Word2Vec: Emoji to Text and Text to Emoji Translations Using Twitter Data - SI630 Final Project Report

Version 1.0

Chris Bredernitz, Tyler Hoff, Amy Newman

Abstract

Emojis are visual representations, or symbols used to denote text, often used in communications through mobile devices and social media communications. In this project, we utilized Twitter data to create a Word2Vec model for translating emoji to text and text to emoji. It worked well when going from text to emoji. It did not work well when going from emoji to text. The main reason for this we saw is that most of the tweets we used had descriptive emojis which were used in tandem with other words instead of substitute emojis. We had an overall BLEU score of 0.39, but certain types of data, like the human translated songs, performed better to the level of 0.43.

1 Introduction

For this project we translated emoji into text and text into emoji using a Word2Vec model. We are choosing a Word2Vec methodology so we could break down and measure the likely words around an emoji to capture words that were used in conjunction with an emoji. We are doing this because as far as we know, it has not been done yet. One company that is doing something similar to what we are interested in accomplishing is Dango, a Canadian-based company that is currently doing machine translation on Android. Specifically, they are utilizing a Recurrent Neural Network to measure the minimum distance that a sequence of words matches with its given emoji. This is somewhat similar to the approach we took, however this system specifically focuses on adding emojis to the end of a sequence, not replacement http://getdango.com/

Since there is no previous research we know of in this area, if we are successful in our task it will introduce a new methodology of comprehensive emoji translation.

The main results are that out of 100 self-annotated tweets from text to emoji, our average BLEU score for each sentence was 0.394579607925. Our average BLEU score for emoji to text was 0.2691104884. The annotated tweets were done in a collective manner based on consensus. Indications show that there is much room for improvement on our methodology with regards to emoji to text. However, our results show that our text to emoji performs fairly well.

2 Problem Definition and Data

There is no actual problem that we are trying to solve. We set out to do something interesting that we believe has not been done before. While Dango is doing something similar, they are not doing exactly what we set out to accomplish.

We used 3,522,564 tweets from public Twitter data produced by UMSI from Professor Jurgens. We also incorporated human translated text to emoji of Disney songs including ”Do You Want to Build a Snowman” and ”Let it go”. We created our own annotations of each song and based the annotations on the ones we found in Buzzfeed. To get our BLEU scores, the group found tweets from the Twitter data and annotated them as a group by consensus both from text to emoji or emoji to text.

The Twitter data is fairly noisy as the spam tweets have not been filtered out. For example, many of the tweets are Also, due to the Olympics taking place during the time of our data, some of our flag related data does not work well, returning country names and not flags.

3 Related Work

There are no published papers we could find tackling our problem specifically. We found three papers on emoji representation from text. The first Emoji2Vec: Learning Emoji Representations

from their Description, is a paper announcing the release of Emoji2Vec, embeddings for Unicode emojis based on their descriptions. These can then be used in Word2Vec applications. The paper then describes the potential applications in sentiment analysis and shows how it outperforms a skipgram model. The second paper we found is Are Emojis Predictable?, and examines how one might predict an emoji based on the text of a tweet. It uses a Long Short-Term Memory network, which outpreforms baselines and humans performing the same task. For our third paper we are using a patent, Emoji for Text Predictions, which was helpful in learning more about emoji to text predictions.

4 Methodology

Data We used 3,522,564 tweets public Twitter data produced by UMSI from Professor Jurgens. We also used ideas from Buzzfeed where we incorporated human translated text to emoji of Disney songs including "Do You Want to Build a Snowman" and "Let it go". We obtained BLEU scores for these songs as well as our Twitter data. To get our BLEU scores for Twitter, the group found tweets from the Twitter data and annotated them as a group either from text to emoji or emoji to text.

The Twitter data is fairly noisy as the spam tweets have not been filtered out. Also, due to the Olympics taking place during the time of our data, some of our flag related data does not work as well, returning country names and not flags.

Implementation For this problem, we implemented a Word2Vec model.

Word2Vec As recommended by Professor Jurgens, we used a Gensim Word2Vec model. For Word2Vec, we implemented the following methods:

1. Create a function for loading and tokenizing using ntlk tweet tokenizer function. The function first isolated the tweet into a string, then split the tweet into a list of tokens while removing stop words, lowercasing all words, removing any tokens that started with a hashtag (to remove any hashtagged phrases), and then converting it back into a string. After, we passed the list of filtered tokens into the "tweetTokenizer" function to create a final list of the cleaned tweet. an overall list then went through each tweet as a line, removed punctuation, a .lower() to lowercase.

2. We then input the tokenized string into Gensims Word2Vec model using sentences as the variable, size = 200 for the hidden layer, context window of 4 around the center token, a minimum count of 5. This was for testing purposes.

3. We upped this to 50 when train the full dataset and 4 workers/or the amount of cores we want to use in the processor.

4. We trained the model using our 100,000 of our overall list of tokenized tweets at 6 epochs. We wanted to get a general likelihood because we werent sure how long it would take to train the model.

5. We trained all of our data simultaneously as per Professor Jurgens.

6. We then loaded the model and inputted sentences that we created to see how well the model did qualitatively. In this step we found multiple errors due to the model not being trained on every possible word in the dictionary. We then added in try/excepts to ensure the working functionality of the main function. Please see the actual code for more details on this step.

7. For results, please see Table 1.

We chose Word2Vec here so we could break down and measure the likely words around an emoji to capture words that were used in conjunction with an emoji. It worked well when going from text to emoji. It did not work well when going from emoji to text. The main reason for this we saw is that most of the tweets we used had descriptive emojis which were used in tandem with other words instead of substitute emojis. This made it so that some words like want and go translated to their descriptive emoji and not something that directly means the word used.

5 Evaluation and Results

Table 1: Word2Vec Model Results

Case	Mean BLEU Score
Emoji to Text	0.2691104884
Text to Emoji	0.394579607925
Snowman Lyrics	0.4366835442847812

We have a Gensim Word2Vec model that we implemented in the following manner: We built a function for loading and tokenizing using ntlk tweet tokenizer function, created an overall list then went through each tweet as a line, removed punctuation, a .lower to lowercase, used that new string to throw into the tweet tokenizer function and appended that to the overall list. From there

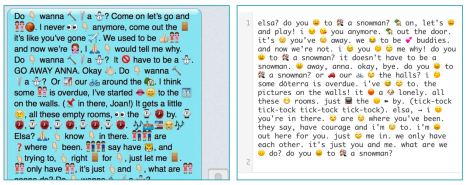


Figure 1: Human translations on the left, our Word2Vec model translations on the right.

we used Gensims Word2Vec model using the variables=senctences size=200 for the hidden layer, context window of 4 around the center token, a minimum count of 5, 4 workers/or the amount of cores we want to use in the processor. Then we trained the model using our 100,000 of our overall list of tokenized tweets at 5 epochs. We wanted to get a general likelihood because we werent sure how long it would take to train the model. We trained our entire dataset.

We decided to test our model on two baselines, songs human translated to include emoji, and tweets we annotated ourselves to have emoji or text translations as appropriate. BLEU score results from our Word2Vec model seem promising for the "Do You Want to Build a Snowman" lyrics at 0.43, and decent for Text to Emoji translations at 0.39. However, our Emoji to text translations show room for improvement at 0.26. A good BLEU score will be near 0.5, meaning our scores show room for improvement.

The reason we used Gensim is because the people who built Keras, individuals on StackOverflow, and Professor Jurgens pointed us in that direction.

6 Discussion

We have learned that Word2Vec is an effective method for assigning meaning and determining word similarity for emojis, which is knowledge we can use to create our synthetic tweets and also evaluate the meaning of the created tweets. However, there were a number of difficulties resulting from the design of emoji in unicode as multibyte objects, preventing us from differentiating between emojis of different colors. Overcoming those difficulties would allow for more accurate predictions. Looking into next steps, we feel that if we had removed more spam tweets or had a language strucutre added to the emoji to text translations, our results could have been imporved upon.

After training our model and evaluating the BLEU scores from our annotated tweets and song comparison, we found that a Word2Vec model trained on tweets containing emojis does fairly well when doing text to emoji translation, and does not perform well doing text to emoji translation. We feel this is due to the fact that many different words in our model were trained around many words that could mean the same emoji. However, in terms of text to emoji translation, when finding a replacement word to an emoji, our model finds the word with the smallest cosine distance between them. In doing this, the word that is being appended into the sentence could not be grammatically correct.

These results are what we expected from the start. We didn't have high hopes for the text to emoji translation and were pleasantly surprised by the BLEU score we recieved. Twitter data is noisy and most of the tweets contained improper spelling and grammar which means that some of our token weights were not as accurate as they could be. The Snowman lyrics score between our translated text to what Buzzfeed users created was also very suprising. We didn't think the translation we would get back would be as close as it was. The main differences were where 'want' and 'go were used and the 'tick-tock' string. The reason we dont see 'tick' and 'tock' being translated since it did not appear enough in our data set and most likely was filtered out by our minimum wordcount threshold.

The main mistake we see our model making is substituting words like 'go and 'want' get replaced with a whining face. This is due to the nature of tweets in general where people like to complain about their daily lives.

7 Other Things We Tried

We initially were trying to use a Seq2seq model to do emoji to text and text to emoji translations. We were going to use the book, Emoji Dick, as our initial training data. Due to not being able to get a copy of Emoji Dick, we ended up using Twitter data and a Word2Vec model instead. We spent a bit of time creating a Seq2seq model, but in the end did not use it with our final dataset. It would have given us the most accurate results, but we did not have the time or resources to do this.

For Seq2Seq, we tokenized, did a minimum threshold, and created chunks for gradient descent. For tokenizing, we had tried using different emoji

libraries and various types of encoding including: the emoji library, Latin 1, Iso-8859-1, and utf-8 encoding. We found that for our purposes, utf-8 encoding worked best.

8 What You Would Have Done Differently or Next

If we had the opportunity to do our project over, we think tokenizing and cleaning more, however there is only so much time in a semester. Specifically, ensuring data included only natural uses of Twitter and not advertisements and updates would probably improve our results. Emoji encoding is challenging and in the future we may have only used tweets with emojis that are neutral and have a default meaning. Its challenging using emojis with overlays because they ran the potential of getting stripped away during tokenization. Skin pigmentation also has proven to be a challenge. Possibly latin-8 encoding would help, but there is no way to visually see what is happening. ISO possibly could have worked as well, but utf-8 was the only way we could visually qualify our results.

For cleaning, removing the spam tweets may help more in the future. We believe the spam tweets had an effect on our training.

Overall, for the time and resources we had, we think the project went well and creating interesting results in a topic that doesn't seem to have any published research. While our BLEU scores are below 0.5, which seem to be a goal of obtaining, we believe there is room for improvement in terms of data cleaning and creating more synthetic tweet annotations.

One way to improve on our current system would be to also add a language model. Ideally, if we had unlimited time and resources, we would used a fully annotated dataset and a Seq2seq model. A Seq2seq model would probably return the best results.

9 Group Effort

This project was a group effort. Everyone's roles were as follows:

Christopher Bredernitz: Chris did the backend portion of the project and generating all the BLEU scores of the results with .txt output files.

Tyler Hoff: Tyler created the Flask application for the project and did research on models and resources and data necessary for the implementation of our Word2Vec model.

Amy Newman: Amy helped with the Flask application and did research on the different models and resources and data necessary for the implementation of our Word2Vec model.

The entire group created the annotated tweets necessary to create the BLEU scores.

Acknowledgments

We would like to thank Professor David Jurgens, Rohail Syed, and Yize Zang for their help with throughout this course and with this project. We would specifically like to acknowledge Professor Jurgens for his guidance on our methodology and Rohail and Yize with their help in our implementations.

References

Barbieri F., Ballesteros M., Saggion H., Are Emojis Predictable?, European Chapter of the Association for Computational Linguistics Valencia, 3-7 April 2017.

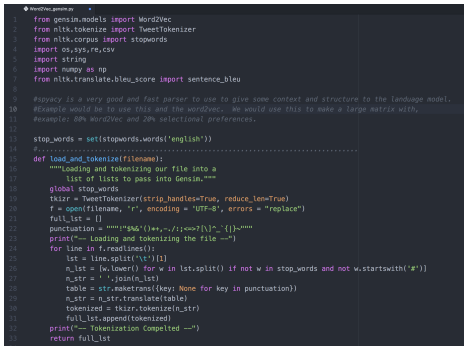
Brownlee, Jason. A Gentle Introduction to Calculating the BLEU Score for Text in Python. Machine Learning Mastery in Natural Language Processing.

Grieves, Jason A., Itai Almog, Eric Norman Badger, James H. Cook, Manuel Garcia Fierro. Emoji for Text Predictions. US20150100537A1. Microsoft Technology Licensing LLC, assignee. Patent Microsoft Corp. 3 Oct. 2013. Print.

Leminh, Trans. Your Favorite Disney's "Frozen" Lyrics Reenacted In Emojis. Buzzfeed. 24 February, 2014.

Shea, Christopher. Text Me, Ishmael: Reading Moby Dick in Emoji. Smithsonian Magazine. March 2014.

A Supplemental Material



```python
from gensim.models import Word2Vec
from nltk.tokenize import TweetTokenizer
from nltk.corpus import stopwords
import os,sys,re,csv
import string
import numpy as np
from nltk.translate.bleu_score import sentence_bleu

#spyacy is a very good and fast parser to use to give some context and structure to the landuage model.
#Example would be to use this and the word2vec.  We would use this to make a large matrix with,
#example: 80% Word2Vec and 20% selectional preferences.

stop_words = set(stopwords.words('english'))
#-------------------------------------------------------------------------------
def load_and_tokenize(filename):
    """Loading and tokenizing our file into a
        list of lists to pass into Gensim."""
    global stop_words
    tkizr = TweetTokenizer(strip_handles=True, reduce_len=True)
    f = open(filename, 'r', encoding = 'UTF-8', errors = "replace")
    full_lst = []
    punctuation = """!"$%&'()*+,-./:;<=>?[\]^_`{|}~"""
    print("-- Loading and tokenizing the file --")
    for line in f.readlines():
        lst = line.split('\t')[1]
        n_lst = [w.lower() for w in lst.split() if not w in stop_words and not w.startswith('#')]
        n_str = ' '.join(n_lst)
        table = str.maketrans({key: None for key in punctuation})
        n_str = n_str.translate(table)
        tokenized = tkizr.tokenize(n_str)
        full_lst.append(tokenized)
    print("-- Tokenization Completed --")
    return full_lst
```

Figure 2: A short example of the beginning of our Word2Vec model can be seen here. For a closer look, please see the file Word2Vec_gensim.py which can be found in the Emoji_Translation_Final_Project_SI630W18 folder on Canvas.