Predicting HashTag For Social Media Posts And Suggesting Similar Profiles

Sai Nikhil Dogiparty
Rohan Jahagirdar

Northeastern University
dogiparty.s@husky.neu.edu
jahagirdar.ro@husky.neu.edu

ABSTRACT

This research is about recommending Hashtags for a post on Social Networking Sites (like Twitter, etc). With the growing influence of microblogging services, the number of users on these platforms has exponentially increased. The growing content on these platforms present an interesting problem for finding posts with a specific theme or content. To make this lookup easy, social media uses meta tagging by using hashtags (#). This dynamic tagging by users sometimes creates multiple hashtags for similar theme which spreads the data across different tags making it difficult to aggregate posts. In this work we instead leverage the wealth of information available from users: namely the posts and other metadata information. By building a knowledge graph using machine learning algorithms, we predict hashtags for a post and recommend likeminded users to follow. We explore two ways of combining these heterogeneous features into a learning framework: 1. Finding the closest hashtag to a user's post using word embeddings; and 2. Finding similar posts and users using SpaCy; an industrial level NLP.

INTRODUCTION

The world of social media websites and applications enable users to create and share content to participate in social networking. The combined effect of data digitalization, social media, big data and wireless mobile telecommunications has facilitated a fertile field of investigation. The easy storage of large scale information on human interaction in the digitized format has made microblogging industry a good platform for researchers to study human behavior. The possibility of freshness and ubiquity of data generated by these applications makes it profitable to study them to be able to ask questions. With the large stream of data comes additional problems for aggregating information and finding out common themes. While meta-tagging posts solves the problem of lookups and grouping similar themes to a degree, the ambiguity in the format and their dynamic usage makes it difficult for end users to find the perfect hashtag related to the post. Being a frequent user of these platforms, we too find the overhead of coming up with relevant hashtags a cumbersome task. This problem interested us, and we wanted to provide an appropriate solution to create an impact in the society. While this is a problem across all platforms such as Facebook, Twitter, Instagram et al our primary focus for this research is on Twitter.

We keep our approach platform independent, so the same approach can be applied to different platforms apart from Twitter as well.

Following is an example of hashtag ambiguity, where same theme has different hashtags:

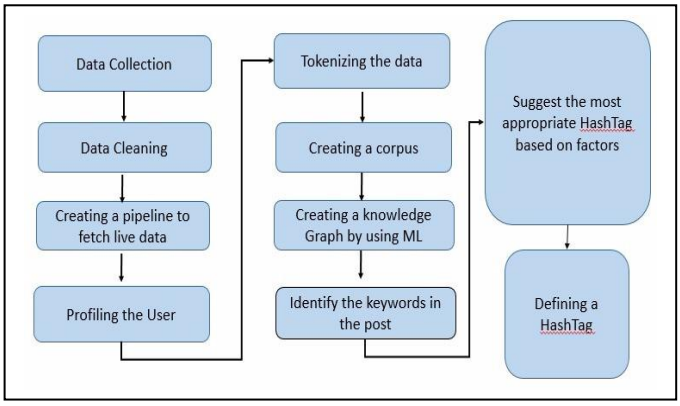
A similar problem that can be solved by our approach of word embedding is finding similar posts and people and provide connection requests.

In this paper, we develop a method for recommending relevant hashtags to users in real-time. Hashtag recommendation has two main challenges that set it apart from traditional document tag recommendations. First, the content of a Tweet is very short and often includes abbreviations, misspellings, and incorrect grammar. The limited number of words in a Tweet makes traditional document classification techniques such as TF-IDF (Term Frequency and Inverse Document Frequency) ineffective because of the TF=1 challenge which means that words are rarely repeated in a Tweet. This makes it difficult to determine which words are most important to the meaning of the Tweet. The second challenge is the sheer volume and speed at which data is produced. Tweets are produced constantly so the incoming Tweets need to be treated as an infinite length stream. Our approach to recommendation attempts to handle these challenges with a variety of preprocessing techniques and a recommendation model based on the popular context similarity algorithms Word2Vec and SpaCy. Word2Vec is a particularly computationally-efficient predictive model for learning word embeddings from raw text.

It comes in two flavors, the Continuous Bag-of-Words model (CBOW) and the Skip-Gram model. These models are shallow, two-layer neural networks that are trained to reconstruct linguistic contexts of words. Word2Vec takes as input a large corpus of text and produces a vector space, typically of several hundred dimensions, with each unique word in the corpus being assigned a corresponding vector in the space. Word vectors are positioned in the vector space such that words that share common contexts in the corpus are in close proximity to one another in the space. This model has been built and used to predict HashTag's for user's post. SpaCy features neural models for tagging, parsing and entity recognition. A novel bloom embedding strategy with subword features is used to support huge vocabularies in tiny tables. Convolutional layers with residual connections, layer normalization and maxout non-linearity are used, giving much better efficiency than the standard models. This model is used to identify similar users based on post's content.

METHODS AND CODE

Our pipeline to achieve our goals is as:



Our approach to the project is to start with real time live datasets. For each location we fetch the popular topics called trends. A trend on Twitter refers to a hashtag-driven topic that is immediately popular in a particular place at particular time. To fetch trends, we began by accessing the trends based on locations and then scrapping live tweets for these trends using Twitter API and the Tweepy library for making requests to the API.
For keeping the data fresh, we run the code periodically to fetch the latest trends and tweets for the trends.

We have a Cronjob (Scheduler that runs a script at regular intervals) that keeps fetching live data at periodically. The Twitter API has a restriction on the amount of information that can be accessed every 15min window. The Cronjob makes sure to not cross the limit.

Accessing Tweepy API:

```
In [134]: consumer_key = 'CUSUMER_KEY'
          consumer_secret = 'CONSUMER_SECRET'
          access_token = 'ACCESS_TOKEN'
          access_token_secret = 'ACCESS_TOKEN_SECRET'

          auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
          auth.set_access_token(access_token, access_token_secret)
          api = tweepy.API(auth,wait_on_rate_limit=True)
```

Fetching the trends and the hash-tags from the trends for Boston:

```
In [160]: trends = api.trends_place('Boston')
          hash_trends = []
          for trend in trends[0]['trends']:
              if(trend['name'].startswith('#')):
                  hash_trends.append(trend['name'])cab)
```

Once we get the trends, we fetch all the tweets for the trend and to dataset. We do this for all locations:

```
In [6]: for tweet in tweepy.Cursor(api.search,q=trend,count=100,
                                   lang="en",
                                   since="2018-01-01").items(1000):

            if trend.lower() not in tweet.text.lower():
                csvWriter.writerow([tweet.created_at,
                                    text.encode('utf-8'), tweet.place])
```

Next, we pickup all the csv files from the relative data folders to create a corpus.

```
In [179]: allFiles = glob.glob("Data/*.csv", recursive=True)
          frame = pd.DataFrame()
          # To store filenames
          filenames = []
          """Looping all the files and creating a data frame"""
          for file_ in allFiles:
              df = pd.read_csv(file_, encoding = "ISO-8859-1")
              filenames.append(file_.split("\\")[-1:][0].split(".")[0])
              frame = pd.concat([frame,df[df.columns[1]]])
```

Next, we preprocess the data.

Preprocessing: As Natural language text requires lot of preprocessing to clean data, remove stop words and noise.

We have created modular functions which handle these unwanted noises in the corpus. These modules can be reused in future projects.

```
In [181]: def preprocessing(frame):
              frame = frame.apply(lambda x: x.astype(str).str.lower())
              frame = frame.apply(lambda x: x.astype(str).str.replace('b\'','\''))
              return frame
```

Function to build stopwords to remove them

```
In [184]: def build_stopwords():
              stop_words = (stopwords.words('english'))

              list_remove = ['.','/','b\'','b\"','xe2','x80','xa6','x99t','x99s','x87','xb4','xaa',
                             'x9a','x9b','x93','xb2','x8c','x9c','xb5','xab']

              stop_words = stop_words +  word_tokenize(string.punctuation) + list_remove

              return stop_words
```

Knowledge Graph: We vectorize the corpus using Word2Vec word embeddings which gives us a 1000 dimensional vector for each word. The similarity between the entities is found using the cosine similarity between them. Cosine similarity is a measure of similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them.

We use this approach after validating different distance measuring techniques and chose this because it keeps into account the direction of the vector, something that is important for establishing word similarity.

Building a word2vec model

```
In [187]: import timeit
          start_time = timeit.default_timer()

          model = Word2Vec(corpus, size=1000, window=3, min_count=1, workers=4)

          elapsed = timeit.default_timer() - start_time

          print(elapsed)
```

```
0.6757893854519352
```

For validating the model, we tested the similarity:

```
In [173]: model.similarity('utd','#facup')
Out[173]: 0.983841470513196
```

Suggesting Connections

To aggregate similar users and suggest connections we used SpaCy. We chose SpaCy for its excellent performance for vectorizing entire posts and finding out similarity between posts.

To do this, we find out the cosine similarity between the user input text against all the tweets in our dataframe. Comparing all similar tweets, we find out the tweet most similar to the user tweet.

We import SpaCy and the Model for English Language Vocabulary, Syntax and entities en_core_web_sm. We load all the data for the tweets we want to find similarity of the user tweet against.

```
In [59]: import spacy

         spacy.load("en")
         import en_core_web_sm
         nlp = en_core_web_sm.load()

         location_df = pd.DataFrame()
         locationFiles = glob.glob("Data/*/*.csv", recursive=True)

         frames = []
         frame_words = []

         for file in locationFiles:
             df = pd.read_csv(file, encoding = "ISO-8859-1")
             location_df = pd.concat([location_df,df[df.columns[1]]])

         for index,frame in location_df.iterrows():
             frames.append(nlp(frame[0]))
             frame_words.append(frame[0])
```

The call to load spacy returns a Language object which contains all the components and data needed to process the data.

Using this Language Object we process our data. This processed Model will use the Language object to find similarity between posts.

```
user_input_nlp = nlp(input)
mostsimilarword = ""
mostsimilarword_similarity = 0
reference = {}
index = 0

for idx, doc in enumerate(frames):
    similarity_of_word = doc.similarity(user_input_nlp)
    if (similarity_of_word > mostsimilarword_similarity):

        print(doc)
        similar_tweet = frame_words[idx]
        if "@" in similar_tweet:
            mostsimilarword_similarity = similarity_of_word
            mostsimilarword = similar_tweet
            reference = set([i[1:].rstrip(':') for i in similar_tweet.split() if i.startswith("@")])
```

Finding Out Most Talked About Organization Or Person

We dug deep into user posts to tag entities using Stanford NLP to be able to identify the most talked about people or organization and suggest it to users for them to stay updated of latest news.

```
In [110]: sentence = "Tim Cook are working at Google facebook Apple."

          print (ne_chunk(pos_tag(word_tokenize(sentence))))
```

```
(S
  (PERSON Tim/NNP)
  (ORGANIZATION Cook/NNP)
  are/VBP
  working/VBG
  at/IN
  (ORGANIZATION Google/NNP)
  facebook/NN
  Apple/NNP
  ./.)
```

RESULTS

With the idea to solve a real-world problem which everyone experiences while using social media platforms we validated the results from our Word2Vec model by predicting hashtags for a real-world tweet:

```
In [203]: input = 'man utd victory fa cup cup win tottenham football'
          HashTagSuggestion = []
          suggestions = getHashTagSuggestion(input)

          pprint.pprint (suggestions)

          [('#facup', 0.9790136429028582)]
```

Defining a HashTag - These words define what a particular HastTag means

We also validated the results from our SpaCy NLP model by running an unknown tweet against all tweets to find out the most similar tweet and recommend the profile to follow based on the similarity of the untagged tweet to the similar tweet from the corpus.

For a real user tweet, We can it through our SpaCy

```
In [60]: import operator

input = "Self employment is best!! I wnt my dghtr to learn 2 be a role model!"


user_input_nlp = nlp(input)
mostsimilarword = ""
mostsimilarword_similarity = 0
reference = {}
index = 0

for idx, doc in enumerate(frames):
    similarity_of_word = doc.similarity(user_input_nlp)
    if (similarity_of_word > mostsimilarword_similarity):

        print(doc)
        similar_tweet = frame_words[idx]
        if "@" in similar_tweet:
            mostsimilarword_similarity = similarity_of_word
            mostsimilarword = similar_tweet
            reference = set([i[1:].rstrip(':') for i in similar_tweet.split() if i.startswith("@")])

print("Similar Tweet: " + mostsimilarword)

top reference = reference.pop()
```

Model:

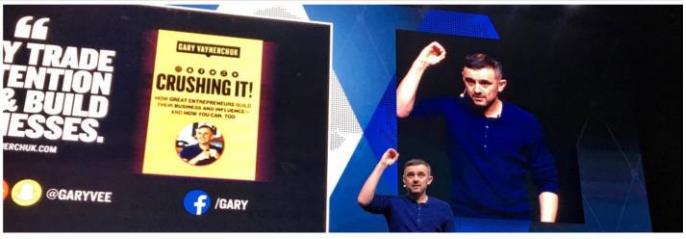
The results were the top most similar tweet and a profile suggestion based on the similar tweet:

```
Similar Tweet: b' #GaryVeeBreakfast @garyvee I am self-employed I still wear hand me downs I\xe2\x80\x99ve had my own place of
 business for nine years I\xe2\x80\x99m 34 and\xe2\x80\xa6 https://t.co/fFqP84Z2iv'

 You might like to befriend:
@garyvee
```

```
User Name:Gary Vaynerchuk
User Status:Family 1st! Entrepreneur 2nd. CEO of @Vaynermedia and @VaynerSports. Investor in Twitter, Snapchat, Uber, Venmo. 5X
NYTimes best . Die hard NYJets fan
User Location:NYC
https://pbs.twimg.com/profile_banners/5768872/1519958066

from IPython.display import Image
from IPython.core.display import HTML
Image(url= status.user.profile_banner_url)
```



Defining a HashTag

We also defined a hashtag based on what the posts have been referring about. We did this by selecting entities which are close to the hashtag in the knowledge graph of word-embeddings.
For eg - #FACup: Man Utd, Chelsea , Tottenham, Football where the words which define it.

CONCLUSION

In this paper we discussed the challenges that come up with exponential increase in social networking platforms and proposed solutions to overcome them. We introduced a set of embedding models that predict highly diverse and relevant hashtags for real-world tweets. We then showed how user metadata could be combined with their posts to suggest them to follow users with similar interests. We also showed how popular entities in the posts can be dug using entity recognition techniques. Particular care has to be taken when working on real world datasets rather than curated ones. We addressed the highly skewed hashtag distribution observed in our dataset by down sampling the more frequent hashtags by choosing only trending hashtags in particular locations. The resulting system is highly scalable and could be used in a number of applications.

REFERENCES

[1] Radim Řehůřek's Word2vec Tutorial, https://rare-technologies.com/word2vec-tutorial/
[2] Tomas Mikolov, Facebook Research Scientist https://research.fb.com/people/mikolov-tomas/ . Creator of Word2Vec model at Google
[3] SpaCy documentation, https://spacy.io/usage/#tests
[4] M. Hermann, D. Das, J. Weston, and K. Ganchev. Semantic frame identification with distributed word representations. In Proc. of ACL, 2014
[5] User Conditional Hashtag Prediction for Images. Manohar Paluri ,Facebook AI Research ,mano@fb.com .http://www.thespermwhale.com/jaseweston/papers/imagetags.pdf