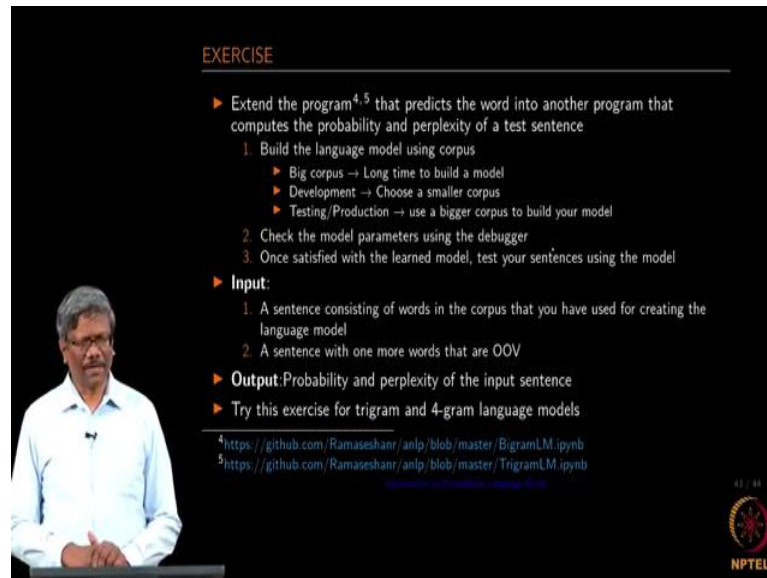


Applied Natural Language Processing
Prof. Ramaseshan Ramachandran
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Lecture - 28
Exercise

(Refer Slide Time: 00:15)



EXERCISE

- ▶ Extend the program^{4,5} that predicts the word into another program that computes the probability and perplexity of a test sentence
 1. Build the language model using corpus
 - ▶ Big corpus → Long time to build a model
 - ▶ Development → Choose a smaller corpus
 - ▶ Testing/Production → use a bigger corpus to build your model
 2. Check the model parameters using the debugger
 3. Once satisfied with the learned model, test your sentences using the model
- ▶ **Input:**
 1. A sentence consisting of words in the corpus that you have used for creating the language model
 2. A sentence with one more words that are OOV
- ▶ **Output:** Probability and perplexity of the input sentence
- ▶ Try this exercise for trigram and 4-gram language models

⁴<https://github.com/Ramaseshanr/anlp/blob/master/BigramLM.ipynb>
⁵<https://github.com/Ramaseshanr/anlp/blob/master/TrigramLM.ipynb>

43 / 44
NPTEL

So, this is the last part. I am going to be asking you to do this as an exercise. I am sure you would have noticed that I have spoken about the language modulus two things, one is predicting the next word, the second one is finding the probability of an input sentence, ok. I have shown some examples of how you can compute the probability of a sentence using a bigram model earlier, but now you have to really build one using the code that I have given. So, I have given you code to build the bigram and trigram model.

So, what you need to do now you have to input a sentence and then see what is the probability of a sentence that you have input to the given model. So, in this case, you have to do a few things. So, I am giving you step by step item that you need to do. One is you first built the model using a corpus, ok. So, take a small corpus as I mentioned earlier the earliest for your development and then once you are done with the testing you can go for a bigger model, ok.

And then check the model parameters in the debugger as I had shown earlier and then once you are satisfied with the learned model test your sentence using it. So, what you

are going to be doing here is once you build, the model building model is very similar to what we have seen earlier for the prediction, ok. So, you can use the same code that is available.

The input is going to be different here. In the cases of bigram and trigram word prediction the inputs are either one word or pairs of words for trigram, ok. So, in the case your input is going to be a sentence, you initially provide input with respect to words that you find in the context, ok. And then later you can try building the OOV and then create a model that includes OOV, and then later you can try the probability of a given sentence and so on, ok.

The output should be the probability of the input sentence, ok. So, you have to try this exercise for the trigram and four-gram language model, ok. So, do not try this for bigram. If you want to really get the hang of how it is done, you can go with the bigram try your exercise, the same exercise, but the exercise that you going to be the building here should be done for trigram and four-gram language models, ok. So, the code for the bigram language model is available and you can see at the footnote. I have given the link for those files. We can pick those files and then build your language model as mentioned in the exercise, ok, alright.

So, with this I conclude the language model building.