ISSN:0975-9646

Spell Checker for OCR

Yogomaya Mohapatra, Ashis Kumar Mishra, Anil Kumar Mishra

Department of Computer Science & Engineering, Orissa Engineering College
Bhubaneswar, Odisha, Pin-752050, India

***Abstract*— the implementation focuses a systematic approach to the design the Spell Checker for OCR. In this a spelling correction system, is designed specifically for OCR-generated text, that selects candidate words through the information gathered from multiple knowledge sources and automatically replaces with the correct word. This system for text correction based on approximate string matching, which uses a statistical model that incorporates techniques like Confusion Matrix and N-gram Analysis. The ability to accurately recognize characters by scanning hard copy images is extremely important for many forms of automated data processing and has wide application. A great deal of effort has been devoted to correcting errors which invariably result from commercially available OCR devices. Besides error patterns like substitution, transposition, insertion and deletion, emphasis is given on modifiers and their positions with respect to the consonants and conjuncts being modified. The system is developed using file management system through java and java Swing for the Windows operating system.**

***Keywords*— Spell Checker, OCR, OCR-generated text, Confusion Matrix, N-gram Analysis.**

I. Introduction

Recent advances in printed document digitization and processing led to large scale digitization efforts of legacy printed documents producing document images. To enable subsequent processing and retrieval, the document images are often transformed to character-coded text using Optical Character Recognition (OCR). Although OCR is fast, OCR output typically contains errors. The introduced errors adversely affect linguistic processing and retrieval of OCR documents. Depending on the application of the optically scanned text, large post processing effort can be necessary. Since OCR is often used to move large amounts of text to electronic form, the proofreading is a task both demanding and dull. This makes the need for good tools of spell checking and correction large and urgent. When a human being reads a document, he uses a wide spectrum of knowledge in making sense of what is written. On the other hand, a machine when presented with optically digitized text, in the absence of such knowledge, is prone to making mistakes in reading the text. The goal of OCR is to transform a document image into character-coded text. This usual process is to automatically segment a document image into character images in the proper reading order using image analysis heuristics, the applying an automatic classifier to determine the character codes that most likely correspond to each character image, and then exploit sequential context (e.g., preceding and following character and a list of possible words) to select the most likely character in each position. The character error rate can be influenced by reproduction quality (e.g., original documents are typically better than photocopies), the resolution at which a document was scanned, or noise either inherent in the document or introduced by the digitized process, any mismatch between the instances on which the character image classifier was trained and rendering of characters in the printed document. Due to the noise, the machine reading process may not be able to recognize a character (reject error) or may recognize a character incorrectly (substitution error). The noise some times also cause character fusions (i.e. two or more character images merge to appear as a single connected component) and or character fragmentation (i.e. a character image is fragmented into more than one sub image). The character fusion and fragmentation may lead to substitution and rejection errors besides making the word length shorter or longer than the actual length. One of the common ways of correcting these errors is to make tagged corpora of the language of the text to check if the OCR output word is valid word. A still higher level of knowledge such as grammar of the language of the text may also employed to check the output of the OCR. In this implementation, some of the issues concerning correction of optically read Oriya character strings using an Oriya tagged corpora is examined and a correction strategy with results of experimentation is suggested. Spell checker provide a ready mechanism for post processing the OCR output for corrections. A spell checker takes into account the process by which spelling errors usually get introduced while suggesting the corrections. A spell checker tries to model this process. A model is usually based on character phonetic proximity, character key label proximity on the key board, character interchange, missing character, an extra character, character repetition etc. in another approach, one estimates the likelihood of a spelling by it's frequency of occurrence that is derived from the transition probabilities between characters. This requires a priori statistical knowledge of the language [3]. One of the most common used models incorporating the OCR process is a statistical model that incorporates Confusion Matrix. The confusion matrix captures the substitution errors that OCR makes during the testing phase. The confusion matrix thus obtained is representative only when tested over a large sample space. It also depends upon the OCR methodology and feature vector space classification. This confusion matrix is used for hypothesizing the substitution errors and suggesting correction. The new hypothesized words have to be checked in the corpora. The correction process, while correcting the substitution errors, has to take the individual character confidence figures or probabilities into account. The character fusion and fragmentation pose further problems for the correcting process. The correction process needs an appropriate criterion to select the most likely candidate when more than one hypothesized words find match with the corpora. A trie-structure has been employed to obtain a set of next possible characters of the corpora while traversing the OCR output word character by character. In another approach, a string matching technique is used to find the best match. It is reasonable to assume an accuracy of 70% or more at the OCR level. For a word length 3, on an average at least two character are correctly recognized. The rest of character may or may not be in error. The other approach, popularly used in the OCR problems, is the N-gram approach. Construction of appropriate N-gram from raw text data is an important issue in this approach. The N-gram frequencies of the corrected text and the optically scanned text were compared and N-grams that showed large frequency differences between text versions were displayed to the editor, together with a concordance of all the occurrences of the N-gram. This allowed the editor to formulate a correction rule for the N-gram under consideration [1][2][3][4].

A BRIEF OVERVIEW OF INDIAN LANGUAGES

The use a language for any application its characteristics are required to be known. Once this is known, the application can make use of languages in a most uniform manner. Indian scripts have a very different structure and have communality amongst them. They follow almost same rules; but the way of representing them is different. Indian Scripts have tremendous applications in day-to-day life. These applications include Word Processing, Database Management, machine Translation, OCR, etc. Once the characteristics of these scripts are known, making use of them for any of these scripts are known, making use of them for any of these application is possible. These are 15 officially recognized Indian Scripts. These scripts are broadly divided into two categories namely Brahmi scripts and Perso-Arabic scripts consists of Devanagari, Gurumukhi, Gujarati, Oriya, Bengali, Assamese, Telugu, Kannada, Malayalam and Tamil. The Perso-Arabic scripts include Urdu, Sindhi and Kashmiri. Devanagari script is used by Hindi, Marathi and Sanskrit languages. The characteristics of the languages within the family are quite peculiar. They have a common phonetic structure, making the common character set.

A.STRUCTURE OF INDIAN SCRIPTS

Since the origin of all these scripts is same, they share a common phonetic structure. The alphabet may vary slightly and also the graphical shapes. Using these characteristics a transliteration facility between any Indian scripts is possible. Typically the alphabets get divided in to following categories: **the Constants**: all Indian scripts use 5 types of constants groups called varga. Some vowel like ‘a’ is included in the constant category. Each varga has 5 consonants, with primary and secondary pairs. The second consonant in each pair is derived from the first consonant with ‘h’ sound and have separate graphical representation. Other consonants not present in this category are YA, RA, LA, VA, SA, HA and invisible consonants like, RA (halant) and (halant) RA, get formed differently.

Vowels: all the vowels are represented by separate symbols. These vowels are placed on the consonants either in the beginning or after the consonant. Each of these vowels is pronounced separately. Typical vowels are:

Vowel: A, I, Ee, u, U, ru, Ee
Usage: Ka, Ki, Kee, Ku, KU, Kru, and Kee
Vowel: e, E, a, o, O, au, ao
Usage: Ke, KE, KA, Ko, KO, Kau, Kao

Halant: while forming the conjuncts use of broken consonants is activated by halant. On mixing of two or more consonants the shape of the conjunct varies. Many a times halant is required to indicate the vowel-less ending. e.g. Ramnathan.

Punctuations and Numerals: all the punctuations and numerals are common between English and Indian Scripts.

OCR SYSTEM

When the page of a text is scanned into a PC, it is stored as an electronic file made up of tiny dots, or pixels; it is not seen by computer as text, but rather as a “picture of text”. Word processors are not capable of editing bit map images. In order to turn the group of pixels into editable words, the image must go through a complex process known as Optical Character Recognition (OCR). OCR research began in the late 1950’s and since then, the technology has been continually developed and refined. In the 1970’s and early 1980’s, OCR software was still very limited-it could only work with certain typefaces and sizes. These days, OCR software is far more intelligent, and can recognize practically all typefaces as well as severely degraded document images [3].

One of the earliest OCR techniques was something called matrix or pattern matching. OCR programs, which use the pattern matching method, have bitmaps stored for every character of each of different front and type sizes. By comparing a database of stored bitmaps the program attempts to recognize the letters.

Feature extraction was the next step in OCR’s development. This attempted to recognize characters identifying their universal features, the goal being to make OCR typeface-independent. If all the character could be identified using rules defining the way that loops and lines join each other, then individually letters could be identified regardless of their typeface. For example: the letter “a” is made from a circle, a line on the right side and an arc over the middle. So, if a scanned letter had these “features” it would be correctly identified as the letter “a” by the OCR program. Feature extraction was a step forward from matrix matching, but actual results were badly affected by poor-quality print. Extra makes on the page, or stain in the paper, had a dramatic effect on accuracy. The elimination of such “noise” became a whole research area itself. Once noise can be identified, the reliable character fragments can then be reconstructed into the most likely letter shapes.

A.DIFFERENT STAGES OF OCR SYSTEM

Optical recognition devices are currently used to convert printed material into ASCII text for automated information retrieval.OCR system consists of four major stages: (1) Pre-processing (2) Segmentation (3) Feature Extraction (3) Classification (4) Post-processing.

APPROACHES TO ERROR CORRECTION FOR OCR

The problem of detecting error in words and automatically correcting them is a great research challenge. It’s solution has enormous application potential in text and code editing, computer Aided Authoring, Optical Character Recognition(OCR), Machine Translation(MT), Natural Language Processing(NLP), Database Retrieval and Information Retrieval Interface, Speech Recognition, Text to Speech and Speech to Text Conversion, Communication

System for the disabled (e.g. blind and deaf), Computer Aided Tutoring and Language Learning, and Pen-Based Computer Interface. The word error can belong to one of the two distinct categories, namely nonword error and real-word error[3][4].

Let a string of character separated by spaces or punctuation marks be called a candidate string. A candidate string is valid word if it has meaning. Else, it is nonword. By real word error we mean a valid but not the intended word in the sentence, thus making the sentence syntactically or semantically ill-formed or in correct. In both cases the problem is to detect the erroneous word and either suggest correct alternatives or automatically replace it by the appropriate word.

There are several issues to be addressed in the error correction problem. The first issue concerns the error patterns generated by different text generating media such as typewriter and computer keyboard, typesetting and machine printing, OCR system, speech recognizer output, and of course, handwriting. Usually, the error pattern of one media does not match with that of the other. The error pattern issue of each media concerns the relative abundance of insertion, deletion, substitution and transposition error, run-on and split word error, single versus multiple character error, word length effect, positional bias, character shape effect, phonetic similarity effect, heuristic tendencies etc. the knowledge about error pattern is necessary to model an efficient spellchecker.

Another important issue is the tagged corpora which concerns the size of the corpora, the problem of inflection and creative morphology, word access techniques and so on. The other approach, popularly used in OCR problems, is the N-gram approach. Construction of appropriate N-gram from raw text data is an important issue in this approach. The detection of real word error needs higher level knowledge compared to the error detection of nonword error. In fact, detection of real word error is a problem that needs NLP tool to solve. Several approaches based on minimum edit distance; similarity key rules, N-grams, probability and neural nets are proposed to accomplish the task. Of these, minimum edit distance based approaches are the most popular ones. The minimum edit distance is the minimum number of editing operations (insertion, deletions and substitutions) required to transform one text string to another. The distance is also referred to as Damerau-Levenshtein distance after the pioneers who proposed it for text error correction. In its original form, minimum edit distance algorithms require 'm' words. After comparisons, the words with minimum edit distance are chosen as correct alternatives. To improve the speed, a reverse minimum edit distance is used where a candidate set of words is produced by first generating every possible single error permutation of the misspelled string and then checking the corpora for any make up valid word. Spell checker are either stand-alone applications capable of operating on a block of text, or as a feature of a larger application, such as a word processor, email client, electronic dictionary, or search engine[11][12].

Simple spell checker operate at the word level, by comparing each word level, by comparing each word in a given input against a vocabulary (often referred to as a dictionary). If the word is not found within the vocabulary, it is designated erroneous, and algorithms may be run to detect which word the user most likely meant to type. One simple such algorithm is listing words from the dictionary with a small Levenshtein distance from the typed word.

As already outlined, a spell checker customarily consists of two parts: (1) A set of routines for scanning text and extracting words (2) A corpora against which the words found in the text are compared.

An interactive spell checker can also be helpful in correction. In the simplest case, the checkers remember all tokens which the user has indicated should be replaced and the words with which the tokens are to be replaced. After the first such replacement, future occurrences of the misspelled token can be automatically corrected. A similar approach is including common misspellings in the dictionary with the correct spelling. This approach has not been included in any current spell checkers, probably because of the lack of an obvious source of known misspellings and the low frequency of even common misspellings. 80% of all spelling errors are result of:

- Transposition of two letters
- One letter extra
- One letter missing
- One letter wrong

The basic algorithm for correction is for each token which is not found in the corpora, construct a list of all words which could produce this token by one of the above rules. These are called candidate spellings. If the list has exactly one candidate, guess that word. If the list contains several words, then N-gram approach is applied. Transpositions can be detected by transposing each pair of adjacent characters in the token, one at a time, and searching for the resulting token. If the resulting token is found, it is a candidate and is added to the list. For a token of WL characters, this requires **WL-1** searches of the dictionary. Checking for extra letter requires deleting each character one at a time, and searching for the resulting token. This requires additional **WL** searches. Most tokens are short so this need not be expensive. The remaining two types of errors (**one missing letter and one wrong letter**) are difficult to detect. A search with a match any character feature cannot stop when the first word match is found, but most continue, since many words may match the token. This requires **WL+1** searches for a missing letter and **WL** searches for a wrong letter error.

For a **wrong letter** in the **third** or **subsequent character**, all words which are candidates must exist on the same chain that the suspect token hashes to. Hence, each entry on that chain is inspected to determine if the suspected differs from the entry by exactly one character. This is accomplished by an exclusive-or (XOR) between the suspect and the dictionary. Then a JFFO instruction selects the first nonzero byte in the XOR. This byte is zeroed and if the result is all zero, then the dictionary words differs from the suspect in only one letter. All such words are listed at CANDBF, where they can be inspected later. For a **wrong letter** in the **first** or **second character**, the program tries varying the second later through all **26** possible values, searching for an exact match. Then all **26** possible values of the first letter are tried, after setting the second letter to its original value.

This means that **52** more chains are searched for possible matches. To correct transposed letters, all combinations of transposed letters are tried. There is only **WL-1** such combinations, so it is fairly cheap to do this. To correct one extra letter, **WL** copies of the token are made, each with some letter removed. Each of these is looked up in the dictionary. This takes **WL** searches. To correct one missing letter, **WL+1** copies of the token are made, each time inserting a NULL character in a new position in the suspect. The NULL character is never part of any word, so the suspect token augmented by an embedded NULL can be thought of as a word with one wrong letter (the NULL). Then algorithm for matching one wrong letter is used. If the first character is omitted. Counting, we find that a total of **3*WL+103** chains must be searched, with **WL** such chain searches requiring a special algorithm to look for (exactly) one wrong character.

The implementation describes a new automatic spelling correction approach to deal with OCR generated errors. The method used is based on three principles.

- Approximate string matching between the misspellings and the term occurring in the corpora.
- The use of confusion matrix, which contains information inherently specific to the nature of errors caused by the particular OCR device.
- The other approach, popularity used in OCR problems, is the N-gram approach. Construction of appropriate N-gram from raw text data is an important issue in this approach.

A .DETAILED DESCRIPTION OF THE PROCESS

The system includes a document scanning device, which may comprise an optical scanner or a facsimile machine. Scanning device scans an input original document and generates an image signal that is representative of the characters appearing on document. After performing an OCR algorithm on the image signal, OCR module creates an electronic document that includes recognized words intended to correspond exactly, in spelling and in arrangement, to the words of the original document.

Although the recognized words of the electronic document should match all the corresponding words of the original document, a complete match some times does not occur. Any incorrect words in the electronic document that is flagged by one of these algorithms as incorrect is referred to as a misrecognized word. The spell checking algorithm is capable of generating at least one alternative word for each incorrect word. These alternative words are referred to as reference words, in the next step is to select the reference word that is most likely the correct word for replacing the identified incorrect word. This selection is accomplished by calculating a replacement word value for each reference word. The next step replaces incorrect word with reference word that has been assigned the highest replacement word value [3][4].

CONFUSION MATRIX

In the field of artificial intelligence a confusion matrix is a visualization tool typically used in supervised learning (in unsupervised learning it is typically called a matching matrix).

Each column of the matrix represents the instances in a predicted class, while each row represents the instances in an actual class. One benefit of confusion matrix is that it is easy to see if the system is confusing two classes (i.e. commonly mislabelling one as another) [5].

When the data set is unbalanced (when the number of samples in different classes vary greatly) the error rate of a classifier is not representative of the true performance of the classifier. This can easily be understood by an example. If there are for example 990 samples from class 1 and only 10 samples from class 2, the classifier can easily be biased towards class 1. If the classifier classifies all the samples as class 1, the accuracy will be 99%. This is not a good indication of the classifier's true performance. The classifier had a 100% recognition rate for class 1 but a 0% recognition rate for class 2.

N-GRAM APPROACH

N-grams are sequences of characters or words extracted from a text or an N-gram is a sub-sequence of n items from a given sequence. N-gram are used in various areas of statistical natural language processing and genetic sequence analysis. The items in question can be letters, words or base pairs according to the application. An N-gram of size 1 is a "unigram"; size 2 is a "bigram"(or, more etymologically sound but less commonly used , a "digram"); size 3 is a "trigram"; and size 4 or more is simply called an "N-gram". Some language models built from N-gram are "(n-1)-order Markov models".

N-grams can be divided in two categories: (1) character based and (2) word based. A character N-gram is a set of n consecutive characters extracted from a word. The main motivation behind this approach is that similar words will have a higher proportion behind this approach is that similar words will have a high proportion of N-grams in common. Typical values for n are 2 or 3; these correspond to the use of bigrams or trigrams, respectively. There are n+1 such bigrams and n+2 such trigram in a word containing n characters. Character based N-gram are generally used in measuring the similarity of character strings. Spell checker, stemming, OCR error correction are some of the applications which use character based N-grams. Word N-grams are sequences of n consecutive words extracted from text. Word level N-gram models are quite robust for modelling language statistically as well as for information retrieval without much dependency on language[4][5][6].

N-GRAM BASED LANGUAGE MODELLING

Informally speaking, a language is modelled by making use of linguistic and common sense knowledge about language. Formally, a language model is a probability distribution over word sequences or word N-gram. Specifically, a language model (LM) estimates the probability of the next word given preceding words. A word N-gram language model uses the history of n-1 immediately preceding words to compute the occurrence probability P of the current word. The value of N is usually limited to 2 (bigram model) or 3 (trigram model). If the vocabulary size is M words, then to provide complete coverage of all possible N word sequences the language model needs to consist of M^N-

grams (i.e. sequence of N words). This is probability expensive (e.g. a bigram language model for a 40,000 words vocabulary will require 1.6 x 10^9 bigram pairs), and many such sequences have negligible probabilities. Obviously, it is not possible word pairs. Typically, an N-gram LM lists only the most frequently occurring word pairs, and uses backoff mechanism to compute the probability when desired word pair is not found.

When used for language modelling independence assumptions are made so that each word depends only on the last *n* words. The Markov model is used as an approximation of the true underlying language. This assumption is important because it massively simplifies the problem of learning the language model from data. In addition, because of the open nature of the language, it is common to group words unknown to the language model together [3][4].

N-gram models are widely used in statistical natural language processing. In speech recognition, phonemes and sequences of phonemes are modeled using a N-gram distribution. For parsing, words are modeled such that each N-gram is composed of *n* words. For language recognition, sequences of letters are modeled for different languages. For a sequence of words, (for example "the dog smelled like a skunk"), the trigrams would be: "the dog smelled", "dog smelled like", "smelled like a", and "like a skunk". For sequences of character, the 3-grams(sometimes referred to as "trigram") that can be generated from "good morning" are "goo", "ood", "od ", "d m", " mo", "mor" and so forth. Some practitioner's pre-process strings to remove spaces, most simply collapse whitespace to a single space while preserving paragraph marks. Punctuation is also commonly reduced or removed by pre-processing. N-grams can also be used for sequences of words or, in fact, for almost any type of data. N-gram models are often criticized because they lack any explicit dependency range is (n-1) tokens for an N-gram model, it is also true that the effective range of dependency is significantly longer than this although long range correlations drop exponentially with distance for any Markov model[9][10].

The hypothesis is that differences in observed frequency between correct text and optically scanned text for a character N-gram would indicate that the N-gram in the question was incorrectly recognized by the scanning process. The N-gram frequencies of the corrected text and the optically scanned text were compared and N-grams that showed large frequency differences between text versions were displayed to the editor, together with a concordance of all the occurrences of the N-gram. This allowed the editor to formulate a correction rule for the N-gram under consideration. Two sets of rules were formulated, one with rules that replaced a character trigram with another string of optional length, the other with rules that replaced a string of optional length with another. The rules that rewrite trigrams were generated with the support of a graphical tool that generated a list of suspect trigrams, for each trigram showed a concordance of all the occurrences of the trigram, and with a correction given by the user, could generate a correction rule. The rules were then used to correct both the article that has been used to generate the rules and the other to see if the rules were useful in another context than the one they had been generated in. the number of errors generated in the correction process were counted separately to keep track of over correction[12].

N-GRAMS FOR APPROXIMATE MATCHING

N-grams can also be used for efficient approximate matching. By converting a sequence of items to a set of N-grams, it can be embedded in a vector space (in other words, represented as a histogram), thus allowing the sequence to be compared to other sequences in an efficient manner. For example, if we convert string with only letters in the English alphabet into 3-grams, we get a 26^3 – dimensional space(the first dimension measures the number of occurrences of "aaa", the second "aab", and so forth for all possible combinations of three letters). Using this representation, we lose information about the string. For example, both the string "abcba" and "bcbab" give rise to exactly the same 2-grams. However, we know empirically that if two strings of real text have a similar vector representation (as measured by cosine distance) then they are likely to be similar. Other metrics have also been applied to vectors of N-grams with varying, and sometimes better results [5][6].

N-GRAM APPLICATIONS

N-gram finds use in several areas of computer science, computational linguistics, and applied mathematics.

They have been used to:

- Design kernel allows machine learning algorithms such as support vector machines to learn from string data.
- Find like candidates for the correct spelling of a misspelled word.
- Improve compression in compression algorithms were a small area of data requires N-grams of greater length.
- Assess the probability of a given word sequence appearing in text of a language of interest in pattern recognition systems, speech recognition, **OCR** (optical character recognition), intelligent character recognition(ICR), machine translation and similar applications.
- Improve retrieval in formation retrieval system when it is hoped to find similar "document" (a term for which the conventional meaning is sometimes stretched, depending on the data set) given a single query document and a database of reference documents.
- Improve retrieval performance in genetic sequence analysis as in the BLAST family of programs.
- Identify the language a text is in or the species a small sequence of DNA was taken from.
- Predict letters or words at random in order to create text, as in the dissociated press algorithm.

.DETECTING MISSSPELLED WORDS IN ORIYA CORPORA USING SYLLABLE N-GRAM FREQUENCIES

Here, a system is designed and implemented which decides whether or not a word is misspelled in Oriya Corpora. Firstly, a database of syllable, bigram and trigram frequencies is constructed using the syllables that are derived from different Oriya corpora. Then, the system takes words in Oriya text as an input and computes the probability distribution of words using syllable, bigram and trigram frequencies from the database. If the probability distribution of a word is zero, it is decided that this word is misspelled. For testing the system, two text databases are constructed with the same words. One text database has 685 misspelled words. The other has 685 correctly spelled words. The words from these text databases are taken as input for the system. The system produces two results for each word: "Correctly spelled word" or "Misspelled word". The system that is designed with monogram and bigram frequencies has 86% success rate for the misspelled words and has 88% success rate for the correctly spelled words. According to the system designed with bigram and trigram frequencies, there is 97% success rate for the misspelled words and there is 98% success rate for the correctly spelled words[6][7][8][9].

In automatic spelling correction, the probability of the sequence of characters Y produced by a possibly imperfect typist given the hypothesized word series W is estimated by using a mistyping model for the word series W.

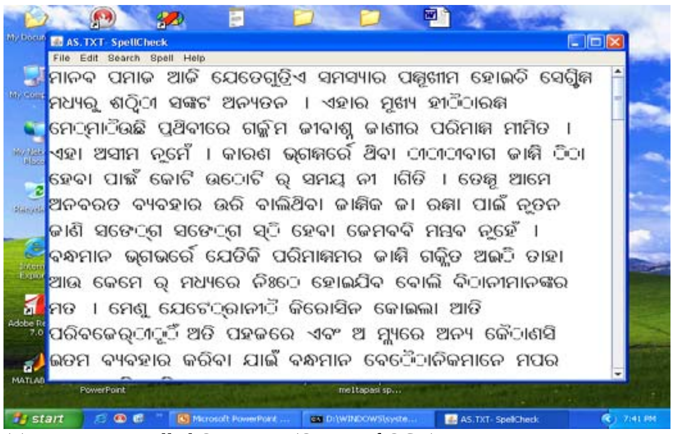
In these types of applications, the probability of the word series W can be modelled according to the equation:
$P(W_1^k) = P(W_1)\,P(W_2|W_1)\ldots\ldots\ldots\ldots..P(W_k|W_1^k\text{-}1)$ (2)
Where W_1^k represents a series of words W_1, W_2... Wk.

In the conditional probability $P(W_k|W_1^k\text{-}1)$ the term $W_1^k\text{-}1$ is called the history or the predictor feature and represents the initial (k-1) words of the series. Each word in the history is a predictor word. The term W_k is called the predicted feature or the category feature [12].

The mechanism for estimating the conditional probabilities in Equation (2) is a language model. A language model estimates the conditional probabilities from limited training text (training data). The larger the training text, and the larger the number of parameters in the language model, the more accurate and precise are the predictions from the language model [11][12].

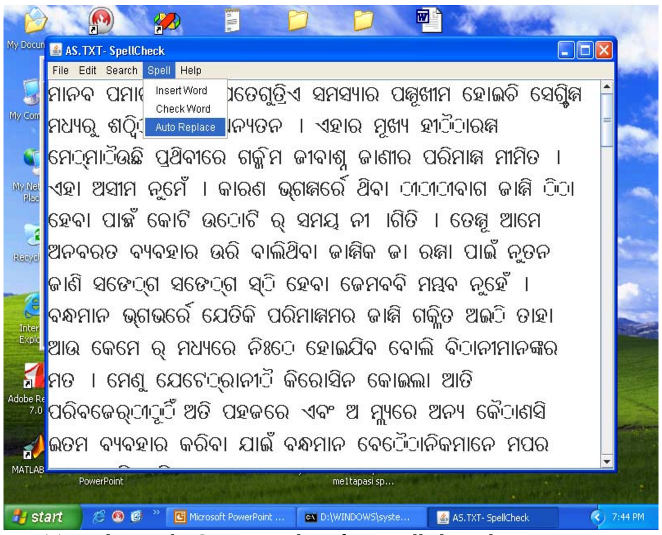
As stated above, a purpose of language model is to assign probabilities to a word series, e.g., the probability of a trigram W_1 W_2 W_3; given that bigram W_1 W_2 has just occurred.

Recently the successful model is the trigram model. The model is based upon deleted interpolation. This model requires the storage of records that identify: (a) a trigram id W_1 W_2 W_3 and its count C (W_1 W_2 W_3); (b) a bigram identification W_2 W_3 and its count C (W_2 W_3); and(c) a unigram identification W_3 and its count C (W_3). The count of a given trigram is the number of occurrences of this given trigram in the training data.



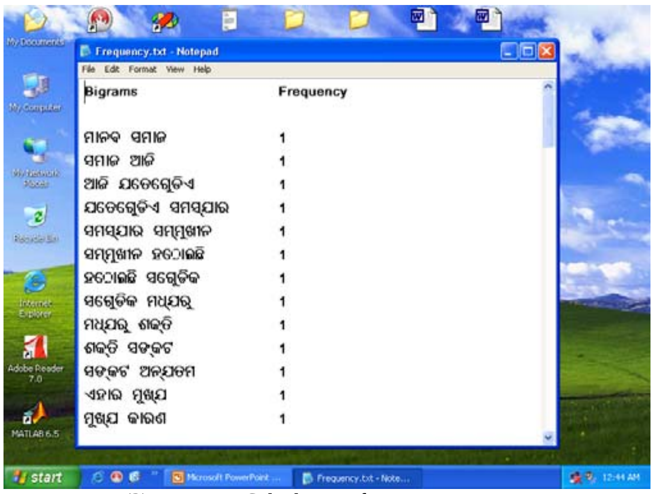
ମାନବ ପମାଜ ଆଜି ଯେତେଗୁଡ଼ିଏ ସମସ୍ୟାର ପକ୍ଷୁଖୀନ ହୋଇଚି ସେଗୁଡ଼ିକ
ମଧ୍ୟରୁ ଶକ୍ତିଏ ସଙ୍କଟ ଅନ୍ୟତନ । ଏହାର ମୁଖ୍ୟ ହୀଁ ାରଣ୍
ମେ୍ମାିଉଛି ପୃଥିବୀରେ ଗଚ୍ଛିମ ଜୀବାଶ୍ମ ଜାଣୀର ପରିମାଣ ମୀମିତ ।
ଏହା ଅସୀମ ନୁହେଁ । କାରଣ ଭୂଗର୍ଭରେ ଥିବା ଏଏଏବାଗ ଜାଣି ିା
ହେବା ପାଇଁ କୋଟି ଉଡୋଟି ର୍ ସମୟ୍ ନୀ ।ଗିତି । ତେଣୁ ଆମେ
ଅନବରତ ବ୍ୟବହାର ଉରି ଚାଲିଥିବା ଜାଣିଜ ଜା ରକ୍ଷା ପାଇଁ ନୂତନ
ଜାଣି ସେଡେ୍ଗ ସେଡେ୍ଗ ସୃି ହେବା ଡେମବର୍ବି ମମ୍ଭବ ନୁହେଁ ।
ବର୍ତ୍ତମାନ ଭୂଗଭର୍ରେ ଯେତିକି ପରିମାଣମର ଜାଣି ଗଚ୍ଛିତ ଅଇି ତାହା
ଆଉ କେମେ ର୍ ମଧ୍ୟରେ ନିଃଚେ ହୋଇଯିବ ବୋଲି ବିାନୀମାନଙ୍କର
ମତ । ମେଣୁ ଯେଟେ୍ରାନୀି କିରୋସିନ ଜୋଇଲା ଆଡି
ପରିବଜେର୍ଏୂିଁ ଅତି ପହଜରେ ଏବଂ ଅ ମୂଲ୍ୟରେ ଅନ୍ୟ କୌାଣସି
ଇତମ ବ୍ୟବହାର କରିବା ଯାଇଁ ବର୍ତ୍ତମାନ ବେଚୈାନିକମାନେ ମପର

(1) Misspelled Oriya Text(Output of OCR)



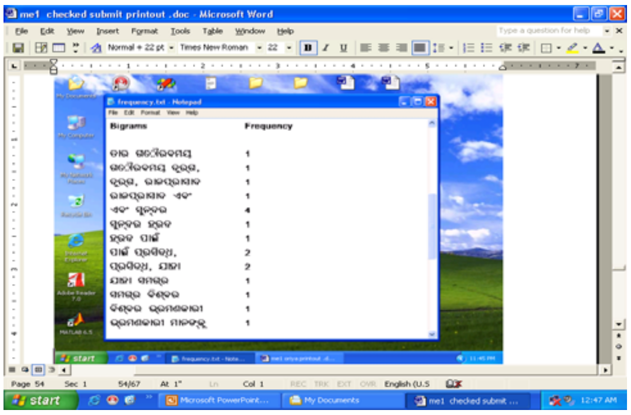
ମାନବ ପମା ଯେତେଗୁଡ଼ିଏ ସମସ୍ୟାର ପକ୍ଷୁଖୀନ ହୋଇଚି ସେଗୁଡ଼ିକ
ମଧ୍ୟରୁ ଶକ୍ତି ନ୍ୟତନ । ଏହାର ମୁଖ୍ୟ ହୀଁ ାରଣ୍
ମେ୍ମାିଉଛି ପୃଥିବୀରେ ଗଚ୍ଛିମ ଜୀବାଶ୍ମ ଜାଣୀର ପରିମାଣ ମୀମିତ ।
ଏହା ଅସୀମ ନୁହେଁ । କାରଣ ଭୂଗର୍ଭରେ ଥିବା ଏଏଏବାଗ ଜାଣି ିା
ହେବା ପାଇଁ କୋଟି ଉଡୋଟି ର୍ ସମୟ୍ ନୀ ।ଗିତି । ତେଣୁ ଆମେ
ଅନବରତ ବ୍ୟବହାର ଉରି ଚାଲିଥିବା ଜାଣିଜ ଜା ରକ୍ଷା ପାଇଁ ନୂତନ
ଜାଣି ସେଡେ୍ଗ ସେଡେ୍ଗ ସୃି ହେବା ଡେମବର୍ବି ମମ୍ଭବ ନୁହେଁ ।
ବର୍ତ୍ତମାନ ଭୂଗଭର୍ରେ ଯେତିକି ପରିମାଣମର ଜାଣି ଗଚ୍ଛିତ ଅଇି ତାହା
ଆଉ କେମେ ର୍ ମଧ୍ୟରେ ନିଃଚେ ହୋଇଯିବ ବୋଲି ବିାନୀମାନଙ୍କର
ମତ । ମେଣୁ ଯେଟେ୍ରାନୀି କିରୋସିନ ଜୋଇଲା ଆଡି
ପରିବଜେର୍ଏୂିଁ ଅତି ପହଜରେ ଏବଂ ଅ ମୂଲ୍ୟରେ ଅନ୍ୟ କୌାଣସି
ଇତମ ବ୍ୟବହାର କରିବା ଯାଇଁ ବର୍ତ୍ତମାନ ବେଚୈାନିକମାନେ ମପର

(2)Displaying the Suggestion list of Misspelled words



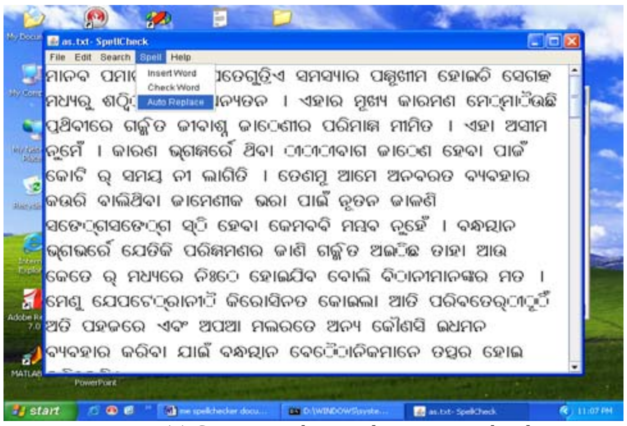
Bigrams	Frequency
ମାନବ ସମାଜ	1
ସମାଜ ଆଜି	1
ଆଜି ଯେତେଗୁଡିଏ	1
ଯେତେଗୁଡିଏ ସମସ୍ୟାର	1
ସମସ୍ୟାର ସମ୍ମୁଖୀନ	1
ସମ୍ମୁଖୀନ ହୋଇଛି	1
ହୋଇଛି ସେଗୁଡିକ	1
ସେଗୁଡିକ ମଧ୍ୟରୁ	1
ମଧ୍ୟରୁ ଶକ୍ତି	1
ଶକ୍ତି ସଙ୍କଟ	1
ସଙ୍କଟ ଅନ୍ୟତମ	1
ଏହାର ମୁଖ୍ୟ	1
ମୁଖ୍ୟ କାରଣ	1

(3)Frequency Calculation of Bigrams


me1 checked submit printout .doc - Microsoft Word
File Edit View Insert Format Tools Table Window Help
Normal + 22 pt
Times New Roman
22
frequency.txt - Notepad
File Edit Format View Help
start
Page 54 Sec 1 54/67 At 1" Col 1 English (U.S
Microsoft PowerPoint...
My Documents
me1 checked submit ...
12:47 AM

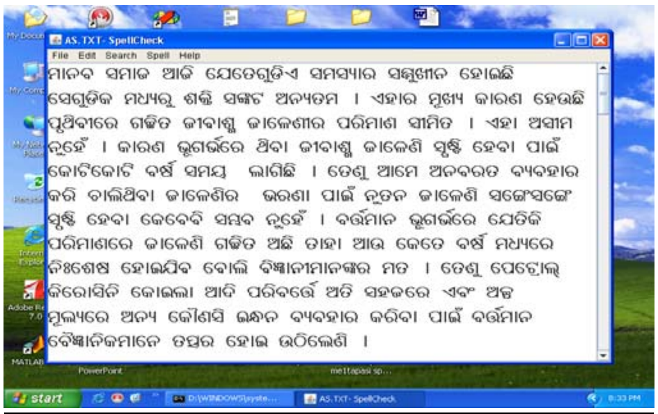
Bigrams	Frequency
ତାର ଖଣ୍ଡିତବାସ୍ତୁ	1
ଖଣ୍ଡିତବାସ୍ତୁ ସୂରଣ,	1
ସୂରଣ, ରାଜପ୍ରଭାବର	1
ରାଜପ୍ରଭାବର ଏବଂ	1
ଏବଂ ସୃଜନର	4
ସୃଜନର ହଇବ	1
ହଇବ ପାଇଁ	1
ପାଇଁ ପ୍ରଶିକ୍ଷ,	2
ପ୍ରଶିକ୍ଷ, ଯାହା	2
ଯାହା ସମାଜର	1
ସମାଜର ବିକାଶର	1
ବିକାଶର ଉପାଦେୟତା	1
ଉପାଦେୟତା ସାମଗ୍ରୀ	1

(4)Frequency Calculation of Bigrams



ମାନବ ପମା ଯେତେଗୁଡ଼ିଏ ସମସ୍ୟାର ପଶ୍ମୁଖୀମ ହୋଇଚି ସେଗୁଡ଼
ମଧ୍ୟରୁ ଶଡ଼ି ଅନ୍ୟତନ । ଏହାର ମୁଖ୍ୟ କାରମଣ ନେହୁମାଁଉଛି
ପୃଥିବୀରେ ଗଜ୍ଜିତ ଜୀବାଶ୍ମ ଜାଡେଣୀର ପରିମାକ୍ଷ ମୀମିତ । ଏହା ଅସୀମ
ନୁମେଁ । କାରଣ ଭଗ୍ଗକ୍ଷର୍ରେ ଥିବା ଏଏଏବାଗ ଜାଡେଣ ହେବା ପାଙ୍ଗ
କୋଟି ର୍ ସମୟ ନୀ ଲାଗିତି । ତେଣମୁ ଆମେ ଅନବରତ ବ୍ୟବହାର
କଉରି ବାଲିଥିବା ଜାମେଣୀକ ଭରା ପାଇଁ ବୁତନ ଜାଳଣୀ
ସଡୋଏ୍ରଗସଡୋଏ୍ରଗ ସ୍ଠି ହେବା କେମବବି ମମ୍ଭବ ନୁହେଁ । ବନ୍ଧଯ୍ରାନ
ଭଗ୍ରଭର୍ରେ ଯେତିକି ପରିକ୍ଷମଣର ଜାଣି ଗଜ୍ଜିତ ଅଇଁଛ ତାହା ଆଉ
କେତେ ର୍ ମଧ୍ୟରେ ନିଃଡେ ହୋଇଯିବ ବୋଲି ବିଠାନୀମାନଙ୍କର ମତ ।
ମେଣୁ ଯେପଟେଏ୍ରାନୀଁ କିରୋସିନତ କୋଇଲା ଆତି ପରିବଡେର୍ଏଡୁଁ
ଅତି ପହଜରେ ଏବଂ ଅପଆ ମଲରତେ ଅନ୍ୟ କୌଣସି ଇଧମନ
ବ୍ୟବହାର କରିବା ଯାଇଁ ବନ୍ଧଯ୍ରାନ ବେଠୌାନିକମାବେ ତତ୍ପର ହୋଇ

(5) Correction phase under Execution head



ମାନବ ସମାଜ ଆଜି ଯେତେଗୁଡ଼ିଏ ସମସ୍ୟାର ସମ୍ମୁଖୀନ ହୋଇଛି
ସେଗୁଡ଼ିକ ମଧ୍ୟରୁ ଶକ୍ତି ସଙ୍କଟ ଅନ୍ୟତମ । ଏହାର ମୁଖ୍ୟ କାରଣ ହେଉଛି
ପୃଥିବୀରେ ଗଚ୍ଛିତ ଜୀବାଶ୍ମ ଜାଳେଣୀର ପରିମାଣ ସୀମିତ । ଏହା ଅସୀମ
ନୁହେଁ । କାରଣ ଭୂଗର୍ଭରେ ଥିବା ଜୀବାଶ୍ମ ଜାଳେଣି ସୃଷ୍ଟି ହେବା ପାଇଁ
କୋଟିକୋଟି ବର୍ଷ ସମୟ ଲାଗିଛି । ତେଣୁ ଆମେ ଅନବରତ ବ୍ୟବହାର
କରି ଚାଲିଥିବା ଜାଳେଣୀର ଭରଣା ପାଇଁ ନୂତନ ଜାଳେଣି ସଙ୍ଗେସଙ୍ଗେ
ସୃଷ୍ଟି ହେବା କେବେବି ସମ୍ଭବ ନୁହେଁ । ବର୍ତ୍ତମାନ ଭୂଗର୍ଭରେ ଯେତିକି
ପରିମାଣରେ ଜାଳେଣି ଗଚ୍ଛିତ ଅଛି ତାହା ଆଉ କେତେ ବର୍ଷ ମଧ୍ୟରେ
ନିଃଶେଷ ହୋଇଯିବ ବୋଲି ବିଜ୍ଞାନୀମାନଙ୍କର ମତ । ତେଣୁ ପେଟ୍ରୋଲ୍
କିରୋସିନି କୋଇଲା ଆଦି ପରିବର୍ତ୍ତେ ଅତି ସହଜରେ ଏବଂ ଅଳ୍ପ
ମୂଲ୍ୟରେ ଅନ୍ୟ କୌଣସି ଇନ୍ଧନ ବ୍ୟବହାର କରିବା ପାଇଁ ବର୍ତ୍ତମାନ
ବୈଜ୍ଞାନିକମାନେ ତତ୍ପର ହୋଇ ଉଠିଲେଣି ।

(6) Output after Correction Phase

CONCLUSIONS

Research into algorithmic techniques for detecting and correcting spelling errors in text has a long, robust history in computer science. As an amalgamation of the traditional fields of the Artificial intelligence, pattern Recognition ,String Matching, Computational Linguistics, and others, this fundamental problem in information science has been studied from the early 1960's to the present. As other technologies matured, this area of research has become more important than ever. This statistical and language independent nature of N-gram model seems suitable for dealing with a multilingual collection of texts. Improving retrieval efficiency from Indian language document by N-gram will be my future effort. The use of N-gram language modeling for information retrieval, text categorization and Machine Translation has to be investigated.

The development of valuable system "Spellchecker for OCR" was of unique experience. In course of carrying out the project work, I found myself in growing field of software development area. There is always a room for improvement in any S/W package however efficient it may be. But the most important things are that the system should be flexible enough for future modification.

The system "Spellchecker for OCR" has been designed in such a manner that modifications may be incorporated without affecting the behavior and working of any modules. Work can also be extended to context dependent error correction research that requires information from the surrounding context for detection and correction. The preference is given to the mappings that are known OCR confusions. The evaluation results cover a variety of corpora and shows that post correction improves the quality even for scanned texts with a very small number of OCR errors. There are two important points to be considered in future work. First the influence of the sentence context ranking to the correction result should be studied in more depth. It is desirable to complete a series of experiments in order to clarify the dependence on the domain and language used and how much improvements it yields. The second point is the further exploration of the correction based on the output of two different OCR engines. Further development and evaluation of this technique would be very valuable for the whole OCR field.

REFERENCES

[1] Sandor Dembitz, Peter Knezevic Mladen Sokele," Developing a Spell Checker as an Expert System", Journal of Computing and Information Technology-ICCIT, 2004.

[2] Dustin Boswell, "Language Models for Spelling Correction", CSE256, Spring 2004.

[3] Li Zhwang, TaBao, Xiaoyan Zhu, Chunheng Wang, Satoshi Naoi " OC R spelling Check Approach Based on Statistical Language Models", International Conference on Systems, Man and Cybernetics, Hague, Netherlands, IEEE, Oct 2004.

[4] Gerasimos Potamianos, Frederick Jelinek, "A Study of N-gram and Decision tree Letter Language Modeling Methods", Speech Communication, 1998.

[5] K. Kukich, "Techniques for Automatically Correcting Words in Text", ACM Computing Surveys, 1992.

[6] Munirul Mansur, Analysis of N-gram Based Text Categorization in a Newspaper Corpus, Undergraduate thesis (Computer Science) BRAC University, August 2006.

[7] William B. Canvar. John M. Trenkle "N-gram based Text Categorization" Proceeding of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval.

[8] R.Anglell, G. Freund, and P.Willett, "Automatic spelling correction using a trigram similarity measure", Information Processing & Management, 19,(4),305-316, (1983).

[9] C.Y.Suen, "N-gram statistics for Natural Language Understanding and Text processing", IEEE Trans. On Pattern Analysis & Machine Intelligence. PAMI, 1(2), pp.164-172, April1979..

[10] Naira Khan, Md. Tarek Habib, Md. Jahangir Alan, Rajib Rehman, Naushad Uzzaman and Mumit Khan, "History (forward N-gram) or Future (Backward N-gram)? Which model to consider for N-gram analysis? Proc. Of 9th international Conference on computer and information Technology (ICCIT 2006), December 2006.

[11] Research paper on post editing through approximate global correction By…. Julie Borsach (Information Science and Research Institute).

[12] Bruno Martins, Mario J.Silva, "Spelling Correction for Search Engine Queries", 2004.