Evaluation of Language Models Trained on Tweet Data

Zachary Bebbington zbebbington@umass.edu
Quan Bui qbui@umass.edu
Frick Shao fshao@umass.edu

1 Problem statement

The goal of our project was to evaluate how well a number of language models could emulate the writing style of multiple Twitter users. We felt that this was an interesting topic as it helps us understand the workings of language models. If we could find a language model capable of accurately emulating the way individuals write on twitter then we would be discovering a language model that is capable of two feats. For starters, to properly emulate the writing of any individual Twitter user the model would be required to learn and replicate the writing style of that individual. It would have to learn how the individual transitions between thoughts, the particular words that the individual uses, and how that individual incorporates Twitter specific strings into their writing such as @'s, #'s, and links. Additionally, the model would need to categorize the various subjects that the Twitter user is writing about. Tweets can be about anything, which means that in one post the user can be talking about the new dog they got. But in their next post they could be talking about politics or another heavily disconnected topic. Depending on how the model chooses to generate its output strings this could potentially lead to both of these topics getting mashed together, creating something that lacks any coherency. And because no two people write the same way it would have to accomplish both of these feats using the limited data set that is the users Tweet history.

2 What you proposed vs. what you accomplished

- ~~Collect data and create training/test sets~~
- ~~Find user created language models online~~
- ~~Train each language model 4 times, once per training set~~
- ~~Generate 100 outputs per model per training set~~
- ~~Humanly evaluate all outputs to rank the language models we used~~
- ~~Obtain perplexity for some of our language models to rank them empirically~~
- *Improve the performance of the top performing language model, trying to boost its human evaluation and perplexity*: We failed to do this as we underestimated how long it would take to get out language models working/train them as well as the time needed to humanely evaluate all of the output that we had gathered

3 Related work

In one system called Knowledge Based Natural Language Processing System, Jim Barnett, Kevin Knight, Inderjeet Mani, and Elaine Rich (1990) attempted to divide language into two kinds of information (Barnett, 1990). The purpose of their work was to train a model to learn about the knowledge and rules of a language. The first kind of information are the facts about the world. The second are linguistic rules, which are the rules used to understand a sentence. To the authors, it doesn't make sense to group certain objects into classes that potentially only have a very niche application. Instead, they try to define relations between groups using linguistic rules as a reference. As a result, we are no longer required to have an explicit rule for every combination of words. On the other hand, since this system is heavily based on knowledge of the world and rules, these must be accurate. If the system contains incorrect assumptions, then it may derive incorrect conclusions. This means that there must be an excessive amount of data for the system to train over. This

is pretty close to our work in the sense that both are trying to understand a "language", but we were also trying to replicate it.

A second work that applies similar concepts is by Elena Not, Massimo Zancanaro, Mark T. Marshall, Daniela Petrelli, and Anna Pisett (Not, 2017). Their purpose of their work was to create personalized postcards given a set of logs for a specific place, such as a museum. It is similar to our goal as given some information, we output a tweet that is stylized to resemble a specific person. However, there is a difference in that the postcard is not attempting to emulate a person, and instead is outputting information in a informal and natural way. The approach they utilize is shallow template-based text generation. This means that given a specific museum, the system will use a specific set of prepared sentences with gaps that are to filled with information from the provided logs. The sentences are assembled dynamically in order to make it sound more natural (4).

Another related work is iParticipate. It proposes the use of government data to get more people involved in local politics. Copious amounts of data are collected by the government such as building plans, meetings, and much more. Because of the huge amount of data, it is impractical to have someone go through it all. They propose we combine this data with the web through social media such as Twitter. To make it feasible with the large amounts of data, something like tweets can be auto-generated (Christoph Lofi, 2012).

Next, BSMA-Gen is a "synthetic social media data generator." Which is designed to generate timeline structures of social media. Such as a Twitter timeline. It's purpose is to build a realistic and flexible timeline for fake users to be used in any process that requires huge amounts of data (Zhou, 2014).

Spellbound is a twitter bot created by Wilson N. and Rajani N.generates multiple tweets using user location and past tweets, ranks them and replies back with the highest ranking to communicate like a human. (Nick Wilson, 2013)

Another related work is "A Context-aware Convolutional Natural Language Generation model for Dialogue Systems". It describes an approach to natural language generation using a convolutional neural network based Seq2Seq learning. (Sourab Mangrulkar, 2018)

O Poeta Artificial is a bot created to generate poems based on trends using trending tweets/topics. (Oliveira, 2017)

In another paper, Gathering and Generating Paraphrases from Twitter with Application to Normalization, the authors describe paraphrasing informal user-generated text like tweets on Twitter and the beneficial uses of it like phrase-based text normalization. (Wei Xu, 2013)

"Data-Driven Response Generation in Social Media" by Alan Ritter, Colin Cherry, and William B. Dolan describe an approach to generating responses to Twitter status posts using phrase-based Statistical Machine Translation and how it outperforms Information Retrieval. (Alan Ritter, 2011)

"How Noisy Social Media Text, How Diffrnt Social Media Sources?" is a paper relating to the realm of social media text (Twitter, YouTube comments, etc.) and explores the claim of how noisy it is. (Timothy Baldwin, 2013)

4 Dataset

Our data is comprised of tweets from four Twitter users. The Twitter users we looked at are two well known political figures, @BarackObama, @realDonaldTrump, and two active political posters, @adamcbest and @iheartmindy. We utilized Twitter accounts with strong political connections in hopes that the Tweets generated stay within the realm of politics. President Obama and Adam served as more left leaning data sets and President Trump and Mindy served as more right leaning sets.

What makes Tweets unique from other forms of text is that they contain hashtags, tweets, mentions, and links, for simplicity we will be referring to these four items as Tweet Attributes. In addition, tweets are limited to 280 characters to get a message across (with some exceptions of using multiple tweets that continue from each other). The character limit does bring in worries in how well a generated tweet can get across a coherent message.

Additionally, using links in the correct context may be difficult since the models will not know what these links are for, other than that they sometimes appear - especially for models at a character level. For example, the string https://t.co/kovcij4fwu is contained in a Tweet by Donald Trump. In the context of his Tweet the link makes perfect sense. If someone wants to support a certain cause or learn more about that specific

topic, then they should click the link. The model, though, will likely not recognize when it is appropriate to insert a link. It may decide to put a link in the middle of a sentence thus disrupting the flow of said sentence. This is obviously not desired, unless the person the model is being trained on has a tendency to do so.

We also have worries for other aspects of people's online language. Slang, text emoticons, and abbreviations are common: b/c, b4, lol, ftw, ggnore, etc.. Or even misspellings. And the model will be providing different probabilities for each phrase despite them meaning the same thing, i.e. ftw and for the win, or Masachcusetts vs Massachusetts. An ideal model would be able to handle issues such as misspelled words. For phrases like slang and abbreviations we decided it would not be too much of a problem, as it is a part of a user's writing style if they write like that.

Another potential problem in the data is that some people use non-words: owo, :),XD, 1337 emojis, etc.. Some users will use text like this in every tweet and the model might put them in outputs that do not make sense. For our dataset, we also have some non-English tweets. Obama tweeted out

```
usted y su familia merecen la
tranquilidad de saber que estn cubiertos
. el mercado de seguros abre maana.
https://t.co/9ircklrgzd.
```

For anything at a character level, these tweets might give us random combinations of English and Spanish words or at word level we might get an English sentence with an out of place Spanish word.

4.1 Data Statistics

Total number of tweets per user in the dataset will be listed below and Figure 1 will contain info on total number of hashtags, mentions, and links along side tweets.

```
@realDonaldTrump      Total Tweets: 2789
@BarackObama          Total Tweets: 2888
@adamcbest            Total Tweets: 1500
@iheartmindy          Total Tweets: 1500
```

The following figure is the count for unique words (figure 2).

As part of our evaluation, we'll be comparing the use of the twitter attributes (hashtags, mentions, and tweets). Figure 3 is the twitter attributes from the data set for each user. As opposed to figure 1, this graph counts the number of tweets that use each attribute at least once in it (so a tweet using one hashtag, two, three, etc.)

Figure 1: General info: Total Tweets used, Total number of mentions, hashtags, and links

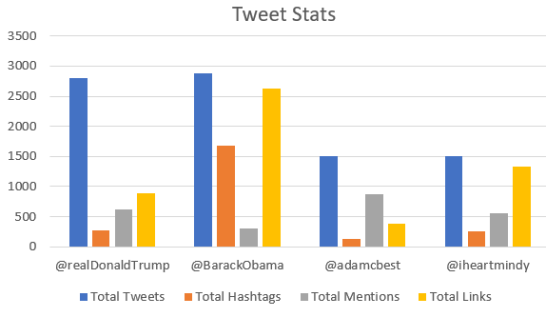


Figure 2: Unique words per user

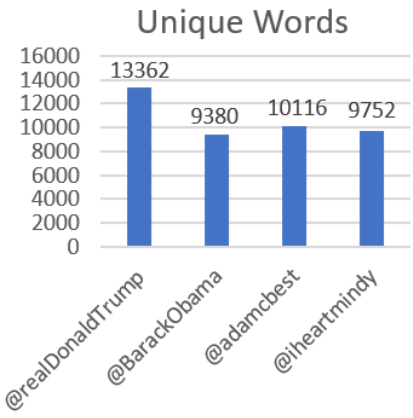
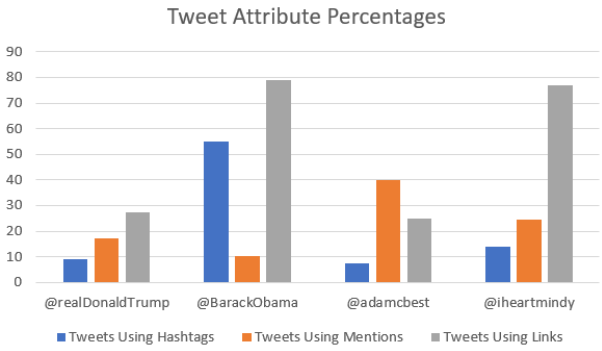


Figure 3: Twitter Attributes in the dataset



5 Baselines

One of the models that we've worked with for this project is the tweet generator package on GitHub created by Olivier Morissette (Morissette, 2018). Working with this language model provided us with the ability to easily implement a simple language model that works under the design of a simple markov state distribution.

Upon running the below code:

```
from tweet_generator import
    tweet_generator
    TPCK = '<public_consumer_key>'
    TSCK = '<secret_consumer_key>'
    TPAK = '<public_access_key>'
    TSAK = '<secret_access_key>'
    twitter_bot = tweet_generator.
        PersonTweeter('user_name',TPCK,
        TSCK,TPAK,TSAK)
```

At this point the model scrubs through all of the tweets that it can access under the restrictions of the Twitter API. With each tweet that it runs over it updates a JSON file that contains a markov chain state model from phrases to phrases. For example, after looking at tweets from the twitter user iheartmindy the JSON file contained some of the following data:

```
"\"But there": ["we"],
"\"Democrats in": ["Georgia"],
"\"For many,": ["this"],
"\"People you": ["may"],
"\"violated state": ["and"]
"#CampFire #CampFireJamesWoods": ["#
    WoolseyFire"],
"#CampFireJamesWoods #WoolseyFire": ["#
    ParadiseFire"],
"#DaysOfMindy Day":
    ["45:","44:","42:","41:"]
```

In this format we can see that the phrase 'violated state' transitions to only one word 'and' where as the phrase '#DaysOfMindy Day' has four transition possibilities '45:', '44:', '42:', and '41:'

From here we can tell the model to perform a walk over these markovian states and from this walk an output will be produced.

```
random_tweet = twitter_bot.
    generate_random_tweet()
    print(random_tweet)
```

Keeping with the iheartmindy account one such tweet that it generated from a walk was:

```
I delete my account the communists
    running Facebook win. I WILL NOT
    BACK DOWN when it comes  t o
```

Another model that we used was a character level RNN created by Sean Robertson that utilized the pytorch library (Robertson, 2017) Working with the RNN already constructed by Robertson provided us with the perfect opportunity to evaluate how well a character RNN would perform at our task of mimicking the tweeting style of an existing twitter user.

The first step here is just like above to train the model on a new data set. To do so we execute the line:

```
python train.py adamcbest_tweets.txt
```

from the terminal. Here the training process occurs over 1000 epochs. In each epoch 100 strings of 200 character are sampled from our training set and with that data the weights within the RNN are updated via a training method that follows the update procedures that we learned about in class regarding gradient ascent. Once the 1000 epochs have run we will be presented with a new .pt file. This file contains all of the weights that were learned from our training data. Because we wanted to see how well the model could perform out of the box all of hyperparameters were kept at their default values.

```
Number of epochs: 1000
Hidden size of GRU: 50
Number of GRU layers: 2
Learning rate: 0.01
Chunk length: 200
Batch Size: 100
```

From here we can run the line python generate.py adamcbest.pt and we will be presented with text generated from our model. In the case of the line provided we obtained output such as this: 'that he was in the gop has to presidential seriously there thing. else for white fightents, lindsey'

Just as was stated in the above paragraph all hyperparameters for text generation were kept to their default values.

```
Prime string = ' '
Output length = 100
Temperature = 0.8
```

Another model used was based off the IBM Developer article published by Uche Ogbuji: *"Using Markov Chains to generate language from letter correlation matrices and N-grams"* (Ogbuji, 2018). Use of this language model allowed us to interact with an character level n-gram model to see how well one would perform at our emulation task. It went through the steps of generating n-gram frequencies at a letter level using the Natural Language Tool Kit library (nltk). From there, output text based on the n-gram frequencies would be reproduced. We followed the article, modifying the code to work with a set of tweets. We collected outputs for several different n-grams shown below for 3-gram and 10-gram.

The first part of this model counted the frequency of each n-gram which simply took in the data and counted the frequency of every n-gram that appeared. For each user, we fed it the csv file containing the respective user's tweets. It output

the n-gram frequencies into nGramFreq.json for each respective n-gram.

The second part of the model took the nGramFreq.json file and followed a similar markov process as show above in Morisette's tweet generator. *Sample output put from this model:*

3-gram model

```
"itur ms bill of they amiliong so la
    paturs don a grocranitice of han the
     thoplegethermorepddrok good of the
    thans the co mocratimers they hug"

" spar cand crageried las oudurly we
    brnothe demills lehour who amp
    rondfinuclawleattpstrulecrincre i
    knot know counch to inny policke
    safeten"

"zokin to mostrult istch nomen
    tolareateresparyoutione the elicamp
    riblif of johe omontaxpled of
    themsyrestrousto ma stodwily wit se
    inthey do"
```

10-gram model

```
" and corrupt cities in honor of
    awarding the inspector general kelly
     book is a scam investigation thats
    because of the war on coal and will
    be f"

"ary and trade many other countries who
    treat the press also comes with a
    really get something i have no fear
    american people hurt so bad for our"

" on the fact that no government funding
     to pass through and it will all
    just have stated mary matalin a
    great sovereign nation we wont
    forget th"
```

When we use 3-gram, the output is gibberish. Increasing n gives output that make sense at first glance; however, with some obvious errors. We also modified this to work at a word level. From this modification we could learn about the differences between character and word level n-gram models. When comparing the output produced by the two models we could easily compare how the two models compare with one another. One output it gave was (7-gram):

```
"you want to protect law-abiding
    americans vote than claudia and she
    is a producer have a president that
    is loyal and just concluded a
    briefing with the fbi pleased to
    announce that matthew g whitaker
    times wrote a long and boring
    article for buddforcongress and
    markharrisnc9 https t c"
```

Which does not make up words, but does include parts of links and gets cut off by the length limit.

For each model we utilized the same training data. 2798 tweets from @realDonaldTrump, 2888 tweets from @BarackObama, and 1500 tweets from both @adamcbest and @iheartmindy. The Donald Trump and Barack Obama training sets utilized every tweet that we were allowed to scrub from Twitter creating a 100/0 training/test split. However, our models were only trained on half of the tweets that we obtained from Adam Best and Mindy, as we needed some equally sized training data to perform our perplexity calculations on. This created a 50/50 training/test split for those two data sets.

6 Our Approach

We have successfully found pre-existing models that can do the desired task. We found models that were already built for the specified task of generating tweets based on a user's past tweets along with general text generation models that existed that we modified to work with tweets. We also found articles that walked through implementing generation models for text. We generated a large set of mock tweets for each respective user and we now need to evaluate them.

We modified the input of models that needed it to be able to be fed tweets. We also modified it so that the outputs could be used in analyzing by us and also by statistical packages.

We collected the Twitter data using Tweepy and the twitter API and parsed the data into json files and csv files to work with each of the models we are looking at.

After giving the models the input on each of our personal computers, we are getting an output of 100 tweets per model. The models we are looking at are:

char-rnn.pytorch

```
"A PyTorch implementation of char-rnn
    for character-level text generation
    ."
```

(Robertson, 2017)

textgenrnn

```
"textgenrnn is a Python 3 module on top
    of Keras/TensorFlow for creating
    char-rnns"
```

(Woolf, 2018a)

oliviersm199 Tweet Generator

```
"The tweet generator uses a Markov chain
     finite state model to take a
    particular user id from twitter and
```

```
    generate tweets based off of that
    person's previous tweets."
```

(Morissette, 2018)
minimaxir Tweet-Generator

```
"Tweet Generator is based off of
    textgenrnn, and trains the network
    using context labels for better
    tweet synthesis."
```

(Woolf, 2018b)
Using Markov Chains to generate language from letter correlation matrices and N-grams

```
"Modeling natural language
    characteristics at the level of the
    word and generating frequency plots"
***We modified this to also work at the
    word level as another model.
```

(Ogbuji, 2018)
Text-Generation-using-Bidirectional-LSTM-and-Doc2Vec-models

```
"Text Generation using Bidirectional
    LSTM and Doc2Vec models"
```

(campdav, 2018)

Notable libraries involved:

Natural Language Toolkit, PyTorch, Tensorflow, tweepy

An issue we ran into with the textgenrnn is that it would produce an empty output most of the time. It occasionally gave us something that we could work with. We were not sure if this was a code issue or a machine issue.

Another issue, which is more of a limitation is the 3200 max tweets we can download from a user's timeline. We're also not including any retweets so that reduces the amount of data we can work with as well. For example, Trump and Obama both only have about 2800 tweets we can access.

We then took the 3200 total outputs we generated with our models and proceeded to compare the 400 outputs of each one. We compared all of the outputs independently while using the judgment criteria described in our problem statement. Once the individual evaluations had completed we came together to create a group ranking of the models that we utilized. We then went on to calculate perplexity for some of the models to have empirical results that also discuss the performance of the models.

7 Error analysis

For one metric of our human evaluation, we examined for the success of the models is how well it matched the number of tweets that use twitter attributes. The following figures were compared to figure 3 for likeness.

Figure 4 for the RNN model was the most similar to figure 3 and figure 5 hardly used any of the

Figure 4: Twitter Attributes for an RNN model

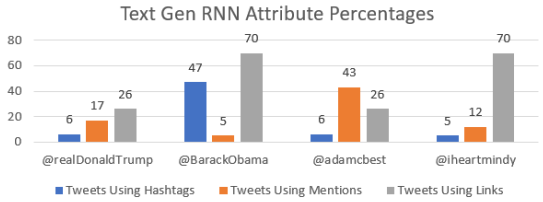


Figure 5: Twitter Attributes for a markov model

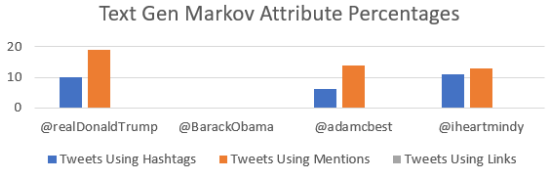
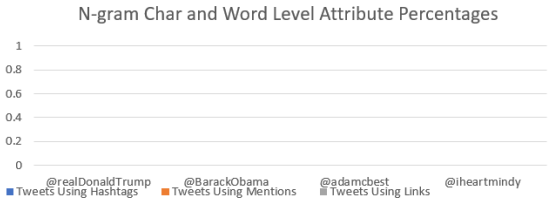


Figure 6: Twitter Attributes for a ngram model



attributes and did not use links. We saw that figure 6 was completely empty, and upon looking at the code for the ngram model, we realized that it was designed for just regular text, leaving out any symbols. As in the input of the model as designed dropped hashtags, /'s, and @ which caused the exclusion of any twitter attributes in the model.

From our evaluation of the one hundred outputs per user per model, we determined several key things about the performance of the models and what kind of outputs work best. One reoccurring thing in tweets we found to be good were that they were short. Tweets like

```
make america great again!

she would. Party.

You d o n t know what "this is....    #
    Injortiestmptically #something

"This is a plan to #GetCovered today: #
    GetCovered

"We have to make the world to do the
    fight climate change. #ActOnClimate
```

were at least almost grammatically correct and had some type of cohesive message.

Another thing we found is that characters models occasionally had a hard time forming words or links. For example,

```
help decide the world are you
```

```
    httpstcobddmyukad what in federal
    government httptcoiznkxlme of the
    administrational statement shutdown
    with pride an

in manronded. make sure over the
    businesses and the past steps
    forward." --president obama https://
    t

move someone upset alreash is surprident
     in a boath text who is what a on in
     anyone on.... https://

o join the front to for progress on
    raishoned more working families.
    https://t.co/yfevmagfbh
```

For the models like the ngram character level, http, https, then the randomized string of letters for link shorteners would all be treated as characters that could form words. Adding some weight to these caused random words. And in general, the character level models would just generate random links or parts of links because the most of the models were not designed with handling links in mind.

For the most part, most outputs made no sense. As in they had no narrative or cohesive message. In the examples below, you'll see that they don't really have a specific topic they're honing in on or that there's random words that just do not belong. And to go along with the previously mentioned finding that short tweets often made sense, we see that a lot of these nonsensical tweets are long and full of multiple tangents. All along side the problems issued before, random links, non-words/broken words, etc..

```
is why it's service by the farme--and we
    'll be jan benefiticany of our below
     better" per promise: ht

women holiday discuss problem. read more
     at the #scotus nomination in
    america." --president obama #v

it is. our economy gamead the weekly
    addrounced step together to help
    build a bold meet president ob


to by out of our children." --president
    obama #acaworks

many dead along with first polls are
    specially on a caravans are for me
    amp greatness is most spirited law
    who have never be bring massively se

will be landmark va account is currently
     repealed to wall on the meeting
    with russians who actually i believe
     that request forced to seeing to

ll that never corporate to clear tester
    early supporters this presidents
    ambassador total phone states
    defense of regs amp the great again
    http

The Supreme Court sentence of the
    movement policy and covered that
    before the country are still before
    the country still senators on
    climate change and that we want to
     give Judge Garland a fair hearing
    and a fair hearing and have a prison
     to talk about the fight to support
    the community policy to


and women but it?s clear republicans are
     merrick garland merrick garland
    merrick garland merrick his dream
    job of serving as a heard that
    because you called on an actually
    counting all the votes realize that
    races need to win to ensure fair
    nation built by immigrants and to
    preten

The Left doess say I was a "white people
     in the wile when they make the most
     difference.
```

Another problem with some of the tweets is that they are not something we'd expect the user to ever tweet.

```
today thousands of young dreamers are
    able to shoot up a classroom when
    you judge garland its highest rating
     senate leaders gonna wish you a
    happy birth-- biden even some senate
     leaders who are now work for the
    good of the american less than 75 a
    month https t who aren?t just
    running against something bu
```

You would never expect Obama to encourage shooting up a classroom. The models probably need more data to be more accurate to what these users would tweet.

From a more concrete and numerical perspective, our models are certainly not the best. Our ngram models at the word level when tested on a set of tweets theyve never seen before give a perplexity of around 800 to 1000. Specifically, the 3gram model has a perplexity of 1005.62. The 5gram model has a perplexity of 924.32. the 7gram model has a perplexity of 862.95. Similar cases arises with our character level models, which are even worse. The 3gram was measured to have a perplexity of 1227.01. The 5gram had a perplexity of 1207.56. The 7gram as with the word level model had the best perplexity of 1188.14.

Overall however, these perplexities are way too high for the models to be considered useful in actual applications. We expect this to be the case because the model drops links, mentions, and hashtags. Unfortunately, we could not get the perplexity of our rnn models as we further explain in our

conclusion.

8 Contributions of group members

List what each member of the group contributed to this project here. For example:

- Frick Shao: Did Language Model Collection / Training / Output Generation, Output Evaluation, Perplexity Calculation, Prior Works Gathering
- Quan Bui: Did Data Collection / Processing, Language Model Collection / Training / Output Generation, Output Evaluation, Data Analysis
- Zachary Bebbington: Did Data Processing, Language Model Collection / Training / Output Generation, Output Evaluation, Lots of Writing

9 Conclusion

9.1 Struggles

Somethings we struggled with while working on this project. The first was interacting with the language models that we found online. Up to this point in our CompSci career the code that we've been provided to work on/with has been relatively straight forward, and if not it was commented or documented well enough to perfectly understand. But that wasn't the case for every model that we were working with. In those cases we had to spend a considerable amount of time reading over the users code to understand how everything worked. Some of the models were given in snippets of code along accompanied by an article. It was time consuming to read through and piece it all together. Because of this it was very difficult to train some of our models and generate outputs from them. To go along with that, some models had to be modified to be fed our data.

We also wanted a bigger dataset, but we were limited by the Twitter API and the timeframe we allowed ourselves to collect the tweets. We just had to make do with the amount of tweets we were allowed. If time permitted, we would have collected tweets overtime just to have a larger dataset to work with.

Another difficulty that we encountered in our project was the evaluations themselves. After looking over the first few datasets it was clear that there were not going to be many great outputs. Figuring out what would be a fair and unbiased way to judge every output that our models had created was a tough task.

When moving further into the evaluation process, we found many problems when attempting to find the perplexities of our models. Our recurrent neural networks for example did not have any obvious weights we could give our test tweets. Since the process of determining the next word is inside the hidden markov model, we had no real way of interpreting the vectors representing the weights. Measuring perplexity the way we measured the perplexities of all the other models would not give an accurate result since the rnns does not follow the same process when generating text.

We also had quite a bit of difficulty modeling our ngram models. Specifically because of the way this implementation would handle new ngrams. Upon encountering a new ngram, it would simply add on to the sentence a random common word it found in the training data that it stored into an array. This means that when measuring perplexity, if given a new ngram, everything would immediately be multiplied by 0. In order to circumvent this problem, we simply made the probability of this specific ngram occurring a very low number.

We had a lot of trouble calculating perplexity. We did not build these models from the ground up. We had a hard time finding where in the code of all the models to implement a way to calculate the perplexity. And we were not able to fully calculate it for most of the models. For the most part, the code wasn't well documented enough for us to figure it out intuitively. If we were to do this project again, we should be using less models to give us more time to calculate perplexity.

9.2 View of Results

Seeing that the RNN was the one that performed the best on this task was something that we had expected, with everything that we had learned in class regarding the ability of neural nets to learn from the data that it receives, especial when the model is tuned properly, it made sense that it would perform better than a markov distribution or n-gram model. I was surprised when we the human evaluations concluded that the markov model performed better than the n-gram model. Perhaps this could be due to the simplicity of how the markov model operates, by doing nothing more than a simple walk over its distribution states there's little room for extreme outputs to occur.

Additionally we did have some issues when setting up the n-gram models. For example, upon closer inspection of how the code worked, we realized that it ignored a lot of characters that would appear in a tweet when it read the data. It could be the case that it was those issues that ultimately led them to performing so poorly.

For the most part, the word level models worked better than character level ones. It's something we expected because the tweets contained things

like links. It would mess up a link by throwing in a word or viceversa, for example: "httpt-coiznkxlme", "roadhttps://", "wh.govern".

We did see that the RNN model output twitter attribute percentages were similar to that of the data set, which was led it to being considered one of the best perfomers at this task. But we also think that these models could do much better with using hashtags and mentions that are more relevant to the content of the tweet. What would be even better is that if it could do links too. What we think could do this task is if we had more data to work with, particularly data that uses a lot of the twitter attributes. Especially with common hashtags that we saw in the dataset like #MAGA, #bluewave, #govote, etc..

9.3 Takeaways

9.3.1 Bebbington's

The biggest take away that I got from this project is the scope of work that the NLP domain contains. Prior to taking this class I had only every hear the name, natural language processing, but never fully understood what the field entailed. When I heard that we were going to have to do a semester long project for the class I thought that there were not going to be many options for what we could do. I expected it to be in depth coding to create something capable of processing language in some way, similar to the binary classifier in our first project. While my groups project did involve a considerable amount of code it was the evaluation process that was the most engaging portion of it for me. Taking the time to personally evaluate 3200 outputs from language models trained on data from Twitter is far from what I was expecting to do as a part of this project. And yet that is exactly what I did! The main idea that I can take away from what I did during this project is that the field of NLP is incredibly expansive, and there are many ways that we can contribute to the field to help progress it.

9.3.2 Bui's

What I took away from this project is that there is so much more progress to be made in the realm of NLP tasks. Just by looking at our output, even though some of it was good, way too much of it was garbage. And it pushes me to want to see or be a part of this field progressing.

This project allowed us to actually look at and work with multiple models. It let me dive more into the inner workings of them as well as the benefits and downsides of them. I always found things like chat bots, the generated Harry Potter book, and generated Obama speeches to be hilarious, entertaining, and interesting. This project gave me a sense of how those all work.

Just to add to that, the amount of work that goes into the NLP field is tremendous. From getting the models to work and then evaluating all of the outputs was time consuming. For tasks like part of speech tagging, it really makes you appreciate people who make publicly available word banks.

9.3.3 Frick's

Overall, I feel like this group project taught me a lot and how much I underestimated the difficulty in completing the task we set out to do. Some of the outputs that our models produced were actually quite good and made sense at least grammatically. However, most of our output did not. When we actually took a look at the models themselves, it was very difficult to tell what we could do to improve our models. If we continued to work on this project in the future, I would like to go more in depth and try to understand a few of the models that were performing better than the others in order to see what makes them different from the ones that did not perform as well.

9.4 Future Work

The main piece of work that we would love to do with this project is to tackle the goal that we were unable to accomplish in our original project design. To take one of the models that we worked with and try to improve its performance on this tweet modeling task. It would be interesting to see just how far we could go with the RNN that is already performing well at the task.

But at the same time it could be good to work with the n-gram models. As we mentioned in the previous paragraph we ran into some issues when setting up that model and its possible that those issues could account for why it performed so poorly, perhaps if we could work to overcome those issues then we would see a boost in performance and try to go further from there.

We would also like to work on getting a model to work well with generating relevant hashtags, mentions, and links. Since as of right now, it's up to chance for any of those twitter attributes to be relevant to the content of the tweet in the models we worked with..

Github Link:
https://github.com/Zbebbington/CS585

References

Alan Ritter, Colin Cherry, W. B. D. (2011). Data-driven response generation in social media. Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing:583–593.

Barnett, Jim, e. a. (1990). *Communications of the ACM*, 33(8):50–71.

campdav (2018). Text generation using bidirectional lstm and doc2vec models. https://github.com/campdav/Text-Generation-using-Bidirectional-LSTM-and-Doc2Vec-models.

Christoph Lofi, R. K. (2012). iparticipate: automatic tweet generation from local government data. *International Conference on Database Systems for Advanced Applications.*, pages 295–298.

Morissette, O. (2018). Tweet generator. github.com/oliviersm199/Tweet-Generator.

Nick Wilson, N. R. (2013). Generating twitter replies based on user location.

Not, Elena, e. a. (2017). *Proceedings of 12th Biannual Conference of the Italian SIGCHI Chapter*, pages 1–9.

Ogbuji, U. (2018). Using markov chains to generate language from letter correlation matrices and n-grams.

Oliveira, H. G. (2017). O poeta artificial 2.0: Increasing meaningfulness in a poetry generation twitter bot. Proceedings of the Workshop on Computational Creativity in Natural Language Generation (CC-NLG 2017):11–20.

Robertson, S. (2017). char-rnn.pytorch. https://github.com/spro/char-rnn.pytorch.

Sourab Mangrulkar, Suhani Shrivastava, V. T. D. A. D. (2018). Ba context-aware convolutional natural language generation model for dialogue systems. Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue:191–2008.

Timothy Baldwin, Paul Cook, M. L. A. M. L. W. (2013). How noisy social media text, how diffrnt social media sources? Proceedings of the Sixth International Joint Conference on Natural Language Processing:11–20.

Wei Xu, Alan Ritter, R. G. (2013). Gathering and generating paraphrases from twitter with application to normalization. Proceedings of the Sixth Workshop on Building and Using Comparable Corpora:121–128.

Woolf, M. (2018a). textgenrnn. https://github.com/minimaxir/textgenrnn.

Woolf, M. (2018b). Tweet generator. https://github.com/minimaxir/tweet-generator.

Zhou, C. Y. X. Z. M. Q. Z. J. (2014). Bsma-gen: A parallel synthetic data generator for social media timeline structures. *Lecture Notes in Computer Science*, 8422:295–298.