CS6370 - Natural Language Processing - Spell Checker

Karthik Abinav (CS10B057) & Abhiram R (CS10B060)

September 22, 2013

Introduction

This report is presents a brief overview of the approach and algorithms that we used in the Spell Checker that we created as part of the first assignment in the course. The Spell Checker was implemented in *JAVA* and all the empirical training that was carried out was carried out using the *Brown Corpus* (with part of speech tagging). The *Dictionary* used is the default dictionary present in ubuntu.

Word Correction

Context free spelling correction - Use of Confusion Matrix

We used the confusion matrix based approach for correcting spelling errors without a context. In this approach we first hardcoded a confusion matrix for substitution,deletion addition of words. We populated a dictionary of English words from the UNIX dictionary. This contained about 65000 words. We then maintaned a Hash Table of length to the words list of words having that length. This structure helped us in pruning the search space when we needed to look at candidate words. We limited the search space to an edit distance of atmost 3 since, beyond this we found it unreasonable to make a good suggestion anyways.

Once we read the input word, we get the ArrayList of words from dictionary which differ in length by atmost 3. Beyond this the edit distance anyway would be greater than 3 and hence, its useless to fetch it. Once we have this list, we now simulate the edit distance algorithm. And we assign scores based on the confusion matrix. For example, suppose the input was "rainin", while performing edit distance algorithm we identify that the optimal way to convert this to "raining" is by inserting a "g" after the "n". Hence, we use this value from the addition matrix and return that as the score. Note that, all the confusion matrix are δ smoothed using Laplace smoothing.

This was the idea taken from the paper "A spelling correction program based on a noisy channel model" by Kerningham , et. al. However, we made a small tweak to the scoring system to accomodate for slightly better results. For example, given the input "rainin", the word

"rain" receives a higher score over "raining", since the values in confusion matrix suggests that adding the two respective characters occurs more frequently than deleting the "g". Hence, we added an offset to the score by adding the edit distance value to the score proportionally. Hence, words at edit distance 1 will receive more weigthage than words with edit distance 2. The proportion that was added was decided based on experimentation.

The key element to the code which increased the speed considerably was maintaining the Hash table structure from the word length to the list of words. This got down the search factor from asymptotic $O(|V|)$ to $O(log|V|)$, where $|V|$ is the size of the vocabulary. Though for the number of words in the dictionary, this is not a significant improvement, this still gives the code more time on finding the edit distance on an average case.

Bayesian Approach

This method was used to do a context based spelling correction for the sentences and phrases. This method was combined along with the trigrams based method, to obtain the final the suggestion. Also, the edit distance was included in the final score calculation for the suggestion.

This method was split into two parts. The first part is the training phase, where we used the Brown corpus to train the context and collocation words. Context, is the set of consecutive words that occurs before the given word. In this approach we used a k-context, that is the k previous words for a given word. After some experimenting, we found for k=20, the performance was the best. We also noted the k previous parts of speech that occured along in the context. Hence, the context feature had the context word and the parts of speech for various words occuring in the context. As for collocation, we used the k words that occured before the given word and the k words that occured after the given word. We found that for more than k=20, the collocation words no longer made sense. This also is a reasonable observation since, an average English sentence is usually about 15-17 words long. Hence, a window of 40 words accounts to approximately 2.5-3 sentences.

Once we trained our code on the Brown corpus, we serialised the various hash maps so that consequent runs of the program will be faster. Now, for an input, we tagged the parts of speech by using the most frequently seen parts of speech for that word in the corpus. This is an approximation since, some words have multiple parts of speech, and the parts of speech used in the sentence may not necessarily be the same as the one most frequently used. But, we found this to be a fair enough approximation in most cases. Once the parts of speech was tagged to the input, we then found the probability of $P(w_i|w_{i-1}.w_{i-2}..)$ and approximated this using the Naive Bayesian assumption (Here w_i is the mispelt word). We performed a similar thing on the parts of speech for the mispelt word. We then found the collocation words of the mispelt word in the given input and found an intersection with the confusion set's collocation in the corpus. The confusion set has been defined as the words with edit distance of atmost 3 from the mispelt word with words having lesser edit distance at a higher priority. We then assigned scores to these words in the confusion sets based on the amount on intersection and

the number of times they were found around the particular word in the corpus. We then combined the above three scores in a ratio of 1:1:1, to obtain a final score for the candidate words.

One assumption our code makes is that only one word in the given sentence is mispelt. Hence, if two words are mispelt in the input sentence, then a new parts of speech will be aassigned to the other mispelt word and then the first mispelt word will be given suggestions. Hence, if the two mispelt words form critical words for giving sense to the sentence, our code will give almost irrelevant suggestions. For example, "Hitt the rod" is the input the words "Hit" and "road" form the critical parts in giving sense and both are mispelt. On the other hand, suppose we have "Hit thee rod", in this case the code performs slightly better, although since "the" is an article and this part of speech plays an important role; this case is better than the previous case.

Part of Speech Trigrams

We also use part of speech trigrams for context sensitivity in spelling correction in sentences/phrases. The Brown corpus is POS- tagged. So we learn all POS trigrams, and use the learnt information to predict which word should replace the misspelt word in the sentence. We do this in the following way.

We first define a *Confusion set* of a misspelt word as the list of all words that are candidate replacements to this word. We obtain this set by passing the misspelt word through the word correction phase. We also then define a probability associated with every sentence. Suppose a sentence S = w_1 w_2 w_3 ... w_n. We associate a probability P(S) with this sentence S. Suppose w_k was the misspelt word in the original sentence. Let w'_k be a candidate replacement from the confusion set. We then associate a probability P(S') with the new sentence S', which has w_k replaced by w'_k. We evaluate these probabilities for every S' (i.e for every w'_k in the confusion set) using a measure defined below, and then suggest the word w'_l for which P(S') gives the maximum score. Rather, we present the confusion set as a list of suggestions, but in the decreasing order of P(S').

In order to evaluate P(S), we use the following method. We first find the parts of speech for every word in the sentence. This is obtained by training initially against a large set of words which are already POS tagged (The Brown Corpus, in our case). In case the part of speech for some word is unknown, we assign a special POS tag to it, called UNKNOWN. Let the part of speech of word w be T(w). We evaluate two measures using this information - The probability of a word occuring, given its part of speech T(w) i.e $P\ (w|T(w))$; and the POS trigram probabilities - $P\ (\ T(w_i)\ |\ T(w_{i-1}),\ T(w_{i-2})\ )$. We obtain these trigram probabilities using counts, as C($T(w_i)\ ,\ T(w_{i-1}),\ T(w_{i-2})$) /C($T(w_{i-1}),\ T(w_{i-2})$), the ratio of trigram to bigram counts. Using these, we then define the probability associated with a sentence S as

$$\mathrm{P(S)} = \prod P(w_i|T(w_i)) \prod P\ (\ T(w_i)\ |\ T(w_{i-1}),\ T(w_{i-2})\ )$$

This model essentially ignores the covariance/co-occurence between the words - the first product assumes that each word's occurence is independent of the words around it, and the second product assumes POS N gram probabilities where N = 3. This model works well to clearly identify the correct part of speech that occurs in place of the misspelt word, and suggests words of that part of speech in decreasing order of their counts within their respective parts of speech. But the model does not perform well in situations when multiple words in the confusion set are of the same part of speech, which is also the recommended part of speech to put in place of the misspelt word. We use the good aspects of this model to contribute 25% weightage to our final measure that we assign to every suggestion. We also use the Bayes Model as another measure, which tackles some of the disadvantages of using part of speech trigrams.

Observations on the Test Data

Here we give the score for some representative sentences.

- The parliament passed the resoltion to discuss the bil.
  Score: $\frac{1}{14}$
- Private hopitals to provide frea treatment to the poor.
  Score: $\frac{1}{5}$
- The fotball match was very interesting.
  Score: $\frac{1}{1}$
- The food served in the restarant was very godd.
  Score: $\frac{1}{1}$
- In great powrr lies great responsibility.
  Score: $\frac{1}{1}$
- To be or to bea is not the question.
  Score: $\frac{1}{14}$
- The crime raet seems to be under control.
  Score: $\frac{1}{6}$
- A great victry has come but at a great cort.
  Score: $0, \frac{1}{1}$
  Note in this case the other error doesnt get corrected. This is consistent with the fact that it looks for just one error in the sentence.