Text Summarization of Twitter Data:

- Abstractive Text Summarization of Twitter Data using Deep Learning
- Extractive Text Summarization of Twitter Data using Bigram Frequencies

Introduction

Our original motivation for this project was to analyze the sentiment of different DC neighborhoods and find what differentiates them. However, upon realizing that only about 1% of tweets are geotagged, we realized that this was not feasible. Therefore, we transitioned to a project focused on text summarization of tweets pertaining to various topics related to Washington DC. Nevertheless, we thought sentiment analysis would be a good complement to our text summarization model as it provides interesting insights into the accuracy of our model. There were a number of difficulties we experienced in finding the proper text summarization model to implement. Many of the resources which we first came across were incredibly simple forms of text summarization that utilized the NLTK library. The text summarizations that resulted from these models were not very accurate so we decided to move to creating our own model. We came to the conclusion that the best way to approach this task would be to perform text summarization by building two models for both extractive and abstractive text summarization. We decided upon a simpler extractive text summarization model using bigram frequencies as well as a more complex abstractive text summarization model using deep learning. We were able to complete a working version of the simpler extractive model on a dataset relating to Trump. However, developing the deep learning model proved to be difficult at first since it was the first time we had ever developed such a model.

We faced many challenges at first in terms of the deep learning model. Most of the resources that teach you how to make text summarization models via deep learning involved extremely large datasets and it was essentially impossible to run these models locally since they would take 50+ hours. Hence, in order to solve this problem, we experimented with running machine learning models on the cloud by making accounts for Amazon SageMaker and Kaggle. Surprisingly, Amazon SageMaker did not run these models very quickly so we decided to use Kaggle to implement and run our model.

The abstractive text summarization models seemed to be more accurate for us to use for a number of reasons since they generate summaries that preserve the overall sentiment of a text. There were a number of models of this kind that we looked into including aided summarization, encoder-decoder models, pointer generator networks, and deep reinforced models. We decided to use an encoder-decoder model since it appeared to be the most effective way to summarize our text based on the research on related work that we did. The most useful resource that we found was a guide that describes how to build an abstractive text summarizer using TensorFlow which is incredibly useful for deep learning applications.

Related Work

In terms of related work, there were a number of guides and papers that we had to read through in order to come to a conclusion on which model was most feasible to use. One of these papers was titled “Extractive Summarization using Deep Learning.” The paper describes an approach to text summarization using a deep learning model. There are three phases to this approach: feature extraction, feature enhancement, and summary generation. These all work together in order to summarize a text in an understandable manner. There were two layers of Restricted Boltzmann Machine used in this model which utilized features to score sentences, provide corresponding scores, and then produce an extractive summary. These features included named entities, proper nouns, numeric tokens, and sentence position. The guide summarized two pieces of text. One was about how an army soldier got injured and another described how divorce in western culture has increased in recent years. The summaries were incredibly accurate and it was impressive how well the model performed. However, many key details were missing from the summary as the model was an extractive one rather than an abstractive one.

After reading through many different guides, papers and various resources, we were finally able to find a resource that was most conducive to the construction of our very own abstractive text summarization model using deep learning. The work that was most related to the model that we built was an abstractive text summarization model built by a user on GitHub whose user name is dongjun-Lee. This model utilizes TensorFlow in conjunction with the seq2seq library in order to summarize the text. It is an Encoder-Decoder model with attention mechanism

(using BahdanauAttention with weight normalization) which utilizes LSTM (Long Short Term Memory) cells with dynamic bidirectional RNN (recurrent neural network) for the encoder portion and an LSTM BasicDecoder for the training of the decoder as well as a BeamSearchDecoder for inference. It yielded accurate summarization results, and as described by a number of sources is one of the most optimized models built for the tasks.

We also found some simpler sequence to sequence modeling tutorials, such as A Ten-Minute Introduction to Sequence-to-Sequence Learning in Keras and Text Summarization Using Keras. These models use a single LSTM for encoding and decoding, and do not have an attention layer implemented. They were however much more easy to implement and follow, and ran much quicker in general (although still very slow, as a result of extremely large datasets and high computational complexity). These more simple models ended up being the basis for how tried to implement our deep learning models, as the state of the art one described above would be too complicated for our current abilities and understanding of machine learning.

Approach

Data Collection

We pulled data from Twitter using a Python library called tweepy. In order to pull the data, we had to make a Twitter developer account and make an app which we called DC Mood Analyzer. This was necessary in order to obtain the API key's needed to extract data from Twitter. We extracted 4 data sets of 10,000 tweets pertaining to the following topics: Donald Trump, the Washington Nationals, the Washington Redskins, and the DC Metro. These can be seen in the csv files provided, and the scripts used for gathering them are in the folder "tweet_gathering". The scripts cannot be run, as we are not including our Twitter API credentials, but they at least provide insight into how we went about it. We performed sentiment analysis on all of the datasets which can be seen in the sentiment column of each csv using textblob which is mentioned below.

For training of the deep learning models, we used the Amazon Fine Food Reviews dataset on Kaggle. For training and testing of the bigram text summarization model, a small subset of the Donald Trump data set was used which included the top 160 retweets of the dataset for the training set and the top 40 retweets of the dataset for the testing set. The other models mentioned earlier were not used as as a result of time constraints relating to researching and attempting the implementation of a number of deep learning models.

Data Preparation

For the encoder-decoder model, we cleaned the Amazon Fine Food Reviews using some regular expressions found in the tutorials, as well as basic text cleaning operations such as making all the strings lowercase.

For the bigram text summarization model, the data was prepared by removing duplicates, dropping na values, and converting certain fields in the data set to numeric values. Additionally, unwanted characters and elements were removed and stylistics changes such as capitalization and changing words to lowercase were performed.

Text Summarization Outline of Approach

Text summarization was performed on a data set based on tweets pulled related to Donald Trump using 140 tweets for a training set and 60 tweets for a testing set. The model can be run by executing "python ngram_text_summarization.py" with the proper datasets in your current working directory.

1. The initial dataset was cleaned up and organized, this involved removing duplicates, dropping na values and converting certain fields in the data set to numeric values.
2. The dataset was sorted into a 70-30 ratio of training to testing and the data was narrowed down to the 140 most retweeted tweets for the training set and the 60 most retweeted tweets for the testing set. Since the text summarization performed was extractive, the tweets needed to be from the same subset of the dataset.
3. The training and testing datasets had unwanted characters and elements removed using libraries such as re and string. In order for the data to appear in an orderly sentence-like fashion, the first letter of every tweet was capitalized and a period was added to the end. The tweets in the training and testing datasets were put into lists and strings and were further cleaned up to remove elements such as digits, stray, whitespaces, urls, and unwanted characters.
4. A bigram frequency table was created for every pair of words in the training set by tokenizing the cleaned up words of each tweet in the training set and adding their frequencies to a dictionary. A dictionary of bigrams

and tweet ids was also created and the frequency of every occurence of a bigram with a specific tweet id was added to it.

5. After tokenizing the testing set into words, the dictionaries created in the previous step as a result of training were used in order to summarize the text. This was done by adding up the overall bigram frequency of every bigram in a tweet and adding this value to a dictionary of tweet ids with their overall frequencies.
6. After deriving this dictionary from the dictionaries developed during the training phase, the average bigram frequency of the tweets in the testing set was calculated. In order to account for frequency values of 0, add-one-smoothing was used. The average bigram frequency was calculated by dividing the sum of all the frequencies of the testing set by the overall amount of tweets in the testing set.
7. Finally, the dictionary developed during the training phase was sorted in descending order of frequency. If the frequency of an element in the dictionary was higher than the average frequency multiplied by a threshold (which can be manipulated if needed on the bottom of the file ngram_text_summarization.py in the variable threshold), then the tweet id in the dictionary was used to access the cleaned up text of tweets and concatenate it to a summary.
8. Using the steps below, a relatively impressive extractive summary of Twitter data pertaining to trump was produced. Modifying the threshold described above has a significant impact on the summary that is generated. The higher the threshold the shorter the summary as can be seen in the table below. For this reason, we only included the summary generated when the threshold is 4, 3.5, and 3. It appears that the summary generated with a threshold of 4 is the most concise and understandable. Something to note is that while the training data set had its capitalization removed, while running the testing set, the words were converted to lowercase. The capitalization seen in the phrases below was left in an attempt to retain the original sentiment of the tweets.

Deep Learning Model

The first step was to just get one of these tutorial models we found working. The Kaggle blog walks through how to make a simple english to french translator using the same type of model as text summarization, so first we just got it working for the translation dataset the provided. The data was very small so it trained very quickly and got sufficient results to show that it worked. We then simply substituted the Amazon data for the language data, in order to train a summary model instead of a translator. Because the models are both sequence-to-sequence, encoder-decoder models, this should have gotten at least some good results (the summarization model examples we found also used essentially the same LSTM setup as the translator, so this was a fair assumption). The first issue we ran into was that because the reviews are so much larger, we were running out of memory both on Kaggle and locally. This was fixed by limiting our training data to just 10000 points, instead of the full ~54000. The next issue was that even with this limited set, training was taking too much time, even on Kaggle. We finally cut the data down to 1000 entries and trained the model locally, which took about 6 hours.

The tutorial models use a character level model (each time step takes in a char and tries to predict the next one). In the real world, for both summarization and translation, a word level model is much more common. We attempted to adapt the model in this direction, changing the input vectors to a word mapping instead of character, and adding embedding layers to the encoder and decoder. Unfortunately, we ran out of time before getting this model to run successfully. The code for the word level model can be seen in wordSummarizer.py, the char model in charSummarizer.py, and the tutorial translator in translator.py. The character model we successfully trained is saved in 1000_data_char_model.h5.

Performance Measurement

The deep learning model was compiled using the rmsprop optimizer and categorical cross entropy for measuring loss, as is the standard for NLP models.

Since the dataset which was used for the bigram text summarization model was a combination of 200 tweets which do not have already existing summaries, a measure of performance for this model could be execution time. The model took approximately 17 seconds to run on average.

Sentiment Analysis

We performed sentiment analysis on our datasets using the Python library textblob. The underlying model behind textblob utilizes a Naive Bayes classifier which is trained using a large data set of movie reviews. The sentiment of the tweets in all four of the datasets available in our directory was generated using this method and the corresponding values can be seen in the CSV files.

Experimental Setup(s)

- Extracting tweets with the highest bigram frequencies from a set of tweets to summarize them
- Training deep learning models on large datasets with already provided summaries
- Generating summaries for our the tweets that we pulled using those models
- Comparing the sentiment of the summaries generated to the sentiment of the original tweets

Results

Bigram Text Summarization Model: An example of the text summarization produced with varying thresholds can be seen below. While it is clear that the grammatical accuracy of the summarizations are not perfect, the general idea of the set of tweets is captured fairly well. For the original text, please run the Python file "ngram_text_summarization.py" as it is not reasonable to put the full length of it in this file.

Example of Summaries

t*	Summary
4	Pres Trump and his defenders have repeatedly claimed that aid to Ukraine was fully paid But the impeachment report says million in military aid continues to be withheld Here this means for Ukraine and for Trump defense. The WH won participate in the process and has no response to House Intel report So what are Rs relying on in voting No That impeachment can be dismissed out of hand Given the real time reactions of Bolton and others including Trump no quid pro quo how is this credible.
3.5	Nadler President Trump abused his power betrayed our national security and corrupted our elections all for personal gain The Constitution details only one remedy for this misconduct impeachment In America no one is above the law not even the President. The Los Angeles Times editorial board is calling for President Trump impeachment. Mueller found Trump obstructed the Russia investigation which impacted Mueller ability to prove conspiracy Now Trump is obstructing the impeachment inquiry If Trump tries to argue there insufficient evidence of wrongdoing IT DIRECTLY DUE TO HIS OBSTRUCTIONIST WRONGDOING. Ex federal prosecutor Anne Milgram Trump Is Clear and Present Danger to the Election We are faced with direct threat If left unchecked the president abusive behavior stands as clear and present danger to the future of our democracy. The disdain of Democrats for Trump has led to obsessive demands for his impeachment since the moment he took office. Pres Trump and his defenders have repeatedly claimed that aid to Ukraine was fully paid But the impeachment report says million in military aid continues to be withheld Here this means for Ukraine and for Trump defense. The WH won participate in the process and has no response to House Intel report So what are Rs relying on in voting No That impeachment can be dismissed out of hand Given the real time reactions of Bolton and others including Trump no quid pro quo how is this credible.
3	This is Orwellian doublespeak The transcript IS the nightmare What happened with Ukraine and why is obvious The only real debate is the consequence By ignoring the obvious the trump party has made impeachment must when it could have been maybe. There is value in slowing down the impeachment process and letting key pieces of evidence sink in for the public Let the public catch up Let the stories marinate while Tell them over and over again via politico. tribelaw has it right Ukraine is part of pattern realDonaldTrump narcissistic compulsion to put his interests above the nation Trump efforts to obstruct Mueller reflected that too Impeachment charges should plead the pattern as well as the specific offenses. It is virtually certain that Trump cell phone calls are monitored and recorded by Russia China and several other countries That an extremely serious national security threat They know more about who Trump talks to and what he says than the. Word is that Pelosi is completely ignoring her moderate members read freshmen in Trump Districts on impeachment There is no messaging or ad support and the RNC is blowing these districts up with ads To me this says she wants this to fail in the House. It unbelievable to me the open way in which the administration and Giuliani are still pursuing this Ukraine matter said Jeffrey Edmonds who served as Russia director at the White House National Security Council under both Barack Obama and Trump. Nadler President Trump abused his power betrayed our national security and corrupted our elections all for personal gain The Constitution details only one remedy for this misconduct impeachment In America no one is above the law not even the President. The Los Angeles Times editorial board is calling for President Trump impeachment. Mueller found Trump obstructed the Russia investigation which impacted Mueller ability to prove conspiracy Now Trump is obstructing the impeachment inquiry If Trump tries to argue there insufficient evidence of wrongdoing IT DIRECTLY DUE TO HIS OBSTRUCTIONIST WRONGDOING. Ex federal prosecutor Anne Milgram Trump Is Clear and Present Danger to the Election We are faced with direct threat If left unchecked the president abusive behavior stands as clear and present danger to the future of our democracy. The disdain of Democrats for Trump has led to obsessive demands for his impeachment since the moment he took office. Pres Trump and his defenders have repeatedly claimed that aid to Ukraine was fully paid But the impeachment report says million in military aid continues to be withheld Here this means for Ukraine and for Trump defense. The WH won participate in the process and has no response to House Intel report So what are Rs relying on in voting No That impeachment can be dismissed out of hand Given the real time reactions of Bolton and others including Trump no quid pro quo how is this credible.

Deep Learning Model:

In terms of results for our deep learning models, unfortunately we were never able to get something usable. When we finally managed to train the character level encoder-decoder on the Amazon data, an example of the results were:

#Input sentence: i have bought several of the vitality canned dog food products and have found them all to be of good quality the product looks more like a stew than a processed meat and it smells better my labrador is finicky and she appreciates this product better than most

#Decoded sentence: great for the kings

For some reason the decoded sentence for every single review, regardless of content, was “great for the kings”. We were unable to figure out why in the time we had, and so did not even bother applying this model to the twitter data as it would obviously not give worthwhile results. As stated before, the word level model ran into a syntax bug in training that we also could not figure out in time, so that model was never able to train. However, the english-french translator, which uses the same model as our attempted character level summarizer, does work. It trains very quickly and gives ok results, see the README for instructions on running. We do not take credit for this code, it is pulled straight from the Keras tutorial. However, it does provide reasonable proof that the model should work when applied to summary datasets.

Analysis

Bigram Text Summarization Model

The bigram text summarization model yielded relatively accurate summarizations of tweets despite being poorly formatting. Adjusting the threshold in the model to be higher makes summarizations of sets of tweets more accurate while lowering the threshold tends to include tweets with more redundant ideas.

Deep Learning Model

Due to lack of prior experience in the topic and difficulties faced in building a deep learning model for text summarization, we were not able to build a model that produces relevant summaries of text. We genuinely put an immense amount of effort into attempting to build an accurate one using a multitude of sources but nevertheless our efforts were not successful. For this reason, we decided to build a second model which is the one described above. Hence, in terms of analysis, there is not much to say other than we greatly underestimated the difficulty of this task.

Conclusion

In conclusion, we learned a lot about a number of topics in the process of building text summarization models, extracting and cleaning up data pulled from Twitter. The end result was that were able to build two text summarization models, one of which was abstract and one of which was extractive, and were able to analyze the sentiment of various datasets of tweets. The extractive text summarization model which utilizes bigram frequencies to extract relevant tweets from a set of tweets combined into a block of text was the model which produced the most accurate summaries. Despite the difficulties we faced, it was a great learning experience especially since it was our first time building text summarization models.

Insights

Deep Learning

Our main insight into deep learning is that it is very hard to successfully set up and train a usable model. Part of that is the time constraint, where every iteration takes hours to train, and if that model then fails that is a significant amount of time spent with no usable output. The other difficulty was that neither of us had any experience with Keras or Tensorflow, nor a particularly large amount of knowledge about how neural networks work in general. However, we were both very interested and gave it our best shot, and although we did not get a usable summarization model, we do now have a much better understanding of how encoder-decoder models work at a high level.

Text Summarization

Since the bigram summarization model implemented was only applied to the Trump data set, it was interesting to see how the summary provided an accurate depiction of the current political situation in the United States. There is an incredible divide between people who despise Trump and people who support him uncircumstantial. This is not surprising considering the political discourse occurring regarding the topic of Trump in the United States but through data, it validates that the political divide indeed does exist.

General Insights

In our project submission folder, there is a folder called visualizations where there are images of some of the insights that we visualized when experimenting with code. There is also a folder called experimentationFiles which has some code that we experimented with prior to implementing the deep learning and bigram text summarization models.

Reference List

- https://www.analyticsvidhya.com/blog/2019/06/comprehensive-guide-text-summarization-using-deep-learning-python/
- https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4636345/
- https://arxiv.org/abs/1708.04439
- https://hackernoon.com/build-an-abstractive-text-summarizer-in-94-lines-of-tensorflow-tutorial-6-f0e1b4d88b55
- https://github.com/dongjun-Lee/
- https://blog.keras.io/a-ten-minute-introduction-to-sequence-to-sequence-learning-in-keras.html
- https://medium.com/@dev.elect.iitd/neural-machine-translation-using-word-level-seq2seq-model-47538cba8cd7