METL Project - Hypothesis Report

Tientso Ning

The Setup

The setup itself is that we are aiming to take text data, and train the model to then produce the soundspel version of the text, effectively implementing a character-based machine translation. I'm guessing that the intuition here is that certain characters have relations to each other, and appear more frequently given the words that are in a given task. In this way, the character based translation should be possible given our training data.

Brief Summary

Here we detail some notes regarding the variants and some information about their uses as well as drawbacks.

RNN is the normal type of model that we're familiar with, whereas the GRU and the LSTM are gated versions of the RNN setup, whereby the unit decides at each timestep whether or not the output is fed to the next layer or reset/discarded.

Attention is the mechanism that exists "between" the encoder and decoder, and is a system that gives information to the decoder at each time step regarding the encoder output information. This allows the decoder to have better performance, acting essentially as a "context" provider. The attention mechanisms are usually trained separately.

Bidirectionality is the idea that we have another set of hidden states that feeds the output at each timestep for the hidden state backwards in time rather than forwards in time. This bidirectionality is useful in cases where the future state informs the past states. I.e: "I am X" along with "I am X hungry" will yield different things for X given whether or not you know that hungry exists. In the first case, we will use a noun (emotion for instnace, I am happy), or maybe a pronoun (name, I am Kense). However, when we know the existence of hungry, we will change it to an adverb most likely (I am very hungry). This is an example of where the future state informs the previous past states.

Stacking is the process of increasing the depth of the model. In a usual machine learning model, we increase the depth by increasing the number of layers. However, in the case of RNN/LSTM/GRUs, we are moving forwards in time, and thus the depth of the model cannot be increased by increasing the number of layers in the same sense (since that would just be increasing the progression of time). And thus, the way to increase the depth is to "stack" multiple RNN/LSTM/GRU models on top of each other, and have each timestep information feed to the next timestep in the same RNN, as well as feed it to the next

stack RNN. In practice this architecture increases performance, and intuitively it makes sense since it increases depth the way an ordinary ML model would.

Dropout is where you drop certain unity completely during training. This leads to better generalization in practice. The details on why it works is probably not worth detailing here.

The Considerations

In terms of choosing RNN/GRU/LSTM, we have to consider the context of the problem. We are essentially doing a character-based translation (going character by character from the source language English, to the target language Soundspel). In the sense of a character based translation, it doesn't seem necessary to keep a memory structure although it could increase the performance in practice, the additional overhead may not be worth it.

In terms of whether or not attention mechanism would assist the character-based translation task is also similar to the considerations above with regards to the model architecture. It seems like the characters themselves don't really influence anything too long-term besides themselves. It would seem that attention mechanism would be more trouble than its worth with regards to overhead.

Bidirectionality is interesting in that if we think about the task as a character by character translation process, then the bidirectionality would suffer from the issues that it does in language models traditionally: that we don't have the future states during testing time, and that would severely hurt the performance of the model. However, the interesting thing is that if the task is thought about from the perspective of taking the text, character by character replicating the text, and then character by character editing it, then we would have access to the "future states" in this setup, and the bidirectionality could prove useful the same way it proves useful in tasks like Named Entity Recognition. However, given the current understanding of the setup of the task, this bidirectionality will most likely prove to be more trouble than it's worth.

Stacking, since we are increasing depth, is concerned about training time as well as efficiency in terms of convergence. However, I am assuming that our task is still considered relatively simple in the world of machine learning, and the task seems like it would most likely benefit from the additional depth.

Dropout has lots of literature on how it improves performance. I see no reason to why it would not be the case in this task, hopefully allowing our model to generalize better, in a task that seems to require generalization power (reason why we are approaching this problem using a Machine Learning model in the first place, see section 2 paragraph 5).

Conclusion

We are most likely aiming for a stacked RNN structure with dropout. In the case where I have been too conservative in regards to the training time, addition of attention as well as using GRU/LSTM architecture might provide additional performance, and the cost of time may be negligible.