Costrita Alexandru – GP3 – ESITech

Report

Sequence modeling

Introduction/Problematics

This work represents the use of the deep learning into 2 type of problems related to the generation of names and sentiment analysis.

First problem consists into generating a name by only giving an origin and the first letter of the name. For example, we set the origin “Russian” and the first letter “V” and the algorithm would give us a generated Russian name: “Vladimir”.

Second problem consists into sentiment analysis. To better solve this problem, we will divide it into two separate experiments. The first experiment is to find out if a comment about a movie is positive or negative. For example, we feed to the algorithm: “I really liked this movie” and from that our trained system will give the verdict that is a positive commentary.

The second experiment consists categorization of posts in social media. Let’s take an example, we feed to our algorithm “Where will happen the next Olympic games?” – the trained system will categorize it to the “Sports”. We can see various applications of this algorithm, classification of topics in a field or if we take a specific and very common example, if someone writes a post on the social media – the algorithm can classify it to a group of posts with the same topic, works a little bit like the “#”.

Method used

We will work mainly with the RNN (Recurrent Neural Networks) and find a solution to the asked questions.

Why RNN? - RNN has shown very good results with problems related to sequential data. It allows previous outputs to be used as inputs. It can also process inputs of any length without increasing the model size.

Downside of the RNN? - it has issues with long range dependencies and the computation is extremely slow. To compensate that we will use LSTM – a variation of the RNN called Long-Short Term Memory that adds gates which modifies the input with a fraction of the hidden state.

Problem 1

Create a Name Generator using a LSTM

First of all, as we mentioned we will use LSTM network for better long-range dependencies. The “Name generator” is basically a database where each origin has a group of names that we can generate from. All this network needs is an origin (the input) which is the first letter in the name. The algorithm returns the name consisted of the first letter we gave, plus the rest of the letters chosen by the hidden layer (the data on which the network trained on).

The size of the input-output that our LSTM network will give depend on the origin and hidden layer (our case 128). We train our algorithm by taking a random word and origin, the model will predict the origin of the name by looking at the words we fed. This procedure will give us the weights of the model and the loss function.

LSTM model is generating us the following examples:

Russian

Ranen
Uanen
Sanen

German

Garer
Eaner
Raner

Spanish

Saran
Parana
Aranan

Chinese

Can
Han
Ian

First reparable observation is that the majority of the letters in each origin are the same. Moreover, the length determined by the model is almost the same for each origin, in Russian – 5, German – 5, Chinese – 3.

Second observation is that it isn’t functioning very good since these names are very rare, almost inexistent. I can speak at least for the Russian names since I am Russian and I never heard these names before and the dictionary of names isn’t giving me examples like “Uanen” or “Sanen”.

Problem 2

Sentiment analysis Model

In the following part we will dive into the resolution of the sentiment analysis problem. We will use the bidirectional LSTM Network and furthermore we will use a database of reviews on IMDB – which is an online database of information related to films, television programs, home videos, video games, and streaming content online – including summaries, trivia, ratings, fan and critical reviews which can be marked up to 10. This marking system (out of 10) will be our input for the network. We feed to the network a review and it gives us back a value between 0 to 1 with 1 the maximum points which we can consider equivalent to 10 in our problem. The input is determined by the length of the review and the output is a float that corresponds to the predicted rating.

How do we train this model?

Firstly, we have to feed the network reviews which are related to the rating. This will give enough information to the model to attribute a "positiveness weight" to the commentary which basically means that some words as: bad, garbage, sucks, lost my time, worst, etc. will be attributed a low "positiveness weight" and words like: majestic, wonderful, beautiful, best, amazing, etc. will have a high "positiveness weight".

We observe an accurate prediction from the model. Positive and good comments which raise the movie's rating have a high mark and vice-versa for bad comments and lowering the rating.

In any case this model can't be precisely applied because we have to take into consideration the point of view of the reviewer – the reviews that aren't nor bad or good, the meanings that only a human will understand. Also, a good point can be the understanding of sarcasm and how people often use nice words but jokingly.

Multi-Class Text Classifier

In the second part of the problem, the database used for the multi-class Text Classifier is the forum on Yahoo Answers. We will work with the LSTM model in which we will change the loss function with the CrossEntropyLoss() function. We make that change because we will need some changes in the training loop and an output dimension equal to 10.

On the Yahoo-Answers forum (the one that we took is almost 10 y.o.) there are 10 categories, and each question-answer is associated to a category.

How does the network work?

The network works with strings which are the questions associated with the answers (usually the best ones) and returns a category (1 out of 10). We use the same techniques to train this network and to measure the accuracy we will simply feed to the model the real category and we will compare with the category the network predicted.

After a long wait, we can observe that the model is not perfect but can give some correct predictions.

This example our model gave us after the compilation:

Being a Human Resources Manager how will formulate an effective HRP process? **– 2/10**

Regarding the subject details your question does not make sense. If you want to be a HR manager, you need to learn to spell and use the language properly. **– 4/10**

We observe an almost accurate prediction from the model. Both question and answer are in the related to HR topic, the categories are slightly different though.

There are still some issues related to questions that aren't precise enough to categorize them even as a human. These questionable choices present a difficulty for the model and it takes only one of them.

Conclusion

In conclusion we can definitely say that the Recurrent Neural Networks are a powerful tool that can give accurate predictions on specific topics. However, if the choice is difficult or the model misunderstands some values due to the sarcasm factor or other values that have a level of uncertainty like the words: "normal, bizarre, weird, exceptional, different" – which basically can be bad or good meanings, the results can be wrong and not relatable.

In any case the LSTM model that we used on our specific problems has shown good results and we can certainly say that all the questions and objectives of the Sequence Modeling TP have been successfully completed and the obtained results are satisfactory.