Russian song generator

Ildar Nurgaliev, Innopolis university

Abstract

In the report we present special case of Text Generation task as lyrics generation. We are going to learn the dependencies between characters and the conditional probabilities of characters in sequences so that we can in turn generate wholly new and original sequences of characters. Therefore we could generate new lyrics for a song, it is clear that the lyrics will not be unique, but most of the songs are the same in lyrics, especially in gangster rap.

I. Problem Statement

The challenge is to build a language model that generates and outputs song lyrics. The generation should be non-deterministic so that each time the program is run, we get a unique output to any previously generated lyrics. As a genre we chose rap, as there is some related works on rap generation that are based on n-gram in English where they have captured the possibility ro maintain rhymes.

II. Data & Crawler

To study the language model of typical Russian songs' lexicon we decided to crawl http://muzoton.ru/ site. It consists of huge list of Russian lyrics of 12 different genres as:

1) songs of 9th may as the day of WW2 victory;
2) kid songs;
3) new year;
4) rock songs;
5) pop music;
6) rap;
7) Russian shanson (mafia-prisoners music)
8) wedding music;
9) songs about war;
10) jokes;
11) jokes from KVN (Russian well-known TV-show).

The crawler source code could be found on my github https://github.com/ILDAR9/musicinfo. It is fully written in Java, the crawled site could be restored by generated sql file for MySQL and Postgres databases. After restoring lyrics and schema by sql scripts you could generate CSV files with lyrics associated with a genre using simple Select statement and Copy command.

The data is organized as follows:

- **AUTHOR**: id, name;
- **GENRE**: id, name;
- **MUSICINFO**: id, name, lyrics_text, author_id, genre_id, url.

In the work we will use just Russian lyrics for rap genre as it is expected to have more words used in a song. We have the limitations over a genre in order to reduce the words variability. The text was filtered by all the punctuations thus it has just Russian letters and spaces in total giving 34 characters. We decided to use 255 different songs that after filtering in total gives 526148 characters of text.

III. Lyrics generator model approaches

One of the first rap generator model was implemented by quadrogam model to build language model, for future work they propose manually create several "gold standard" parses based on rap lyrics [1]. Once they have generated these parse trees, they could count the part-of-speech $\rightarrow$ word pairs to create probabilities of a word given a part of speech also count the parent-node $\rightarrow$ child-node-list pairs. Starting at the root, they could recursively build a tree downward randomly based on a calculated probabilities until they end up with a tree with only parts of speech at the bottom. Then, using this structure, they could generate a word for each part of speech. This actually will generate sentences with generally good grammar and good vocabulary.

It is time consuming and takes a lot of effort to write the rules manually. What if we can build a generator that learns from existing textbook, lyrics or conversations between humans. These models that automatically learn from data, Intelligent models. They are difficult to train, as they need to learn the proper sentence structure by themselves. One of well know model is Generative models. However, once trained, the generative models outperform the retrieval-based models.

The more advanced idea of sentence generation was proposed in [2]. They train an encoder- decoder model that tries to reconstruct the surrounding sentences of an encoded passage. Sentences that share semantic and syntactic properties are thus mapped to similar vector representations. Unfortenuatly they face with such problem as the vocabulary size. The decoder has to run softmax over a large vocabulary of say 20,000 words, for each word in the output. That is going to slow down the training process, even if your hardware is capable of handling it. Representation of words is of great importance. How do you represent the words in the sequence? Use of one-hot vectors means we need to deal with large sparse vectors due to large vocabulary and there is no semantic meaning to words encoded into one-hot vectors. Lets look into how we can face these challenges, one by one.

IV. Building model

We decided to concentrate on generative language models as seq2seq or c2c that are based on deep neural networks. After text analysis of Russian lyrics it became clear that most of the words except closed sequence and stop words are rare to repeat, thus seq2seq model is not appropriate for such a little dataset as text of genre limited lyrics. We study over the joint probability between characters in a word, as the grammatic in Russian is hard, thus we will use history based neural network as LSTM layers with RNN. As we work with joint probability we should choose softmax activation function. Softmax function is typically used only in the output layer of a neural net to represent a probability distribution of possible outcomes of the network. The experiment is conducted over one layer and two layers network of 256 units of LSTM. The optimization loss function is "categorical crossentropy" allso known as multi-class logloss. In RNN each hidden state influences the next hidden state and the final hidden state can be seen as the summary of the sequence. This state is called the context or thought vector, as it represents the intention of the sequence. From the context, the decoder generates another sequence, one symbol at a time (c2c). Here, at each time step, the decoder is influenced by the context and the previously generated symbols.

V. Challenges & solutions

One of the limitations of seq2seq framework is that the entire information in the input sentence should be encoded into a fixed length vector, context. As the length of the sequence gets larger, we start losing considerable amount of information. This is why the basic seq2seq model doesn't work well in decoding large sequences. As it was mentioned before we decided to use c2c model.

In order to increase the quality of language model we decided to reduce character set up to 34: we remove all punctuation from the source text, and therefore from the models' vocabulary.

As the next challenge it is study time for RNN, it takes quite a lot of time even to study one layer of LSTM thus we tried different activation function as rectified linear (ReLU) that become the fastest one but it gave qualitatively very bad generation like the permutation of 3 characters that are most used, thus we stayed on softmax activation function as giving clear probability of 34 classes (34 characters).

For future work we could experiment with scale factors (temperature) when interpreting the prediction probabilities.

VI. Qualitative evaluation

One layer network of 256 LSTM units with softmax activation function and 'adam' optimization gave us not clear words but even in this case we already had rhymes, it was something like Beyonce song having a lot of repeats with clear periodicity.

The two layer network result was significantly better, us grammatically and with some kind of sense, it was clear from iteration to new iteration that the model become more rich with words. While in first iteration the two layer model could generate a lot of repeats, even in the 4th iteration it generated as follows: Seed is "твоя слеза это моя вина я навсегда только с тобою малыш не грусти только меня люби одного меня люби"

Generated rap: твое сердце в себе не понять со мной и постоянно не понять со мной и не понять ты меня не стоишь просто прости меня не понять сорни не поддолить со мной и постоянно не понять со мной ты меня не стоишь ты меня не стоишь

Actually this is just the result of the 2 layer model with 4 epoch with final loss = 2.0216, in order to achieve better 'flow' we expect to study model at least having loss nearly 1.02 for this we should have 40 epochs, but it would take 2 days.

References

[1] Brian Hieu Nguyen. Rap lyric generator, 2009.

[2] Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302, 2015.

[3] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.