APPROXIMATE STRING MATCHING AND

APPLICATIONS TO INDIAN SCRIPTS

By

VENKATA RAVINDER PARUCHURI

Bachelor of Technology in Computer Science and Engineering

JNT University

Hyderabad, India

2009

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
July, 2011

APPROXIMATE STRING MATCHING AND APPLICATIONS TO INDIAN SCRIPTS

Thesis Approved:

Dr. Subhash Kak

Thesis Adviser

Dr. Johnson Thomas

Dr. David Cline

Dr. Mark E. Payton

Dean of the Graduate College

ACKNOWLEDGMENTS

It gives me immense pleasure in acknowledging all those people, without whom, this thesis would have been impossible.

I owe my deepest gratitude and respect to my Thesis Advisor Dr. Subhash Kak who has been a guiding light throughout. It was his enthusiasm, inspiration and patience which have always prompted me into exploring this topic further. It was his timely advice and guidance that led me this far. I could not have imagined having a better advisor and mentor for my Masters degree.

I would like to take this opportunity to extend my gratitude and thanks to Dr. Bill Picking and Wade Price for supporting me through my period of study at OSU. I have worked on one of the most interesting projects under them. They mentored and taught me how research is done for which I am grateful to them.

I would also like to thank Dr. Johnson Thomas for encouraging me with his valuable comments and thought provoking questions which led me to a conclusive research. I would also like to thank Dr. David Cline for his patience and support as the graduate coordinator who has helped me on numerous occasions.

I would like to thank my friends, Anvesh Reddy Aileni, Singireddy Rohith, Yashwanth Kothapalli, Pradeep Kumar Dantala, Siddarth Eechampati, Nitesh Reddy Rentam, Bindu Visireddy, Harikishan Kotha, Ravi Teja Gunda, Sanath Chilakala, Vijay Singh, Akila Devi, Sindhujaa Chandrababu, Rohit Vaidya, Shashank Sadalia for always being there for me.

I would like to thank my entire extended family for always believing in me and giving me constant support in everything I intended to do. I would like to thank my sister, Laxmi Ravali Paruchuri whom I could totally depend on for moral support.

Lastly and most importantly, I would like to dedicate my thesis to my parents Vidya Sagar and Raja Kumari, for believing in me at every point of my life and teaching me to be a better human being.

TABLE OF CONTENTS

LIST OF TABLES

Table Page

LIST OF GRAPHS

Figure Page

CHAPTER I

INTRODUCTION

Approximate String Matching

Approximate string matching is one of the major problems in classical string matching and it is used to find techniques suitable for extracting information from database where there may be a spelling mistake or an error in a keyword [1]. This is an important issue for information retrieval, pattern recognition, web search, computational biology and text mining. Looking for a person's name in a database is the most common operation in information systems. It gives exact results for the lookup if the name is given correctly, but if the person's name is misspelled, then there should be some procedure to obtain the correct result. In approximate string matching the match is measured in the number of operations that are performed to match the exact string. The goal is to perform string matching of a pattern in a text where one or both of them have suffered some kind of corruption [2].

In this thesis, the concept of approximate string matching is analyzed and then it is extended to two Indian languages, namely Hindi and Telugu. It starts with analyzing how humans can easily read words where the letters have been jumbled in a certain way. Several modifications were made to the text based to properties that are analyzed, such as

breaking the text without functional words and breaking the context of sentence, for which independent words are taken. Several edit distances are studied in the process of associating the distance measure with the jumbling process. Edit distance between two strings is the number of operations performed in the process of transferring one string into another. The operations that can be performed are insertion, deletion, substitution and transposition of adjacent letters. There are different ways of performing the edit distance such as Levenshtein distance, Damerau-Levenshtein distance, Hamming distance, Jaro-Winkler distance, Longest common subsequence problem etc. some of them are explained in detail in the subsequent sections. In this thesis, the modifications to the text are generated according to Damerau- Levenshtein distance. Several modifications were made based on position of letters, such as using the Damerau- Levenshtein distance without disturbing the first letter, first and last letters and according to the placement of keys in the QWERTY keyboard. Also the relationship between the edit distance and the time taken is derived.

Our objective is to extend the concept of approximate string matching to some of the Indian languages like Hindi and Telugu. We look into the characteristics of the Indian languages, their classification and variation from English. We then try to jumble the letters based on the properties of the language and look at how humans are able to read them. Then the modifications to the text are made based on the Damerau- Levenshtein distance to see its effect on reading. The modifications were made with the vowels, consonants and syllables in the respective languages.

Entropy

We also look into the information theory and entropy, a key measure of information which is used to measure the uncertainty associated with the system [11]. The entropy of the English language is calculated according to the Shannon's entropy. A study is made on Telugu language on how to classify its alphabets and then its entropy is calculated. Telugu language is

first converted into English script by the use of available software's and then the entropy is calculated. We do it in two ways, first- by converting it to English and then considering them as English alphabets and second- by converting it to English and then considering them as Telugu alphabets. We then derive the relation between entropies of English and Telugu languages.

The rest of the document gives the detailed description of the various edit distances, the characteristics of the Indian languages and how they differ from English and a brief description about entropy and how to calculate it for English and Telugu. The results are simulated and represented graphically.

CHAPTER II

REVIEW OF LITERATURE

Jumbling of letters

The text below has circulated on the Web for several years to show how powerful the human mind is in making sense of jumbled spellings. It may be viewed from the perspectives of joint error correction and coding [3] that are done simultaneously and automatically by the mind, or from the point of view of approximate string matching [2], [4]-[7].

> *Aoccdrnig to a rscheearch at Cmabrigde Uinervtisy, it deosn't mttaer in waht oredr the ltteers in a wrod are, the olny iprmoetnt tihng is taht the frist and lsat ltteer be at the rghit pclae. The rset can be a toatl mses and you can sitll raed it wouthit porbelm. Tihs is bcuseae the huamn mnid deos not raed ervey lteter by istlef, but the wrod as a wlohe*

It has been proposed that the human brain is able to read the words even when they are jumbled because of the following properties:

1. The grammatical structure of the sentence is not disturbed in the above sentence, that is the small words [of 2 or 3 letters] or the function words [by, the, is etc] are not jumbled.

   Since the grammatical structure is preserved, the user is able to predict the next word in the sentence. The jumbled text not only preserves the grammatical structure, it leaves almost 45-50% of the words correct. (In the above paragraph that we took 46% of the words are unchanged.)

2. People generally tend to notice the first and last letters more easily than they tend to observe the middle letters. So there is less possibility of finding errors in the middle letters than the initial and last letters.
3. Although the words are jumbled in the paragraph, the jumbled words are not new words, thus making the task of the reader easier.
4. The sound of the original word is preserved in the jumbled words. This also makes reading easy as people tend to read the word by its sound.
5. People read the jumbled text because of the context of the sentence.

The two things that interested me are the use of function words and the context that plays a part in guessing the next word in the sentence. I have decided to remove the function words from the paragraph and then use the same jumbling technique to study the effect of this change. Also, to break the context of the sentence, I have taken 100 independent words that are commonly used in everyday life and then applied the jumbling technique.

Approximate String matching

Approximate string matching is the technique of performing string matching to the pattern of text. The match is measured in the number of operations that are performed to match the exact string. The most common operations that are performed to match the string are insertion, deletion and substitution. The number of operations performed is measured in terms of edit distance [8].

Examples of the operations are shown below:

Insertion: monkey → monkeys

Deletion: monkey → money

Substitution: monkey → donkey

All the above operations the number of edit distances performed are one. Some string matchers also consider transposition of two adjacent letters in the string [9].

Transposition: lost →lots

Approximate string matching has applications in many fields. Some examples are recovering the original signals after their transmission over noisy channels, finding DNA subsequences after possible mutations, and text searching where there are typing or spelling errors [2].

Most approximate string matchers assume same cost for all the operations performed in string matching, but some matchers do assign different weights to different operations. A more detailed description about edit distance and distance functions are explained in the distance measures section.

Applications of Approximate String Matching for English

Suffix Automata

Suffix automata are an efficient data structure for representing the full index of a set of strings. They are minimal deterministic automata representing the set of all suffixes or substrings of a set of strings. The suffix automata of a finite word w over automata A are the minimal automata that recognize the language Suff (w) of the suffixes of w. It is minimal such that every other (deterministic) automata recognizing Suff (w) has a larger number of states. It can be built in linear time O (|w|) [14]. The automaton is showed in Graph 1.

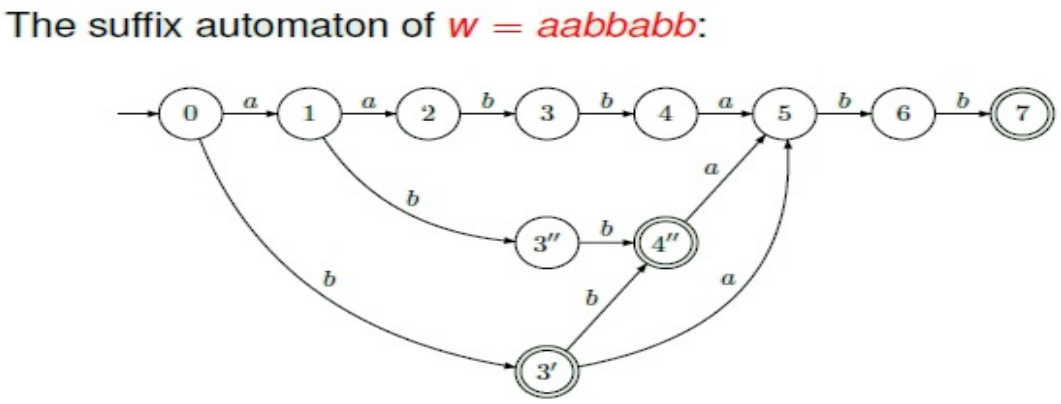
Suffix Context Congruence

For a factor f of w we denote by Endsetw (f) the set of positions of w in which f ends.

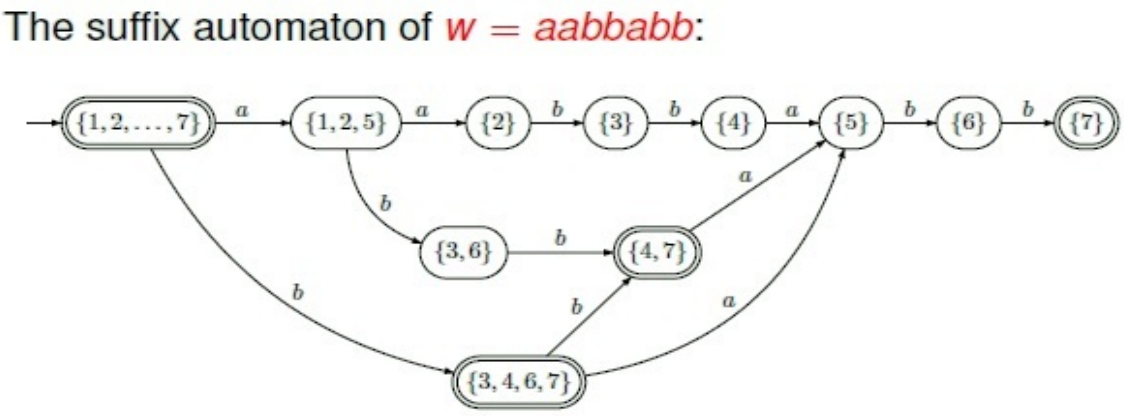
Let W= abbaab. We have:

$Endset_w(ab) = \{2, 6\}$, $Endset_w(b) = \{2, 3, 6\}$.

The states of the suffix automaton of w are in fact the classes of the suffix context congruence defined on the factors of w by: $u \equiv f$, $Endset_w(u) = Endset_w(f)$



Graph 1: Suffix Automaton of w=aabbabb.



Graph 2: Suffix Automaton of w= aabbabb.

The number of states N verifies: $|w| + 1 \leq N \leq 2|w| - 1$.

Suffix Tree

A suffix tree is a tree-like data structure that allows the storage of all substrings of a given string in linear space. It solves the string pattern matching problem in linear time. A suffix tree ST for an n-character string S is a rooted directed tree with exactly n leaves numbered 1 to n. Each internal node, other than the root, has at least two children and each edge is labeled with a nonempty substring of S. No two edges from a node can have edge-labels beginning with the same character. The key feature of the suffix tree is that for any leaf i, the concatenation of the edge-labels on the path from the root to the leaf i exactly spells the suffix of S that starts at position I.

Example:

Suffixes of 'mississippi'

– 'mississippi'

– 'ississippi'

– 'ssissippi'

– 'sissippi'

– 'issippi'

– 'ssippi'

– 'sippi'

– 'ippi'

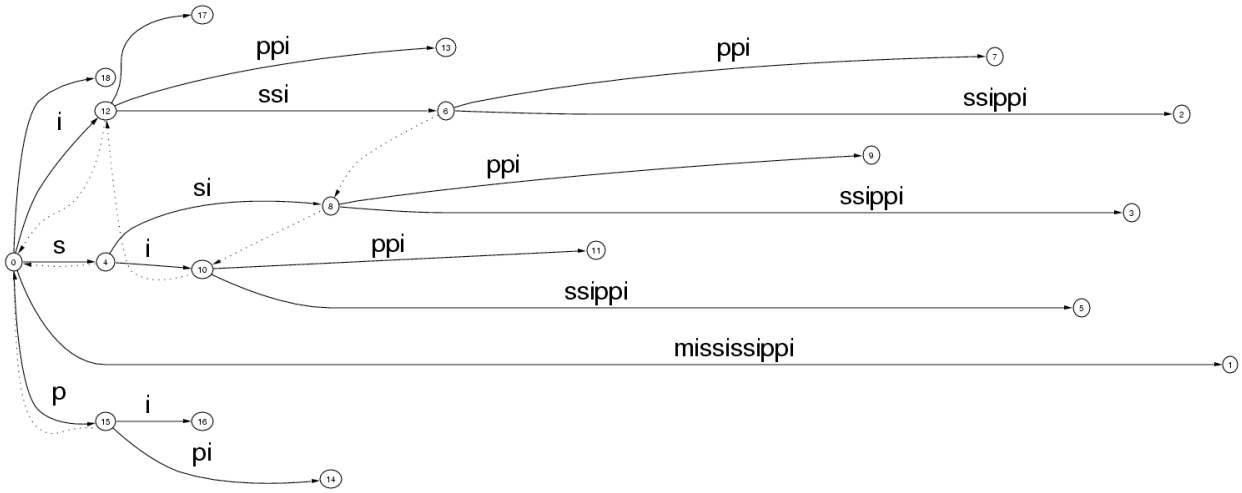
– 'ppi'

– 'pi'

– 'i'

– ''

It is assumed that the string terminates with some character like '\0'.



Graph 3: Suffix tree for string Mississippi.

Bit Parallelism

Bit-parallelism refers to the execution of several operations over a set of bits or numbers stored in a single computer word simultaneously. This technique permits searching for the approximate occurrences of a pattern of length l in a text of length n in time **O ([l/b]n)** where b is the number of bits in the computer word. Although this is asymptotically the optimal speedup over the basic O(mn) time algorithm, it wastes bit parallelism's power in the common case where l is much smaller than b, since b-l bits in the computer words get unused. It is the technique of packing several values in a single computer word and updating them in a single operation [15]-[16].

Dynamic Programming

In the following € denotes the empty string. To compute Levenshtein distance ed(A,B), the dynamic programming algorithm fills an (|A| + 1) x (|B| + 1) table D, in which each cell D[i, j] will eventually hold the value ed(A1..i, B1..j). Initially the trivially known boundary values D[I, 0] = ed(A1..i, €) = i and D[0, j] = ed(€, B1..j) = j are filled. Then the cells D[i, j] are computed for i = 1 … |A| and j = 1 …. |B| until the desired solution D[|A|, |B|] = ed(A1…|A|, B1…|B|) =

ed(A,B) is known. When the values D[i – 1, j - 1], D[I, j - 1] and D[i – 1, j] are known, the value D[i, j] can be computed by using the following well-known recurrence.

D[i, 0] = i, D[0, j] = j.

$$D[i, j] = \begin{cases} D[i-1, j-1], \text{ if } A_i = B_j . \\ 1 + \min(D[i-1, j-1], D[i-1, j], D[i, j-1]), \text{ otherwise.} \end{cases}$$

This distance computation algorithm is easily modified to find approximate occurrences of A somewhere inside B [9]. This is done simply by changing the boundary condition D[0, j] = j into D[0, j] = 0. In this case D[i, j] = min(ed(A1…i, Bh…j), h < j), which corresponds to the earlier definition of approximate string matching if we replace A with P and B with T.

The values of D are usually computed by filling it in a column-wise manner for increasing j. This corresponds to scanning the string B (or the text T) one character at a time from left to right. At each character the corresponding column is completely filled in order of increasing i. This order makes it possible to save space by storing only one column at a time, since then the values in column j depend only on already computed values in it or values in column j - 1.

Some properties of matrix D are:

-The diagonal property: D[i, j] - D[i - 1; j - 1] = 0 or 1:

-The adjacency property: D[i, j] - D[I, j - 1] = -1, 0, or 1, and D[I, j] - D[i – 1, j] = -1, 0, or 1

Applications of Approximate String Matching for Indian Languages

The Suffix tree for మహామహులు

The suffixes of మహామహులు

-మహామహులు

-హామహులు

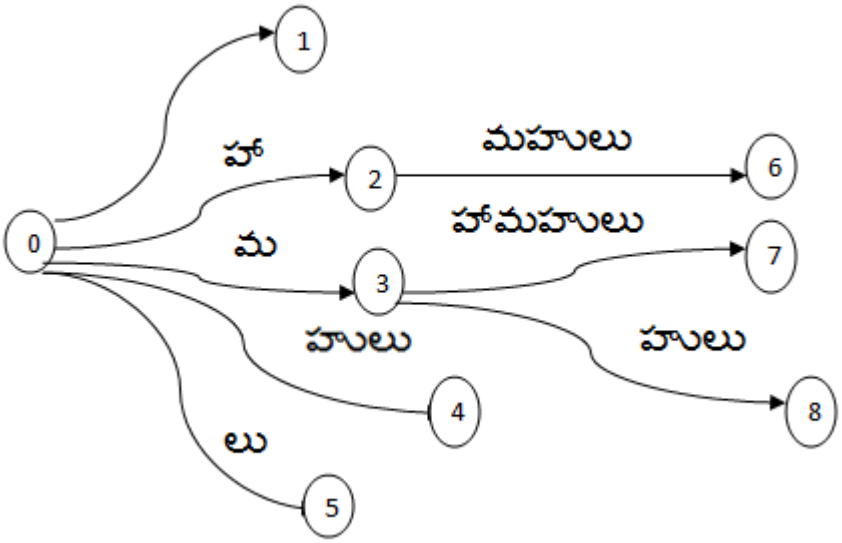
-మహులు

-హులు

-లు

-“ “

The Suffix tree is:



Graph 4: Suffix tree for string మహామహాలు

The Suffix tree for लाइलाज

The Suffixes for लाइलाज are

-लाइलाज

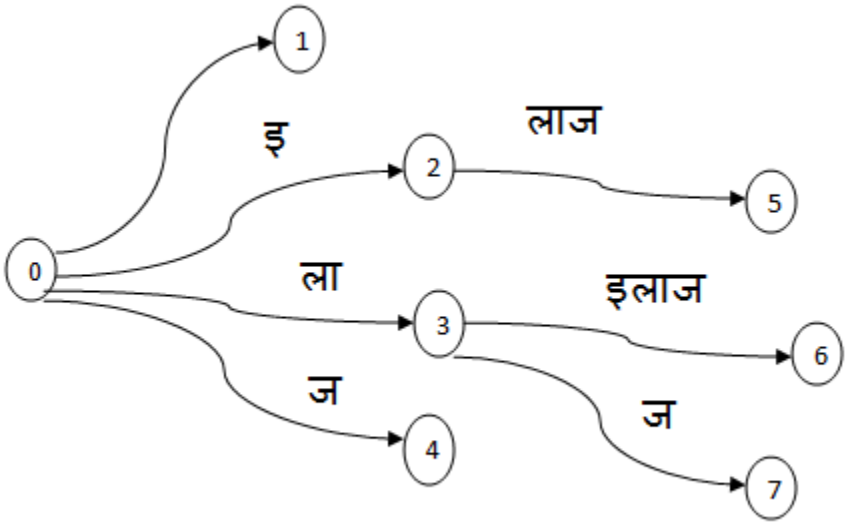
-इलाज

-लाज

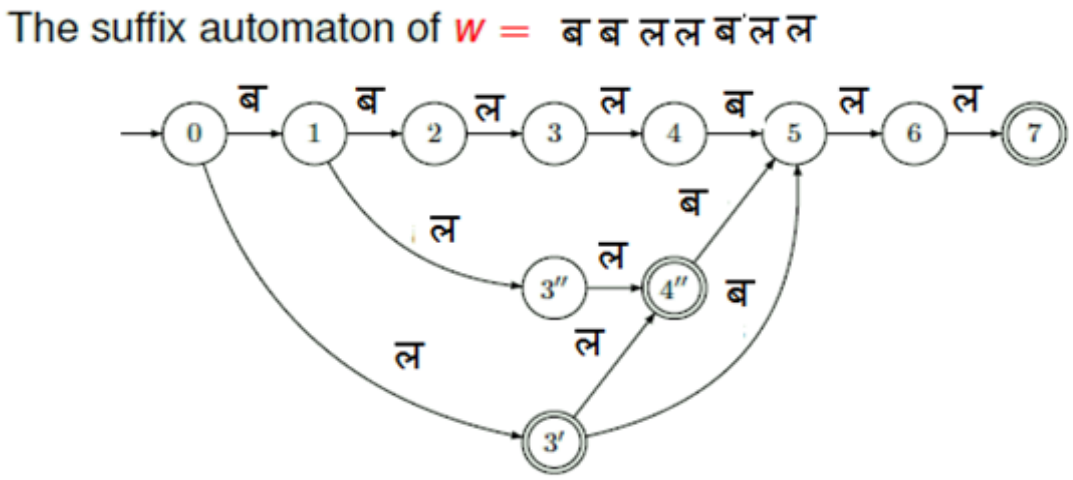
-ज

-“ “

The Suffix tree is:



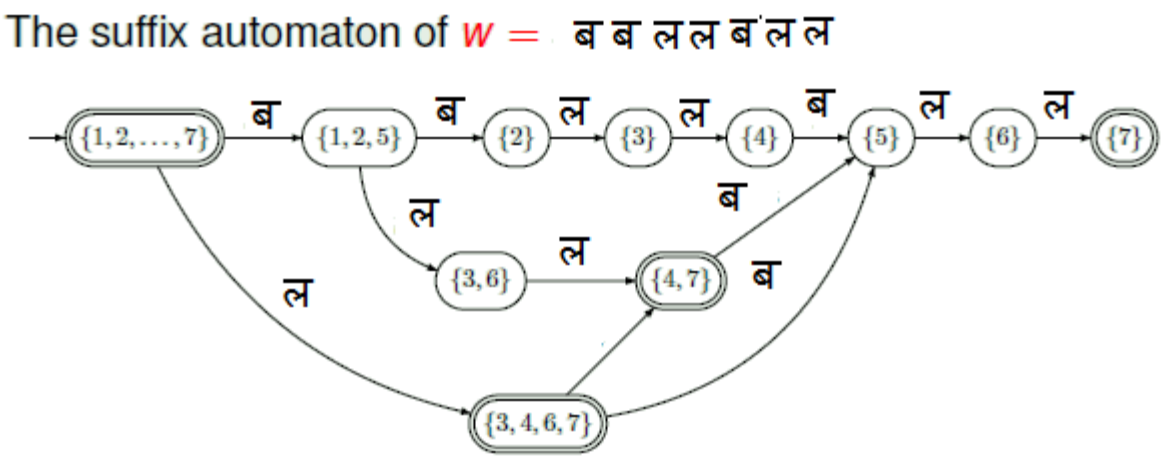
Graph 5: Suffix tree for string लाइलाज



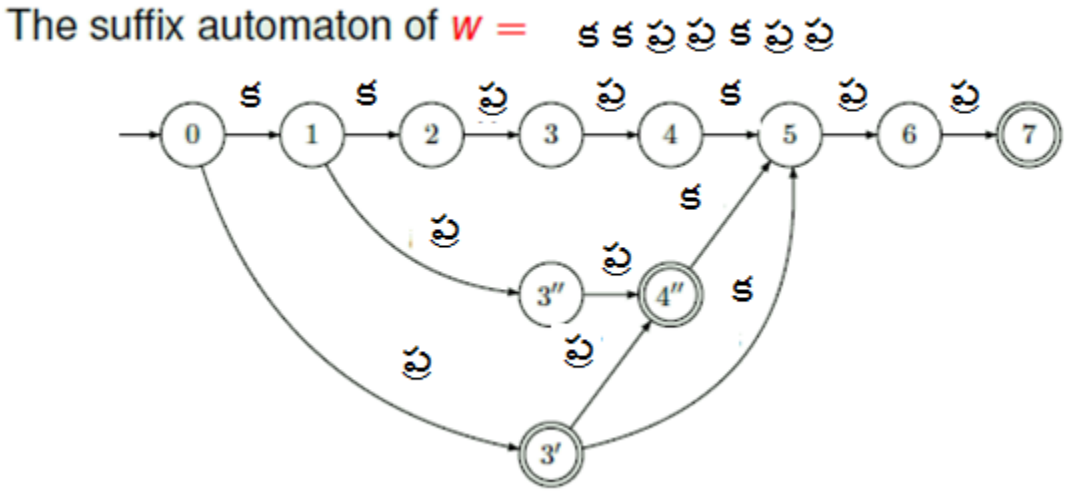
Graph 6: Suffix Automaton for w= बबललबलल

Let $w =$ ब ल ल ब ल

$Endset_w$ बल $= \{2, 6\}$, $Endset_w$(ल) $= \{2, 3, 6\}$.



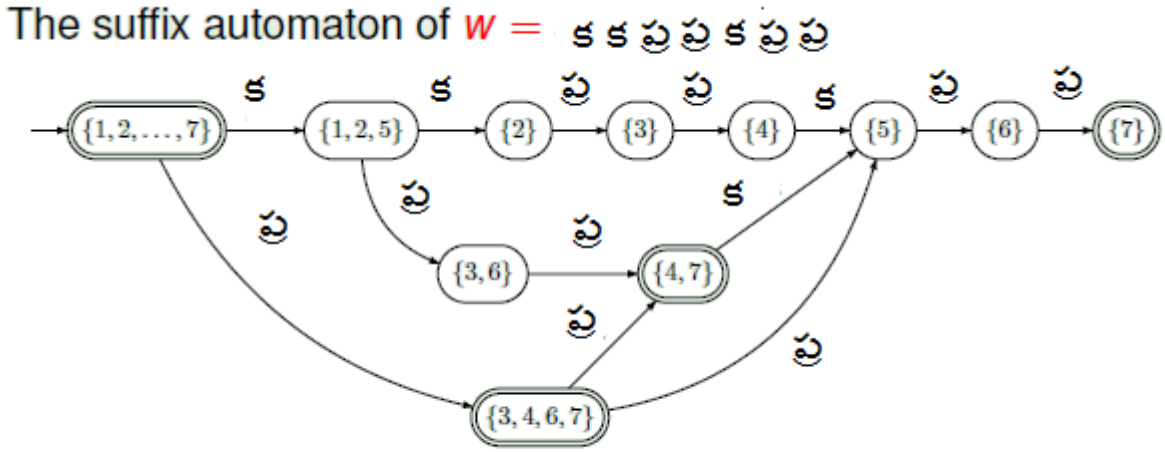
Graph 7: Suffix Automaton for w= బబలలబలల



Graph 8: Suffix Automaton for w= కకప్రపకప్ర

Let $w =$ కపపకప

$Endset_w$ కప $= \{2, 6\}$, $Endset_w$(ప) $= \{2, 3, 6\}$.



Graph 9: Suffix Automaton for w= కకప్రపకప్రప

Distance measures

The distance, d(x, y) is the minimum number of operations that are needed to perform in order to transform 'x' into 'y'. 'x' and 'y' are two strings and the distance is a non negative real number. Once the operation has converted 'x' into 'y', no other operations can be performed on 'y'. The operation is represented by $\partial(x, y) = z$, where 'z' is the non negative real number. The minimal number of operations required to transform one string into another is also called the cost of the operation [2].

The cost of sequence of operations is equal to the sum of cost of individual operations. If for every operation $\partial(x, y)$, there is a respective operation $\partial(y, x)$ for the same cost, then the distance is said to be symmetric.

That is $d(x, y) = d(y, x)$

For all strings 'x', 'y' and 'z', the below expression holds true

$d(x, z) \leq d(x, y) + d(y, z)$, that is it satisfies the triangle inequality.

The set of possible operations can be expressed as:

- Insertion (□, a), represents the insertion of element ‘a’.
- Deletion (a, □), represents the deletion of element ‘a’.
- Substitution (a, b), represents the substitution of element ‘b’ for element ‘a’.
- Transposition (ab, ba), represents the transposition or swapping of elements ‘a’ and ‘b’.

Some of the most commonly used distance measures are explained below:

- Hamming distance:

  Hamming distance allows only substitution of letters, which cost one unit. It is applied only to the strings of similar length. In other words, it is the minimum number of substitutions required to transform one string into another or the number of errors that transform one string to another. In the literature, the search problem is called the string matching with k mismatches. If the lengths of two strings are equal then the distance is symmetric. It is applied in error detection and correction [12], [13].

  $0 \leq d(x, y) \leq |x|$

  Examples:

  The hamming distance between

  Sunday and Monday is 2 (one substitution of M for S, i.e. ∂(S, M) and one substitution of o for u, i.e. ∂ (u, o))

  Monkey and Donkey is 1 (just one substitution of D for M, i.e. ∂ (M, D))

- Longest common subsequence distance:

This allows insertions and deletions, which cost one unit each. It is often used to find the longest sequence that is common to all the sequences in the set of sequences. It measures the length of longest pairing of characters that can be made between the two strings such that the pairing respects the order of letters. The distance is the number of unpaired characters and is also symmetric. It has wide range of applications in the areas such as molecular biology, file comparison and screen redisplay etc [10].

$0 \leq d(x, y) \leq |x| + |y|$

Examples:

The longest common subsequence between

Human and Chimpanzee is 4 (HMAN)

ABCBDAB and BDCABA is 4 (BCBA or BCAB)

The longest common subsequence is not always unique.

- Episode distance:

  Episode distance allows only insertions, which cost one unit. In literature, the search problem is called the episode matching. Since it only allows insertions, the conversion of one string into another is not always possible. Also, the distance is not symmetric. Since not always the distance can be calculated, the mathematical expression is given as,

  $d(x, y) = |y| - |x|$ or ∞

  Examples:

  The episode distance between

  Money and Monkey is 1 (∂ (□, k))

  Sunday and Monday is ∞

- Levenshtein distance:

Levenshtein distance is a metric used to measure the difference between two sequences. This measure between two strings is defined by the number of edit operations used from transforming one string to another. The edit operations may be insertion, deletion and substitution of a single character. Here all the operations cost one unit. It is the minimum number of insertions, deletions and substitutions that are required to make two strings equal. In literature, the search problem is called the string matching with k differences. The distance is symmetric and is given by,

$0 \leq d(x, y) \leq \max(|x|, |y|)$

Levenshtein distance has a wide range of applications in areas such as spell checkers, dialect pronunciations and used in software's for natural language translations.

Examples:

The Levenshtein distance between

Sunday and Tuesday is 3 (one substitution of T for S, one substitution of e for s and one insertion of s)

ABCAB and BDAB is 2 (one deletion of A, one substitution of D for C)

- Damerau- Levenshtein distance:

  Damerau-Levenshtein distance is similar to Levenshtein distance except that it includes an extra edit operation called the transposition of adjacent letters, where all the operations cost one unit. It is the minimum number of insertions, deletions, substitutions and transpositions required to make two strings equal. The distance is symmetric and is given by,

  $0 \leq d(x, y) \leq \max(|x|, |y|)$

  Damerau-Levenshtein distance has its applications in fields of fraud vendor name detections, where it can detect the letter that has been deleted or substituted, in DNA, where the variation between the two strands of DNA can be found out by this distance.

Examples:

The Damerau- Levenshtein distance between

AC and CAB is 2 (one transposition of C and A and one insertion of B)

Tried and tired is 1(just one transposition)

In this thesis, for all the modifications, the Damerau- Levenshtein distance is used as the operations also contains transposition of adjacent letters apart from the regular operations such as substitution, insertion and deletion. This is because 80% of the errors that occur during database search or web search can be corrected by just one insertion, deletion, substitution or transposition.

Algorithm for Damerau- Levenshtein distance:

```
int damerau_levenshtein_distance(string a, string b) {
   int alen = strlen(a);
   int blen = strlen(b);
   mixed array dist = allocate(alen + 1, allocate(blen + 1));
   for(int j = 1; j <= blen; j++)
      dist[0][j] = j;
   dist[alen][0] = alen;
   for(int i = 0; i < alen; i++) {
      dist[i][0] = i;
      for(int j = 0; j < blen; j++)
         if(i && j && a[i] == b[j - 1] && a[i - 1] == b[j])
```

```
                dist[i + 1][j + 1] = min(dist[i][j + 1] + 1, dist[i + 1][j] + 1, dist[i][j] + (a[i] !=
                b[j]), dist[i - 1][j - 1] + (a[i] != b[j]));
        else
            dist[i + 1][j + 1] = min(dist[i][j + 1] + 1, dist[i + 1][j] + 1, dist[i][j] + (a[i] != b[j]));
    }
    return dist[alen][blen];
}
```

Characteristics of Indian scripts

Indian languages are highly phonetic; i.e. the pronunciation of new words can be reliably predicted from their written form. A background to Indian scripts is given in [17]-[22]. Indian scripts are highly systematic in their arrangement of sounds, and the milieu in which they arose is provided in papers on early Indian science [23]-[26]. From the evidence available at this time it may be assumed that the 3rd millennium BC Indus script evolved into the Brahmi script of late centuries BC which, in turn, evolved into the different Indian modern Scripts. Structurally, all the Indian scripts are thus closely related although their forms may look quite different. The Brahmi script is also the parent to Southeast Asian scripts.

The alphabets of Indian languages are classified into consonants, vowels and other symbols. An akshara or syllable consists of 0, 1, 2 or 3 consonants and a vowel or other symbol. Each akshara can be pronounced independently. Alphabets of all the Indian languages are derived from the Brahmi script. In all the Indian languages there are 33 common consonants and 15 common vowels. In addition, there are 3-4 consonants and 2-3 vowels that are specific to each language, but not very significant in practice. Words are made up of one or more aksharas. If an akshara consists of more than one consonant, they are called samyuktakshara.

The similarity in alphabet is not extended in graphical form which is used for printing. Each language uses different scripts, which consists of different graphemes. There are about 10-12 major scripts in India among which Devanagari is the most widely used. Different languages have different statistical characteristics. Some have an over line for the entire word, while some have not touching graphemes. The vowels and the supporting consonants in samyuktakshara can appear on the left, right, above, below or in combinations to the main consonant.

Examples:

Over line □□□□□-□□□□□□□

Non Touching □□□□□□□□□□□

Entropy

Information according to Shannon is viewed probabilistically. It is carried discretely as symbols, which are selected from set of possible symbols. The information carried by the symbol depends upon its probability [11]. The amount of information gained decreases with the increase in uncertainty. Entropy is a key measure of information. It is the measure of uncertainty or disorder that is associated with the system. Since information is a decrease in uncertainty, entropy is regarded as the information required to construct the correct set of symbols. The information required to reconstruct the correct set of symbols increases with the increase in entropy [11]. Entropy and information are sometimes used interchangeably although they do not always mean the same thing. Entropy in a source is equal to the information per symbol needed to reconstruct its output.

For a random variable X with n outcomes $\{x_i : i = 1,2,3 \dots n\}$, the Shannon entropy, a measure of uncertainty and denoted by H(X), is defined as

$$H(X) = -\sum_{i=1}^{n} x_i \log_b x_i$$

The entropy of a language is a statistical measure which measures the average information produced for each letter of text in a language [11].

CHAPTER III

METHODOLOGY

In all the experiments that I have conducted, I recorded the time 10 readers took to read the text individually. This time was then averaged.

String Matching for English

From the analysis of the properties of the jumbled letters, the two things that seemed important to me are the use of functional words and the context of the sentence which is helping in reading. In this thesis, I have tried to remove the functional words from the paragraph and then use the jumbling procedure. The text after jumbling looks like this,

> *Accdroing resaecrh elgnsih uvinsreity deosn't mtaetr waht oredr letrets wrod olny iopmrtant tnihg frist lsat lteetr rihgt plcae. rset tatol mses sitll raed whtiuot pborelm. Tihs baceuse raed ervey lteter istlef wrod wohle.*

In order to test the importance of context through which the readers are able to read the jumbled sentence, I took some 100 independent words that are commonly used and then jumbled them with their first and last letters in their original position. The words after jumbling are,

> *sutdy bisas ecxiting feild utlizie gerat inretesitng exapidnng many cnortubietd*

nomral sceond bolew geneitc grdataue nopetad jbmule coapmbitle giivng cmapnig

sohcol cmoemnt sipmle sclorl atcion aihcveeemnt braod psate naitanol esstneial

firend dsiaml dimiinsh grteeing divdie conimg exetnral puord aitctave rcenet

moeny rdaeer saecrh iivnte cpmotetioin seicntsit eelavte porargms ietnnrtaioanl coisnetsntly

rpealce smysopuim aacdimec flloewod pporreites addsers pltarofm konelwgde widnwos itnecartoin

porcudt cnolose fcartion paitrpicate ganiing high ppoele rteial aevrgae dollar

wsbeite wtsae headnig sevearl editnig ptonetial fargile spennidg fuutre suohedlrs

bruedn scetor inofmritaon cfnoused ucpmoing sreuois assist sbusnataitl qlautiy bceome

cmmoon matniian reuqrie griwong hmuor ainmal gniog finance ietnnret wemon.

Then the Damerau- Levenshtein distance of one is applied to the original paragraph and the result of it is,

Accrding to recearch at an Enlgish univiersity, it dosn't matetr in wiat ordier the lettes in a werd are, the only impurtant thng is that the fist and last lette is at the rijght place. The rect can be a totul mess and you can stillt raed it wihout probllem. Tihs is becase we do not raed evry lette by it slf but the wurd as a whule.

Damerau- Levenshtein distance of one is also applied to the independent words to see its effect on reading. The operations are performed with several combinations such as

- To the entire word
- Without disturbing the first letter in the word

- Without disturbing the first and last letter in the word
- The operations being performed by the neighboring letters in the QWERTY keyboard.

String Matching for Indian Languages

In this thesis, the languages considered are Hindi (Devanagari script) and Telugu. Since these are phonetic languages, they are different from English language.

The jumbling of words was based on the following combinations:

- Vowels
- Consonants
- Syllables
- Syllables but without disturbing the first and last one

The paragraphs that are considered for examination are,

Telugu

క్రికెట్ దేవుడు సచిన్ టెండూల్కర్ కు భారత రత్న ఇవ్వాలని ప్రధాని కార్యాలయం ప్రతిపాదించినట్లు సీ ఎన్ ఎన్ ఐబీఎన్ వెల్లడించింది. అయితే, ఆయనకీ అవార్డు ఇవ్వటం పట్ల కొన్ని అభ్యంతరాలు ఉన్నట్లు సమాచారం. ముఖ్యంగా భారత దేశ అత్యున్నత పౌర పురస్కారం అందుకునేంత వయస్సు ఆయనకు లేదనే వాదన బలంగా వినిపిస్తోంది. టెండూల్కర్ రెండేళ్ల కిందటే పద్మ విభూషణ్ అందుకున్నారని, ఇంతలోనే భారత రత్న ప్రకటించడం తొందర పాటు అవుతుందని వ్యతిరేకించే వర్గాలు చెబుతున్నాయి. మంగళవారం ముంబాయిలో ఒక కార్యక్రమంలో పాల్గొన్న టెండూల్కర్ ఈ

వూహాగానాలకు సాధ్యమైనంత దూరంగా ఉన్నా రు.

Hindi

जम्मू-कश्मीर समेत देशभर में गणतन्त्र दिवस समारोह धूमधाम और पारम्परिक उल्लास से मनाया गया, जबकि राष्ट्रीय राजधानी दिल्ली में विविधता में एकता का नज़ारा उकेरती रा-बिरी झाँकियों और सैन्य बल की क्षमताओ के जानदार प्रदर्शन ने गणतन्त्र दिवस समारोह में ऐसा समा बा, मानो मिनि इडिया राजपथ पर उतर आया हो. गणतन्त्र दिवस के अवसर पर हिा प्रभावित राज्यों के मुख्यमियों ने अलगवादियों और उग्रवादियों से वार्ता कि लिए आगे आने का आह्वान किया.

Also Damerau- Levenshtein distance of one is applied to these texts to see the effect of reading. Modifications are made to the texts based on the following combinations:

- Vowels
- Consonants
- Vowels without disturbing first and last letters
- Consonant without disturbing first and last letters

The timings are noted and are represented graphically in the results section.

Entropy

Entropy of the language is the measure of disorder associated with the system. For a random variable X with n outcomes $\{x_i : i = 1,2,3 \dots n\}$, the Shannon entropy, a measure of uncertainty and denoted by H(X), is defined as

$$H(X) = -\sum_{i=1}^{n} x_i \log_b x_i$$

Entropy of the English language is calculated by taking into consideration, the 26 alphabets and space character and leaving out the punctuation. The frequencies of the letters are calculated and are shown in the table 1.

Alphabet	Frequency (%)
A	7.80
B	1.43
C	2.72
D	3.19
E	9.26
F	1.64
G	1.89
H	4.05
I	5.88
J	0.10
K	0.70
L	2.87

M	2.03
N	6.42
O	6.09
P	1.71
Q	0.11
R	4.84
S	5.56
T	7.72
U	2.34
V	0.64
W	2.00
X	0.08
Y	1.27
Z	0.01
Space	17.54

Table 1: Frequencies of alphabets in English language

The above results are calculated based on 10,000 characters and the frequencies are rounded off to 2 decimal points.

Entropy of Telugu language is computed by converting the Telugu language into English language using some software's and then using the above mentioned formula to compute it. The entropy in Telugu language is computed in two ways.

- By converting it to English and then considering them as English alphabets
- By converting it to English and then considering them as Telugu alphabets.

To understand the conversion of Telugu into English font, here are some of the examples:

కార్యాలయం	kAryAlayaM
ప్రతిపాదించినట్లు	pratipAdiMcinaTlu
పద్మవిభూషణ్	padmavibhUShaN^
వయస్సు	vayassu
సాధ్యమైనంత	sAdhyamainaMta

In the first method, after converting them into English font, they are considered as English alphabets, i.e. padmavibhUShaN^ is considered as a sequence of characters containing p, a, d, m, a, v, i, b, h, U, S, h, a, N, ^. In the Telugu language, the alphabets are case sensitive as each has different meaning. The frequencies of each letter are shown in the table 2.

Alphabets	Frequencies (%)	Alphabets	Frequencies (%)
A	13.15	A	6.13
B	0.99	B	0.00
C	1.98	C	0.00
D	2.47	D	1.35

E	1.30	E	1.77
F	0.04	F	0.00
G	1.52	G	0.00
H	3.31	H	0.00
I	6.81	I	1.03
J	0.78	J	0.04
K	3.92	K	0.00
L	3.57	L	0.69
M	2.01	M	3.67
N	4.66	N	0.25
O	0.49	O	1.46
P	2.68	P	0.04
Q	0.00	Q	0.01
R	3.31	R	0.06
S	2.46	S	0.29
T	2.97	T	1.41
U	5.65	U	0.07
V	2.43	V	0.01

W	0.00	W	0.00
X	0.00	X	0.00
Y	1.86	Y	0.00
Z	0.00	Z	0.00
Space	12.05	^	0.46

Table 2: Frequencies of alphabets in Telugu language

The above results are calculated based on 10,000 characters and the frequencies are rounded off to 2 decimal points.

In the next method, after converting them to English font, they are considered as Telugu syllables as opposed to English alphabets in method one. In this approach, the words are partitioned on the basis of Telugu syllables. Some examples are shown below.

padmavibhUShaN^	pa, dma, vi, bhU, Sha , N^
kAryAlayaM	kA, ryA, la, yaM
sAdhyamainaMta	sA, dhya, mai, naM, ta

The frequencies of each syllable according to the given text are computed and then the entropy of the language is calculated.

In this case we are considering one syllable at a time and the entropy of the language calculated is approximately 5.98 for the 10,000 characters that I have considered.

We have continued this method for finding the entropy of Telugu language by considering two syllables at a time to decrease the entropy of the language.

In this approach, the words are partitioned on the basis of Telugu syllables. Some examples are shown below.

padmavibhUShaN^	padma, dmavi, vibhU, bhUSha, ShaN^
kAryAlayaM	kAryA, ryAla, layaM
sAdhyamainaMta	sAdhya, dhyamai, mainaM, naMta

In this case the entropy of the language is calculated to be 3.98 approximately for the same 10,000 characters.

In the next step, we found the entropy of Telugu language by considering three syllables at a time. In this approach, the words are partitioned on the basis of Telugu syllables. Some examples are shown below.

padmavibhUShaN^	padmavi, dmavibhU, vibhUSha, bhUShaN^
kAryAlayaM	kAryAla, ryAlayaM
sAdhyamainaMta	sAdhyamai, dhyamainaM, mainaMta

In this case the entropy of the language is calculated to be 2.739 approximately for the same 10,000 characters.

In the next step, we found the entropy of Telugu language by considering four syllables at a time. In this approach, the words are partitioned on the basis of Telugu syllables. Some examples are shown below.

padmavibhUShaN^	padmavibhU, dmavibhUSha, vibhUShaN^

kAryAlayaM	kAryAlayaM
sAdhyamainaMta	sAdhyamainaM, dhyamainaMta

In this case the entropy of the language is calculated to be 2.077 approximately for the same 10,000 characters.

We continued this process for up to six syllables and the entropy of the language when 5 syllables and 6 syllables are considered is calculated.

The entropy of language for 5 syllables is 1.699.

The entropy of language for 6 syllables is 1.39.

All the results are represented diagrammatically using graphs in the results section.

CHAPTER IV

FINDINGS

Approximate String Matching

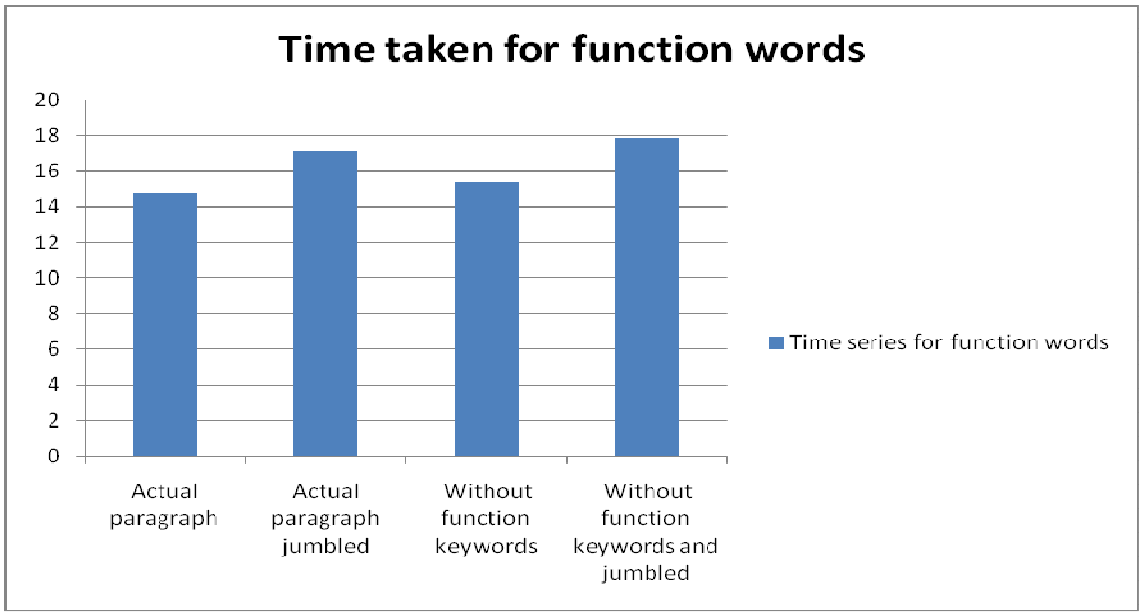
For English

In this section I considered the actual paragraph and then removed all the function words from the paragraph to find the effect on the reader.

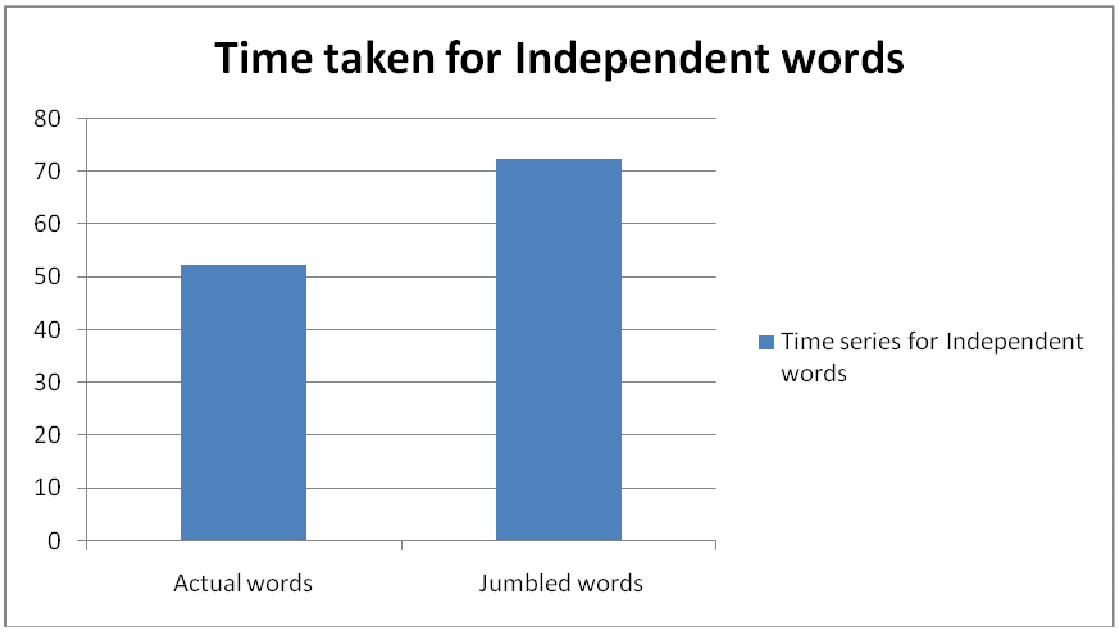
Without function words

> *According research English university doesn't matter what order letters word only important thing first last letter right place. Rest total mess still read without problem. This because read every letter itself word whole.*

In this section, in order to test the importance of context through which the readers are able to read the jumbled sentence, I took some 100 independent words that are commonly used and then jumbled them with their first and last letters in their original position.



Graph 10: Time taken for function words.

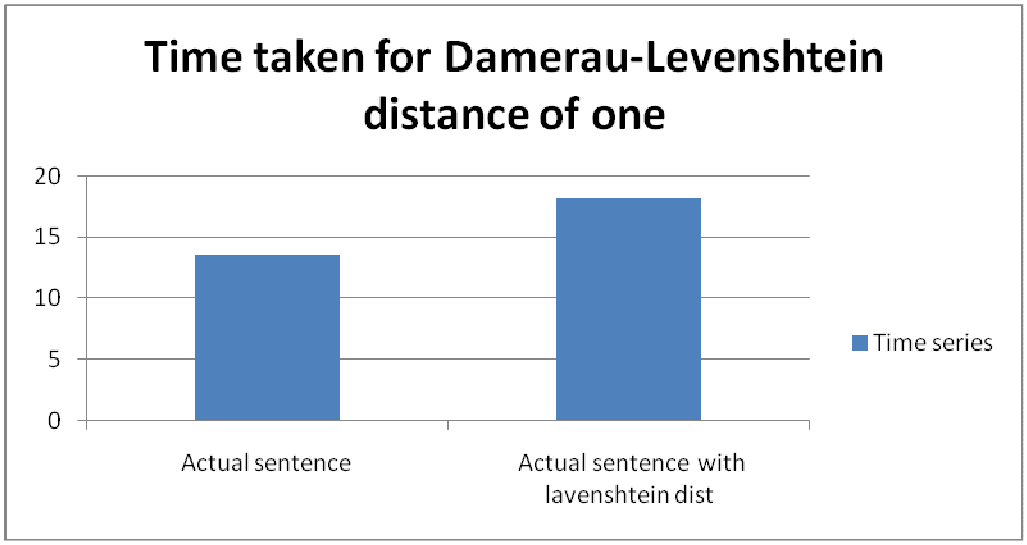


Graph 11: Time taken for independent words.

I applied the Damerau-Levenshtein distance of one to the original paragraph, obtaining the following text:

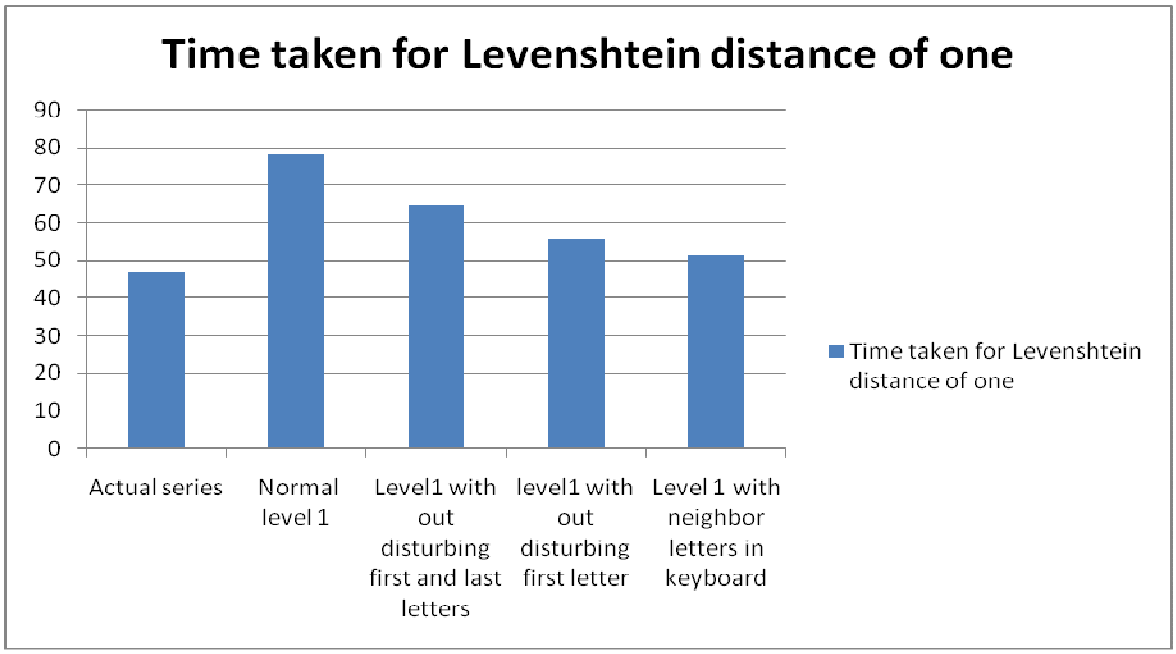
Accrding to recearch at an Enlgish univiersity, it dosn't matetr in wiat ordier the lettes in a werd are, the only impurtant thng is that the fist and last lette is at the rijght place. The rect can be a totul mess and you can stillt raed it wihout probllem. Tihs is becase we do not raed evry lette by it slf but the wurd as a whule.

The edit operations performed are addition, deletion, substitution and transposition of neighboring letters. The timings were then recorded.



Graph 12: Time taken for Damerau-Levenshtein distance of one.

Then I took the same 100 independent words and used the Damerau-Levenshtein distance of one to find the effect on reading.

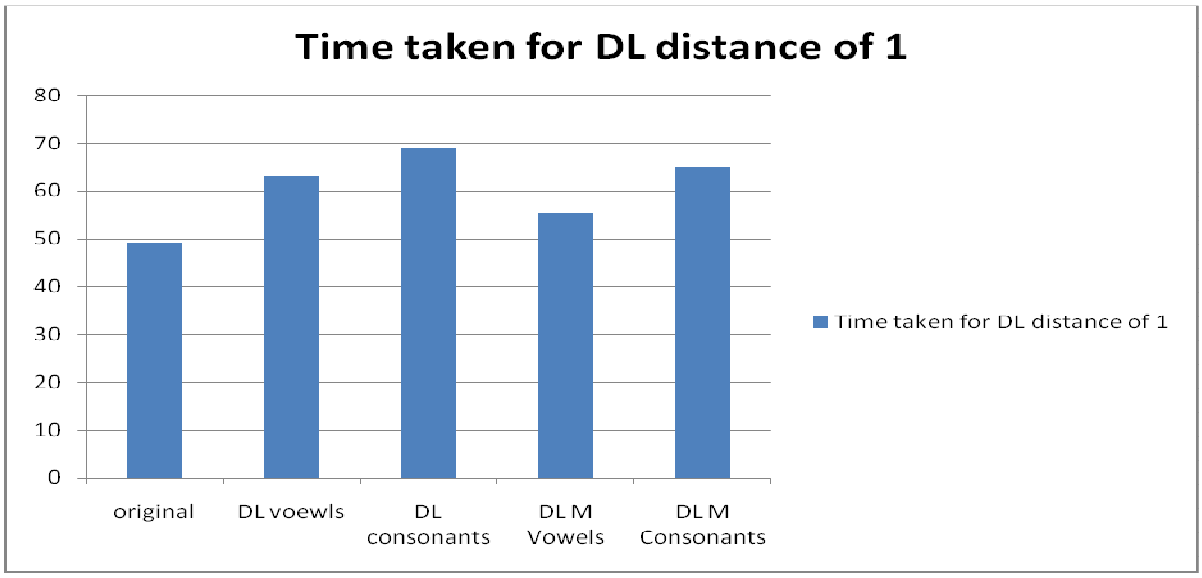


Graph 13: Time taken for Levenshtein distance of one.

For Telugu

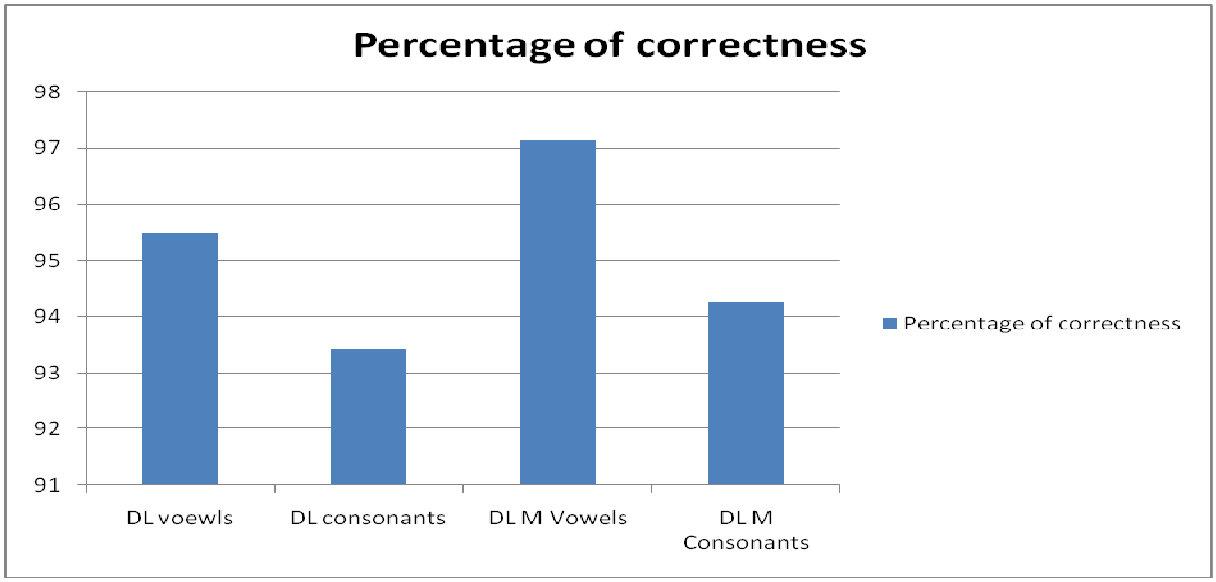
The Damerau- Levenshtein distance of one is applied for the Telugu text and the combinations were based up on the following:

- Vowels
- Consonants
- Vowels but without changing the middle ones
- consonants but without changing the middle ones



Graph 14: Time taken for Damerau Levenshtein Distance of one.

The next section shows the percentage of words that are correctly read by the users. The graph shows that when the middle vowels/ consonants are not changed, the users are able to read it with much accuracy than if they are changed. Also, vowels have much less importance than the consonants.

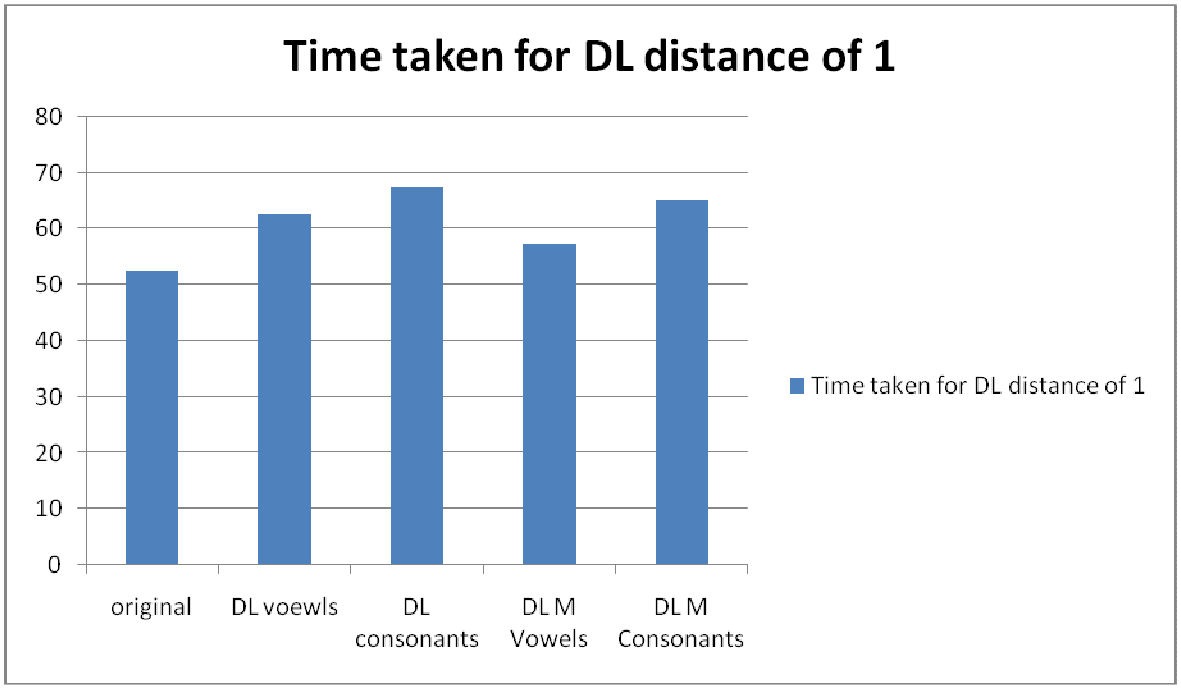


Graph 15: Correct percentage of words read.

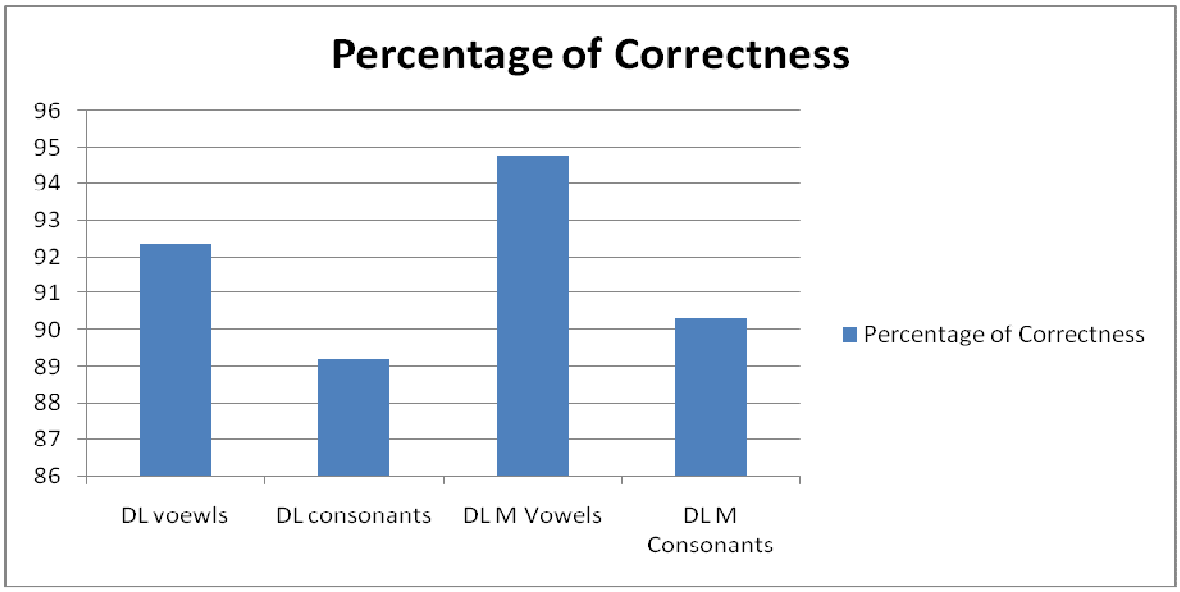
For Hindi

The Damerau- Levenshtein distance of one is applied for the Hindi text and the combinations were based up on the following:

- Vowels
- Consonants
- Vowels but without changing the middle ones
- consonants but without changing the middle ones



Graph 16: Time taken for Damerau Levenshtein Distance of one.



Graph 17: Correct percentage of words read.

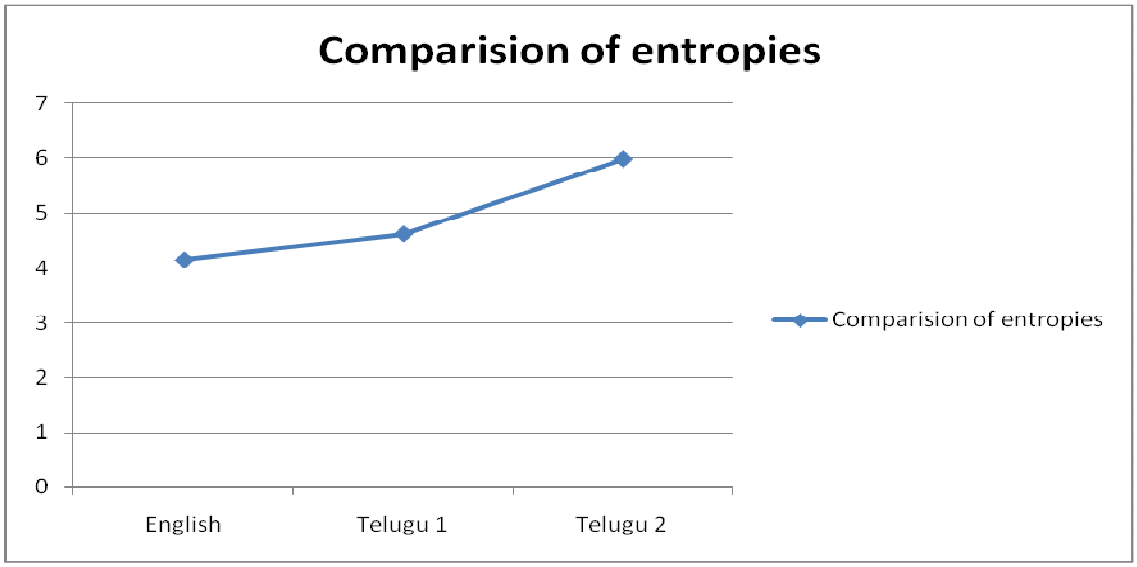
The above section shows the percentage of words that are correctly read by the users. The graph shows that when the middle vowels/ consonants are not changed, the users are able to read it with much accuracy than if they are changed. Also, vowels have much less importance than the consonants.

Entropy

This graph compares the entropies of English and Telugu language. Entropy of the English language is calculated by taking into consideration, the 26 alphabets and space character and leaving out the punctuation.

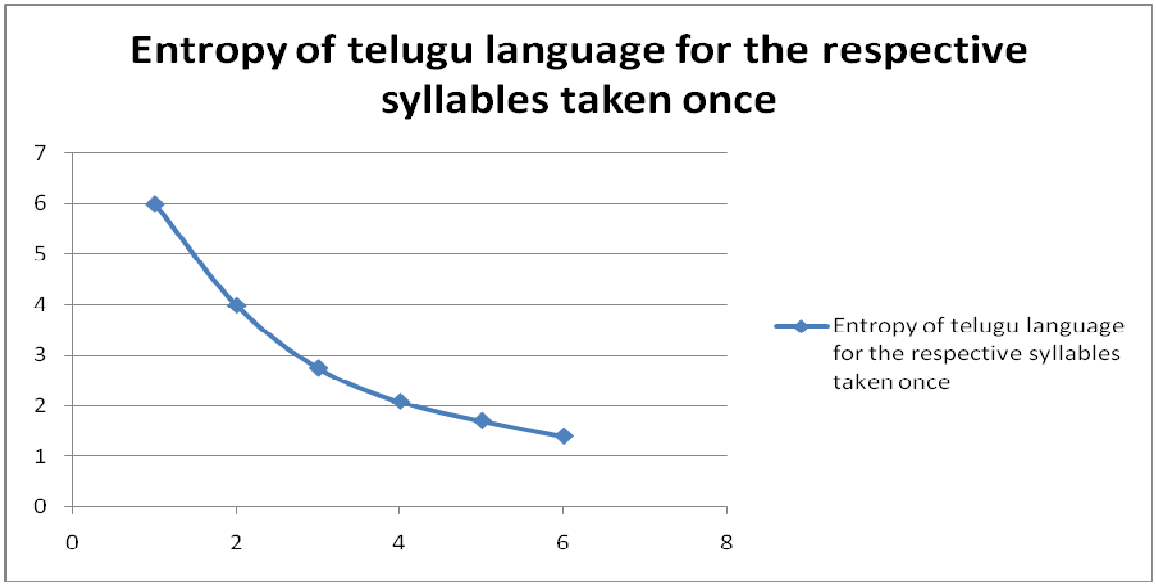
The entropy in Telugu language is computed in two ways.

- By converting it to English and then considering them as English alphabets (Telugu 1).
- By converting it to English and then considering them as Telugu alphabets (Telugu 2).



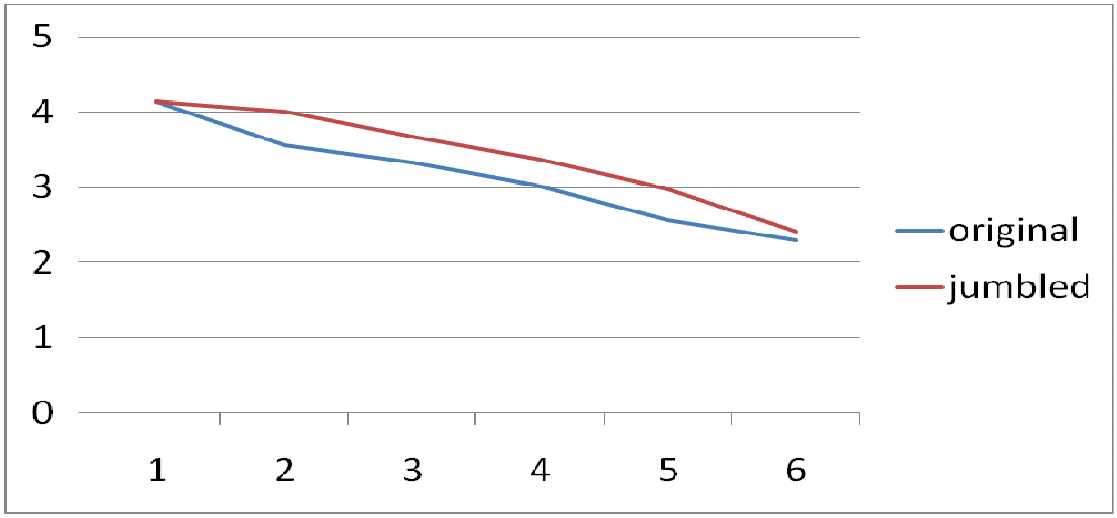
Graph 18: Comparison of entropy calculated.

The next section is about Telugu 2 where the alphabets are considered as in Telugu language. We have considered one syllable at a time, two syllables at a time etc till we have reached a value which doesn't change much. Entropy decreases as the number of syllables considered increases.

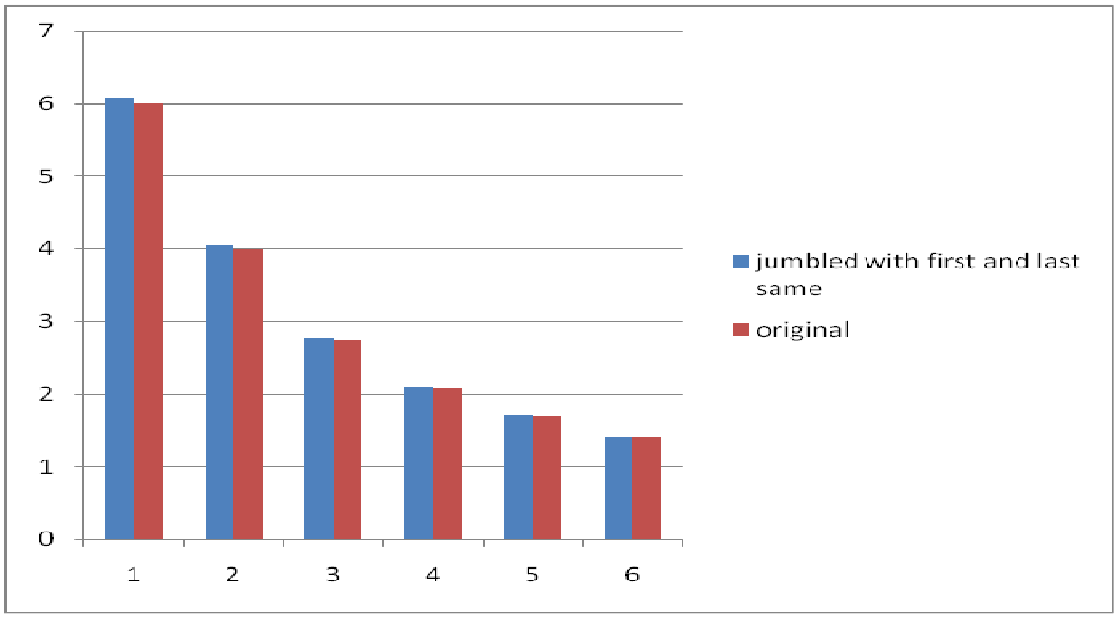


Graph 19: Entropy of language when considered according to syllables.

The next graph shows the comparison of entropies of English language for the original text and the jumbled text with first and last letters kept constant. This graph shows that there is much less difference in the entropies of original text and the jumbled one and hence people are able to read the jumbled text with not much difficulty.



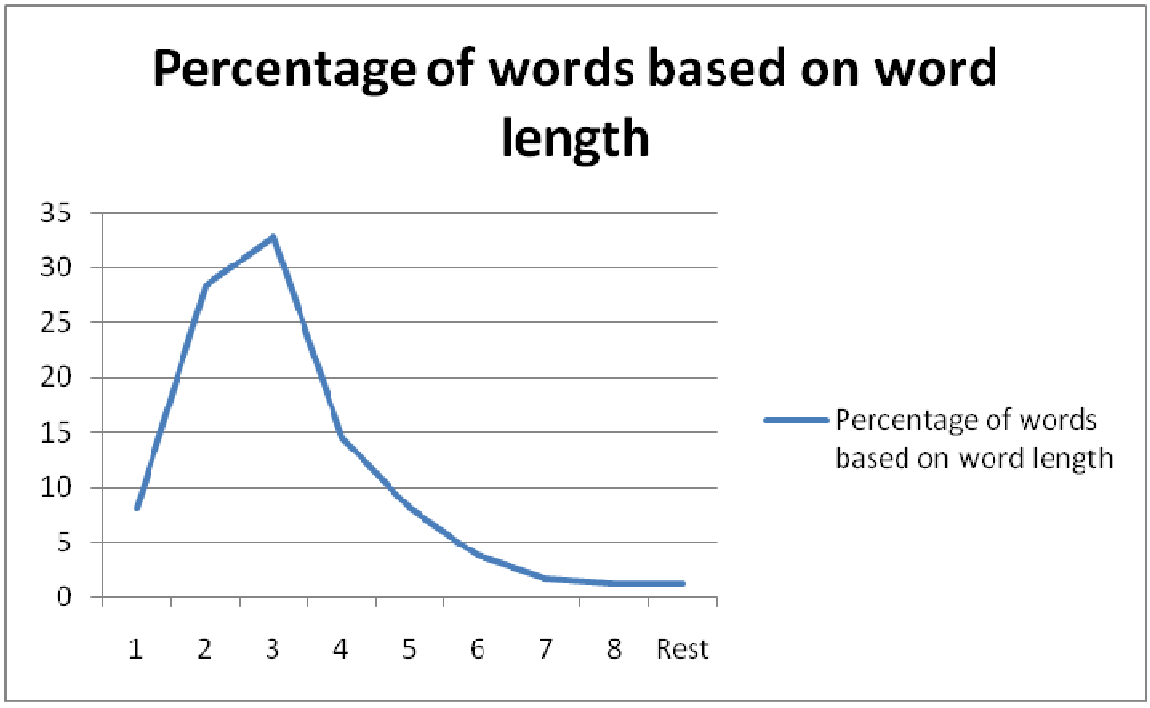
Graph 20: Entropy of English language for original text and jumbled text.



Graph 21: Entropy of English language for original text and jumbled text.

The above graph shows the comparison of entropies of Telugu language for the original text and the jumbled text with first and last letters kept constant. This graph shows that there is much less difference in the entropies of original text and the jumbled one and hence people are able to read the jumbled text with not much difficulty.

The difference is very little in Telugu language compared to that of English is because of the word length in Telugu language. In jumbling process we only consider the words of length greater than three, but the graph shows that approximately 70% of the words in Telugu are of length less than four.



Graph 22: Percentage of words based on word length

CHAPTER V

CONCLUSION

The results show that the importance of functional words in reading these words is lesser than proposed before. Nevertheless, context plays an important role in helping the user read such words. In addition, this thesis applied the Damerau-Levenshtein distance of one to the words and found that the words can be read if the first and last letters are left in their places.

For Indian languages like Hindi and Telugu, which are phonetic so that the pronunciation of new words can be reliably predicted from their written form, consonants carry more information than the vowels. This can be inferred from both the time taken to read the text and also from the percentage of correctness. The words can be read with much ease if the first and last letters are left in their places.

Finally, the entropy of Telugu is higher than that of English, which means that the Telugu is more succinct than English and each syllable in Telugu (as in other Indian languages) contains more information compared to English. The comparison of entropies of the language for original and jumbled texts confirms that the users can read jumbled text with not much difficulty.

REFERENCES

1. D. Michailidis and G. Margaritis, “Flexible approximate string matching application on a heterogeneous distributed environment”, PDPTA. 164-172, 2003.
2. G. Navarro, “A guided tour to approximate string matching”, ACM Computing Surveys 33, 31-88, 2001.
3. S. Kak, “Joint encryption and error-correction coding”, Proceedings of the 1983 IEEE Symposium on Security and Privacy, pp. 55-60, 1983.
4. R. Lowrance and R.A. Wagner, “An extension of the string-to-string correction problem”, Journal of the ACM, 22(2):177-183, 1975.
5. R. L. Kashyap and B. J. Oommen, “An effective algorithm for string correction using generalized edit distances -I. description of the algorithm and its optimality”, *Inform.Sci.*, 23(2), 123-142, 1981.
6. T. Okuda, E. Tanaka, and T. Kasai, “A method of correction of garbled words based on the Levenshtein metric”, *IEEE Trans. Comput.*, C-25, 172-177, 1976.
7. A. Marzal and E. Vidal, “Computation of normalized edit distance and applications”, *IEEE Trans. on Pat. Anal. And Mach. Intel.*, PAMI-15, 926-932. 1993.
8. S. Skiena. Algorithm design manual (1st Ed.). Springer, 1998.
9. T. Ottmann, P. Widmayer (in German). “Algorithmen und datenstrukturen (4th ed.). spektrum akademischer verlag”. pp. 636–9, 2002.
10. Daniel S. Hirschberg, “Algorithms for the longest common subsequence problem”, Journal of the Association for Computing Machinery, vol. 24, pp. 664-675, Oct 1977.

11. C.E. Shannon, “Prediction and entropy of printed English”, Bell System Technical Journal, vol. 27, pp. 379-423, 623-656, Oct 1948.
12. A. Jarrous and B. Pinkas, “Secure Hamming distance based computation and its applications”, ACNS 2009, LNCS 5536, pp. 107–124, 2009.Springer-Verlag Berlin Heidelberg 2009.
13. S. Kak, “Encryption and error-correction coding using d sequences.” IEEE Transactions on Computers, C-34, 803-809, 1985.
14. M. Mohri, P. Moreno and E. Weinstein, “General suffix automaton construction algorithm and space bounds”, Theoretical Computer Science, 410(37), Sept 2009.
15. S. Tata, R.A. Hankins and J.M. Patel, “Practical suffix tree construction”, Proceedings of the 30th VLDB Conference, Toronto, Canada, 2004
16. H. Hyyro, K. Fredriksson and G. Navarro, “Increased bit-parallelism for approximate string matching”, In Proc. 3rd Workshop on Efficient and Experimental Algorithms (WEA’04), LNCS 3059, pp. 285–298, 2004.
17. S. Kak, “A frequency analysis of the Indus script. cryptologia”, vol. 12, pp. 129-143, 1988.
18. S. Kak, “Indus and Brahmi – further connections. cryptologia”, vol. 14, pp. 169-183, 1989.
19. S. Kak, “The evolution of early writing in India. Indian Journal of History Of Science”, vol. 28, pp. 375-388, 1994.
20. P.G. Patel, P. Pandey and D. Rajgor (eds.), “The Indic scripts. D.K. Printworld”, New Delhi, 2007.
21. S. Kak, “Knowledge of planets in the third millennium BC”, Quarterly Journal of the Royal Astronomical Society, vol. 37, pp. 709-715, 1996.

22. S. Kak, “The astronomy of the age of geometric altars”, Quarterly Journal of the Royal Astronomical Society, vol. 36, pp. 385-396, 1995.
23. S. Kak, “The Paninian approach to natural language processing”, International Journal of Approximate Reasoning, vol. 1, pp. 117-130, 1987.
24. S. Kak, “Astronomy of the Vedic altars, vistas in astronomy”, vol. 36, pp. 117-140, 1993.
25. S. Kak, “The structure of the Rig Veda. Indian Journal of History of Science”, vol. 28, pp. 71-79, 1993.
26. S. Kak, “The nature of physical reality”, Peter Lang, New York, 1986.

VITA

Venkata Ravinder Paruchuri

Candidate for the Degree of

Master of Science

Thesis: APPROXIMATE STRING MATCHING AND APPLICATIONS TO INDIAN SCRIPTS

Major Field: Computer Science

Biographical:

Education:

Completed the requirements for Master of Science in Computer Science at Oklahoma State University, Stillwater, Oklahoma in July, 2011.

Completed the requirements for Bachelor of Engineering in Computer Science and Engineering at JNT University, Hyderabad, India in 2009.

Experience:

Graduate Assistant *Oklahoma State University, Stillwater, OK,* January 2010 to present

- Responsible for recording, editing and posting media files online.
- Extensively converted media formats for online viewing using conversion tools and web development tools.

Graduate Assistant *Oklahoma State University, Stillwater, OK,* October 2009 to present

- Designed and maintained the website of Micro Biology Department.
- Helping the students in the lab with computer trouble shooting, installing, configuring and upgrading operating systems and software.

Professional Memberships: Phi Kappa Phi

Name: Venkata Ravinder Paruchuri Date of Degree: July, 2011

Institution: Oklahoma State University Location: Stillwater, Oklahoma

Title of Study: APPROXIMATE STRING MATCHING AND APPLICATIONS TO INDIAN SCRIPTS

Pages in Study: 46 Candidate for the Degree of Master of Science

Major Field: Computer Science

Scope and Method of Study:

This thesis is concerned with the analysis of approximate string matching and its applications to Indian scripts. It starts with the analysis of the problem of the readability of jumbled text. Several modifications of jumbling are considered based on position of letters, such as using the Damerau- Levenshtein distance without disturbing the first letter, first and last letters and according to the placement of keys in the QWERTY keyboard. The process is applied to the Indian scripts and modifications are based on vowels, consonants and syllables in the respective languages. A study is made on the entropy of English language and then the entropy for Telugu language is calculated and compared to that of English language.

Findings and Conclusions:

Through simulations and analysis, it is observed that the functional words are not that important in reading the jumbled words, but context plays a crucial role. Also, for both the English language and Indian languages when the first and last letters are not moved, users are able to read them even when the text is modified through Damerau-Levenshtein distance of one. Also, the entropy of Telugu is much higher than English, which implies that Telugu is much more succinct than English.

ADVISER'S APPROVAL: Dr. Subhash Kak