POSHF: Twitter Hashtag Recommendation

Matthew Lau
University of Texas
Department of Computer Science
mattlau@outlook.com

Johahn Wu
University of Texas
Department of Computer Science
johahnwu@cs.utexas.edu

***Abstract*—On Twitter, hashtags are used by users to tag the content of their 140 character long tweets which aids in categorization and indexing. Unfortunately due to the vague way hashtags are chosen and defined, many tweets are tagged incorrectly or are not tagged at all resulting in a loss of potentially useful information. We propose Part-Of-Speech+Hashtag-Frequency (*POSHF*), a supervised learning hashtag recommendation system that relies on a dual ensemble approach where one implementation generates appropriate hashtags based on the tags of existing training data and the other implementation extracts hashtags based on Part-Of-Speech tags of word phrases in the tweet. We tested our implementation on a Twitter data set consisting of approximately 97,000 tweets and demonstrated that *POSHF* is successful in recommending relevant hashtags by achieving a 90% precision when asked to find the top 3 most likely hashtags for a tweet.**

Keywords*—*Twitter, NLP, HashTag Recommendation, POS

I. Introduction

Twitter's ever increasing popularity and establishment as the most popular micro-blogging platform has led it to become a very popular source of current, up-to-date information. Unlike traditional blogging platforms, Twitter limits the size of their user's publications (called Tweets) to 140 characters forcing the content of the tweets to be precise and most importantly short. This combined with their user base of over 236 million active users per month allows it to be an extremely useful potential source in tracking real-time trends. For example, Johns Hopkins researchers were able to use the geographical data provided by users along with the content of the tweets to accurately determine the spread of the flu through New York City in 2014[1]. To aid in the tracking of trending topics, Twitter utilizes hashtags which are inserted within the tweet itself to help identify the topic of the tweet. Researchers can track rising trends by simply looking at which hashtags are suddenly being used frequently. A hashtag can be identified if it is prefixed by a # symbol followed directly without white space by a string and can be placed anywhere within the body of the tweet [2]. For example, the annual bicycle race that takes place in France could result in the hashtag #tourdefrance, #tdf or #france, all of which would be appropriate hashtags for the event. Our example also demonstrates the difficulties for a user in choosing the most correct hashtag as there are an arbitrary number of possible relevant hashtags which may result in two tweets about the same topic being tagged completely differently. In many cases, the user may forgo using hashtags altogether which would make it impossible to track trends.

As the use of hashtags becomes more popular in blogging sites, the need for a hashtag recommendation system becomes more prevalent to keep the information organized. Tumblr, another blogging platform, has a simple recommendation system based on the content of the user's blog [4]. Various other researchers have proposed similar recommendation systems for Twitter to solve the tagging problem. These algorithms can be generally categorized into two groups. The first includes algorithms that recommend pre-existing hashtags either from a set of predetermined topics or based on the existing hashtags of training tweets. While this system is effective for text categorization problems, using predefined hashtags ignores rising trend tweets as overall more popular hashtags may be favored over the more correct and timely hashtag. In addition, due to the arbitrary nature of hashtags, the correct hashtags may often not exist within the corpus of known hashtags and an incorrect hashtag may be assigned instead. The second group attempts to extract hashtags directly from the words and phrases of the tweet itself. This solves the aforementioned rising trend problem as the hashtag extracted will not be from an older and possibly irrelevant tweet. In section II of the paper, we will discuss past implementations that we have based our approach on.

Our recommendation system, fully described in section III, is a combination of an implementation that generates hashtags from learned data as well as an implementation that extracts hashtags from tweet content based on part-of-speech tags. In section IV-B we will discuss our evaluation methods and metrics. Section IV-C and IV-D will detail and discuss our results, and finally we will conclude in section VI.

II. Related Works

The most naive approach to hashtag recommendation is to retrieve the hashtags of the training tweet(s) that best matches the test tweet. Various studies have applied metrics popular in information retrieval to the problem with varying degrees of success [5][7]. A proposed hashtag recommendation system by Zangerle, Gassler and Specht applied *tf-idf* to the problem. Treating their testing tweet as a vector, their algorithm found the k most similar tweets from an existing training set based on the *tf-idf* weight. They then extracted the hashtags from the retrieved tweets and weighted each with the *tf-idf* value of its containing tweet in order to rank the hashtags.

The problem with using traditional information retrieval is that these approaches ignore the fact that documents/tweets to hashtags is a 1 to many relationship which greatly decreases the number of hashtags that can be retrieved. Otsuka, Wallace and Chiu proposed a novel approach that combats this problem while still loosely sharing a similar structure to the *tf-idf* approach called Hashtag Frequency - Inverse Hashtag

Ubiquity (*HF-IHU*). However, unlike *tf-idf* which looks at the association between words and documents, *HF-IHU* looks at the association between the words of a tweet and the hashtags associated with a tweet. The most significant limitation to Otsuka's approach was a problem with "out-of-vocabulary" hashtags due to the arbitrary nature in which users pick hashtags. As mentioned in their paper, even a perfect algorithm could only achieve a recall ceiling of 80% on their data set. We hope to address this problem by extracting possible hashtags from the content by parsing the tweet based on part-of-speech tags. In addition, we modified their implementation of *HF-IHU* to attempt to better "retrieve" relevant hashtags.

III. Our Approach: POSHF

A. Defining the Task

POSHF is a supervised algorithm for predicting hashtags, as such we train on labeled data. This consists of (preprocessed) tweet text and hashtag sets as input, and finally our algorithm outputs a predicted hashtag set. For the paper, we define notation as follows:

T – the the Tweet Corpus

t – a tweet object of T

$text_t$ – the preprocessed text of the tweet t

a_t – the actual hashtags of tweet t as from the original tweet text

p_t – the predicted hashtags for tweet t

Our final goal is to create an ensemble system combining *POS* and *HF* with two distinct methods.

1) $train(T)$ which trains a model given a corpus of tuples $(text_t, a_t)$
2) $predict(text_t, k)$ which returns p_t, a prediction set of k hashtags based on the trained model and $text_t$.

HF, as described in section III-C, is a supposed upgrade from the HF-IHU as described by Otsuka [3]. Additionally, we hope to see the results of training on POS-tagged tweets. POS-tagging has only been vaguely used as a feature in previous work but never directly for predicting hashtags as in *POSHF* [13]. We believe that *HF* and *POS* bring different qualities to the the ensemble, mainly being that the former is generative while the latter is extractive. With the two combined, the system will perform well, regardless of how much or little relevant data it has trained on.

B. Extraction - Part of Speech

The extractive half of *POSHF* utilizes POS-tagging in order to determine which Parts of Speech are most likely to be hashtagged. As tweets often contain terms and phrases that do not appear in grammatically correct English, we decided to utilize CMU's Tweebo Parser to POS-tag the tweets [6]. In the most naive fashion, we hypothesize that nouns and proper nouns are the most likely to be hashtagged. However we want to rely on statistical data rather than mere intuition.

In this approach, for each tweet in the training corpus, we POS-tag each term. If that term is a hashtag, then we increment its POS by 1. We do similarly for consecutive terms, creating POS-Unigram and POS-Bigram models. The pseudocode can be found in Fig. 1.

T – corpus of tweets to train on
U – POS-Unigram Model
B – POS-Bigram Model
for all $t \in T$ **do**
  $a_t \leftarrow$ actual hashtags of t
  $W_t \leftarrow$ list of POS-tagged terms in $text_t$
  $\alpha \leftarrow$ START_TAG
  for all $w \in W_t$ **do**
    $\beta \leftarrow POS_w$
    if $term_w \in a_t$ **then**
      Increment U_β by 1.
      $\gamma \leftarrow \alpha\beta$
      Increment B_γ by 1.
    end if
    $\alpha \leftarrow \beta$
  end for
end for
Normalize U and B
return U, B

Fig. 1. Alogrithm for training models in POS approach

$text$ – text to predict for
k – number of predictions requested
U – trained POS-Unigram Model
B – trained POS-Bigram Model
S – score of terms
$W \leftarrow$ list of POS-tagged terms in $text$
$\alpha \leftarrow$ START_TAG
for all $w \in W$ **do**
  $\beta \leftarrow POS_w$
  $\gamma \leftarrow \alpha\beta$
  if $term_w \in S$ **then**
    $S_{term_w} \leftarrow$ Interpolate(U_β, B_γ)
  else
    $S_{term_w} \leftarrow max(S_{term_w}$, Interpolate($U_\beta, B_\gamma$))
  end if
  $\alpha \leftarrow \beta$
end for
return k terms with the highest score in S.

Fig. 2. Algorithm for scoring hashtags in POS approach

Predicting hashtags is simply a matter of applying the POS-Unigram and POS-Bigram models to a given tweet. For each POS-tagged term (or pair) we look up the corresponding value in the models and interpolate the scores. Then the terms are sorted by score and the k terms with the highest scores are returned as hashtag suggestions. Some terms appear multiple times in a tweet and thus we account for this by simply taking the highest scores for those terms. The pseudocode is detailed in Fig. 2.

C. Generation - Hashtag Frequency

The generative half of *POSHF* consists of a modified version of the *HF-IHU* system proposed by Otsuka [3]. To differentiate our approach from Otsuka's we will refer to

ride	#tdf	1
with	#tdf	1
pros	#tdf	2
	#france	1
cons	#tdf	1
	#france	1

#tdf	ride, with, pros, cons
#france	pros, cons

ride	1
with	1
pros	2
cons	1

Fig. 3. Data structures created from training on example tweets THF(left) HWP(middle) ITF(right)

our modified implementation simply as *HF* (Hashtag Frequency) in the paper. *HF* relies on a training step that creates three data structures from the training tweet corpus: Term-to-Hashtag-Frequency, Hashtag-Word-Popularity and Inverse-Tweet-Frequency. Then the predicting step ranks the top k most likely hashtags for a tweet. Before the data structures are created, we perform the following additional pre-processing steps to the corpus to minimize noise:

1) Porter Stemming: Used the tartarus Porter Stemmer to remove common morphological endings from words [10].

2) Stop Words: Removed common words since words used ubiquitously won't provide any additional information. We attained the list of words used from an open source Google project [11].

The first data structure, called Term-to-Hashtag-Frequency (THF), is a mapping from terms to another map. This second map is from hashtags to counts. We explain this process further in the example. Hashtag-Word-Popularity (HWP) maps each hashtag to a set of all unique terms that appear with the hashtag. Finally, Inverse-Tweet-Frequency (ITF) maps all the terms of a corpus to the number of unique tweets that contains the term.

In Fig. 3, we demonstrate the three data structures created from a corpus of two tweets: "ride with pros #tdf" and "pros cons? #france #tdf". In respect to TDH, for each hashtag in a tweet, that hashtag mapping is incremented by 1 for each term. So [ride, with, pros] would all get an increment of 1 for #tdf. Moving onto the second tweet, [pros, cons] would all get an increment of 1 for both #tdf and #france. Thus we get the resulting data structure.

HWP is almost a reverse mapping of TDH as described above. Thus the first tweet would cause #tdf to gain [ride, with, pros], and the second tweet would cause both #tdf and #france to gain [pros, cons].

The predicting step relies on the calculations of three weights that correspond with each of the data structures: Hashtag Frequency ($hf_{te,h}$), Hashtag-Word-Popularity (hwp_h), and Inverse-Tweet-Frequency (itf_{te}) for each hashtag associated with a term in $text_t$.

Given a processed testing tweet $text_t$, we first iterate through each term te in $text_t$. itf_{te} is calculated using:

$$itf_{te} = \log \frac{|T|}{|ITF[te]|} \tag{1}$$

where $|T|$ represents the size of the tweet corpus and $|ITF[te]|$ denotes the number of tweets that term te appears in. Exactly the same as idf in *tf-idf*, itf_{te} discounts common terms that appear in many different tweets. Therefore hashtags that are associated with a commonly tweeted terms like "and" will weigh less than hashtags associated with more uncommon words like "France". [1]

Then for each term te, we iterate through all of the hashtags h associated with the term by looking at the THF data structure. To calculate $hf_{te,h}$, we use the following equation:

$$hf_{te,h} = \frac{THF[te][h]}{\sum_{h' \in THF[te]} THF[te][h']} \tag{2}$$

where THF[te][h] represents the frequency hashtag h occurs with term te. This value is then normalized by dividing by the total hashtag frequencies of all hashtags associated with te. The purpose of $hf_{te,h}$ is to reward hashtags that are commonly associated with a term in the tweet which is similar to how *tf* functions in *tf-idf*.

In addition, for each hashtag h, we calculate the weight htp_h with the equation:

$$hwp_h = \log \frac{|T_{Terms}|}{HWP[h]} \tag{3}$$

where $|T_{Terms}|$ denotes the total number of unique terms in the corpus and the denominator represents the number of unique terms associated with h. hwp_h functions similarly to itp_{te} in that its weight decreases as h is used with more terms. Its purpose is to discount popular hashtags that can be used in a wide range of tweets and instead favor hashtags that are more unique to the specific term in the tweet.

The score for the hashtag h associated with term te is calculated as $hf_{te,h} * hwp_h * itf_{te}$. Scores for all hashtags associated with each term in the test tweet are summed together and ranked to find the most likely predicted hashtags for the tweet. Then pseudocode for the *HF* algorithm can be seen in Fig. 4.

D. Combination - POSHF

Since our two approaches, *POS* and *HF*, rely on different algorithms, they should be independent and distinct enough that we can employ ensemble learning to obtain better predicted hashtags. In order to compare the predicted hashtags of each approach, we first had to normalize the values by dividing each hashtag score S_h by the maximum score. The most naive approach to combining is to simply merge the two predicted hashtag lists together and add up S_h for all hashtags that were predicted by both approaches. The intuition behind this is that hashtags predicted by *POS* and *HF* will automatically be given high scores and be chosen as the top k predicted hashtags. While this works relatively well, it ignores the strengths of both approaches as it weighs them equally regardless of scenario. Through testing, we found that *POS* performed significantly better than *HF* when trained with a small subset of the training data while *HF* performed better when trained with a larger data set.

[1] "and" will actually be removed during the stop word step in preprocessing. It is only used as an example of a common word

$Tw = \{t_1, t_2, ..., t_n\}$ where t_i is a term in tweet T
k – number of predictions requested
S – score of terms
THF – Term-to-Hashtag-Frequency map
HWP – Hashtag-Word-Popularity map
ITF – Inverse-Tweet-Frequency map
for all $t_i \in Tw$ **do**
  $itf_{t_i} \leftarrow \log \frac{|T|}{|ITF[t_t]|}$
  for all $h_j \in THF[t_i]$ **do**
    $hf_{t_i,h_j} \leftarrow \frac{THF[t_i][h_j]}{\sum_{h' \in THF[t_i]} THF[t_i][h']}$
    $hwp_{h_j} \leftarrow \log \frac{|T_{Terms}|}{HWP[h_j]}$
    $S_{h_j} \leftarrow S_{h_j} + itf_{t_i} * hf_{t_i,h_j} * hwp_{h_j}$
  end for
end for
return k terms with the highest score in S.

Fig. 4. Algorithim for scoring hashtags in HF approach

k – number of predictions requested
p – arbitrary number larger than k
S_POS_p – top p predicted hashtags for a tweet using POS
S_HF_p – top p predicted hashtags for a tweet using HF
Normalize S_POS_p
Normalize S_HF_p
$S\_Fin_k \leftarrow S\_POS_p$
for all $h\_hf_i \in S\_HF_p$ **do**
  if $h\_hf_i \in S\_Fin_k$ **then**
    $S\_Fin_k[i] \leftarrow max(h\_hf_i, S\_Fin_k[i])$
  end if
end for
return k terms with the highest score in S_Fin_k.

Fig. 5. Algorithm for scoring hashtags in POSHF approach

In order to tailor our algorithm to better take advantage of strengths of each approach, we decided to employ a different method of combination. Instead of adding the scores together when we come across a duplicate predicted hashtag, we take the max of the two scores. This is analogous to how duplicates are handled within the *POS* approach, which results in the combined $POSHF$ doing well both when training on a small amount of data and on a larger data set. The combination method can be summarized by looking at the pseudocode in Fig. 5

IV. Experiment Evaluation

A. Tweet Corpus

1) Scraping: We ran into a few obstacles when looking for data sets to experiment on. Currently, it is against the Twitter API's terms of service to distribute data sets containing tweets. Thus we decided to scrape our own data set [14]. A second challenge we faced was how sparsely hashtags were used. Twitter's API provided a rate limited stream function to scrape random tweets. However only 13% of tweets actually contain hashtags, which made collecting a sufficiently sized tweet-hashtag pair data set difficult [3]. To solve this, we employed a data set collection method utilized by Zangerle in which words from a dictionary are used as hashtag search queries for collecting tweets [7]. The selected words came from

Fig. 6. Example of non-alphanumerical character use in tweets

a dictionary called 6of12 [9]. This dictionary is a compilation of 12 different dictionaries, and the words in 6of12 are words that appeared in at least 6 of the 12 dictionaries. We believe by choosing this, we would have words that are commonly used as hashtag searches while also getting a good spread. Of the words in the 6of12 dictionary, we randomly selected 5,000 words to use as seed hashtags for our data set.

The final tweet corpus consisted of approximately $97,000$ tweets retrieved from Twitter at various times from 1 May 2015 to 7 May 2015. We wanted to get a wide variety of tweets to use while at the same time have enough tweets for each hashtag. The reason for the latter is so that the generative part of *POSHF* has more data to recognize a certain hashtag from than just a single tweet. Thus the general methodology used was to have a set of words and search Twitter for tweets hashtagged with those words.

We used Twitter4J's search functionality to interface with the Twitter API and searched for tweets hashtagged with the given word [12]. Getting approximately 20 tweets per word, we arrived with our dataset of 97,000 tweets.

2) Preprocessing: We preprocessed each tweet text to make it suitable for our system. This consisted of removing '#' characters and keeping track of hashtags for each tweet. For example, "I am so #happy #today!" would be stripped to "I am so happy today!" with a hashtag set of [happy, today].

In addition for Hashtag Frequency we also changed all words to lowercase and removed all non-alphanumerical characters. While it could be argued that keeping non-alphanumerical characters (i.e, emoticons) could provide additional information, their use is often so inconsistent that it may just introduce unnecessary noise into the model. Fig. 6 shows an example of a tweet consisting of ascii art that would result in random characters being associated with the #HNY hashtag.

We left the text in its original format for POS-tagging because the use of capitals and other punctuation could give valuable details as to the POS. For example, "Rangers" can be identified as a proper noun while "rangers" would just be identified as a plural noun.

B. Experiment Evaluation Methodology

Evaluating how well our recommending system predicts hashtags is a fairly difficult task to quantify. As mentioned in

section I, there are many "relevant" hashtags to a particular tweet. Even more so, many hashtags, while syntactically different, are semantically synonymous. Thus we present a relaxed version of precision and recall.

For evaluation notation, let p represent the set of predicted hashtags and a represent the set of actual hashtags.

1) Naive Precision: For precision, the normal formula of precision for a tweet is the number of correctly predicted hash tags divided by the number of predicted hashtags [8].

$$P(p,a) = \frac{\mid p \cap a \mid}{\mid p \mid} \tag{4}$$

However, we relax this to $NP(p,a) = 1$ if there exists any hashtag in the predicted set that also exists in the actual set.

$$NP(p,a) = \begin{cases} 1 & : \exists x \mid x \in p \wedge x \in a \\ 0 & : otherwise \end{cases} \tag{5}$$

In total, for a corpus T of tweets we find the total NP score by finding the mean of all the NP scores.

$$NP(T) = \frac{1}{\mid T \mid} \sum_{t \in T} NP_t(p_t, a_t) \tag{6}$$

2) Recall: The traditional recall formula is defined as the number of correctly predicted hashtags divided by the number of actual hashtags [8]. However, since POSHF takes in $k = \mid p \mid$, the number of hashtags to predict, it's possible that k is smaller than the actual hashtag set. Thus we modify the equation and make the divisor the smaller of the two. Simply put:

$$R(p,a) = \frac{\mid p \cap a \mid}{min\left(\mid p \mid, \mid a \mid\right)} \tag{7}$$

Similarly to Naive Precision, we find the total Recall score for a corpus of tweets by taking the mean score.

$$R(T) = \frac{1}{\mid T \mid} \sum_{t \in T} R_t(p_t, a_t) \tag{8}$$

3) Tenfold Cross Validation and Iterative Testing: For each $NP(T)$ and $R(T)$, we had two different ways of utilizing the evaluation methods: Iterative Testing and Tenfold Cross Validation. Tenfold Cross Validation was simply a matter of breaking up our Tweet corpus into 10 different sets: $T = [T_0, \cdots T_9]$. For each $T_i \in T$, we would train on the other 9 sets and test on T_i, evaluating using both $NP(T_i)$ and $R(T_i)$. This was done so that we would get a good grasp of the true accuracy of our evaluation by reducing variance. A single skewed testing set would be overshadowed by the other nine.

For Iterative Testing, we randomly choose one-tenth of T to be the testing set T_{test}. The rest went into the training set $T_{train} = T - T_{test}$. We iteratively trained our models and tested on T_{test}. For example, we chose the first 500, 1000, then 1500 and so forth of T_{train} to train and test on T_{test}. While evaluating each method, we made sure to use the same testing and training sets for each iteration in order to get a true comparison.

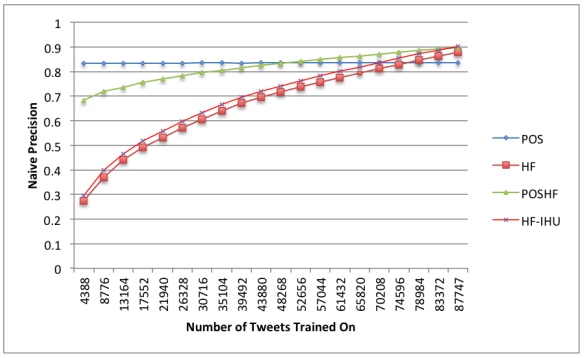

Fig. 7. Comparisons of different methods using Iterative Training and Naive Evaluation with k=3

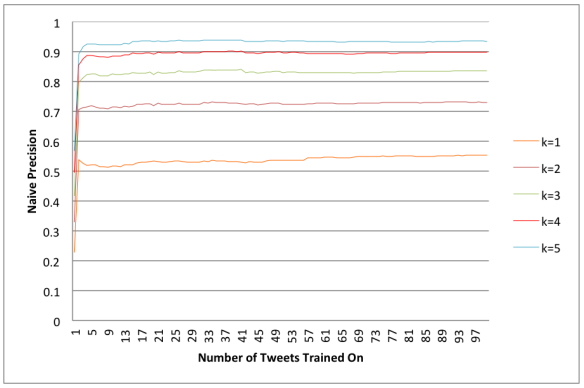

Fig. 8. POS Iterative Training using Naive Evaluation

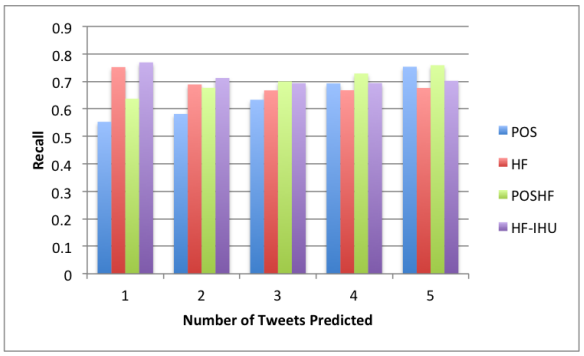

Fig. 9. Comparison of different methods using Ten Fold Testing and Recall Evaluation

C. Experiment Results

1) Comparisons for Naive Evaluation: The graph in Fig. 7 depicts how well each method did when trained on an increasing number of tweets, evaluated using Naive Iteration. In general it seems that all of the methods show a positive correlation between number of tweets trained on and testing score. A curious thing to note is that the variance in score for our *POS* approach is very little. However, it does start fairly high. Also note that *POSHF* starts off worse than *POS* but better than the other two. The *POSHF* Naive Precision continues to increase as the model trains on more data until the Naive Precision converges with that of *HF-IHU* at the very end.

Fig. 8 takes a closer look at the iterative training for the *POS* approach. Note that there is a huge jump between 1 and 5 tweets. After that, training on more tweets yields diminishing returns.

2) Comparisons for Recall Evaluation: Fig. 9 compares how well each method, when evaluated using recall, did using $k = [1, 2, 3, 4, 5]$, the number of hashtags predicted for each tweet. Remember that our formula for recall is slightly different, as noted in section IV-B2. Notice that *POS* and *POSHF* start low but perform increasingly well as k grows while *HF* and *HF-IHU* decrease as k grows.

D. Analysis of Results

1) Extraction - Part of Speech: As noted in section IV-C1, *POS* has a huge jump in correctness between training on the first couple tweets, after which the returns are minimal. We attribute this to the fact that most hashtags are nouns, as we hypothesized in section III-B. Looking at the models trained, nouns and the like (plural, proper, etc.) have by far the highest scores. Thus by training on a few tweets, our model quickly learns to suggest nouns as the most likely hashtags.

Since there are multiple nouns per tweet, many of the higher scored hashtag predictions are nouns. Our training model does not really take effect until after the first few predictions. At that point the order begins to matter. However, since this is an extractive method, as we begin to predict 10 or more hashtags, the Naive Precision is extremely high. We attribute this to the fact that tweets, limited to 140 characters, do not have an excess of words. After a certain point, it seems that we are just suggesting every word as a possible hashtag.

2) Generative - HF and HF-IHU: Unfortunately *HF-IHU* actually performs fractions of a percent better than our modified *HF* across all training data set sizes. While this difference isn't significant, it also means that our modifications to *HF-IHU* did not improve the approach, but instead made it perform slightly worse. This can possibly be attributed to the addition of the itf weight, which may have discounted hashtags from popular words too much resulting in the *HF* metric favoring incorrect hashtags. Another possible explanation is that there could be a relatively small amount of entropy in the hashtags of the data set. Since we iterated through a dictionary for hashtags to collect our tweets, uncommon hashtags that would have plagued the original *HF-IHU* would not have been tested on.

3) Iterative Testing Comparisons: Here we discuss the results pointed out in section IV-C1. As expected, due to the generative nature of *HF* and *HF-IHU*, training on a small corpus understandably yields lower accuracy. This is because with fewer tweets the pool of hashtags to choose from is just smaller when it comes to generating predictions for a tweet. Training on more tweets yields better results. As explained in the previous section IV-D1, *POS* yields a steadily high Naive Precision rate across the board.

Our combined *POSHF* approach seems to have a significant advantage when we train on a smaller corpus. This is due to the fact that *POSHF* exhibits both extractive and generative qualities. The fewer tweets *POSHF* trains on, the less confident the *HF* part of the system is in its predictions. Thus those scores are lower. However, the *POS* part is more confident in its scores, consistently (and often correctly) recommending nouns as hashtags. However as *POSHF* trains on more tweets, *HF* gains more confidence in its predictions and the scores being to match and slightly eventually outweigh that of the *POSHF* predictions. Understanding this, it is expected that *POSHF*, *HF*, and *HF-IHU* begin to converge as we train on a bigger corpus.

4) Tenfold Cross Validation Testing Comparisons: To reiterate, for each method we trained on 9 parts and tested on 1 part ten different times as explained in section IV-B3. This resulted in an accurate look at the performance of each method. Looking at Fig. 9, we see that *POS* and *POSHF* steadily get higher scores as k increases while *HF* and *HF-IHU* seem to decline as k increases.

Remembering our definition of recall (equation 7), these results imply a few things about the methods tested. For *POS*, it implies that nouns typically are hashtagged. This approach simply might not choose the correct one as the first suggestion, but its increasing score as more hashtags are suggested indicates that nouns are indeed hashtagged quite often.

HF and *HF-IHU* have fairly good recall for $k = 1$. This implies that their first suggestion is often correct. However, the decreasing scores suggest that the next few suggestions are not correct as often. This could be due to the nature of our corpus. Each tweet was chosen specifically because it contained a single hashtag. All the extra hashtags were arbitrary and only included miscellaneously. This means that these methods did not have enough data to train on when attempting to suggest hashtags past the first, most-likely one.

Finally, *POSHF* gains the best of both underlying methods, inheriting the good first prediction from *HF* while getting good extra guesses from *POS*. The performance in recall shows just that.

5) High Evaluations: Compared to previous evaluations, our scores are unexpectedly high. In the original *HF-IHU* paper, the recall scores are well below 50% for all the presented algorithms. However, our implementation of *HF-IHU* seems to do fairly well when tested on the corpus. We believe this could be the result of several factors. First, Naive Precision as presented in equation 5 is a more lenient form of traditional precision (equation 4). Additionally, the way we compiled our training and testing corpus resulted in a hashtag set with low entropy. In this case, it would be about 5000 hashtags plus other hashtags that the tweets containing the searched words had. In training and testing on a closed set of hashtags, we are guaranteed that "correct" answers exist in the pool of hashtag candidates.

V. Future Work

One of the short-comings of *HF* and *HF-IHU* is that they can only predict hashtags that they have been trained on. If we trained on a certain set of hashtags, but then tested on a separate corpus that contained completely different hashtags, the algorithms will often suggest incorrect hashtags because the "correct" hashtags would not be in the pool of candidates. Additionally, we believe that some of our results may be skewed due to the nature of our corpus collection. We hope to

test on tweets randomly collected from Twitter. This would allow us to get good breadth and not target any specific hashtags. With such a corpus, we would be able gain a better understanding of how well *HF* and *HF-IHU* really compare to *POS*.

A major feature of our *POS* approach is that training on 5 tweets yields very similar results to training on 100 tweets, as depicted in Fig. 8. Although not mentioned, Tweebo Parser actually has confidence values given along with the POS-tags suggested [6]. We could utilize these confidence values to get more fine-grain scores than the current method of just taking the POS-tag scores.

One feature we experimented with but did not actually document was dealing with duplicate hashtag suggestions. Some words appear multiple times in a tweet and as such get tagged with multiple scores by *POS*. In our current algorithm, we simply take the highest score. However, we hope to find a way to account for duplicates such as adding the scores and weighting by some derivation of the number of duplicates. However, we want to avoid simply taking the an average, as a score of 1 and .5 would result in an end score of .75, which is actually less than the original maximum score. Thus, we hope to experiment and find a good way to weight duplicate suggestions.

In our current approach, the POS-tagger exists completely independently of the *HF* side of the approach, and the results are only combined at the very end. A possible future avenue is moving the merging of the two approaches to an earlier step where the POS-tags can be used as an additional feature or a weight when scoring the hashtags associated with a term.

VI. Conclusion

In this paper we have presented two different hashtag recommendation systems: one extractive and the other generative. Our experiments and analysis have shown that while both are effective at recommending hashtags independently, an ensemble approach that combines the two systems performs even better– achieving a precision accuracy of almost 90% when asked to find the top 3 most likely hashtags. While *POSHF* seems to perform similarly to the existing recommendation system *HF-IHU*, *POSHF* really shines when trained on a smaller corpus as the *POS* half of the ensemble solves the "out-of-vocabulary" hashtag problem that *HF-IHU* faces when sparsely trained.

The way that we as humans connect with one another is constantly evolving and growing. Communication forms in this increasingly technology-oriented world are becoming more succinct, and more social networks are utilizing hashtags to categorize this data. We hope this recommendation system can make but a small dent in helping make data more organized and accessible.

References

[1] P. Sneiderman. (2014). *Johns Hopkins researchers go local with their Twitter flu tracking efforts*. [Online] Available: http://hub.jhu.edu/2014/03/18/twitter-data-flu-tracking-new-york

[2] *Using hashtags on Twitter*. [Online] Available: https://support.twitter.com/articles/49309-using-hashtags-on-twitter

[3] E. Otsuka, S. Wallace and D. Chiu. (2014). *Design and Evaluation of a Twitter Hashtag Recommendation System*. [Online] Available: http://mathcs.pugetsound.edu/ dchiu/Papers/otsuka-IDEAS2014.pdf

[4] *How to Use Tags*. [Online] Available: https://www.tumblr.com/docs/en/using_tag

[5] T. Li, Y. Wu and Y. Zhang. (2011). *Twitter Hash Tag Prediction Algorithm*. [Online] Available: http://www.worldcomp-proceedings.com/proc/p2011/ICM3338.pdf

[6] Carnegie Melon. *Tweet NLP*. [Online] Available: http://www.ark.cs.cmu.edu/TweetNLP/

[7] E. Zangerle, W. Gassler and G. Specht. *Recommending #-Tags in Twitter*. [Online] Available: http://ceur-ws.org/Vol-730/paper7.pdf

[8] Sirahi. (2010). *Retrieval by Content*. [Online] Available: http://www.cedar.buffalo.edu/ srihari/CSE626/Lecture-Slides/Ch14-Part1-Precision-Recall.pdf

[9] Scowl. (2014). *12Dicts Introduction*. [Online] Available: http://wordlist.aspell.net/12dicts-readme/

[10] *The Porter Stemming Algorithm*. [Online] Available: http://tartarus.org/martin/PorterStemmer/index-old.html

[11] *Stop-Words*. [Online] Available: https://code.google.com/p/stop-words/

[12] *Twitter4J*. [Online] Available: http://twitter4j.org/en/index.html

[13] F. Goden, V. Slavkovikj and W. De Neve. *Using Topic Models for Twitter Hashtag Recommendation*. [Online] Available: http://www.academia.edu/4601149/Using_Topic_Models_for_Twitter_Hashtag_Recommendation

[14] *Developer Agreement & Policy*. [Online] Avaliable: https://dev.twitter.com/overview/terms/agreement-and-policy