NLP Assignment 2 by Suyash Vardhan Mathur Apr 7, 2021

NLP Assignment 2 Report

The algorithm used here is the LSTM based Neural Network, which is a state-of-the-art approach for Language Modelling tasks nowadays. The corpus given consisted of ~50,000 sentences, which were earlier split in the ratio 7:2:1, and thus `train.txt` consists of 35,000 sentences, `valid.txt` consists of 10,000 sentences and `test.txt` consists of 5,000 sentences.
The model was trained for various epochs, but due to some issues with Google Colab, not all of them could be stored. Further, at the time of calculating perplexity of the sentences, loading all the sentences was causing the Notebook to crash, and so in the perplexity files, only the first 30,000 tokens have been considered to avoid the crashing. **NOTE** that all the sentences in `train.txt` were used to
Handling Unknown Words: Unknown words were handled by using the **<unk>** tag for all the words in training set whose count was lesser than or equal to 3. Then, while calculating perplexity of testing/validation datasets, an out of vocabulary word was mapped to the lemma as well. For handling sentence boundaries, <sent> tag was used.

Perplexity Scores

LM1-LM4 are differnt LSTM-based LMs created during different epochs. **LM1-train**: 757.0506443612416
LM1-validate: 400.3966369714608
LM1-test: 371.1640456588943

LM2-train: 571.94596827277
LM2-validate: 308.6732827864598
LM2-test: 285.60263978426696

LM3-train: 753.3754500463355
LM3-validate: 400.64373346423673
LM3-test: 372.3392684145832

LM4-train: 786.5858743139001
LM4-validate: 419.6364948551919
LM4-test: 388.39031217165075

As we can see, the perplexities of Train, Test and Validate are of similar order. On the other hand, in case of statistical models with Kneyser-Ney and Witten Bell Smoothing, the perplexities were really low for training data, but high for testing data. For example,
Kneser-Ney Training Perplexity: 3.4081123363458143

Kneser-Ney Validation Perplexity: 16778.791735467417
Kneser-Ney Testing Perplexity: 26457.949957689852

Witten Bell Training Perplexity: 2.468052151268716
Witten Bell Validation Perplexity: 293.50395706835184
Witten Bell Testing Perplexity: 462.81717604751435

Thus, as we can see, Neural Models are learning on Semantic data, rather than just the counts of words occuring after one, which is the case in Siatistical Models. Thus, Neural Models give comparable performance acr training, testing and validation data, as they are not learning which words occur after which, but rather correlate it with the semantic content. Thus, we can even see worse performance of Training perplexities in Neural LMs as compared to Validation and Training, as it is not learning just on the counts from the training data.

We can also see that statistical neural models might also perform really good when there are few/no "unseen" n-grams and the unknown n-grams are handled well. It appears that Witten Bell works really good, but that might have been an error on my part on penalizing it lesser in unknown words and unknown contexts, as Kneyser-Ney seems to be a better estimate of what we would expect from a statistical model. As we can see in Kneyser-Ney, the perplexities on Brown Corpus are really small(single digit) for training,as it KNOWS for the training data what word will most likely occur based on its counts.However, it gives ~16k and ~26k errors for Training and Validation. It might also be the case that since the training corpus is quite large(35k sentences), the model has learnt enough to give good estimates, and Witten Bell is the true picture. However, even then, there is a huge difference in training errors and testing/validation errors.