Hashtag Similarity based on Tweet Text

Ryan Cebulko
cebulko.r@husky.neu.edu

William Silva
silva.w@husky.neu.edu

Introduction

Since its inception, Twitter has grown into a social media powerhouse that sees hundreds of millions of monthly users [1]. With so many diverse groups using the service, it is important to consider what sort of content is being tweeted out by these different groups and how they may be related. For example, some seemingly innocuous hashtags may share a high degree of content with hashtags promoted by hate groups and certain ideologies. Thus, this paper considers the problem of how similar hashtags are based on the content of the accompanying tweets. Using a large dataset and varying models, we analyze the most popular hashtags in an effort to determine if two theoretically unrelated hashtags actually have similar tweet contexts. Specifically, given two hashtags, we output a single numerical value from 0 to 1 stating how similar the two hashtags are, with 0 being not similar at all and 1 being the same hashtag. For example, #cat and #unitednations would likely have a similarity closer to 0 due to the topics being completely unrelated. However, #cat and #cute would likely have a similarity closer to 1 theoretically because of the Internet’s penchant for felines. Additional uses of knowing hashtag similarity could include the generation of suggested hashtags for a new tweet as well as identification of clusters of hashtags.

Related Work

Given the popularity of Twitter and the ease of obtaining large sets of tweets, it is no surprise that there is some existing work related to hashtag similarity. One example of this is the work posted by Paul Mineiro on the Machined Learnings blog [2], which used term frequency vectors to determine which hashtags were essentially synonyms. While it is interesting to see synonymous hashtags, our work here goes further by
using a multitude of different models to determine even if hashtags are just somewhat similar.

Other work we spotted included that of Github user *linguanchen*, who posted his or her hashtag similarity work on Github [3]. They took a dataset consisting of 1 million tweets and determined hashtag similarity through co-occurrence counts. The results they posted only indicate that most hashtags are similar to #fb, and we hope to produce more expansive results than this.

Lastly, we found another Github user with username *yyl* who also posted their project on Github which attempts to determine the similarity of two hashtags [4]. Using tf-idf techniques, they determine how similar six hashtags are to one another. This project in particular is probably most similar to the work in this paper, but we have chosen to use the top 1000 hashtags with millions of tweets at our disposal rather than using just the six hashtags with 400 tweets each that the user *yyl* analyzed.

Data

For the work in this paper, we wanted to use a data set that was large enough to get accurate results while not being so large it would be impossible to train models on the data in a reasonable amount of time. After some deliberation, we decided that the Twitter datasets provided by the ArchiveTeam group [5] would be a good fit. We ended up using the “ArchiveTeam JSON Download of Twitter Stream” for June 2017 [6]. This dataset includes JSON data grabbed from the Spritzer version of the Twitter stream. Spritzer is a random ~1% sample of the full 100% Twitter stream [7]. Unfortunately, not all tweets will have hashtags or be suitable for our project, but this dataset was 45 GB and contained over 124 million tweets so we were certain it would contain enough tweets suitable for our purposes.

Methodology

Data Preprocessing

Unsurprisingly, our data needed a massive amount of filtering and preprocessing before it was ready to be given to any of our models. Some tweets were marked as deleted and had no text, so they could not be used. Additionally, significant numbers of tweets used non-Latin characters such as tweets in Arabic and the use of emojis. To simplify the problem, we decided to only handle Latin characters. Another issue was the inclusion of URLs and user “mentions” in tweets. Given that virtually no semantic meaning can be retrieved from a URL or usernames in mentions, we filtered these elements from tweets. Finally, the hashtags themselves were removed from each tweet as the goal of the project was to determine how similar hashtags were based on the content of the associated tweet.
At this point in preprocessing, we had mostly suitable input for models. However, we found a large portion of the tweets were in Spanish and other non-English languages. Suppose two tweets discuss “cats,” but one is in English and the other is in Spanish. Without a translation, there is no way to know that these tweets are semantically related. For this reason, we attempted to remove all non-English tweets using Python’s “langdetect” library, which is capable of determining which language a tweet is in [8] with questionable accuracy. Thankfully, the majority of non-English tweets were filtered out using this method.

After filtering to a reasonable set of tweets, we now had many thousands of hashtags to consider. If we wanted to retrieve the similarity of N hashtags, that would involve computing $N^2$ relationships between the hashtags. For a large $N$, this would be extremely time-consuming, so we pruned these hashtags to the top 1000 most popular hashtags. This operation left us with approximately 3.9 million individual tweets with over 5.5 million hashtag usages. We do not need to separate this data into a training set and a test set because we are trying to predict existing relationships among hashtags in the current data.

Evaluation Metrics

We had two metrics for evaluating our models that each considered different aspects of similarity: co-occurrence and human-based semantic similarity ratings.

Co-occurrence Matrix

One clear ground-truth we have for hashtag similarity is how often these hashtags actually occur together in the same tweet. A co-occurrence matrix shows how often each hashtag co-occurred with each other hashtag. Using this information, one can generate clusters showing which hashtags appear together.

![Co-occurrence Matrix Diagram](image)

Figure 1: One cluster produced from our co-occurrence data
The cluster above shows that many technology-based hashtags appear together. While hashtags occurring together is a good indication that they are similar, it is possible that hashtags that rarely co-occur are actually quite similar, so this metric is somewhat flawed but it is the only ground-truth that can be extracted directly from the data.

To get the similarity of hashtag $h_i$ and hashtag $h_j$ using co-occurrence, we take the average of the ratio that $h_i$ appeared in $h_j$’s tweets and the ratio that $h_j$ appeared in $h_i$’s tweets. For our metric, we use the Pearson correlation coefficient of all of the calculated co-occurrence similarities to the model’s.

**Human-supplied Data**

A potentially much better approach to evaluating our models would be comparing it to what humans consider to be similar. To do this, we took 10 hashtags from our dataset that we thought would be most interesting for real people to rate the similarity of. These hashtags were #teenchoice, #veranomtv2017, #rt, #dogs, #news, #maga, #trump, #uk, #brexit, and #cute. Then, we took five random and representative tweets from each of these hashtags and removed the hashtags from the tweets so that the people being surveyed would only see the tweet content similarly to our model input. While we initially wanted to use Amazon Mechanical Turk [9], we ended up asking our friends and family to fill out short surveys describing how similar they thought each group of tweets was to each other group on a scale of 1 to 10. The groups were labeled with numbers rather than the hashtags so that the hashtags would not influence how similar they thought the tweet text was.

Once we received the survey data back, we took the average for each group comparison and called that the similarity between those two hashtags. These similarities were compared to the similarities for the same hashtags for each of our models. Once again, we used the Pearson correlation coefficient as metric for how successful our model was, with 0 correlation being unsuccessful and 1 correlation meaning the model mimicked the human answers. For the remainder of this paper, we will refer to this metric as the “human-survey correlation.”

**Differences between our Evaluation Metrics**

We found that the co-occurrence similarities we generated correlated with the human-survey data by a Pearson correlation coefficient of 0.6. While this shows significantly high correlation, one would think what humans think are similar would be what actually occurs on Twitter. This inconsistency could just be the result of the small set of hashtags we chose for humans to compare. If we chose a larger list of hashtags to compare, it is possible that the correlation between our metric data would increase as we have more sample data. The fact that our metrics are not perfectly correlated means it should not be possible to get very high correlation for both metrics at the same time.
Models

Naive Bayes - Baseline

Our baseline model was Naive Bayes with Laplace smoothing [10]. We used Python’s NLTK library [11] to tokenize each of the tweets into words for the model. To compute hashtag similarity for hashtag $h_i$ and hashtag $h_j$, a random sample of 20 hashtags is taken for each tweet. For each of $h_i$’s tweets, we predict the probability that the tweet belongs to $h_i$ and vice versa. The average of all these probabilities is the hashtag similarity. The logic behind this is that if the Naive Bayes model has trouble discerning between two hashtags, it is possible that they are related in some way or potentially use similar wording.

<table>
<thead>
<tr>
<th>Co-occurrence correlation</th>
<th>0.287</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human-survey correlation</td>
<td>0.325</td>
</tr>
</tbody>
</table>

Figure 2: Evaluation results of Naive Bayes Classifier

While not excellent, having significant positive correlation for both of our metrics for such a simple model is a great baseline.

Sequence Matcher

Following our baseline, we used the Sequence Matcher class from Python’s difflib library [12], which is used to determine text similarity. Given that we want to determine how similar hashtags are based on their tweet text, this tool was very helpful for the work in this paper. The SequenceMatcher algorithm uses a pattern matching algorithm similar to that of Ratcliff and Obershelp’s pattern matching paper [13]. However, this algorithm adds in the idea of “junk” elements with patterns being the “longest contiguous matching subsequence” that do not contain these elements [12].

To compute similarity of hashtag $h_i$ and hashtag $h_j$ using this method, our Python script takes a random sample of 30 tweets for each hashtag. These hashtags are then appended together, and we run the Sequence Matcher algorithm to compare these two sets of hashtags. The algorithm outputs the similarity on a scale of 0 to 1, which is exactly the output we want for the hashtag similarity.

One final note for this algorithm is that it is particularly CPU-intensive. In order to get results within a reasonable timeframe, we used methods provided in the library that let you get a faster upper bound on the hashtag similarity. We ran the algorithm with both “real_quick_ratio,” which returns an upper bound “very quickly,” and “quick_ratio,” which provides an upper bound “relatively quickly”.
<table>
<thead>
<tr>
<th></th>
<th>real_quick_ratio</th>
<th>quick_ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Co-occurrence correlation</td>
<td>0.0575</td>
<td>0.067</td>
</tr>
<tr>
<td>Human-survey correlation</td>
<td>0.521</td>
<td>0.428</td>
</tr>
</tbody>
</table>

Figure 3: Evaluation results of Sequence Matcher

As one can see, there was very little correlation with co-occurrence data while there was significantly high correlation with the data provided by the human-survey. The high correlation with the human data is expected because this algorithm typically outputs similarities that “look right” to people according to its documentation [12]. However, the co-occurrence correlation is low, which seems to indicate that this model is saying that some number of hashtags which are semantically similar do not co-occur together.

Lastly, the more accurate method “quick_ratio” seemed to fare worse with human-survey correlation. This could be just random chance based on the random samples that were picked. Another explanation is that the “quick_ratio” over-fitted and was better able to differentiate between tweets that were somewhat semantically similar.

**Random Forest Classifier**

We use the RandomForestClassifier from Python’s sklearn library to determine hashtag similarity [14]. We opted for the default configuration, which trains the forest with 10 trees and nodes are expanded until all leaves are pure or until all leaves contain less than 2 samples. We use many models that classify in our work because it is our hypothesis that if a classifier has difficulty discerning the tweets of two hashtags, the hashtags are more similar. Random forest classifiers are an ensemble method that use several decision trees to arrive at a result. To train the random forest classifier, we gave took 25 random tweets for each hashtag and produced feature vectors for each. We decided on using a relatively simple set of features for tweets, which merely involved word counts using a fixed vocabulary. To prevent our feature vectors from being massive, all words that were used less than 20 times in our dataset were replaced with the unknown flag symbol UNK. The random forest classifier was given the feature vectors along with its hashtag classification for training.

After training, to determine the similarity of two hashtags $h_i$ and $h_j$, we take a random sample of 5 tweets for each hashtag and create feature vectors for them. Then, we have the random tree classifier attempt to predict the class probabilities for all the feature vectors. We use the “predict_proba” method for the RandomForestClassifier class which explicitly provides the class probability for every class for a given tweet [14]. After we get these results, we get the average probability
that $h_i$’s tweets are $h_j$’s and vice versa. We then take the average of these two probabilities to obtain the hashtag similarity.

<table>
<thead>
<tr>
<th>Co-occurrence correlation</th>
<th>0.566</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human-survey correlation</td>
<td>-0.070</td>
</tr>
</tbody>
</table>

Figure 4: Evaluation results of Random Forest Classifier

Clearly, the model does well with co-occurrence correlation but has significant failures regarding human-survey correlation. It is possible that using different features for tweets would lead to better outcomes. One of the clusters produced by our random tree model is shown in the figure below.

![Cluster produced by Random Tree Classifier](image)

Figure 5: Cluster produced by Random Tree Classifier

#job, #jobs, and #hiring are all clearly closely related, so the output of our random tree classifier appears to be quite reasonable.

**Multilayer Perceptron**

We also used the Multi-Layer Perceptron (MLP) from the same Python sklearn library [15]. We configured our MLP to use 6 hidden layers with the layers having (300, 200, 100, 50, 25, 10) neurons in each layer in that order. Using this number of layers with so many nodes led this particular model to take over 12 hours to get hashtag similarities. The features used for this were the same features as given to the random forest classifier.

To get the hashtag similarities for this model, we used the same technique as described for the random tree classifier with “predict_proba”.

<table>
<thead>
<tr>
<th>Co-occurrence correlation</th>
<th>0.459</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human-survey correlation</td>
<td>0.202</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-------</td>
</tr>
</tbody>
</table>

Figure 6: Evaluation results of Random Forest Classifier

With the MLP, we get some of the best results overall with a relatively high co-occurrence along with a smaller but sizable human-survey correlation. This makes sense given the power of neural networks and the amount of time put into training the MLP compared to other models.

**Linear SVC**

We used the LinearSVC from Python’s sklearn library as a model with its default configuration [16]. We used the LinearSVC mostly because a standard SVC does not scale to datasets with more than 10000 samples [17], and our dataset is certainly significantly larger even when only using small samples sizes for our 1000 hashtags. LinearSVC scales better for larger sample sizes. The input features were the same as we gave to the random forest classifier.

To obtain hashtag similarities, we follow the same steps as we did for both the random tree classifier and the MLP. However, the “predict_proba” method is replaced with the “decision_function” method, which provides the same exact output.

<table>
<thead>
<tr>
<th>Co-occurrence correlation</th>
<th>0.009</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human-survey correlation</td>
<td>0.052</td>
</tr>
</tbody>
</table>

Figure 7: Evaluation results of Random Forest Classifier

With both metrics being very close to 0, it is apparent that our use of Linear SVC was a complete failure. Given more time, we would modify the configuration of the Linear SVC to determine if that would improve its results at all.

**Ensemble - Weighted Averages**

For one final model, we attempted to combine the best of both worlds with some of our top-performing models. We decided to use a weighted average between the results of the random forest classifier and the Sequence Matcher. For each hashtag similarity, we create new hashtag similarity by multiplying the similarity produced by the random forest classifier by some alpha. We add this to the result of multiplying the similarity produced by the Sequence Matcher by (1 - alpha) and then divide this sum by 2. After much trial and error, we found that the alpha that produced the best results was 0.975.

| Co-occurrence correlation | 0.518 |
By a significant margin, the ensemble model gives us the best results so far. As we hoped for, we have gotten good results thanks to taking advantage of the results produced by two very different models.

### Analysis

Overall, we saw varying results depending on the model that was used. However, the only model which seemed to give us good human-survey correlation results was the Sequence Matcher (as well as the ensemble model, which used the Sequence Matcher results). All other models had much better results for our co-occurrence correlation metric. This is likely a result of the features given to train these models rather than a failing of the models themselves since models like MLP are capable of solving many problems. In the end, maximizing the human-survey correlation is likely the top priority, as our original goal coming into this project was to determine hashtag similarity based purely on tweet text. If humans say that the text is similar, then it means the hashtags must be similar. Just because hashtags do not co-occur does not mean that they are not similar. The main goal of our work was to find hidden similarities of hashtags that may not always be used together.

For this reason, the Sequence Matcher and ensemble model were our most successful models overall. If we want to determine if two hashtags have some semantic relation, it is most likely best to use these models.

### Examples of Hidden Similarities

We can use the results of our models to determine if there are hashtags that rarely co-occur but are semantically similar. To do this, we will take hashtags that have high similarities according to the Sequence Matcher and low rates of co-occurrence according to the co-occurrence matrix. We have provided some examples in the figure below.

<table>
<thead>
<tr>
<th>Human-survey correlation</th>
<th>0.463</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 8: Evaluation results of Random Forest Classifier</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>#theresamay, #trumpcare</th>
</tr>
</thead>
<tbody>
<tr>
<td>#parisagreement, #queenspeech</td>
</tr>
<tr>
<td>#inspiration, #islam</td>
</tr>
<tr>
<td>#energy, #finance</td>
</tr>
<tr>
<td>#democrats, #lgbt</td>
</tr>
</tbody>
</table>
For all of the hashtag pairs in the figure, there were extremely low rates of co-occurrence and similarities above 0.935. Interestingly, most hashtag pairs with these qualities were political in nature while the top 1000 hashtags were far less political in nature in general. These results appear to be very reasonable. The first row in the table shows that tweets discussing Theresa May, the UK’s conservative Prime Minister, are similar to discussions of the Republican healthcare bill in the United States. We also see #blacklivesmatter and #lgbt are semantically similar, which is no surprise as both groups are related to marginalized groups and have plenty of relevant activism.

**Conclusion and Future Work**

Overall, we believe that our project was mostly successful. We found several useful models that were good at determining hashtag similarity using different metrics, and a simple ensemble model combined our best models to create a model doing well in both of our metrics.

Some possible future work could be done using more data from the dataset or using a larger dataset. We only used the most popular 1000 hashtags from this dataset, so much of our original given dataset was filtered out. Additionally, many of the hashtags we saw were relevant to events that happened during the time our dataset was created, June 2017. It would be interesting to take data from another time period and see if different hashtag similarities are found. Lastly, many of our models were trained only using a relatively small random sample for each of the 1000 hashtags we observed. Despite this, each model still took many hours to produce our desired results. If we had more time or more computing power, we could have trained our models with more data and produced better results.

**Acknowledgements**

We would like to thank Professor Lu Wang for teaching an interesting Natural Language Processing class as well as our former partner Andrew Singley who worked on a portion of this project with us.

**References**


