Vernacular Language Spell Checker & Autocorrection

Aniket Mishra, Tapish Sharma, Vividha Shinde

Students, Dept. of Computer Engineering, Rajiv Gandhi Institute of Technology, Mumbai, India

Abstract - *Spell checking applications are important part of several applications such as word processor, electronic dictionary, editors and search engines. It is a software tool which is used to detect and correct the misspelled words in a text document. Misspelled word can be a word that exists in the existing dictionary that is not correctly spelled or in detection component identifies the errors. After detecting the error, the suggestions are provided by the suggestion prediction component that are closer to the misspelled word. Spell checkers can be combined with other research areas focusing on linguistics like machine translation, information retrieval, natural language processing etc. or they can be clubbed with other software's like compilers, word processor software's. We will be developing a spell checker for Hindi language*.

Index Terms:

1. INTRODUCTION

As we know India is going through a transition and in this age of digitization more and more people have started using the internet. The apps and websites provide support for almost all languages. In India most people prefer Hindi language so this makes a need to develop a Hindi spell checker and autocorrection tool.

2. AIMS AND OBJECTIVES

To use natural language processing to implement spell checker. To provide easy to use User Interface. To increase the accuracy of spell-checking Objectives to ease the scanning the text and extracting the words contained in it. To compare each word with the known lists of correctly spelled words. This might might contain just a list of words, or it might also contain additional information, such as hypentation points or lexical and grammatical attributes. An additional step is a language-dependent algorithm for handling morphology. For many other languages such as those featuring agglutination and more complex declension and conjugation this part of the process is more complicated. This project focuses on correcting the real word errors and non-word errors. The project will help the users to make their documents error free.

3. RELATED WORK

A. *Spell Checker for Non-Word Error Detection: Survey By-Hema P. H.*, Sunitha C Computer Science and Engineering Vidya Academy of Science and Technology Thalakkotukara, Kerala, India*

Summary Abstract Spell checker is a software tool which is used to detect the spelling errors in a text document. A spell checker can also provide suggestions to correct the misspellings. The error can be either non word error or real word error. Detecting real word error is really difficult task and requires advanced statistical and Natural Language Processing (NLP) techniques. Currently we have many methods for detecting non word errors in a text document. This paper mainly deals with the non-word error detection methods for various languages.

B. *A study of spell-checking techniques for Indian Languages By-*Rakesh Kumar1, Minu Bala2, Kumar Sourabh3*

Summary Spell checker is one of the most important tools for any language to be digitized. A spell checker is a software tool / plugin that identifies and corrects any spelling mistakes in a text in a particular language. Spell checkers can be combined with other research areas focusing on linguistics like Machine translation, Information retrieval, natural language processing etc. or they can be clubbed with other software's like compilers, word processor software's. In this paper authors have made a study on the developmental approaches as well as roles of spell checker with respect to various applications based on Indian languages

C. *Using the Web for Language Independent Spellchecking and Autocorrection By-Casey Whitelaw and Ben Hutchinson and Grace Y Chung and Gerard Ellis Google Inc. Level 5, 48 Pirrama Rd, Pyrmont NSW 2009, Australia whitelaw, benhutch, gracec, ged@google.com*

Summary we have designed, implemented and evaluated an end-to-end system spellchecking and autocorrection system that does not require any manually annotated training data. The World Wide Web is used as a large noisy corpus from which we infer knowledge about misspellings and word usage. This is used to build an error model and an n-gram language model. A small secondary set of news texts with artificially inserted misspellings are used to tune confidence classifiers. Because no manual annotation is required, our system can easily be instantiated for new languages. When evaluated on human typed data with real misspellings in English and German, our web-based systems outperform baselines which use candidate corrections based on hand curated dictionaries.

4. IMPLEMENTATION PLAN

Error Detection Correction For error detection each word in a sentence or paragraph is tokenized by using a tokenizer and checked for its validity. The candidate word is a valid if it has a meaning else it is a non -word. Two commonly used techniques for error detection is N-gram analysis and dictionary/Wordnet lookup. Error correction consists of two steps: the generation of candidate corrections and the ranking of candidate corrections. The candidate generation process usually makes use of a precompiled table of legal n-grams to locate one or more potential correction terms. The ranking process usually invokes some lexical similarity measure between the misspelled string and the candidates or a probabilistic estimate of the likelihood of the correction to rank order the candidates. These two steps are most of the time treated as a separate process and executed in sequence. Some techniques can omit the second process though, leaving the ranking and final selection to the user. The isolated-word methods that will be described here are the most studied spelling correction algorithms, they are edit distance, similarity keys, rule-based techniques, n-gram-based techniques, probabilistic techniques, neural networks and noisy channel model. All of these methods can be thought of as calculating a distance between the misspelled word and each word in the dictionary or index. The shorter the distance the higher the dictionary word is ranked.

Error Correction method Correcting spelling errors in a text document is an important problem. The error correction mainly includes two steps, they are generating candidate words and ranking the candidate words. The spelling correction mainly focuses on the isolated words and will not consider the context in which the word appears. Today we are having the following types of error correction algorithms; Minimum edit distance method, Soundex algorithm, Rule based Technique, N-gram based technique, Probabilistic method, Neural net techniques

Minimum Edit Distance Method Minimum edit distance is the number of operations required to transform one text to other. The operations can be insertion, deletion, substitution etc. Most of the edit distance algorithm gives the integer distance scores. Edit distance is useful for correcting errors formed from keyboard input. It is not preferable for correcting phonetic error

N-gram based Technique N-Grams are n letter sub sequences of words or strings where n usually is one two or three. This can be used for error correction in text document. N gram can be used along with dictionary or without a dictionary. Without dictionary N-gram can be used to find the position where the misspelled word occurs. Together with a dictionary, n- grams are used to define the distance between words, but the words are always checked against the dictionary. This can be done in several ways, for example check how many n-grams the misspelled word and a dictionary word have in common, weighted by the length of the words.

Rule Based Technique The method will have a list of rules from common spelling error patterns, and it will be applied on the misspelled words, and will generate a list of candidate words for the error correction.

Soundex Algorithm, the Soundex system is mainly used in the phonetic spelling correcting situations. The method in which a Soundex code will be generated for all words in the dictionary. And the algorithm will generate a Soundex code for a misspelled word and will retrieve words from the dictionary which having the same code.

Neural networks is an important technique. The current methods are based on back-propagation networks, using one output node for each word in the dictionary and an input node for every possible n-gram in every position of the word, where n usually is one or two. Normally only one of the outputs should be active, indicating which dictionary words the network suggests as a correction.

Probabilistic method it is a simple method in which two common methods are used they are transition probabilities and confusion probabilities. Transition probability depends on the language. It gives the probability of a letter followed by another letter. The probability can be estimated by calculating n gram frequency for the words from a corpus. Confusion probability gives the value of how often a letter is mistaken or substituted for another letter. They are source dependent.

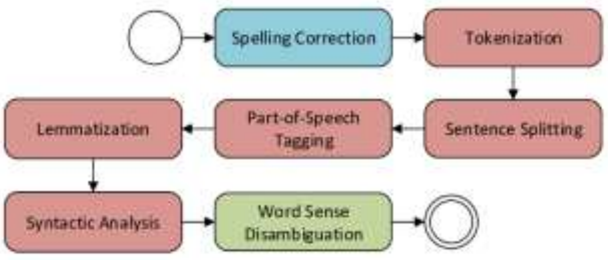

Fig. 1. Implementation Plan.

1) *Unigram Language Model:* This model is defined by

$P(w_1 .... w_k ..... w_K) = P(w_k) = count(w_k)/N$

That is, it ignores all context and simply returns the probability of the focus word. count(wk) is the number of times the word wk occurred in the corpus and N is the size of the corpus. Again, this model is not a true probability function either since the sum over all sentences is much greater than 1. Using this model results in a system that picks the most frequent word of the candidate replacements.

1) *Before Bigram Language Model:* This model includes the previous word as context:

$P(w_1 .... w_k .... w_K) = P(w_{k-1} w_k) = count(w_{k-1} w_k)/N$

Where count uses a distance sensitive bigram count.

is the number of times that bigram appeared in the corpus.

1) *After Bigram Language Model:* This model includes the following word as context:

$P(w_1....w_k........w_K) = P(w_k w_{k+1}) = count(w_k w_{k+1})/N$

where count (wk wk+1) is the number of times that bigram appeared in the corpus.

The rest of the language models are more complex and use all K words of context. We start by conditioning the probability on wk:

$P(w_1....w_k....w_K) = P(w_1....w_{k1} w_{k+1}..................w_K|w_k) * P(w_k)$

Now technically our notation is sloppy: a probability should have both a random variable and its value. We will show both from here on, as it will cause confusion later on if we don't. We say Wi is the random variable which word is in the Ith slot of the K-word window? wi is the particular word that Wi is taking on.

To repeat:

$P(W_1 = w_1, ..., W_k = w_k, ..., W_K = w_K) = P(W_1 = w_1, ..$

$, W_{k1} = w_{k1}$

$W_{k+1} = w_{k+1}, ..., W_K = w_K | W_k = w_k) * P(W_k = w_k)$

Now, we make the Naive Bayes Assumption:

$P(W_1 = w_1, ..., W_{k1} = w_{k1}, W_{k+1} = w_{k+1}, ... ,$

$W_K = w_K | W_k = w_k)$

$= \quad P(W_i = w_i | W_k = w_k)$

$if=k$

Notice that this is not the Bag of Words assumption. The locational distance between the words (—k i—) is still there. We now propose our first model that uses this directly.

The Spaced-Bigram Model: The model is simply:

$P\ (W_1 = w_1...., W_k = w_k\ ...., W_K = w_K) = P\ (W_k = w_k) *$

$P(W_i = w_i | W_k = w_k)$

$if=k$

where we compute

$P\ (W_k = w_k) = c(w_k)/N$

as just a simple unigram count estimated from a training corpus and

$P\ (W_i = w_i | W_k = w_k) = P\ (W_i = w_i$

$, W_k = w_k)/P\ (W_k = w_k) = c(w_i, w_k, ki)/c(w_k)$

This model has the advantage that it will learn different statistics for adjacent words. However, the parameter space is effectively K times larger than a traditional bigram, and this model may suffer from under-training. All models we propose may benefit from smoothing and other techniques to adjust the estimated probabilities. Indeed, in the experiments we use Laplace Smoothing. Notice the case of K = 3 effectively becomes the application of traditional bigrams, using a window of size 3.

5. METHODOLOGY

Clearly, our methodology is highly dependent upon the corpus that we use. There were nearly 30M words in the corpus with around 1.17 Lac unique words. The corpus is noisy i.e. it contains mis-spelled words. So, we try to eliminate noise by not considering with words with low frequencies. We have also implemented a basic GUI in which users can type in Hindi and check for non-word and real word errors. All our code and implementation has been done in python. We use a dictionary with word, frequency pairs as our language model. A lookup into the dictionary categorizes a word as correct or erroneous. To produce candidate corrections, we calculate strings at edit distance one and two from the identified erroneous string and further filter out those strings that are not present in the dictionary. The edit distance used is Damerau- Levenshtein edit distance. This gives us a set of words that are possible corrections for the erroneous word. To produce a ranking among these words, we sort these candidates in increasing order of edit distance. Words at same edit distance are sorted in order of their frequencies. To deal with real word errors we create 2-grams and 3-grams along with their frequencies of occurrence. To check for real word errors, every 2-gram. 3-gram and 4-gram of the sentence is checked in the created set. If the frequency of the gram is low, it is raised as an error. To produce corrections for the erroneous gram, we calculate edits of each of the word in the gram and construct valid candidate grams from these. Again, ranking is done based on edit distance and frequency of the grams.

REFERENCES

[1] rams.

http://en.wikipedia.org/wiki/N-gram, Accessed on 21/8/2011.

[2] George A. Miller and Katherine Miller. Introduction to wordnet: An on-line lexical database. 1993.

[3] Eleni Katsoulotou. Master thesis: Semi-Automatic Marking of Diagrams.