NLP Final Project Proposal

Elliot D Rappaport

April 10, 2014

The goal of this project was to research methods used to correct mis-typed words along with coding a program to accomplish this task. This is an example of the noisy channel problem in which the intended or correct word is replaced by another word that is not correct in the context that it is used. Some basic examples of this would be the use of the phrase "fifteen minuets" instead of "fifteen minutes," or "there car" instead of "their car." While both "minuets" and "minutes," and "there" and "their" are all words, they are not all correct in context.

1 Background

The noisy channel model is used a wide variety of applications in several subjects. It can be used in communication theory, for example, to predict the message that is sent over a channel with Gaussian or other noise. It is also used in NLP for several possible purposes. It can be used in speech recognition where the noisy channel is between one's mouth and the microphone and over the microphone itself. For the purposes of this project, the noisy channel is essentially the combination between one's hand and the keyboard. One word is intended, but that word is slightly modified over the channel. It can similarly be used to correct the results of an OCR system.

In order to predict the intended word there is essentially a single approach, with several variations, that dominates the literature. In order to predict what the intended word is, we want to maximize the probability $P(w|x)$, where w is the actual word and x is the noisy word that the algorithm receives. Using Baye's Rule we can equivalently maximize the product $P(x|w)P(w)$ [1, 2].

This can be described as a product between the channel model and the dictionary model. The channel model tells us the probability that a specific word could have turned into the received word, x. The dictionary model tells us the probability of a specific word (or N-gram) w showing up [1, 2].

In it's simplest form the channel model will give a higher probability for the lowest number of edits (where an edit is an insertion, deletion, substitution, or transposition). However, we

can also approach the probability more specifically. One option is to use some training set of labeled mistakes in order to determine which mistakes are more likely. Another option is to develop some heuristic to explain which mistakes are more likely. For example, we would say that a substitution of keys that neighbor each other is more likely than other substitutions [1, 2].

The dictionary model can also take a simpler or more complicated form. We can look at the probability of a single word showing up, or we can look at the probability of some N-gram showing up. Using a bigram or trigram will give us context and allow us to more intelligently determine the actual word [1, 2].

2 Implementation

The first step in the spelling correction process was to produce all words within one edit-distance of each of the words from the input sentence. This project used one edit-distance as the maximum because in general, about 80% of all spelling mistakes could be found in that one edit-distance. If there were reason to assume that the channel was particularly "noisy," the same algorithm could be used to add all words an additional edit-distance away to the candidate list.

After a list of candidate words is generated, each of these words is plugged into the model discussed above. For each specific edit, the probability of each edit is calculated. This is done using data from [3] which provides the number of each specific edit (for example the number of times ie is typed as ei: ei—ie). [3] also provides a count of all two-letter-grams allowing us to calculate a $P(x|w)$ where w is the actual word and x is the word with noise.

Next we must take the dictionary model into account. In order to do this we want to look at the 2 bigrams from either side of the candidate word - $P(word|prevWord)$ and $P(nextWord|word)$. Originally, I was planning on using requests to Google's N-gram database; however, this failed because Google cut me off (Error 429) from performing too many requests. I found several other large pieces of text and combined them all to create my own bigram probabilities with the help of the NLTK toolbox [4]. Unfortunately, this results in a significantly smaller corpus than the giant Google database, but still proved to work for the most part. Additionally, due to the smaller size of the available text, smoothing proved to be more important.

The probabilities of each candidate word and the original word are all computed, and at first the word with the maximum probability is chosen to be the intended word. In order to properly analyze the results, and edit probability (the channel model) must be assigned to the actual word typed. I tested several different options of this probability before settling

on assigning the typed word with a probability of 1e-7 while other edit probabilities were in the range of 1e-8 to 1e-9. After correcting some of the words from the first run, the same method is repeated until no more changes are made. This is performed so that noisy words that are surrounded by other noisy words are also corrected.

The final program outputs the top three possibilities for each word in the sentence, in increasing order of probability. These are printed for each successive try to edit the sentence.

3 Instructions for Use

The user inputs to the program a text file with the lines that need correction and an output file. The user may optionally use an additional argument specifying a directory that contains additional text files for the corpus. Adding more text files may help the program's performance. For most of my tests I used all the data found at http://corpus.byu.edu/full-text/formats.asp.

python SpellingCorrector.py ¡Input¿ ¡Output¿ [More Training]

4 Results

Testing this program provided two main challenges. Firstly, I could not locate a corpus of mistaken sentences. Therefore, in order to test the system I found some single examples and made up several other myself, a total of about 30 sentences with a varying number of mistakes in each sentence. While the test set is not large, the successes on this test set provide insight as to the successes and failures of the system. The second issue with testing was deciding how to quantify results. This analysis cannot simply count the number of mistakes corrected, because while trying to correct some mistakes occasionally other mistakes are added. Therefore, most of the analysis of this system is qualitative with some examples.

The following are examples of a sentence with different mistakes, including substitution, deletion, insertion, and transposition. With this example and all following examples, the caps words are meant to identify words of note.

```
I will go to bed in FITEEN minutes
I will go to bed in FFITEEN minutes
I will go to bed in FIFTAEEN minutes
I will go to bed in FIFTAEN minutes
I will go to bed in fifteen MINUETS
```

All of these examples were corrected properly to

```
I will go to bed in FIFTEEN MINUTES
```

The sentence below contains several mistakes all in a row.

```
Do you NED ANYTING ALSE FRM ME
```

Due to the fact that many of the incorrect words are surrounded by other incorrect words, the bigram model cannot fix them on the first loop. Therefore the first loop corrects it to the sentence below before correcting it fully.

```
Do you ned anyting alse FROM me
```

The following sentences provide a few more examples of sentences that were corrected properly.

```
My house is the one with the porch up FROUNT
Sometimes in the middle of the night I go to WACH tv
If you can read this you must be RAELLY smart
If you can read this you must be really SMRAT
I THING that this is important
he lived on the DESSERT island
```

Unfortunately, due to the limited corpus the system failed with some of the following examples.

```
he lived on the DESSERT ISLAN
she was a stellar and versatile ACRESS whose combination
```

The above sentences were corrected to

```
he lived on the DESSERT ISLAM
she was a stellar IND versatile ACRESS WHOLE combination
```

5 Conclusion

There is certainly room for improvement on this system, but it does seem to correct a majority of the mistakes. Many of the mistakes that do come up from the program's output seemingly could have been corrected with the Google N-Gram corpus instead of the smaller ones that are used.

References

[1] D. Jurafsky, "The noisy channel model of spelling - stanford nlp," Apr. 2012, https://www.youtube.com/watch?v=RgHr2KVXtiE.

[2] D. Jurafsky and S. Weiss, *NLP*. Hoboken, NJ: Wiley, 2010.

[3] P. Norvig, "Natural language corpus data: Beautiful data," Jul. 2011, http://norvig.com/ngrams/.

[4] E. L. Steven Bird and E. Klein, "Natural language processing with python," 2009, oReilly Media Inc.