Capstone Project Proposal

For the final project, I would like to work on an NLP application. Natural Language Processing has gained tremendous momentum in the previous year. My idea falls under the generative part of NLP. Scrolling through Kaggle datasets, I found a dataset that had 55000+ song lyrics and other data like an artist, song Name and Link to the song. I thought of creating a web app that generates song lyrics once we input a random song name.

The problem is a text generation problem. The algorithm is trained in a supervised classification fashion.

Problem Domain/Background
The problem falls under NLP (Language Generation). Core NLP tasks like Language modeling, Named Entity Recognition, Translation, Sentiment analysis were being solved using traditional statistical models like Markov Chains, Conditional Random Fields, Traditional Machine Learning models like Logistic Regression and SVMs. With the advent of Deep Learning, these tasks are being solved much more effectively given huge data and computational power.

Seq2Seq models are used in Neural Machine Translation (NMT). They essentially comprise of two Sequential models (RNN/LSTM/GRU) in the form of encoder-decoder architecture. In NMT, the model takes input in one language which is processed by an encoder, which generates a latent representation fed to the decoder. The decoder is a sequential model like encoder, but its job is to generate the output in the target language (in case of NMT).

Literature survey:
Many eminent NLP scientists like Christopher Manning, Dzimitry Bahdanau have successfully implemented NMT (paper). Stusvekar et al implemented one of the early seq2seq architecture (paper) and scored better BLEU scores than traditional models. Yan LeCunn, another eminent researcher experimented CNNs on text for classification purposes (paper). Traditionally CNNs are used for image-based tasks.

In our case, we feed the song title as the encoder input, and output song lyrics from the decoder.

There can be a couple of experiments we can perform. Since the song titles are considerably smaller in length than the actual lyrics, the encoder can be a simple CNN instead of an RNN/LSTM. CNNs have shown promise in a number of NLP tasks like text classification. In addition, they train faster than an RNN.

Solution brief

Consider an encoder which can be a sequential model or a CNN (for experiments sake).

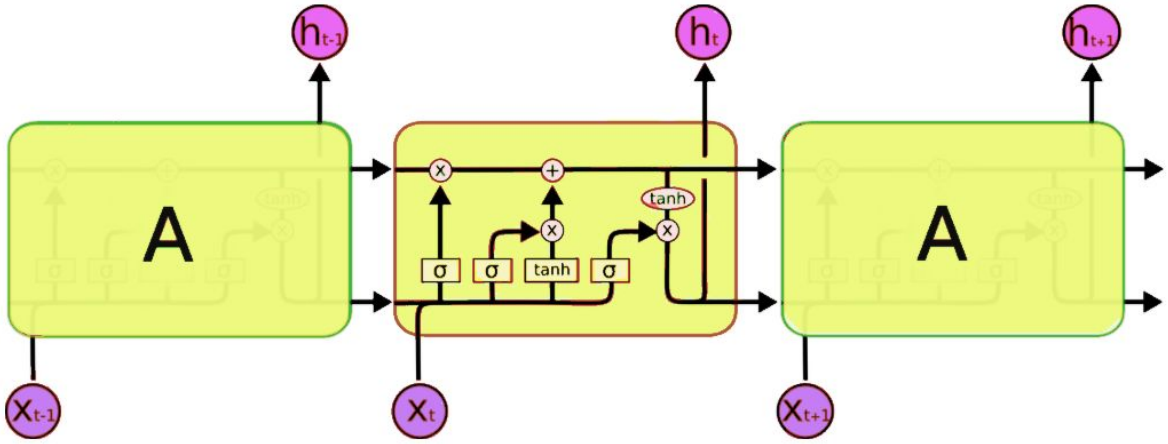

Image credits:
https://s3-ap-south-1.amazonaws.com/av-blog-media/wp-content/uploads/2017/12/10131302/13.png

An LSTM is a sequential model consisting of hidden states just like RNNs in addition to memory cells which help solve the vanishing problem in Vanilla RNNs. They are trained like RNNs.

In case of a CNN encoder,

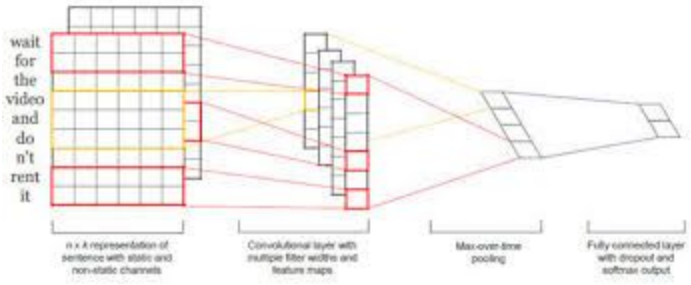

Image credits:
https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcRGFsJfUTQv4arKqKexN-EiQ09YaPkXRTs5YfbsFjYZzxOvHbxOsA

In any case, we obtain a latent vector fed to a decoder which essentially is an LSTM/GRU. Complete encoder-decoder architecture is shown below.

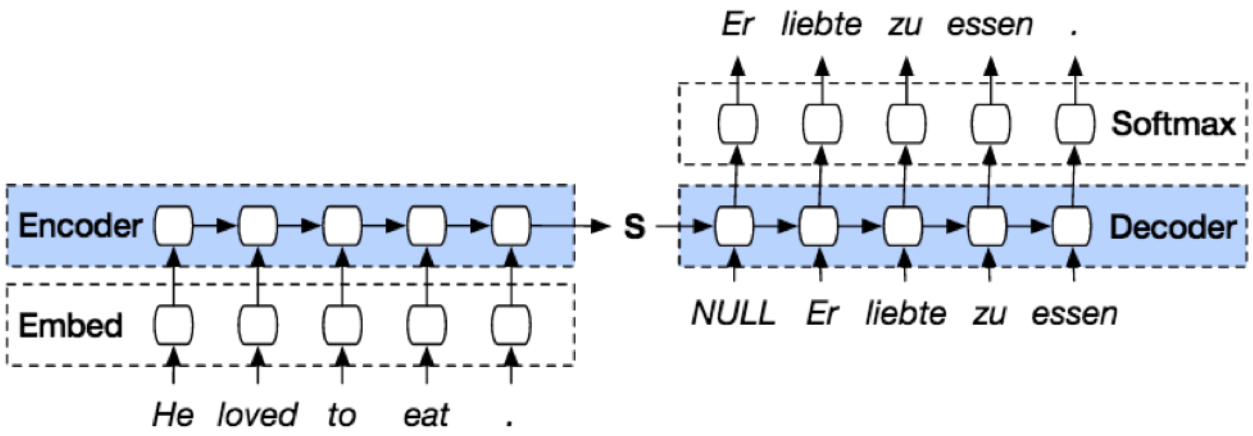

Image credits: https://miro.medium.com/max/1200/1*vl0HqHVsLB_hvxxiwX-0dQ.png

Dataset
The dataset is obtained from Kaggle. Link.

Example input:

Title: `Ahe's My Kind Of Girl`

Lyrics:

```
Look at her face,
it's a wonderful face
And it means something special to me
Look at the way that she smiles when she sees me
How lucky can one fellow be?
She's just my kind of girl,
she ma...
```

Many Kaggle kernels have implemented lyrics generation. On of which (Link) got a sample output as

rolling down a backwoods tennessee highway
but we pray the storm that you leave you
outs a little heart of god
more than in your eyes
yeah

the one is a little start
i'm gonna see your life is too
i just want to see you understand

and i want to love you
i know you are my words
can't you see you say
but i think i stop in the sun
and you'll be a storm
the light of the stars and you can be
someone to take you

Implementation Strategy

I intend to use the PyTorch framework for constructing my model. The training part is explained above. The inference part will comprise of a topic and number of lines to be generated. Lines will be separated by <NEW> token just as traditionally <PAD>, <START>, <END> tokens are used in NLP tasks. If possible, I also intend to use attention mechanism for better matching of topics and text (Link)

Evaluation Metric

Perplexity. This metric is widely used in NLP tasks like Language modelling. A broad overview of what perplexity is can be understood from the infographic below.

Perplexity

- In practice we don't use raw probability as our metric for evaluating language models, but a variant called perplexity.
- The perplexity (sometimes called PP for short) of a language model on a test set is the inverse probability of the test set, normalized by the number of words. For a test set $W = w_1, w_2, \ldots\ldots, w_N$:

$$\begin{aligned} \text{PP}(W) &= P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}} \\ &= \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}} \end{aligned}$$

Image credits:
https://image.slidesharecdn.com/nlp09-180507133413/95/nlpkashkevaluating-language-model-8-638.jpg?cb=1525700442

Solution Steps

1) Collect the data from Kaggle
2) Extract song title and lyrics
3) Perform EDA – Most frequent words, statistics on the number of lines in a song, etc
4) Perform data preprocessing – lowercasing, removal of punctuations, etc
5) Add <PAD>, <START>, <STOP>, <UNK> tokens.
6) Create unique indices for tokens
7) Create batches, input a title and output as lyrics
8) Build an encoder-decoder architecture
9) Perform training job
10) Evaluate and experiment with hyperparameters, add attention if necessary.

11) Create an endpoint for deployment
12) Create a lambda function for triggering the model.
13) Create an API for the web app.