Content Optimization and Creation using LSTM Modeling and Sentiment Analysis

*Note: This is a Project of NEU Skunkworks, Social Butterfly, under the guidance of Dr. Prof. Nik Bear Brown

Manali Sharma
Northeastern University, Information System Program
NEU Skunkworks
Boston, United States of America
manali2251@gmail.com

Rushabh Nisher
Northeastern University, Information System Program
NEU Skunkworks
Boston, United States of America
rushabhnisher123@gmail.com

Abstract— **Social Butterfly is a social media engagement software project which is a part of the NEU AI Skunkworks Project team. We propose to create a software that performs the task of content optimization and then publishes the same optimized content on one or more social media platform. Publishing content on social media is the fastest way to reach audience so the content that should be posted and the timing at which it should be posted is of prime importance. Finding new content is incredibly hard, we are also proposing to employ machine learning methods to create new content after training the models on the given data. These are the two major tasks that we propose to tackle in our Project. We use Tweepy to scrape data from twitter and apply LSTM algorithm on the same scraped data. We also perform some sentiment analysis on the same using Gensim, Textblob and other NLP methods.**

Index Terms—**Content Optimization, Content Creation, Tweepy, LSTM, Sentiment Analysis, Gensim, Textblob, NLP**

I. Introduction

We are living in the day and age where Social Media is being flooded by new content everyday. Organizations and Media outlets are pouring hundreds and thousands of dollars to get hold of the latest news and trends. All of them are in a rat race to get attention of the common man. But what does a layman like? Is it worth for these organizations to pull so many of their resources for this task? Is there a limit for generation of new content, or is that resource exhaustible? Can machine learning provide a solution for the same? These are a few questions which we are proposing to tackle in this project.

A. Problem Overview

To know what does a layman like, we need to take a deeper look into their everyday activities, and what better medium than social media to see the engagement of people. Although statistics derived on the basis of social media patterns may not show the inclination of entire population, it is a very good place to begin, and with number of people and time spent by them on social media being a rising trend, it gives us a good sample size to work with. So having established the importance of social media, the next question that comes up is, what happens on social media platforms, and how to grab people's attention. Social media platforms generally see two major types of behaviors, people voicing their opinions or thoughts, and people liking or engaging on other people's opinions or thoughts. From this pattern it can be hypothesized that people spend time on things which appeal to them either in a positive, negative or a neutral manner. People's "Likes" and "Comments" can give us a deep insight about their opinion on different things. This engagement can be studied by a statistical approach or by the use of Natural Language Processing (NLP).

PS- We are using Python to tackle these problems

B. Solution Overview

Now that we know what methods can be used for tackling this problem, we propose to tackle it in different parts.

- **Data collection -** We have chosen Twitter as the social media platform to collect data from. This is because it's relatively easier to scrape data from twitter if you have a developer account.
  PS- We are not choosing an available Dataset as we wanted to work on live data
- **Modeling -** After scraping data from twitter, we apply a Recurrent Neural Network(RNN) algorithm on our data, and generate different models. We have chosen Long Short Term Memory Networks (LSTM) which is a special kind of RNNs. We chose this algorithm because RNNs don't tend to persist data for long term dependencies, which is addressed by LSTMs
- **Sentiment Analysis -** Another widespread and popular method for understanding and analyzing the data. This method is used for building models which are able to identify and extract opinions from the text.

II. Libraries

We are using various libraries available in python. Tweepy, JSON and CSV to extract data from twitter, and storing it in JSON and CSV files respectively. Advertools helps us extract and perform functions on Hashtags. TextBlob is used for sentiment analysis, while Wordcloud is used for visualization of most occurring words(hashtags) in the corpus. Codecs helps us mask the Wordcloud in different shapes. Plot.ly helps us visualize our sentiments. Keras is a neural network library, so we use modules which pertain to LSTM. NLTK is Natural language toolkit and it is widely used for NLP tasks. Gensim helps us for implementation of Word2Vec algorithms. Textblob states the polarity of the text, helping us classify it as a Positive, Negative or Neutral.

III. Challenges Tackled

We encountered many problems while doing this project.

- **Need for Metadata -** We needed to collect Metadata, so that we know what hashtags to use while actual scraping of data.
- **Tweet Length -** When we first scraped data, it scraped only 140 characters of text(previous limit of twitter). So we had to turn off the truncation of tweets to extract the whole text of tweet.
- **Inclusion of Retweets -** While retweets might tell us about the popularity of a particular post, we don't include it in our data as it might have many repeated texts.
- **LSTM Modeling -**
  - **Window size** Depends on the size of the dataset, needs to be bigger for larger amount data for better modeling.
  - **Seeds** Needs to be determined on the basis of the dataset.
  - **Length of sequence** Needs to be determined on the basis of the dataset.

  These are the 3 things that cannot be predetermined and need to be tuned according the dataset.

IV. Metadata

A. Need of Metadata

Metadata describes other data. It provides information about a certain item's content. Before we begin content scraping from twitter, we need to have basis as to where should we begin from. Scraping data on the basis of hashtags is a popular method, and it makes more sense to scrape data with the filter of hashtags rather than scraping all data and then grouping them by the content we need. In our case we want to scrape data on the topics relating to "DataScience". But what hashtags should we feed to our scraper to get relevant tweets? Is the '#DataScience' enough? For answering these questions, we collect metadata. We use 2 other terms close to 'DataScience' i.e. 'AI' and 'ML', and scrape around 2000 tweets to see the trend of hashtags(metadata). This gives us a good idea as to which are the most popular hashtags used in the context of 'DataScience' and collecting data using these hashtags would probably give us a more comprehensive and better results.

B. Metadata Collection

We scrape data from twitter using the python library tweepy. Tweepy makes the task of data collection very easy, we just need to have a developer twitter account, which is also not that difficult to obtain. The scraped data is stored in a nested JSON format, and it needs to normalized before we can extract hashtags from the tweets. After this we store the data in a CSV format, which can be loaded as a pandas dataframe. After this we need to clean the text, to be able to extract hashtags from the data. The extracted hashtags are then stored in a list. The frequency of hashtag gives us an idea of it's popularity and tells us which hashtags should we use to scrape the data for our analysis. This seems like a strenuous task, just to collect metadata. Thankfully there exists a library in python named advertools, which can make this task very easy. It can extract hashtag from the given dataset and run analysis in it as well. From the sample of data that we collected for the '#datascience', '#ai' and '#ml', we got the result that apart from these 3 hashtags, other top hashtags used alongside these are :- '#machinelearning', '#bigdata', '#artificialintelligence', '#deeplearning' and '#analytics'. So we use these hashtags as a filter for our actual data collection.

V. Algorithm used

Having already collected the metadata, we know which hashtags must be used as a filter while streaming the new Data. After data is collected, next big thing that we need to decide on is the algorithm. Ideally a language model should be able to predict the next word, given the current word and the sequence of words that have already been observed. Neural Network models are the way to go here, as they can generate statistical language models because they use a distributed representation where different words with similar meanings have similar representation and because they can use a large context of recently observed words when making predictions.

A. RNNs

A recurrent neural network (RNN) is a class of artificial neural network where connections between nodes form a directed graph along a temporal sequence. This allows it to exhibit temporal dynamic behavior. Unlike feedforward neural networks, RNNs can use their internal state (memory) to process sequences of inputs. This makes them applicable to tasks such as unsegmented, connected handwriting recognition or speech recognition.One of the appeals of RNNs is the idea that they might be able to connect previous information to the present task. Sometimes we only want to look at the current information (recent) to perform present task and past information is of not much use to our result, but in case of language modeling , where context is important at times and not important at other times the situation becomes trickier. In cases, where the gap between relevant information and the place where it's needed is small, RNN are useful because they can learn to use past information. But there are also cases where we need more context, the gap becomes large between two informations, RNNs is unable to learn to connect the information. And that's when LSTM comes in play.

B. LSTM

Long Short Term Memory networks usually just called "LSTMs" are a special kind of RNN, capable of learning long-term dependencies. LSTMs are explicitly designed to avoid the long-term dependency problem. Remembering information for long periods of time is practically their default behavior. All recurrent neural networks have the form of a chain of repeating modules of neural network. In standard RNNs, this repeating module will have a very simple structure, such as a single tanh layer. LSTMs also have this chain like structure, but the repeating module has a different structure. Instead of having a single neural network layer, there are four, interacting in a very special way.

VI. Working of LSTM

A. Common Notations

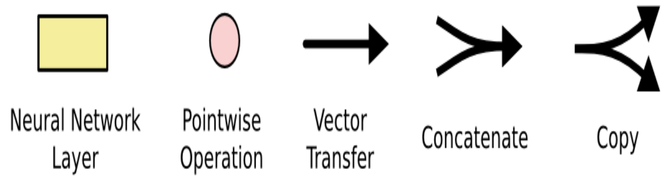


Fig. 1. LSTM Notations

In the diagram 1, each line carries an entire vector from the output of one node to the inputs of others.

- **Pink** circles represent pointwise operations, like vector addition
- **Yellow** boxes are learned neural network layers.
- **Lines merging** denote concatenation
- **Line forking** denote its content being copied and
- The **copies** going to different locations

B. Block Diagram

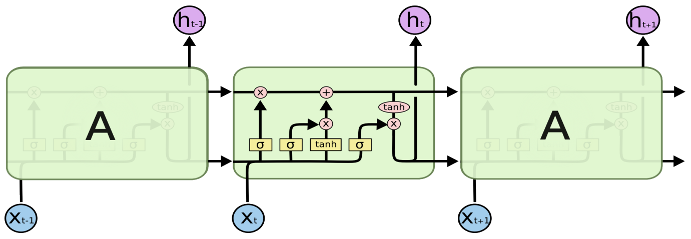
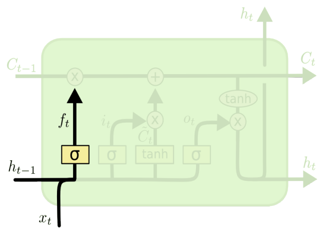


Fig. 2. LSTM Block Diagram

LSTM is a chain like structure containing various cells , each represented by green box in diagram 2. The most important thing in LSTMs is the cell state i.e the horizontal line running through the top of the diagram. The cell state is kind of like a conveyor belt. It runs straight down the entire chain, with only some minor linear interactions. It‘s very easy for information to just flow along it unchanged. The LSTM does has the ability to remove or add information to the cell state, carefully regulated by structures called gates. Gates are a way to optionally let information through. They are composed out of a sigmoid neural net layer and a pointwise multiplication operation. The sigmoid layer outputs numbers between zero and one, describing how much of each component should be let through. A value of zero means “let nothing through,“ while a value of one means “let everything through¡‘

C. Steps in LSTM Algorithm

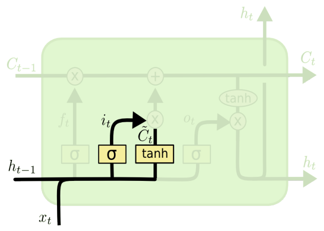
- The first step in LSTM is to decide what information we are going to discard from the cell state. This decision is made by sigmoid layer. It looks at ht-1 and xt, and outputs a number between 0 and 1 for each number in the cell state Ct1. 1 represents “keep this“ while a 0 represents “get rid of this.“



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \;+\; b_f\right)$$

Fig. 3. LSTM Step 1

- The next step is to decide what new information were going to store in the cell state. This has two parts. First, a sigmoid layer called the “input gate layer“ decides which values we‘ll update and which we will reject.
  Next, a tanh layer creates a vector of new values, Ct, that could be added to the new state.



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \;+\; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \;+\; b_C)$$

Fig. 4. LSTM Step 2

- Then we update the old cell state, Ct1, into the new cell state Ct. We multiply the old state by ft, forgetting the things we decided to forget earlier. Then we add it*Ct. This is the new values, scaled by how much we decided to update each state value.

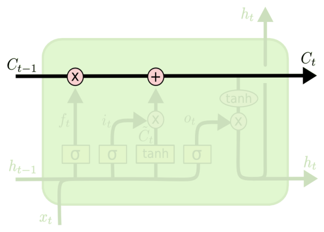
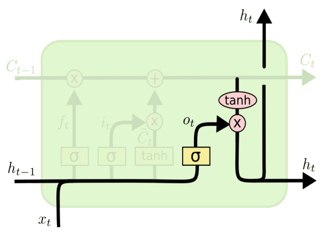


Fig. 5. LSTM Step 3

- Finally, we need to decide what were going to output. This output will be based on our cell state, but will be a filtered version. First, we run a sigmoid layer which decides what parts of the cell state we‘re going to output. Then, we put the cell state through tanh (to push the values to be between 1 and 1) and multiply it by the output of the sigmoid gate, so that we only output the parts we decided to.



$$o_t = \sigma\left(W_o \; [h_{t-1}, x_t] \; + \; b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Fig. 6. LSTM Step 3

We will be fitting and training our model now. The learned embedding needs to know the size of the vocabulary and the length of input sequences as previously discussed. It also has a parameter to specify how many dimensions will be used to represent each word. That is, the size of the embedding vector space. We will use a two LSTM hidden layers with 200 memory cells each. More memory cells and a deeper network may achieve better results.

Important Points to Note:-

- A dense fully connected layer with 200 neurons connects to the LSTM hidden layers to interpret the features extracted from the sequence. The output layer predicts the next word as a single vector the size of the vocabulary with a probability for each word in the vocabulary. A softmax activation function is used to ensure the outputs have the characteristics of normalized probabilities.
- Our model is learning a multi-class classification and this is the suitable loss function for this type of problem. The efficient Adam implementation to mini-batch gradient descent is used and accuracy is evaluated of the model.
- Finally, the model is fit on the data for 100 training epochs with a modest batch size of 128 to speed things up.

Model Parameters:-

- **Activation Function:** We have used ReLU as the activation function. ReLU is a non-linear activation function, which helps complex relationships in the data to be captured by the model.
- **Optimizer:** We use adam optimizer, which is an adaptive learning rate optimizer.
- **Loss function:** We will train a network to output a probability over the 10 classes using Cross-Entropy loss, also called Softmax Loss. It is very useful for multi-class classification.

VII. Sentiment Analysis

Sentiment Analysis is the process of determining whether a piece of writing (product/movie review, tweet, etc.) is positive, negative or neutral. It can be used to identify the customer or follower’s attitude towards a brand through the use of variables such as context, tone, emotion, etc. Marketers can use sentiment analysis to research public opinion of their company and products, or to analyze customer satisfaction. Organizations can also use this analysis to gather critical feedback about problems in newly released products. Sentiment analysis not only helps companies understand how theyre doing with their customers, it also gives them a better picture of how they stack up against their competitors.

In our project we propose to use Sentiment Analysis on the text that we generate using LSTM Model. So when we generate text, we don’t know what impact it might have on the audience. Just generating text is not enough, we need to have a measure of how the people are going to react to it. Sentiment analysis is a good measure of what people think about it, and will it have a positive, negative or a neutral reaction on the people. This part will help us deliver an end to end product, from scraping data, to generating new text, it will also help us validate our result and gauge it’s impact.

A. *Tokenization*

In lexical analysis, tokenization is the process of breaking a stream of text up into words, phrases, symbols, or other meaningful elements called tokens. The list of tokens becomes input for further processing such as parsing or text mining. Tokenization is useful both in linguistics (where it is a form of text segmentation), and in computer science, where it forms part of lexical analysis. But simply, tokenization is a method to simplify content prior to the next step of processing. You replace certain input that you know with ”tokens” which represent the meaning of that input. Tokens are individual terms or words, and tokenization is the process of splitting a string of text into tokens. For example- This is an example of Tokenization - [‘This‘, ‘is‘, ‘an‘, ‘example‘, ‘of‘, ‘Tokenization‘]

B. *Stemming*

Stemming is the process of producing morphological variants of a root/base word. In stemming a word is reduced to its word stem that affixes to suffixes and prefixes or to the roots of words known as a lemma. Stemming programs are commonly referred to as stemming algorithms or stemmers. A stemming algorithm reduces the words “chocolates“, “chocolatey“, “choco“ to the root word, “chocolate“ and “retrieval“, “retrieved“, “retrieves“ reduce to the stem “retrieve“.

C. *Textblob*

Tokenization and Stemming are two widespread method which break down a word into the simplest form possible. But having broken a word down into a small part is not good enough. We need to be able extract information from the same. One of the python libraries for doing the same is Textblob. It can perform basic NLP tasks such as extraction of sentiment from a statement with ease. The sentiment function of Textblob returns two properties, polarity and subjectivity.

- **Polarity -** Polarity is float which lies in the range of [-1,1] where 1 means positive statement and -1 means a negative statement.
- **Subjectivity -** Subjective sentences generally refer to personal opinion, emotion or judgment whereas objective refers to factual information. Subjectivity is also a float which lies in the range of [0,1].

D. Gensim

Another popular python library to extract sentiments from texts is Gensim. It is designed to process raw and unstructured digital texts. It has a wide array of algorithms at its disposal, and one of the algorithms that we're going to use in it is Word2Vec.

Word2vec is a group of related models that are used to produce word embedding. These models are shallow, two-layer neural networks that are trained to reconstruct linguistic contexts of words. Word2vec takes as its 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 located in close proximity to one another in the space. So basically this algorithm is used to convert a word to vector, and it finds similar words on the basis of these vectors.

VIII. Results

- AI, datascience, ML, machinelearning, bigdata, artificial-intelligence and deeplearning were the top 7 hashtags in the observed dataset after scraping for Metadata
- We extract the hashtag count using two methods, manually and with using the library advertools. The top 7 hashtags are same in both the methods, although a count may differ a bit. This may be due to the fact that library extracting some deep hidden hashtags. We don't know the exact working of the library so this is meager speculation.
- The difference in count is not very significant, and we are more interested in the trend of the use of hashtag, rather than the actual count.
- We are not interested in the most accurate (classification accuracy) model of the training dataset. The model is the one that predicts each character and word in the training dataset perfectly. Instead we are interested in a generalization of the dataset that minimizes the chosen loss function. We are seeking a balance between generalization and overfitting but short of memorization.
- As of now we are able to generate text, which seems much more gibberish but is making sense in some parts, the text generated can be tuned by adding different dense layers , building complex neuron network and by increasing epochs and decreasing the batch size.
- Model can also perform well , if we take a large vocab size and our input seed length is good enough, although time consuming, but output can vary differently if we experiment it in that way.
- Adding weights to model with better accuracy results results in making a model (deeper and denser one better), we can keep on iterating that ways.
- The collected data can be tested upon to find its sentiment by our third project notebook (sentiment analysis part 3)
- As we scrapped the data for given list of hashtags, we found that the sentiment is mostly positive or neutral than negative.
- When we employed text blob, it was giving us 70% accuracy in aspect of knowing the data.
- We employed Gensim, and with it's help we were able to find similar meaning words i.e. 'machinelearning' returned python, datascience, deep learing and so on.
- The model on the test set of 3 class sentiment classification provides a result of 53.7 % accuracy.
- We modeled the data and achieved 50% accuracy, but that was not the aim, we are seeking a balance between generalization and on overfitting and getting a 50% accuracy is also a good enough measure.

IX. Code and Documentation

The code and dataset of this project has been uploaded on our Github and can be accessed by anyone.

The code is available under **MIT License** for use.

X. Discussion

Good content is what sets a website, blog or any social media platform apart from the masses and delivers the right message into the hearts and minds of customers. The success of a social media platform is determined primarily by its content. And generating it is not that easy, because in today's world of duplication, one might get content on various topics but the uniqueness of it is a hard quotient. So, it is very important to generate content that formulates some meaning, and that is what our model does. Our LSTM model generates unique content on the basis of content scrapped from various websites (twitter in our case).

A dense fully connected layer of LSTM model with numerous neurons connects to the LSTM hidden layers to interpret the features extracted from the sequence that we have inputted. The output layer predicts the next word as a single vector the size of the vocabulary with a probability for each word in the vocabulary. The generated content is then passed through a sentiment analysis model that tells that whether the content will have a positive or negative or neutral impact on the masses. Not only this it also tells us the context of the content , which is given by gensim and textblob models. These algorithms provide good inference on the type of content generated and can tell us whether the content can be posted on a social media platform or not.

Acknowledgment

This research was supported by Dr. Prof. Nik Bear Brown and the team at Skunkworks who provided insight and expertise that greatly assisted the research.

We thank team of Social Butterfly and especially Vivek Dalal for comments that greatly improved the manuscript.

We would also like to show our gratitude towards all the colleagues who shared their previous experiences during the course of this research.

REFERENCES

[1] https://www.analyticsvidhya.com/blog/2018/03/text-generation-using-python-nlp/
[2] https://wiseodd.github.io/techblog/2016/09/17/gan-tensorflow/
[3] https://towardsdatascience.com/selenium-tweepy-to-scrap-tweets-from-tweeter-and-analysing-sentiments-1804db3478ac
[4] https://ipullrank.com/step-step-twitter-sentiment-analysis-visualizing-united-airlines-pr-crisis/
[5] https://www.youtube.com/watch?v=UNmqTiOnRfg
[6] http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/
[7] https://skymind.ai/wiki/lstm
[8] https://medium.com/explore-artificial-intelligence/an-introduction-to-recurrent-neural-networks-72c97bf0912
[9] https://machinelearningmastery.com/how-to-develop-a-word-level-neural-language-model-in-keras/
[10] https://machinelearningmastery.com/custom-metrics-deep-learning-keras-python/
[11] https://www.codementor.io/mgalarny/using-scrapy-to-build-your-own-dataset-cz24hsbp5
[12] https://www.youtube.com/watch?v=JU2wUgOK4-o&list=PLE50-dh6JzC6dHxpAno-a6W7QpWdAFN20&index=2
[13] https://www.kaggle.com/paoloripamonti/twitter-sentiment-analysis
[14] https://medium.com/@msalmon00/web-scraping-job-postings-from-indeed-96bd588dcb4b
[15] https://monkeylearn.com/sentiment-analysis/
[16] https://machinelearningmastery.com/text-generation-lstm-recurrent-neural-networks-python-keras/
[17] https://github.com/ayushoriginal/Sentiment-Analysis-Twitter
[18] https://realpython.com/twitter-sentiment-python-docker-elasticsearch-kibana/
[19] https://dzone.com/articles/stream-tweets-the-easy-way-in-under-15-lines-of-co
[20] https://stackoverflow.com/questions/8282553/removing-character-in-list-of-strings
[21] https://github.com/bear/python-twitter/blob/master/twitter/parse_tweet.py
[22] https://gist.github.com/dreikanter/2787146
[23] https://docs.python.org/3.4/howto/unicode.html
[24] https://www.kaggle.com/eliasdabbas/extract-entities-from-social-media-posts