Homework 3 – Autocorrect

- Submitted by Pragya Shah (SC15B107)

Problem Statement: In this assignment you will be training a language model to build a spell checker. Specifically, you will be implementing part of a noisy-channel model for spelling correction. We will give the likelihood term, or edit model, and your job is to make a language model, the prior distribution in the noisy channel model. At test time you will be given a sentence with exactly one typing error. We then select the correction which gets highest likelihood under the noisy-channel model, using your language model as the prior. Your language models will be evaluated for accuracy, the number of valid corrections, divided by the number of test sentences.

Implement the following language models:

•Laplace Unigram Language Model

•Laplace Bigram Language Model

•Stupid Backoff Language Model

•A Custom Language Model

Evaluation : Your language models will be evaluated on the development data set. To help with your implementation, we give you the expected performance of each language model on the development set:

•Laplace Unigram Language Model: 0.11

•Laplace Bigram Language Model: 0.13

•Stupid Backoff Language Model: 0.18

•Custom Language Model: at least as good as Stupid Backoff, so 0.18.

Solution :

Making use of the starter codes provided, I implemented the following models -

Laplace Unigram Language Model is a unigram model with add-one smoothing. It treat all unknown words as a word seen zero times in training.

Laplace Bigram Language Model is similar to Laplace Unigram except that probabilities of current word depend on the previous word as well. It also has add-one smoothing.

Stupid Back-off Language Model uses an unsmoothed Bigram Model. It backs off to to an add-one smoothed Unigram Model when it encounters a pair of words it has never seen occurring together in the training set.

I have implemented an unsmoothed Trigram Language Model that backs off to an unsmoothed Bigram Model and further to an add-one smoothened Unigram Model. Thus it is a Nested Stupid Back-off Language Model starting with Trigram.

Source code:

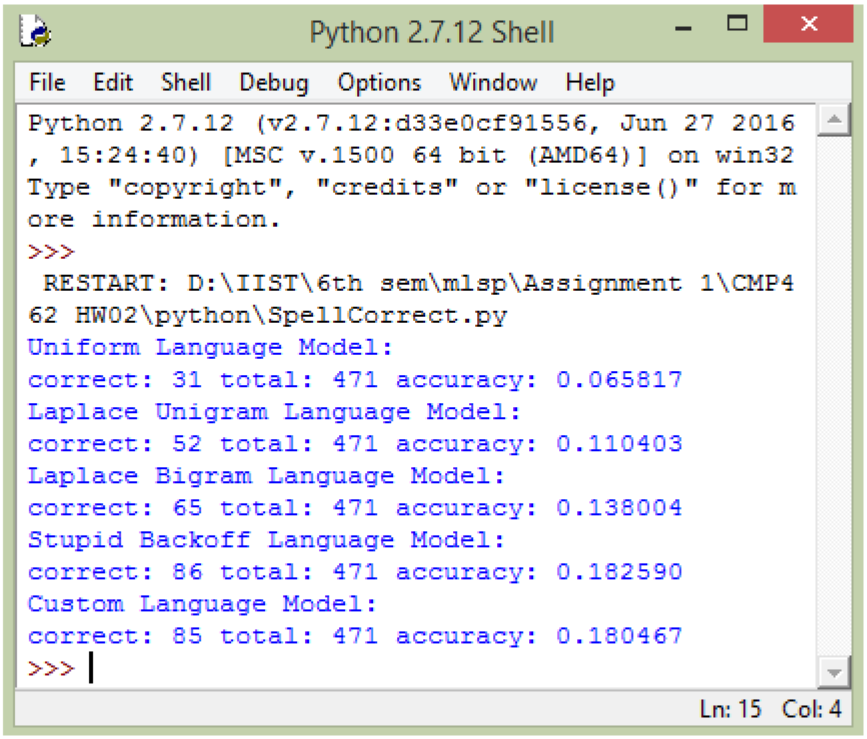
Look at python directory for source code. The important edited codes are present in files

- LaplaceUnigramLanguageModel.py
- LaplaceBigramLanguageModel.py
- StupidBackoffLanguageModel.py
- CustomLanguageModel.py

All other codes and files are identical to those downloaded from the course site, no changes have been made in them.

NOTE: The code uses Python 2.7.12 and might not be compatible with Python 3

Outputs:



```
Python 2.7.12 (v2.7.12:d33e0cf91556, Jun 27 2016
, 15:24:40) [MSC v.1500 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for m
ore information.
>>>
 RESTART: D:\IIST\6th sem\mlsp\Assignment 1\CMP4
62 HW02\python\SpellCorrect.py
Uniform Language Model:
correct: 31 total: 471 accuracy: 0.065817
Laplace Unigram Language Model:
correct: 52 total: 471 accuracy: 0.110403
Laplace Bigram Language Model:
correct: 65 total: 471 accuracy: 0.138004
Stupid Backoff Language Model:
correct: 86 total: 471 accuracy: 0.182590
Custom Language Model:
correct: 85 total: 471 accuracy: 0.180467
>>> 
```

Summary:

•Laplace Unigram Language Model accuracy - 0.110403

•Laplace Bigram Language Model accuracy - 0.138004

•Stupid Back-off Language Model accuracy - 0.182590

•Nested Stupid Back-off with Trigram Language Model (Custom Language Model) accuracy - 0.180467

Inference and Conclusion:

For the custom language model initially I used add-one smoothened trigram model for custom language model but its performance was just a bit better than the uniform language model case (about 0.07). This might be resulting from the fact that many three word combinations are possible and not all of them were present in our limited training data. We need a huge corpus to train such a model.

Using Nested Stupid Back-off improved the performance to a very great extent (more than 0.18). It was seen to be as good as Stupid Back-off Language Model with Bigram. This is obvious as now the algorithm backs off to bigram and unigram when it comes across a set of three words occurring together which it has not seen while training. In previous case this case was handled by add-one smoothing which gives a very low probability even though the word combinations are possible.

We can give the same argument for the better performance of Stupid Back-off compared to Laplace Bigram Language Model.

Laplace Bigram Language Model is better than Laplace Unigram Language Model as it models the fact that in real world, the words we speak are somewhat related to the words coming before them. The more we fit the various subtleties of speech in our language models, keeping in mind the limit on corpus size and computational powers of our systems, the better will be their performance.