1 Project Description

While searching for an idea for the final project, I got hooked by one of the biggest and the most important problem Facebook has had, the problem with the fake news. As it is a social media platform, it is very easy to fake some news and spread it out worldwide in seconds. Facebook has decided to try to detect fake news using AI. I thought, "why don't I try to make a project that will try to generate fake news that would be as real as possible?" In that way, I could potentially use generative adversarial networks (GANs) to try to fool the fake news detector, and at the same time to train fake news generator to generate better (i.e. "more real") news content. However, which technique to use? Should I use neural network or some probabilistic model? That is the question I will answer in this project.

Here, for the comparison of two popular techniques, I implemented both probabilistic and artificial neural network models to generate some text. Specifically, I implemented the n-gram probabilistic model (a.k.a. Markov chain), and the long-short term memory (LSTM) network, type of recurrent neural network.

2 Dataset

My dataset initially consisted of around 50K articles, each with length around 5000 characters. The computational cost to process everything and then to define a model was very high, so I decided to operate only on a subset of this data. I randomly chose 2000 articles just in case to avoid any imbalance and/or bias. This must be enough to generate some data, though it is highly recommended to model on the entire dataset. However, as I do not have access to a lot of computational power, I decided to stick to a small subset for the proof of concept. If you wish to operate on an entire dataset and you have enough computational power, you are more than welcome to do so.

Articles were mostly political ones, rather than a mix-up of scientific, cultural, etc. type of articles. They are not focused on certain political party or against some party (except a few articles), therefore we shall not see any social bias towards or against certain political party, country, or government.

I read the data using *latin-1* encoding as there are a lot of latin characters. I tried removing them all to start using *utf* format, but it would remove essential characters from words. Thus, I just used the *latin-8* to read the articles.

As of data preprocessing, I removed all unrecognizable characters, emojis, ampersands, URI/URLs, etc. using powerful *regular expressions*. I also made sure that if article is of size less than 100 characters, I would remove it as well. There were such "articles" whereas in reality they were just ads (e.g. "Follow us on @instagram"). And, of course, I made everything lowercase.

3 Implementation

I implemented N-gram probabilistic model (a.k.a. Markov chain) from scratch, using python classes and numpy arrays. If you are looking for mathematical definition, take a look at my presentation.

Markov chains directly use words to generate the text. By looking at a present n-gram, I append all words, that come after this n-gram, into probabilities list. Then, I randomly choose the word (if same words reappear in the list, they have a higher probability of being chosen). Thus, by predicting next word, append it to the gram, and (using kind of sliding window technique) seeing what is the next word, and repeating the process, I manage to generate a text.

As of LSTM, I used **Keras**, the open-source neural network learning library. I implemented two LSTM layers, each with 128 neurons, and 1 dense (fully-connected) output layer. Input is characters, not words. For regularization, I used dropout technique in each layer with 20% probability. As of dense layer, the activation function I used is softmax because softmax gives the vector of probabilities of different words (you can think of it as a generalized version of logistic sigmoid function). My loss is *categorical cross-entropy* function, which basically defines the multi-class classification problem (where each class is basically a word in a vocabulary).

We cannot feed characters into LSTM. Thus, I created a mapping that maps characters to integers (in other words, the vocabulary). Having mapped all characters to integers, I feed the text into 2-layer LSTM and 1-layer dense layer network, which then outputs an integer. This integer is encoded character, which I map back using int-to-char mapping vocabulary (see the code for futher implementation details). Thus, the LSTM uses some part of text, converted to integer as an input, and predicts what will be the next character or sequence of characters.

4 Results

Both models generated more or less good text. By "good" I mean that is close to the real news content. Ultimately, the LSTM network generated a bit more credible text, though better text was expected from it.
Here is an example of using 5-gram probabilistic model:

> Figures from britains office for national statistics show that 900.000 EU nationals have migrated to britain since 2010 with many of those now starting families. Women born in romania had the highest average number of children. Simon Ross of campaign group population matters said these figures show that the impact of migration is not simply in the number of people applying over and out of the impact is shown by the number of young people entering the UK and then having a family and that affects the birth rate. Migrants are predominantly people who are in the country group who want to start a family and that is what they do doing they are generally in the age bracket when fertility is at its highest. A growing population is having an impact on a housing transport and healthcare it is putting pressure on resources and affecting the quality of play under added leaked document from George Soros's Open Society Institute network the billionaires level of involvement in attempting to build what his organization describes as a national movement to reform local police forces across the US.

As you can see, in some places the text makes sense. In others, it is non-sense. With a hope to get a more credible text, I generated some using LSTM:

> President Trump ignored his own repeated public statements criticizing the intelligence community a group he compared to nazis just over a week ago he also called journalists among the most dishonest human beings on earth and he said that up to 1 5 million people had attended his inauguration a claim that photographs disproved later at the white house he dispatched Sean Spicer the press secretary to the briefing room in the west wing where Mr Spicer scolded reporters and made a series of false statements he said news organizations had deliberately misstated the size of the crowd at Mr Trumps inauguration on friday in an attempt to sow divisions at a time.

As you can see, the generated text is way more credible than the Markov chains. However, there are some places where the text is not as good.

Thus, we can see that there is a lot of opportunities to improve the text: get more articles, use more GPUs and computational power, do a ”smarter” data preprocessing, etc. And, not surprisingly, the LSTM network generated a way better text than n-gram probabilistic model.