MAIS 202 Deliverable 3: Final Training Results

Michelle Wang

1 Final training results

I made some big changes to the implementation of my model. Instead of using Keras, I am now using the OpenNMT-py toolkit. The reason why I changed is that OpenNMT-py provides a relatively easy to use sequence-to-sequence model with a pre-implemented attention mechanism and beamsearch, which I wanted to use to improve my previous model. The OpenMNT-py model was also still quite flexible, with many parameters that could be configured. Another addition was that the OpenNMT-py model uses embedding layers (instead of one-hot encoding) to let the model learn a relationship between different letters.

The table below contains hyperparameter combinations that I tried out for the OpenNMT-py model and the performance of the corresponding model. The best set of hyperparameters is in bold. Generally, using a bidirectional LSTM ('brnn') gave better results. I ran into problems when I tried to use 1000 nodes because the saved checkpoint files were very large and I was in danger of exceeding the limit for my Google Drive (since I am using Google Colaboratory to train). I have also found that increasing the patience parameter gives better results, so I am now using 10 instead of 1, 2 or 3. I have also not explored all the possible parameters for the model, especially the ones for the embedding layers and the attention mechanism.

Encoder	Layers	RNN size	Dropout	Batch size	Patience	Optim.	Acc.	BLEU
rnn	2	500	0.3	64	10	sgd	0.6573	80.41
rnn	2	500	0.2	32	10	adam	0.6466	79.79
brnn	**2**	**500**	**0.2**	**32**	**10**	**adam**	**0.6684**	**81.45**
rnn	3	500	0.2	32	10	adam	0.6489	79.88
brnn	3	500	0.2	32	10	adam	0.6680	81.35
rnn	2	500	0.3	32	10	adam	0.6477	79.81
brnn	2	500	0.3	32	10	adam	0.6670	81.24
rnn	3	500	0.3	32	10	adam	0.6555	80.07
brnn	3	500	0.3	32	10	adam	0.6660	81.17
brnn	2	500	0.1	32	10	adam	0.6622	81.01
brnn	3	500	0.1	32	10	adam	0.6665	81.17
brnn	4	500	0.1	32	10	adam	0.6574	80.67
brnn	2	600	0.1	32	10	adam	0.6629	81.06
brnn	3	600	0.1	32	10	adam	0.6681	81.32
brnn	4	600	0.1	32	10	adam	0.6651	81.26
brnn	4	500	0.2	32	10	adam	0.6626	80.94
brnn	2	600	0.2	32	10	adam	0.6655	81.13
brnn	3	600	0.2	32	10	adam	0.6681	81.27

The accuracy and BLEU score of the best model are 0.6684 and 81.45 respectively, which is an improvement from my previous best model (accuracy: 0.6195, BLEU: 73.59).

2 Final demonstration proposal

I plan to create a landing-page web application. The user would write a word in an input text box and then the predicted pronunciation will appear. For my model, I am thinking of creating a REST translation server (following this tutorial: `http://forum.opennmt.net/t/simple-opennmt-py-rest-server/1392`). I would then be able to make an API call and get a JSON object containing the predicted pronunciation.

For the front-end of my application, I might use a template or just code it by myself, following online tutorials (I don't think it will be too challenging since my page will be very simple). In any case I will use the Bootstrap library for styling.

For the back-end, I think I will be using a NodeJS server. My experience with web applications is not very extensive, but I have some minimal experience with NodeJS (and also with Flask, but that was a long time ago), and I think I will be able to do this with the help of some tutorials.

Finally, if time permits, I would like to make it so that, when the user hovers on a displayed phoneme (from the predicted pronunciation), a corresponding sound is played. I have found downloadable online audio files of someone pronouncing phonemes (`http://www.phonetics.ucla.edu/course/chapter1/chapter1.html`), so this should be possible to implement. The only potential problem is that these correspond to IPA symbols, but I think there is a one-to-one correspondence between ARPABET and IPA, so it should not be a big problem.