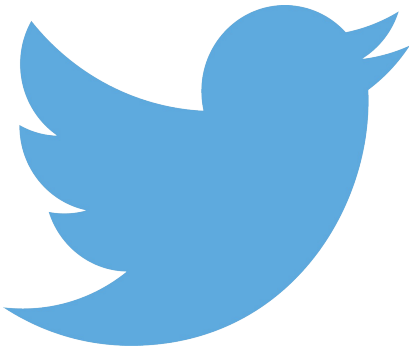
Twitter-Interpreter: Using Big Data to Learn From Natural Language

Pandi Rrapo Asa Axelrod

UMass Amherst CS383 Final Project

https://github.com/prrapo/Twitter-Interpreter



Abstract

With our project, we set out to create a program that would be fed into it a stream of tweets that contained potential errors with the intention of utilizing a neural network Artificial Intelligence interface to have our program learn the most likely possible corrections for consistently reoccurring mistakes. The intuition for us in taking to Twitter for our neural network entries was Twitter's grand scope and its status as a meeting ground of generally candid thoughts alongside news and advertising, in addition to the "cutting-edge" brand of internet slang found amongst its young users. A good amount of fun was to be expected when directing our program, armed with only a regular English dictionary, to face an onslaught of new and novel words. Ultimately, our goal for this project, as students, was to gain heightened familiarity with Artificial Intelligence techniques in order to have a stronger foundation for future endeavors in the fields of Machine Learning and Natural Language Processing.

Keywords: Big Data, Natural Language Processing, Machine Learning.

1.) Introduction

The field of Natural Language Processing has consistently sputtered along while the rest of Artificial Intelligence has made leaps and bounds as technology has improved. Though definitive breakthroughs have occurred, the overall progress made has lagged behind the expectations set by experts and the public at large. In turn, our amateur effort at a NL Processor led us to more than a few head-scratching scenarios where we unsure of how to proceed. Initially, our project idea was to simply scour tweets for errors and provide an overview as to what subject matters and what lengths of tweets were most prone to containing mistakes. It quickly became evident to us that such a scope for this program was limited and we then turned our sights to a more full-fledged Natural Language Processor that would attempt to make sense of an erroneous message and return a fixed version, aided by a neural network.

Our choice of Twitter as the go-to medium was essentially a no-brainer. Twitter is the most sprawling social media site on the web and a very reliable source for genuine human input. Combined with the tendency for its short messages to contain an occasional error and internet or culture specific slang and it is readily evident how promising this medium is for Natural Language Processing, which can be attested to by the appearance of Twitter in other

projects in this field.

Under an ideal set of circumstances, our project would also have checked for grammar syntax mistakes in addition to spelling errors but unfortunately, due to us starting too small, we overcompensated and aimed too high. As opposed to utilizing a machine learning program like Weka to aid us in our neural network, we built one up from scratch with the intention to let it serve as a learning process, allowing us to become more familiar with the systems at work in Artificial Intelligence, and we were pleased with the result in this regard.

2.) Prior Work

Georgetown-IBM Experiment[1]:

Georgetown University and IBM developed a program in 1954 that translated Russian into English. It was primarily created for the purpose of attracting public interest and funding by showing what machine translation is capable of. It had access to only 6 grammar rules and 250 words, which heavily limited what it was able to translate. Despite this, it was considered a success after a demonstration was held in which 60 carefully chosen Russian sentences were translated by it. Funding to similar projects received greatly increased funding because of this and the developers of the program thought machine translation would be a thing of the past in 5 years time. Little progress was made over the next 10 years, though, and funding was cut.

Eliza[2,3,4]:

Joseph Weizenbaum created a program called Eliza in 1964 that attempted to mimic a Rogerian Psychotherapist. Rogerian Psychotherapy is a type of therapy in which the conversation is driven by the client. The therapist responds to a clients answer by posing another question that is formed from the previous answer given. The program mimics this by parsing the input and trying to find synonyms for each word. After that, it puts the key words into a template and prints it for the user. Weizenbaum chose to program it in this way because he didn't want it to rely on large amounts of background knowledge while still being able to hold a conversation, which this type of therapy allowed.

Parry[5,6]:

A program named Parry was developed in 1972 by Kenneth Colby that tried to simulate a paranoid schizophrenic. It was programmed with a conversational strategy that would attempt to deflect questions that it thought were approaching vulnerable areas by using a system of assumptions and emotional responses. It is much more advanced than Eliza and held conversations with it on multiple occasions. When a transcript of a conversation it had had was compared with those of real patients, experienced psychiatrists were unable to determine who was human, meaning this was the first chatbot that passed the Turing test.

Jabberwacky[7,8]:

Rollo Carpenter developed a chatbot in 1981 called Jabberwacky that would hold entertaining conversations with its users. It remembers every conversation it has ever had by storing them all in a very large database. Each time it is asked a question it searches through the database and tries to find a match for that question. If it finds a match, it outputs the response the user gave when it was asked that question. If no match is found, the closest one to the original is used instead and the response to that is printed. Cleverbot is a later variant of Jabberwacky that uses better methods to match the data.

Tweet NLP[9]:

A tweet natural language processor was created by an informal collective of NLP researchers known as the Noah's Ark group. It works by reading in tweets using a text parser named TweeboParser which tries to identify the main grammatical elements in the tweet and structures them into a tree graph in order to determine content. It then assigns a tag to each word that the parser collected based on what category of words it believes it belongs to, with a confidence rating also being generated that shows how likely it is correct.

3.) Methods

The process of obtaining tweets is carried through with the help of the open source

Twitter4J Java library, an unofficial package, that provides all the functionality of being on Twitter as a real user on the site or app. We searched for tweets corresponding to one keyword mention and Twitter4J would return an array of the most recent tweets containing the keyword. In order to make sure that a tweet found was a good potential candidate, it had to pass a series of boolean tests to ascertain that it lacked any extraneous details to it, such as media, hashtags, links, or replies and of course to make sure that the tweet was in English. Afterwards, the tweet text was funneled into a project dependency known as LanguageTool, an open source spellchecker and grammar checker. Our adapted LanguageTool would detect all spelling errors in a tweet and return along with them an array of suggested corrections per error based on its set suggestions.

The neural network is then fed in the misspelling and the first suggested correction. It first has to convert these words into inputs that it can understand, so it sends the mispelled word to the WordConverter class. This class will take a word, split it up into an array of characters and convert each character to a number that represents that character. The array of numbers that is returned to the network is the input for each of its input neurons. The weight of the connections between each neuron is randomized when it is created, so it simply outputs nonsense before it goes through the learning process. The network uses backpropagation to

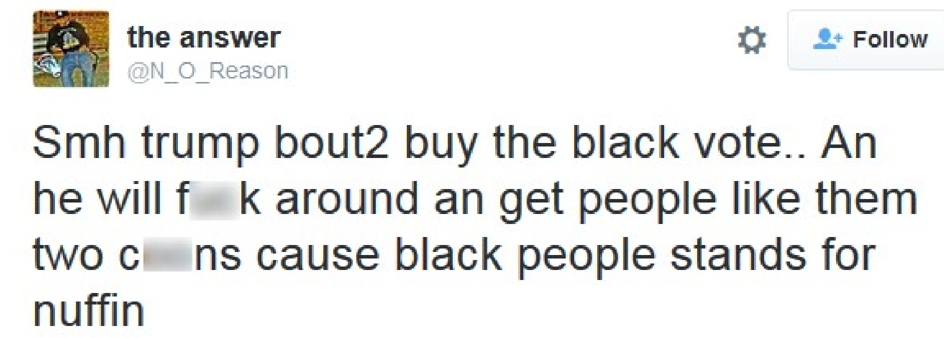
update the weights of each of these connections and, after enough data is processed, it will start outputting the correct numbers. These numbers are compressed with a logistic activation function and returned in an array. The array is sent back through the WordConverter and combined into a string. This string is the answer that the neural network has come to.

4.) Results

Ideally, we would have amassed a large amount of saved corrections per misspelled words and have those saved corrections be subjected to further analysis to make sure our program was selecting the right choice consistently but this was not to be. Our tweet grabbing process was severely hampered by the limits imposed by Twitter's Streaming API, which only allows for 180 total tweets every 15 minutes, and about only 10 or so of these would be proper candidates for parsing with LanguageTool.

It is difficult to teach the network the correct spellings of words since each tweet must be manually spell checked before being sent to it in order for it to learn and it takes hundreds of iterations before it starts to spell them correctly. We decided to use LanguageTool's first suggested spelling correction as the correct spelling to teach it with large quantities of tweets. When taught this way, it is much faster to train, but whenever the LanguageTool is incorrect, it will also give way to the wrong spelling.

We did however manage to come across a few interesting interactions when looking up the very charged world of tweets with the keyword "Trump."



LanguageTool and our neural network decidedly had much difficulty in interpreting the tokens of speech here, as evident in what was determined as a suitable correction for the last word.

```
nfhafmbfdeeeddd
nhhafmbeddeedcd
njhagmbecdddccc
mlgagnbdcdddccc
mngbgnbdcdddccc
mpgbhnbdccccccc
mqgbhnbcccccccc
mqgbhnbccccccbc
mrgbhnbcbcccbbc
mrgbhnbcbcccbbb
msgbinbcbcccbbb
msgbinacbbbbbbb
msgbinabbbbbbbb
msgbinabbbbbbbb
mtfcinabbbbbbbb
mtfcinabbbbbbbb
mtfcinabbbbbbbb
mtfcinabbbbbbbb
mtfcinabbbbbbbb
mtfcinabbbbbbbb
mtfcinabbbbbbbb
mtfdinabbbbbbbb
mtfdinabbbbbbbb
```

```
mtfdinababbbaaa
mufeinabaaabaaa
mufeinaaaaaaaaa
mufeinaaaaaaaaa
mufeinaaaaaaaaa
mufeinaaaaaaaaa
mufeinaaaaaaaaa
mufeinaaaaaaaaa
mufeinaaaaaaaaa
muffinaaaaaaaaa
muffinaaaaaaaaa
muffinaaaaaaaaa
muffinaaaaaaaaa
muffinaaaaaaaaa
muffinaaaaaaaaa
muffinaaaaaaaaa
muffinaaaaaaaaa
muffinaaaaaaaaa
muffinaaaaaaaaa
```

```
muffinaaaaaaaaa
muffin aaaaaaaa
muffin aaaaaaaa
muffin aaaaaaaa
muffin aaaaaaaa
muffin aaaaaaaa
muffin aaaaaaaa
muffin aaaaaaaa
muffin aaaaaaaa
muffin aaaaaaaa
muffin aaaaaaaa
muffin aaaaaaaa
muffin aaaaaaaa
muffin aaaaaaaa
muffin aaaaaa a
muffin a aaa  a
muffin a aaa
muffin a aaa
muffin a
muffin
```

5.) Conclusion

Our initial goal was to check both the spelling and grammar of the tweets that we processed, but as we worked on the project it was becoming apparent that we would have to focus on only one of these things. We decided that making an Artificial Intelligence interface that could act as a spellchecker would be more interesting and were met with success. When manually taught words it will nearly always be correct and when taught with LanguageTool it will only be wrong when the tool itself is wrong. It could use some optimization, but overall we are satisfied with the way it turned out, particularly with the developing of our own neural network as a learning process and a foundation for future endeavors in Artificial Intelligence.

6.) Future Work

A better method for learning from tweets or the use of a different non-AI spellchecker that was always correct and could show our program the correct misspellings of words would help it greatly. The neural network could also be optimized by changing the number of inputs, outputs, and hidden neurons which would make it run more efficiently. Once that is done, it could be improved to also check the grammar of sentences and suggest a correction if

necessary. A Chatterbot could potentially be made with this that is somewhat intelligent unlike current Chatterbots that merely rely on huge amounts of data in databases and matching words to determine output.

7.) Citations

1.) "Georgetown-IBM Experiment." Wikipedia. Wikimedia Foundation, n.d. Web. 13 Dec. 2015.
2.) "ELIZA." Wikipedia. Wikimedia Foundation, n.d. Web. 13 Dec. 2015.
3.) "Person-Centered Therapy (Rogerian Therapy)." Person–Centered Therapy (Rogerian Therapy). N.p., n.d. Web. 13 Dec. 2015.
4.) "Eliza Test." Eliza Test. N.p., n.d. Web. 13 Dec. 2015.
5.) "PARRY." Wikipedia. Wikimedia Foundation, n.d. Web. 13 Dec. 2015.
6.) ”PARRY." Artificial Intelligence. N.p., n.d. Web. 13 Dec. 2015.
7.) "Jabberwacky." Wikipedia. Wikimedia Foundation, n.d. Web. 13 Dec. 2015.
8.) "How Does the Chatbot Jabberwacky Work?" Quora. N.p., n.d. Web. 13 Dec. 2015.
9.) "Tweet NLP." Twitter Natural Language Processing. N.p., n.d. Web. 13 Dec. 2015.