Report of Homework 1: Spelling Correction

Name: Jialun Shen

Student id: 16307110030

My homework is in a single python file “spell_correction.py”, which can mainly be divided into four parts:

I. Language model

In the first part, I trained a language model based on corpose “reuters“ and texts in corpose “brown” whose category is “news”. I built a unigram model, a raw bigram without smoothing, a bigram model with add-1 (Laplace) smoothing., a bigram model with Keneser-Ney smoothing and a raw trigram model without smoothing. For each model, I defined a function to return the probability of a word (given its predecessors): Unigram_prob(*word*), Bigram_prob(*word_i, word_i-1*), Trigram_prob(*word_i, word_i-2, word_i-1*)

In this part, I also counted the total number of 1-letter and 2-letter combinations in my training set, which will be used in part II to calculate P($x|w$) in channel model.

II. Channel model

In the second part , I built a channel model based on the Peter Novig’s “count_1edit.txt” (see https://norvig.com/ngrams/count_1edit.txt). I calculated the confusion matrix from the txt, according to definition of the original article[1] instead of the lecture slide (which may be a bit confusing). The counts of 1 letter or 2-letter combinations on the denominator come from my training set (rather than Peter Novig’s original set), since what we concern is only the relative value of probability.

III. Spell Correction -- non word error

Let *wx* be the non word error word, *wc* be a candidate.

At first, I calculated the edit distance between *wx* and all words in vocabulary, which is extremely inefficient. Therefore, I changed the method by generating *wc*’s 1 or 2 edits away from *wx* first, then using known() function to get all valid candidates.

For each *wc* 1 edit(deletion, insertion, substitution, transposition) away from *wx*, function kind_of _edit(*wx*, *wc*) gives the kind of edit *e* and two letters *x, y* to refer to the confusion matrix. CM_prob(*wx, wc, p*) gives the channel model probability.

In most cases, candidates are with 1 edit. However, if there are no candidates within 1 edit, we have to look for candidates within 2 edits. An example is testdata#7, where “Taiwan” is typed as “Taawin”, which is like a “transposition”, say, “*xay” into “yax”*. A wise way is to look for such candidates in the vocabulary -- by testing if a word in vocabulary is made up of the same letters as *wx*.

Functions like best_candidate(*word, vocabulary, front_word*=None*, p*=0.95) gives the best candidate of word according to its predecessors(if applicable) on the basis of vocabulary. p(default 0.95) refers to P($w|w$) in channel model, the probability of typing correctly, which should have been used in real word correction.

[1] Kernighan, M. D. , Church, K. W. , & Gale, W. A. . (1990). A Spelling Correction Program Based on a Noisy Channel Model. COLING-90. Association for Computational Linguistics.

Function nonword_correction(*sentence, vocabulary*) corrects all the non word errors in *sentence* according to *vocabulary*.

A comparison between performances of different models is as follows:

Model	Accuracy	Time used (sec)
Unigram (raw)	86.80%	384.8067
Bigram (raw)	91.00%	403.0262
Bigram with add-1smoothing	89.20%	391.1983
Bigram with KN smoothing	90.70%	822.7818
Trigram (raw)	89.90%	199.4506

Even the simplest Unigram model has an accuracy of 86.80%. It is a little bit surprising that raw Bigram should perform better than both Bigrams with smoothing in accuracy. A possible explanation is that the size of the training set is relatively small (N=1821455, V=48025, including punctuation), which may also be the reason why raw Trigram does not excel in accuracy. Anyway, the performance of Bigram with Keneser-Ney smoothing is better than that with add-1 smoothing, which makes sense. Therefore, the code I hand in uses raw Bigram for higher accuracy.

IV. Spell correction -- real word error

Real word errors are significantly harder than non word errors to correct. I have tried to implement a real word corrector, but in vain. Worse still, real word correction takes much longer time. In my case, "real word correction" of 50 sentences takes about 130 seconds, though few real word is actually corrected.

For example, the first sentence in testdata.txt is

"They told Reuter correspondents in Asian capitals a U.S. Move against Japan might boost protectionst sentiment in the U.S. And lead to curbs on American imports of their products."

My real word error corrector gives

"They told Reuter correspondents in Asian capitals a U.S. *More* against *Jagan* *mighty* *boosts* *protectionist* *sentiments* in the U.S. And *dead* *go* curbs *own* *Americana* *import* *if* their *product*."

After I enlarged the training set to whole "brown" and "reuters", it gives

"They told Reuter correspondents in Asian capitals a U.S. *love* against *Jagan* *night* boost *protectionist* sentiment in the U.S. *Andy* *head* *do* *curds* *an* American *import* *on* *thei* *produces*."

While the answer in ans.txt is (also what non word error corrector gives)

"They told Reuter correspondents in Asian capitals a U.S. Move against Japan might boost *protectionist* sentiment in the U.S. And lead to curbs on American imports of their products."

The unlined italic words are words being corrected. My real word error "creator" would even create new real word errors.

I have observed mainly two kinds of real word errors in test data.

①Real word errors have something to do with grammar, like plural forms, past participle, or third person forms. For example, testdata#62:{cautiously, cautious}, testdata#65:{trade, traded}, testdata#494:{Germany, German}, testdata#998:{estimated, estimates}, etc.

②Typos. For example, testdata#55:{unions, onions}, testdata#139:{later, latter}, testdata#305:{world, word}, testdata#636:{produce, product}, etc.

By the way, there can also be a combination of non and real word error. An example is testdata#935, the wrong sentence is

"The *pisotl* earlier this week offred to buy the airline for 2.3 billion dlrs, and assume 2.2 billion dlrs of existing debt."

The correct sentence is

"The *pilots* earlier this week offered to buy the airline for 2.3 billion dlrs, and assume 2.2 billion dlrs of existing debt."

My non word error corrector have just corrected "pisotl" into "pistol", but correcting it into "pilots" would require long-range dependency, which is hard to achieve by such a preliminary model.