 Testing

During testing, we classify tweets by finding the tag that yields the highest probability of a word sequence upon conditioning on the tag:

$$\arg\max_{tag} P(t = tag) \prod_{N}^{i=1} P(w_i | t = tag) \quad (8)$$

We are able to do this calculation based on the counts we accumulated during the training stage and we only consider tags we have seen while training.

However, novel words are to be expected at test time since the vocabulary for a given tweet is essentially infinite. Thus when we encounter a novel combination of a word and tag, i.e. where $count(w_i, t) = 0$, we implement add-1 smoothing in order to smooth our estimates and not entirely discount novel (word, tag) pairs.

$$\frac{count(w_i, t)}{count(t)} = \frac{1}{count(t) + V} \quad (9)$$

Where V is equal to the total number of (word, tag) combinations we have seen in training.

5.3 Evaluation

Accuracy is evaluated based on whether each tag in the set of predicted tags is included in the set of original tags that were hashtagged with a given test tweet. If so, the total number of correctly tagged tweets is incremented. Case is ignored, so the tag "#FollowFriday" is considered equivalent to the tag "#followfriday". This is a harsh metric for evaluation because similary tags such as "#ff" and "#followfriday" are deemed distinct tags. Tweets without tags are not considered for evaluation purposes.

6 Code

We wrote our Naive Bayes Classifier in Python. Classes include:

- Naive Twitter
  - If the command is to `set`, Naive Twitter sets (word, tag) counts and tag counts to the given count files
  - If the command is to `count`, Naive Twitter reads in the training data and accumulates counts for each (word, tag) pair and each tag. For each line, the tags are first counted and accumulated. Then the count for (each word in the tweet, each tag in the tweet) is incremented. Duplicate words and duplicate tags in a tweet are not considered, in order to prevent repeated iterations of a word or tag in a tweet to have an unfair advantage (i.e. a tweet of `"Cookie cookie cookie #yum #yum"` will add 1 to $count(cookie, yum)$ and 1 to $count(yum)$. In addition, mentions (i.e. "@username") are ignored but links are taken into account, which can be a descriptive measure if a given link and hashtag become strongly associated with each another.
  - Counts are outputted to a file in the format of

    ```
    word<||>tag count
    ```

    for (word, tag) counts, and

    ```
    tag count
    ```

    for tag counts.
  - If Naive Twitter is called on to test, Naive Twitter finds the tag yielding the maximum sum of the log probability of the tag and log probability of each word given the tag. Optimizations include halting consideration of a tag if the probability is lower than the stored highest probability.

- Parser
  - If the command is to `train`, Parser removes tweets without hashtags in a training file of tweets (in the format as specified above in Processing Data)
  - If the command is to `read`, Parser reads two files–one of (word, tag) counts and another of tag counts–and stores the counts.

- Twitter Tagger
  1. The driver program. We define the training and test data file path in Twitter Tagger, who first hands the training and test data to Parser to remove tagless tweets

2. The processed data is handed to NaiveTwitter to train and accumulate counts
3. Each line in test is handed to NaiveTwitter to test on
4. Accuracy counts are accumulated: if each predicted tag is in the set of original tags for a tweet, the prediction is considered to be correct.
5. Prints percent accuracy over all test data

7 Results

7.1 Training and Testing on Sentiment140 Corpus

We split up the Sentiment140 Corpus into a training corpus composed of 80% of the hashtagged tweets (911,629 tweets with 21,813 hashtags) and a test corpus composed of 20% of the hashtagged tweets (136,947 tweets with 5,423 hashtags). The accuracy achieved on the test tweets was 8.849% over 4170 tweets. This is at first a rather alarming result since the tweets are taken from the same overall corpus so it seems the classifier should do quite well on predicting hashtags, but other things must be taken into consideration. While the tweets originally belonged to one corpus, the tweets were all randomly collected at a set, rather long time period in time (2009-2010). We believe the classifier did poorly for a number of reasons:

- Many tags and (word, tag) counts have a count of 1 since vocabulary of tags and (word, tag) is infinite. However, some tags, such as "#fb" (or colloquially known as Facebook) appear upwards into 1000 times in training data. This tag is "used by people who have installed the Selective Twitter Update application on Facebook. Tweets including the hashtag #fb are automatically imported to Facebook, all others are ignored[7]". Thus, the hashtag of "#fb" provides no information about the content of a tweet besides the fact that it has a parallel status on Facebook. This gives an unfair advantage to the tag and relating words in the tweet to the tag will yield no descriptive correlation between words and tags besides that when in doubt, tag with "#fb".

- Even though we split the hashtagged corpus into 80% train and 10% test, this does not guarantee a large crossover of hashtags between the two corpuses. It is very well possible that there are a large number of original hashtags not included in train or original words not included in train in the test corpus which results in low exact accuracy.

In another experiment, we composed a training corpus consisting of the Sentiment140 Corpus, the Twitter Political Corpus, tweets gathered from 20 random users, and 400,000 randomly sampled tweets and a test corpus consisting of 15,354 randomly sampled tweets. The training corpus has a total of test corpus has a total of 25,967 hashtags. When testing on 15,354 of the tweets, we obtained 25.54% accuracy. This rise in accuracy can possibly be attributed to a number of reasons:

- The classifier had more diverse training data spanning different time period and thus was able to generalize better over the test data

- Many of the tweets in the test data share similar hashtags in the train data, and the similar hashtag had a high frequency count. Therefore, the classifier had a better chance of picking out these kinds of hashtags.

In an experiment to test how the classifier is when it has already peeked at the test data, we trained the classifier on a set of randomly collected 570,112 tweets with 169,329 hashtags and tested the classifer on 15,354 tweets, a subset of the 570,112 tweets. As expected, the classifier scored well–87.20%–since it accumulated counts and built up probabilities based on the same corpus that it tested on.

8 Problems Encountered

8.1 Data Collection

This was initially solved by the discovery of preexisting corpuses such as Sentiment140 and The Twitter Political Corpus. Difficulty was encountered when trying to implement a recursive algorithm using Python Twitter to fetch the tweets of users and

[7]https://www.hashtags.org/analytics/fb/

their friends, which was then resolved by the discovery of twitter_stream_downloader, which facilitated the fetching of a large number of tweets in a reasonable amount of time.

8.2 Noisy Data

Twitter users do not always reliably hashtag their tweets. Often users simply include the empty hashtag, or "#" or render hashtags unsearchable by placing punctuation after a hashtag, such as in the case of "#." or "#?". These hashtags are not of interest since they are unsearchable, do not provide any relevant information about tweets, and should not be considered while determining what hashtags to predict.

8.3 Appropriate Evaluation Metrics

Twitter users are unbounded in their use of hashtags, and the creative boundaries are boundless. Even if we train on a large set of training data, such as the million tweets in the Sentiment140 corpus, it is difficult to accurately predict the exact same set of tags for completely different set of tweets. If the tweets are gathered at different points in time, different hashtags will be trending and thus popular with users, such as "#mtvstars" which is currently popular in December of 2014 due to MTV using the hashtag to collect votes for different pop stars, or "#iranelection" which was popular in 2009.

While our evaluation metric was based on exact matches of tags, we noticed that for "incorrectly-hashtagged" tweets, the hashtags predicted by the classifier were often well-related to the tweet. Examples include `''Protests of police shootings in D.C., NY http://t.co/pygpIKBUad #morningjoe''`, which was given the hashtag of "#millionsmarchnyc". While the original tag is related to the Morning Joe segment on MSNBC and was not correctly predicted by our classifier, the predicted tag of "#millionsmarchnyc" is actually relevant to the tweet and the contents of the link.

Another example is the tweet `''Out with Lisa for a lovely night (@ Beatrice & Woodsley on #Yelp http://t.co/2nAYqnw6dM''`, which was given the hashtag of "#win". While the original tag was generated automatically through Yelp app, the classifier would not have known this backstory. However, the classifier did tease out that the words in the sentence were well correlated with the tag of "#win", which seems appropriate for the situation given that was it was "a lovely night".

Thus begs the question of what is an appropriate evaluation metric? Accuracy based on predicting the exact same set of hashtags especially when the training and test set come from different time periods will suffer from the gap in relevancy. However, it would be beneficial to measure accuracy also in terms of how relevant the predicted tag is to the original tag in terms of synonymity or equivalance, such as the tag "#ff" which should be considered equal to the tag "#followfriday". But to do this, we would have to define a dictionary of synonyms, which could very well be infinite especially if we consider multiple languages. Further, it is very hard (impossible?) to predict a novel hashtag, which is something our method makes no attempt to accomplish.

9 Comparison to Proposal

Our proposal was originally to create a classifier that would predict whether a user has Ebola or not using Logistic Regression based on data describing the spread of Ebola in the United States as well as synthetic data generated to describe healthy individuals. We did not have enough data, since there have been less than 10 cases of Ebola in the United States, and we were unable to generate our proposed synthetic data in a way that would be descriptive, thus we decided to overhaul our original plans completely. Here is what we were able to accomplish

- Collected more than a million tweets to serve as data

- Creating a Naive Bayes classifier that can
  - Predicts approximately 25% of hashtags in tweets
  - Predicts relevant hashtags for tweets based on uncovering (word, tag) frequency as well as tag frequency from a training set of tweets.

References

[Murty et al.2011] Narasimha Murty, M., Susheela Devi, V. 2011. Pattern Recognition: An Algorithmic Approach.

[Marchetti-Bowick et al.2012] Micol Marchetti-Bowick and Nathanael Chambers. 2012. Learning for Microblogs with Distant Supervision: Political Forecasting with Twitter. *In Proceedings of the European Association for Computational Linguistics*, Avignon, France.

[Go et al.2009] Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter Sentiment Classification using Distant Supervision.

[Jurafsky et al.2008] Jurafsky, Daniel and Martin, James H. 2008. *Speech and Language Processing*,