ID2223 - Scalable Machine Learning and Deep Learning

Report
Final Project

Donald Trump Bot

Author :
Sina Erndwein
Samuel Leonardo Gracio

Professor :
Amir Payberah

January 27, 2019

1 Short explanation of the project

The most powerful politician in the world is the American president. That person should have the ability to speak in a proper way to affect the people. To hold a speech is one of the most important tools of politicians to communicate with the public.
Donald Trump is the current president of the USA. He is known for his iconic and very *specific* speeches. But are they also unique? Is a neural network able to reproduce a speech that sounds like a common Donald Trump speech?
This project tries to produce an artificial speech which sounds like Mr Trump. The dataset which is used is a collection of 834 speeches from 2016-2017. A special recurrent neural network called *Long Short Term Memory* (LSTM) is built using Keras. In the end, the network shall be used to see if a politician like Trump can be replaced by a machine.

2 Description of the dataset

The dataset used in this project is available at this link :
https://github.com/LeonardoGracioS/ID2223/blob/master/MrTrumpSpeeches.csv.

The dataset was collected by somebody else, on kaggle, who collected them from subtitles of Youtube videos. We cannot be truly sure of the subtitles' quality but what we saw by picking random samples of the speeches was a perfect transcription correct. Even more since these are official subtitles for the major part of the speeches of the dataset. We cleaned the data and went through it to avoid special signs or noise description for deaf people.
We started our project by doing a short statistical analysis of our dataset. First of all, we wanted to know from which period of Donald Trump's political life the speeches were made. We've divided his political life into different periods and created a graph through the python package "matplotlib" to visualize it. We also made tables with the most common words and a "wordCloud" that will be presented at the end.

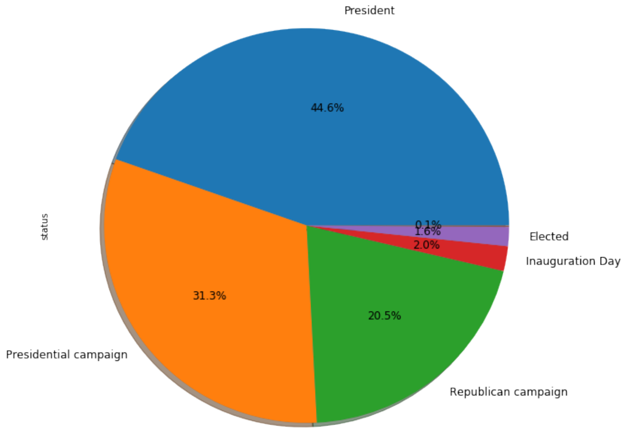

Figure 1: Proportion of Donald Trump's speeches according to the different periods

3 Methods

First of all, we've used Tensorflow and Keras. We started by exploring the dataset with some python libraries in order to put the dataset in order. After that, the data which should be used for learning was put in a single long vector of strings. A pack of 20 to 50 different speeches were used each time for learning due to time restrictions. A mapping

between each character and a number was created since the network will use numbers for word predictions. Further, a one-hot vector was used for input and output of the network. To decide on the architecture of the LSTM, two different models were tried. Since there are no perfect model for LSTM, we tried to figure out which model was giving the best performances for the lowest amount of time. First, a more complex model was built with four layers and dropout between each layer. Adam optimization was used for optimization and a softmax function for categorization.

The other network which was used was a simpler LSTM that consisted of only one layer with dropout. It also used the Adam optimizer and softmax function. This model seems simple but still gives good results.

During learning, the output was plotted every 5 epochs. After 5 epochs, based on a random part of a speech a short new speech was produced to see some results. Since not only the most probable word should be used based on the recent words, a softmax function is added. The diversity of probability for next characters can be controlled by a variable called *diversity* (or temperature). It seems that the best choice is a low diversity, between 0.2 and 0.3.

4 Results

> *And the way the American people to do that the president Trump is a very. may God bless.*
> Donald Trump Bot

The more complex LSTM produced a close to readable speech after only 5 epochs. However, computation time was extremely high and it was not possible for us to learn the model due to that and restrictions due to the available computer power. However, when using the more simple model of the LSTM and after around 50 iterations, the results were comparable to the ones from before and it took a lot less time to learn the network.

A full speech could not be produced by both models though. Only small parts of the speech make sense and it seems like it needs a better adapted network and maybe more data input to get better results. Maybe if the network could learn for some weeks, with the whole dataset for learning, the complex model and a certain number of iterations, the model could be able to produce a longer speech.

When looking at the short text parts of the network, words we found to appear a lot in the dataset like *people*, *country*, *Hillary Clinton* and *United States* is also used often by the Donald Trump Bot.

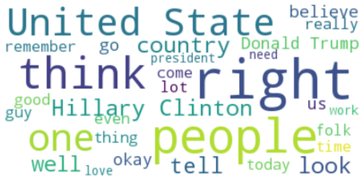

Figure 2: WordCloud of most appearing words in the original dataset

A comparison between a real Trump quote and a Trump bot quote shows that it is hard to differ between them:

> *We're going to do a wall; we're going to have a big, fat beautiful door on the wall.*
> real Donald Trump
>
> *And we're going to start the war the way we're going to start companies.*
> Donald Trump Bot

Trump bots are often used to produce new Donald Trump Twitter posts and maybe the reason for that is the easier learning of the recurrent neural network due to the shortness of Twitter posts.

5 How to run the code

At the beginning of the program, you can change the different parameters, i.e, the number of speeches, the model or the period of Trump's political life. Then, you just have to run the full code. As learning is included in the code, you might want to skip that part completely since it will take up your time.
To create your own Donald Trump Speech, go to the last cell of the code and import the different pre-computed weights in the h5 and json files available in our ZIP file. Further, you can choose if you want to use the complex or simple neural network to produce a speech by only executing the matching cells.