CS 6370: Programming Assignment

Dr. Sutanu Chakraborty

Yogesh B , EE12B066
Sagar J P , CS12B0
R Santhosh Kumar , EE12B101

03/10/2015

Word spell check:

In word spell check, we are given a single word which is not present in the dictionary and corrections are to be suggested for it.
We used Damerau Levenshtein distance as a metric to find the distance between 2 words. It accounts for insertion, deletion, or substitution of a single character, or a transposition of two adjacent characters. Computing this distance for a given word with respect to every word in the dictionary is computationally expensive. So we adopt alternative approaches.

We initially tried to reduce the search space using n-grams approach. We build a inverted index offline on this dictionary using the bigrams or trigrams. This reduces the search space. However, there is a major problem with this. Suppose a query word is of length 5 with the errors at 2nd and 4th position, we will never have to correct word in our search space. Ex: Actual word - HELLO, Query word - HWLKO

With the inverted index built using bigrams or trigrams we are never going to get a word hello. So this method does not work for such cases. To overcome this problem, we used the trie data structure which is built offline on the dictionary. It works only with distances that obey triangle inequality. But DamerauLevenshtein distance does not obey triangle inequality.

So we used Levenshtein distance in the trie data structure which only accounts for insertion, deletion, or substitution of a single character. So in this case substitution will cost 2 instead of 1. So when retrieving the words from trie we added a tolerance level to the actual distance within which we want the words. This tolerance level will take care of the extra cost when the transposition will be treated as 2 substitutions.

Now on this reduced set of words we run the actual DamerauLevenshtein distance to find the edit distance of the word from the query. To order the words in a rank order we used the noisy model approach. Our assumption is that a mistake in the query is independent of another mistake. So now we multiplied the probability of all mistakes to get the final probability of that word being the correction.

We obtained the probabilities from the dataset that is present in the norvig website. We built up the probabilities according to the formulas given below. Matrices are formed for each of the operation.

- del[x,y] is the number of times characters xy were typed as x in the training set
- add[x,y] is the number of times character x was typed as xy in the training set
- sub[x,y] is the number of times y was typed as x in the training set
- rev[x,y] is the number of times xy was typed as yx
- chars[x,y] is the number of times xy appeared in the training set
- chars[x] is the number of times x appeared in the training set

The likelihood is obtained as follows:

$$Pr(t|c) = \begin{cases} \frac{del[c_{p-1},c_p]}{chars[c_{p-1},c_p]} & \text{if } \textit{deletion} \\ \frac{add[c_{p-1},t_p]}{chars[c_{p-1}]} & \text{if } \textit{addition} \\ \frac{sub[t_p,c_p]}{chars[c_p]} & \text{if } \textit{substitution} \\ \frac{rev[c_p,c_{p+1}]}{chars[c_p,c_{p+1}]} & \text{if } \textit{reversal} \end{cases}$$

where c_p is the p^{th} character of c and t_p is the p^{th} character of t_p. The matrices are obtained for the del, add, sub, rev operations by using the list of correct words and the possible errors for that word. The $chars[x, y]$ is obtained by using the bigram frequencies given. $chars[x]$ is obtained from the word counts itself.
Once we have got probabilities for all the words we select the top 10 words from them and output them.

Context based Spelling Correction

Context-sensitive spelling correction is the problem of correcting spelling errors that result in valid words in the lexicon. These errors occur due to a variety of reasons like homonym confusion, typos or usage errors. To accomplish this task, a list of confusion words are first identified. A confusion set $w_1, ..., w_n$ implies that each word w_i in the set is ambiguous with every other word in the set and our task is to choose the best word given the context.
Two standard methods have shown to be effective in case of context based spelling correction - context words and collocations. The context words approach looks at the neighbouring words to correct a specific target word, whereas collocations look at local syntax. Both these methods have complementary advantages and effectively combining the methods would boost the performance. [1] proposes two methods to combine these - decision lists and bayesian classifier. Decision list looks at the best discriminative feature to classify a target word, whereas bayesian classifier looks at all features.

Context words

In the method of context word, given a target word that needs to be disambiguated, we look at the neighbouring words to infer about this word. For instance to disambiguate "desert" and "dessert", we can say that if the neighbours contain words like "arid", it's most probably desert, whereas if the neighbours contain words like "sweet", it is mostly "dessert". In our case, we looked at $+-k$ window around a target word for the task of disambiguation. k was chosen to be 3.

Let w_i be the target word to be disambiguated and c_j be the elements in the set of context words corresponding to the target word w_i. Then, by Baye's rule,

$$P(w_i|c_{-k}....c_k) = P(c_{-k}....c_k|w_i)P(w_i)$$

To simplify our task, we can approximate that the context are independent given the target word.

$$P(c_{-k}....c_k|w_i) = P(c_{-k}|w_i).....P(c_k|w_i)$$

We need to estimate the probabilities $P(c_j|w_i)$ which is done in the training phase. While training, we look at the whole corpus and count the total number of occurrence of the word w_i, which we denote as M. For each occurrence of w_i, we collect the list of words in the $\pm k$ window and also get their overall count. Let us denote each unique context word to be c_i and their corresponding counts to be m_i. Then,

$$P(c_j|w_i) = \frac{m_j}{M}$$

We shall discard those words whose occurrences are either minimum in the corpus (i.e.) we discard those words obeying,

$$\sum_{1 \leq i \leq n} m_i < T_{min}$$

and we also discard those words whose occurrences are very high leading to less discriminative power.

$$\sum_{1 \leq i \leq n} m_i > M - T_{min}$$

T_{min} was selected to be 10. Apart from these, it also becomes essential that we remove all those words which are not discriminating. For example, the word 'the' might occur in the context of almost every word in a confusion set and so it wont help us in disambiguation. To accomplish this task, we look at the entropy of occurrence of a context word given target words , i.e., we compute $P(c|w_i)$ for all $w_i \in$ confusion set and take the entropy of these probabilities. Higher the entropy, better is the discriminative power of a context words.
During testing, we parse through the phrase to identify if the word belongs to any of the confusion sets. If so, we pick all the context words in the k window around the target word and estimate the probability $P(w_i|c_{-k}....c_k)$ for all i in the confusion set as mentioned above. Then we pick the maximum probability among all the confusion words and identify it to be the correct word. If a context word doesn't occur around the target word in the training phase, we would get the probability as 0, which is not desirable. So, we add smoothing to prevent this situation. That is

$$P(c_j|w_i) = \frac{m_j + 0.5}{M + 0.5n}$$

k was selected to be 3.

Collocations

In this method, instead of looking at the context words, we look at the local syntax (i.e.) the pattern of syntactic elements around the target word. For this task, we look at the parts of speech tags of the sequence of $\pm k$ window words around the target word. Unlike the previous case of context words, we can't treat each of the parts of speech tags to be independent and consider a bag of words representation. The sequence becomes very important for collocations.
Apart from the sequence preservation, inference is similar to the context words. Let us assume that s_i be the sequence of parts of speech tags around the target word w_i. Then

$$P(w_i|s_i) = P(s_i|w_i)P(w_i)$$

As before, the priors and likelihoods are estimated in the training phase by collecting the total occurrences of a target word and also the sequences that occur with it. Infrequent sequences and non-discriminative sequences are removed and the residual sequences are used for disambiguation. While testing, the text is parsed to first identify if it belongs to one of the confusion sets. The sequence around the confusion set (s) is extracted, $P(w_i|s)$ is computed for each w_i in the confusion set, and the maximum is chosen. Smoothing is performed so as to take care of newly encountered sequence around the target word. k was selected to be 3.

Bayesian Classifier

This is a method combining Collocations and Context word features. There exists a complementarity between collocations and context words. Context words pick up generalities that are order independent, and collocations capture order dependent generalities. All the features (f_j) are listed and sorted in order of decreasing strength. Then, for a particular correction, evidence is gathered from all possible paths taken to achieve that correction, i.e, different word corrections can be followed by the same context and collocation correction. This way, the evidence is gathered for each correction. The confusion word with the highest evidence is selected as the correction (c_i).

$$score(c_i) = \frac{\sum_j P(c_i|f_j)P(f_j)}{\sum_i \sum_j P(c_i|f_j)P(f_j)}$$

The corrections are sorted by the scores and the top 3 are suggested.

Dataset

In our word check experiments, we used the Natural Language Corpus from ”norvig.com”. In our phrase and sentence check experiments, we used Brown corpus to extract the data needed for likelihood and prior computation. Confusion sets were gathered as follows - We first used the 18 confusion sets given in [1], then we looked at many online sources for list of commonly confused words. We collected all such lists and pruned them based on the number of occurrences in brown corpus. It turned out that Brown corpus was a small dataset as a result of which many confused sets were knocked out.

References

[1] Andrew R. Golding, *A Bayesion Hybrid Method for context-sensitive spelling correction.*

[2] Mark D. Kemighan, Kenneth W. Church, William A. Gale, *A Spelling Correction Program Based on a Noisy Channel Model.*