ANALYSING TWITTER BOT

ABSTRACT

A Twitterbot is a type of bot software that controls a Twitter account via the Twitter API. The bot software may autonomously perform actions such as tweeting, retweeting, liking, following, unfollowing, or direct messaging other accounts. The automation of Twitter accounts is governed by a set of automation rules that outline proper and improper uses of automation. Proper usage includes broadcasting helpful information, automatically generating interesting or creative content, and automatically replying to users via direct message. In this paper, we are examining the automatic replies generated by a text generating machine which generates the quote to the hashtag word (prime word) sent by a sender to the bot. For training, the bot deep neural networks, more specifically RNN (Recurrent neural network) is used for which python is used as the programming environment, and Tensorflow as the machine learning library. On the sender's side, we applied the text summarisation algorithm using PyTeaser library to detect whether the quote generated and tweeted to the sender is by the bot or any human.

Keywords: Twitterbot, Recurrent neural network, Python, Tensorflow, PyTeaser.

1. INTRODUCTION

1.1 Machine Learning

Machine learning is a field of computer science that gives computer systems the ability to "learn" (i.e., progressively improve performance on a specific task) with data, without being explicitly programmed. Machine learning is closely related to (and often overlaps with) computational statistics, which also focuses on prediction-making through the use of computers. It has strong ties to mathematical optimization, which delivers methods, theory and application domains to the field. Machine learning can also be unsupervised and be used to learn and establish baseline behavioral profiles for various entities and then used to find meaningful anomalies.

1.2 Recurrent Neural Network

A recurrent neural network (RNN) is a class of artificial neural network where connections between units form a directed graph along a sequence. This allows it to exhibit dynamic temporal behavior in a time sequence. Unlike feedforward neural networks, RNNs can use their internal state (memory) to process sequences of inputs. This Twitterbot forms the sentences by employing the technique of RNN. The hashtag word sented by the sender is given to the network and then the machine using that work generates the whole quote.

1.3 Long and Short Term Memory (LSTM)

Long short-term memory (LSTM) units (or blocks) are a building unit for layers of a recurrent neural network (RNN). An RNN composed of LSTM units is often called an LSTM network. A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell is responsible for "remembering" values over arbitrary time intervals; hence the word "memory" in LSTM. Each of the three gates can be thought of as a "conventional" artificial neuron, as in a multi-layer (or feedforward) neural network: that is, they compute an activation (using an activation function) of a weighted sum. Intuitively, they can be thought as
regulators of the flow of values that goes through the connections of the LSTM; hence the denotation "gate". There are connections between these gates and the cell.

1.4 TensorFlow

TensorFlow is an open source software library for dataflow programming across a range of tasks. It is a symbolic math library and also used for machine learning applications such as a neural network. This library has been used to train the bot to generate the sentences.

1.5 Text Summarisation

Automatic data summarization is part of machine learning and data mining. The main idea of summarization is to find a subset of data which contains the "information" of the entire set. On the sender side, the text summarization algorithm is run which then matches the summarised text with the database prime words which stores the prime words on which bot generates quotes and its synonyms if the match is found then the summarisation algorithm concludes that the text is generated by the bot otherwise it is generated by a human.

1.6 PyTeaser

PyTeaser is a Python implementation of the Scala project TextTeaser, which is a heuristic approach for extractive text summarization. PyTeaser has been used on the sender side for text summarisation.

2. BACKGROUND

2.1 Recurrent Neural Network

![Recurrent Neural Networks Have Loops](image)

Fig 1: Recurrent Neural Networks Have Loops

In the above diagram, a chunk of the neural network, A, looks at some input xt and outputs a value ht. A loop allows information to be passed from one step of the network to the next.

These loops make recurrent neural networks seem kind of mysterious. However, if you think a bit more, it turns out that they aren’t all that different than a normal neural network. A recurrent neural network can be thought of as multiple copies of the same network, each passing a message to a successor. Consider what happens if we unroll the loop:

![Unrolled Recurrent Neural Network](image)

This chain-like nature reveals that recurrent neural networks are intimately related to sequences and lists. They’re the natural architecture of neural network to use for such data.
2.2 Long Short-Term Memory

The first step in our LSTM is to decide what information we’re going to throw away from the cell state. This decision is made by a sigmoid layer called the “forget gate layer.” It looks at $h_{t-1}$ and $x_t$, and outputs a number between 0 and 1 for each number in the cell state $C_{t-1}$. A 1 represents “completely keep this” while a 0 represents “completely get rid of this.”

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

The next step is to decide what new information we’re 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. Next, a tanh layer creates a vector of new candidate values, $\tilde{C}_t$, that could be added to the state. In the next step, we’ll combine these two to create an update to the state.

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

Finally, we need to decide what we’re 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.

\[
    C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t
\]

2.3 Text Summarisation

TextTeaser associates a score with every sentence. This score is a linear combination of features extracted from that sentence. Features that TextTeaser looks at are:

- **titleFeature**: The count of words which are common to the title of the document and sentence.
- **sentenceLength**: Authors of TextTeaser defined a constant “ideal” (with value 20), which represents the ideal length of the summary, in terms of a number of words. **sentenceLength** is calculated as a normalized distance from this value.
3. RESULT

The end result is a full-fledged bot which works on the request and serving model. Giving reply with a semantic quote, which is bot intended to do.

4. CONCLUSION

The quote generating TwitterBot is completed whose training is based on deep neural networks more specifically RNN (Recurrent neural network) which remembers values over arbitrary intervals, it (bot) receives hashtag word from the twitter account holder which is tweeted to the bot and returns the machine generated quote relating to that hashtag word and retweet it to the sender the quote generated along with the prime word.

Tweepy library is used to make the connection between the bot and the sender's account. It keeps track of the user's account and sends a quote to the correct sender of the particular hashtag.

5. LIMITATIONS

Some of the limitations we came across while developing the bot. Those are, the bot can generate the quotes only for the limited word that are present in the dataset, if the bot has to be made such that it generates quote for the extensive range of words then the dataset has to be extended in order to achieve that and the database on the sender side present to match the summarised text has to be extensive. Second is that the text summarisation algorithm that runs on the sender side need to have an extensive database which contains all the words that form the dataset for the bot to check for the prime words that matches the set of prime words for which the bot generates quote and as the text summarisation algorithm result may deviate from the prime words sometimes so we need to store the synonyms of the prime words on the sender side to keep the track of all the text generated by the bot. Not meeting any of the above requirement will result in abnormal results by the bot.

6. REFERENCES