Tribhuvan University
Institute of Science and Technology
Central Department of Computer Science & Information Technology

Text Similarity Using Corpus Based Semantic Word Similarity and String Similarity for Short Nepali Texts.

Dissertation
Submitted to

Central Department of Computer Science & Information Technology
Kirtipur, Kathmandu, Nepal

In partial fulfillment of the requirements
for the Master's Degree in Computer Science & Information Technology

By

Laxman Manandhar

March, 2013

Tribhuvan University
Institute of Science and Technology
Central Department of Computer Science & Information Technology

Text Similarity Using Corpus Based Semantic Word Similarity and String Similarity for Short Nepali Texts.

Dissertation
Submitted to

Central Department of Computer Science & Information Technology
Kirtipur, Kathmandu, Nepal

In partial fulfillment of the requirements
for the Master's Degree in Computer Science & Information Technology

By

Laxman Manandhar
March, 2013

Supervisor
Prof. Dr. Shashidhar Ram Joshi

Co-supervisor
Mr. Bikash Balami

Tribhuvan University
Institute of Science and Technology

Central Department of Computer Science & Information Technology

Supervisor's Recommendation

We hereby recommend that this dissertation prepared under our supervision by Mr. Laxman Manandhar entitled "**Text Similarity Using Corpus Based Semantic Word Similarity and String Similarity for Short Nepali Texts**" be accepted as partial fulfillment of the requirements for the Master's degree of Computer Science and Information Technology.

... ...	... ...
Prof. Dr. Shashidhar Ram Joshi Department of Electronics and Computer Engineering, Institute Of Engineering, Pulchowk, Nepal (Supervisor)	Mr.Bikash Balami Central Department of Computer Science and IT, Tribhuwan University, Kirtipur, Nepal (Co-Supervisor)
... ...	... ...
Date	Date

Tribhuvan University
Institute of Science and Technology

Central Department of Computer Science & Information Technology

LETTER OF APPROVAL

We certify that we have read this dissertation and in our opinion it is satisfactory in the scope and quality as a dissertation in the partial fulfillment for the requirement of Master's Degree in Computer Science and Information Technology.

Evaluation Committee

... ...
Asst. Prof. Nawaraj Paudel
(Head of Department)
Central Department of Computer Science & Information Technology
Tribhuvan University
Kritipur, Kathmandu, Nepal

... ...

... ...

... ...
(External Examiner)

... ...
(Internal Examiner)

Date: .....

Acknowledgments

It is first a privilege and a pleasure for me to thank my principal supervisor, Prof. Dr. Shashidhar Ram Joshi, Department of Electronics and Computer Engineering, Institute Of Engineering, Pulchowk, Nepal for his constant encouragement, support and advice.

I want to express my deep thanks to my honoured co-supervisor Mr. Bikash Balami of Central Department of Computer Science & Information Technology, TU, Nepal for his inspiration, trust, the insightful discussion, thoughtful guidance, critical comments, and correction of the thesis.

I greatly acknowledge respected teachers of the Central Department of Computer Science & Information Technology, TU, Nepal for their valuable suggestion. I would like to thank my family and friends for their precious direct and indirect support. Finally, I would like to give my special thanks to my wife Mrs. Sneha Shrestha for helping and providing wonderful environment and support to complete this work.

Abstract

Similarity measure for long text, documents have been in research from long time but similarity measure for short text were not been given much emphasis. Short Texts and sentences similarity measures are now considered to be important research topic due to its many applications in the field of Natural language processing and information retrieval. The need to determine semantic similarity, semantic distance between two lexically expressed concepts is a problem that pervades much of natural language processing. This thesis deals with one of Information Retrieval's big interest: Textual Similarity. This thesis includes the study and implementation of short text similarity measure for Nepali language. The semantic text similarity has not been yet studied for Nepali language text. This thesis deals with two main challenges .The first is to determine the similarity of the two short texts having different lexical terms and the second is determining the semantic similarity based on string similarity for considering the minor spelling mistakes of the words in the sentence. Such measures should mostly be considered during web retrieval as users may not always give the right spelling for the words. Nepali language is based on devanagari script and has different literature. This thesis includes the implementation and analysis of the String similarity measures (Modified version of Longest Common Subsequences and String edit distance) and corpus based word similarity measure (Second Order Co-Occurrence Point Wise Mutual Information) for overall semantic Text similarity. Improvement has been done for the integration of word similarity measure and string similarity measure.

Table of Contents

List of Figures

Details
Page

List of Tables

Details Page

Abbreviations

STS	→	Semantic Text Similarity
LCS	→	Longest Common Subsequence
POS	→	Part of Speech
LSA	→	Latent Semantic Analysis
PMI	→	Pointwise Mutual Information
SOCPMI	→	Second Order Co-occurrence Pointwise Mutual Information
HAL	→	Hyperspace Analogues to Language
NLCS	→	Normalized Longest Common Subsequence
NMCLCS	→	Normalized Maximal Consecutive Longest Common Subsequence
IC	→	Information Content
JVM	→	Java Virtual Machine

CHAPTER 1

Introduction to Text Similarity

1.1 Text Similarity

The large amounts of information are now stored and they need processing. The need to retrieve the information more effectively is the major challenge. Semantic Text Similarity (STS) measures the degree of semantic equivalence between two texts. Such measures are used to find the relatedness between two concepts or words [1]. The short text semantic similarity measure has many applications in natural language processing and information retrieval [2], automated answers checking [3], image retrieval on the web (using short text surrounding the image), conversational agent [4], automatic text summarization based on semantics [5], [6], word sense disambiguation, machine translation and documents classification.The strict lexical similarity between the texts gives the similarity measures based on the lexical information only that is string similarity between words. The semantic text similarity deals with computing similarity between the given texts based on the semantic information. For two sentences to be similar, it is not always the case that they share common lexical terms. They can exihibit same meaning even if the lexical terms are not similar. That means words in one sentence may be the synonyms for the words in the other sentence. Also it's not always the case that for two words or concepts to be related they should be synonyms, they can exhibit other relations as hyponyms, meronyms, hyponyms, etc. The concept of the hyponyms, hypernyms, entailment, redundancy, polysemy etc. should be considered to detect the similarity on semantic level.

The texts can also be considered as similar if one text can be derived from the other. Lexical term similarity is the simple concept which deals with matching the exact spellings of words whereas semantic similarity is complex topic based on the human judgment of likeness and judgment of differences.

1.2 Types of Text Similarity

In this section, the textual similarities are described in terms of lexical and semantic basis.

1.2.1 Lexical Similarity

Lexical text similarity can also be termed as exact string similarity measure. Based on the spelling matching of the words, similarity is computed. This is the most basic similarity measure and purely lexical. Given two short texts A and B, the similarity can be computed based on the lexical terms matching. The similarities criteria can be defined as following:

Exactly similarity: A and B are exactly similar i.e lexically equivalent

Phrase: A is the substring of B or vice versa

Subset: A is subset of B if the terms in A are subset of B.

Such measures for two segments of text returns the result as either match ('similar') or they do not. There is no graded score associated with the match. However, if necessary, it is possible to impose such score by looking at various characteristics of the match such as the length of *A* and *B,* or the frequency of the terms in some collection.

1.2.2 Semantic Text Similarity

Semantic similarity or semantic relatedness is a concept whereby a set of documents or terms within term lists are assigned a metric based on the likeness of their meaning or semantic content. Computing semantic similarity is very complex job as the concepts of semantic relatedness are based on the human judgment of likeness and differences. The presence of ambiguities and complexities in the language makes it more complex to determine the exact semantics. Similarity measurements generally take into consideration the standard semantic relationships, such as hyponym, hypernym, holonym, meronym, entailment and redundancy between words [7].

DEFINITION 1 (semantics): Semantics is the study for the meaning of words, phrases, and sentences and the relations between them.

DEFINITION 2 (synonym):W1 and W2 are synonyms for each other if they can be used interchangeably in a context. An example sentence for the synonym relationship is: "Blossoms is a synonym of flower."

DEFINITION 3 (hypernym):W2 is a hypernym of W1 if W1 is a type of W2. An example sentence for the hypernym relationship is: "flower is a hypernym of daisy."

DEFINITION 4 (hyponym):W2 is a hyponym of W1 if W2 is a type of W1. An example sentence for the hyponym relationship is: "daisy is a hyponym of flower."

DEFINITION 5 (holonym):W2 is a holonym of W1 if W1 is a part of W2. An example sentence for the holonym relationship is: “flower is a holonym of petal.”

DEFINITION 6 (meronym):W2 is a meronym of W1 if W2 is a part of W1.

An example sentence for the meronym relationship is: “petal is a meronym of flower.”

DEFINITION 7 (word form): A word form represents all the words or phrases which define entities in WordNet. Example word forms are “car” and “sports utility vehicle.” As we can see the first word form is comprised of one word, and the second word form is a phrase comprised of three words.

DEFINITION 8 (sense): Sense represents the explanation of the meaning for a word form. An example sense for the word form “car” is: “A wheeled vehicle with four wheels usually propelled by an internal combustion engine.”

1.3 Challenges for Semantic Text Similarity

1.3.1 Challenges for String Similarity

Sometimes there are cases when two words tend to have similar meaning or exactly the same meaning even though they have some few differences in spelling of the lexical terms.

The String Similarity measure must deal with following challenges.

- A true reflection of lexical similarity - strings with small differences should be recognized as being similar. In particular, a significant sub-string overlap should point to a high level of similarity between the strings.
- Robustness to changes of characters order- two strings which contain the same characters, in a same order, should be recognized as being similar. On the other hand, if one string is just a random anagram of the characters contained in the other, then it should (usually) be recognized as dissimilar.
- Language independence - the algorithm should work in many different languages.

1.3.2 Challenges for Semantic Similarity

The other case for textual similarity is when sentences have completely different lexical terms but also exhibit the same meaning. The different lexical terms may be synonymns for each other. And also it is not always the case that sentences have the exactly same words or their synonyms for representing the same meaning.The two sentences may refer to same

sense without sharing any of common words or even without synonynms. In section 1.2.2 the different types of semantic relationship have been defined. The consideretions of such relationship for the semantic similarity is the major challenge.

1.4 Approaches to Text Similarity

1.4.1 Corpus Based Measure

In linguistics, a corpus (plural *corpora*) or text corpus is a large and structured set of texts (now usually electronically stored and processed). They are used to do statistical analysis and hypothesis testing, checking occurrences or validating linguistic rules on a specific universe. Corpora are intended to be representative of some specified population. A corpus may contain texts in a single language (*monolingual corpus*) or text data in multiple languages (*multilingual corpus*). Corpus based measures use such corpus for the word level similarity.

1.4.1.1 Latent Semantic Analysis

Latent Semantic Analysis (LSA) [20] is a theory and method for extracting and representing the contextual-usage meaning of words by statistical computations applied to a large corpus of text. The underlying idea is that the aggregate of all the word contexts in which a given word does and does not appear provides a set of mutual constraints that largely determines the similarity of meaning of words and sets of words to each other.

LSA is a fully automatic mathematical and statistical technique for extracting and inferring relations of expected contextual usage of words in passages of discourse. It is not atraditional natural language processing or artificial intelligence program; it uses no humanly constructed dictionaries, knowledge bases, semantic networks, grammars, syntactic parsers, or morphologies, or the like, and takes as its input only raw text parsed into words defined as unique character strings and separated into meaningful passages or samples such as sentences or paragraphs.

The first step is to represent the text as a matrix in which each row stands for a unique word and each column stands for a text passage or other context. LSA applies singular value decomposition (SVD) to the matrix. This is a form of factor analysis, or more properly the mathematical generalization of which factor analysis is a special case. In SVD, a rectangular matrix is decomposed into the product of three other matrices. One component matrix

describes the original row entities as vectors of derived orthogonal factor values, another describes the original column entities in the same way, and the third is a diagonal matrix containing scaling values such that when the three components are matrix-multiplied, the original matrix is reconstructed. [21] [22] suggests that tools such as LSA are unlikely to be necessary in relatively small document collections, as most participants are likely to use a brute force approach, in which all documents are accessed.

1.4.1.2 Hyperspace Analogues to Language

The Hyperspace Analogues to Language (HAL) [23] method uses lexical co-occurrence to produce a high-dimensional semantic space. A semantic space is a space in which words are represented as points, and the position of each word along the axes is related to the word's meaning. Once the space is constructed, a distance measure can be used to determine relationships between words. In HAL, this space is constructed by first passing a window over a large corpus and recording weighted lexical co-occurrences (words closer to the target word are given a higher weight than words farther away). These results are recorded in an $n \times n$ co-occurrence matrix with one row and one column for each unique word appearing in the corpus. Once this is complete, a vector representing each word in $2n$ dimensional space is formed by concatenating the transpose of a word's column to its row. Subsequently, a sentence vector is formed by adding together the word vectors for all words in the sentence. Similarity between two sentences is calculated using a metric such as Euclidean distance. However, the authors' experimental results showed that HAL was not as promising as LSA in the computation of similarity for short texts [21]. HAL's drawback may be due to the building of the memory matrix and its approach to forming sentence vectors: The word-by-word matrix does not capture sentence meaning well and the sentence vector becomes diluted as a large number of words are added to it [14].

1.4.1.3 Pointwise Mutual Information

The pointwise mutual information using data collected by information retrieval (PMI-IR) was first suggested by Turney 2001 [24] as an unsupervised measure for the evaluation of the semantic similarity of words. It is based on word co-occurrence using counts collected over very large corpora. Given two words w_1 and w_2, their PMI-IR is measured as:

$$\text{PMIIR}(w_1, w_2) = log_2 \frac{p(w_1 \& w_2)}{p(w_1) * \text{p}(w_2)} \quad (1.1)$$

Where $p(w_1 \& w_2)$ is the probability that the two words co-occur. If w_1 and w_2 are statistically independent, then the probability that they co-occur is given by the product $p(w_1) * \text{p}(w_2)$. PMIIR (w_1 ,w_2) indicates the degree of statistical dependence between w_1 and w_2, and can be used as a measure of the semantic similarity of w_1 and w_2. Among the four different types of queries suggested by Turney (2001), the NEAR query (co-occurrence within a ten-word window) is used, which is a balance between accuracy (results obtained on synonymy tests) and efficiency (number of queries to be runagainst a search engine). Specifically, the following query is used to collect counts from the AltaVista search engine.

$$\text{PNEAR}(w_1 \& w_2) \cong \frac{hits(w_1 \text{NEAR } w_2)}{WebSize} \quad (1.2)$$

With p (w_i) approximated as hits (w_1) = WebSize, the following PMI-IR measure is obtained:

$$log_2 \frac{hits(w_1 \text{AND } w_2) * WebSize}{hits(w_1) * \text{hits}(\, w_2)} \quad (1.3)$$

1.4.1.4 Second Order Co-occurrence Pointwise Mutual Information Method

Second Order Co-occurrence PMI method is the extension of PMI method with improvised concept necessary for the semantic level similarity detection. The Second Order Co-occurrence PMI (SOC-PMI) [25] method uses Pointwise Mutual Information to sort lists of important neighbor words of the two target words from a large corpus. The main advantage of using SOC-PMI is that it can calculate the similarity between two words that do not co-occur frequently, but because they mostly co-occur with the same neighboring words. SOCPMI is described in more details in chapter 2.

1.4.2 Knowledge Based Measure

Knowledge based measures use lexical resources like WordNets or Synsets available. WordNet is a lexical database which is available online and provides a large repository of lexical items. The WordNet was designed to establish connections between four types of POS (Parts of Speech): noun, verb, adjective, and adverb. The smallest unit in WordNet is synset, which represents a specific meaning of a word. It includes the word, its explanation, and the

synonyms of its meaning. The specific meaning of one word under one type of POS is called a sense. Each sense of a word is in a different synset. Synsets are equivalent to senses: structures containing sets of terms with synonymous meanings. Each synset has a gloss that defines the concept it represents. WordNet synonym and hypernym relations could be used to capture the semantic similarities of tokens. Given a pair of words, once a path that connects the two words is found, their similarity could be determined based on two factors: the length of the path and the order of the sense involved in this path.

1.4.2.1 The Leacock & Chodorow Similarity Measure

The Leacock and Chodorow [8] similarity is determined as:

$$\text{Sim}_{lch} = -log\frac{length}{2 * D} \tag{1.4}$$

where length is the length of the shortest path between two concepts using node-counting, and D is the maximum depth of the taxonomy.

1.4.2.2 The Lesk Similarity Method

The Lesk similarity [9] of two concepts is defined as a function of the overlap between the corresponding definitions, as provided by a dictionary. It is based on an algorithm proposed by Lesk (1986) as a solution for word sense disambiguation. The application of the Lesk similarity measure is not limited to semantic networks, and it can be used in conjunction with any dictionary that provides word definitions. SOC-PMI is defined in more details in chaper two

1.4.2.3 The Wu and Palmer Method

The Wu and Palmer metric [10] measures the depth of two given concepts in the Word-Net taxonomy and the depth of the least common subsume (LCS), and combines these figures into a similarity score:

$$\text{Sim}_{wup} = \frac{2 * depth(LCS)}{depth(concept_1) + depth(concept_2)} \tag{1.5}$$

Where depth (LCS) is depth of the least common subsume and depth(concept) is the depth in the wordnet taxonomy.

1.4.2.4 The Resnik Method

The measure introduced by Resnik [11] returns the information content (IC) of the LCS of two concepts:

$$\text{Sim}_{rex} = IC(LCS) \tag{1.6}$$

Where IC is defined as: $\text{IC(c)} = -\log P(c)$,

and P(c) is the probability of encountering an instance of concept c in a large corpus.

Lin's method which builds on Resnik's measure of similarity, and adds a normalization factor consisting of the information content of the two input concepts:

$$\text{Sim}_{lin} = \frac{2 * IC(LCS)}{IC(concept_1) + IC(concept_2)} \tag{1.7}$$

1.4.2.5 The Jiang & Conrath Method

The Jiang &Conrath method [12] returns the similarity score as :

$$\text{Sim}_{jnc} = \frac{1}{IC(concept_1) + IC(concept_2) - 2 * IC(LCS)} \tag{1.8}$$

1.4.3 Hybrid Methods

Hybrid methods use both the corpus based method and knowledge based measures. Knowledge based measures are based on the lexical resources like WordNets and Synsets. The combined method for measuring the semantic similarity of texts by exploiting the information that can be drawn from the similarity of the component words is suggested for more effective measure.

[13] uses two corpus-based measures: PMIIR (Pointwise Mutual Information and Information Retrieval) and LSA (Latent Semantic Analysis) and six knowledge-based measures [Jiang and Conrath; Leacock and Chodorow; Lesk; Lin 1998; Resnik; Wu and Palmer] of word semantic similarity, and combine the results to show how these measures can be used to derive a text-to-text similarity metric.

Sentence Similarity Based on Semantic Nets and Corpus Statistics [14] by Yuhua Li, David McLean, Zuhair A. Bandar, James D. O'Shea, and Keeley Crockett made use of a lexical database to enable the method to model human common sense knowledge and the incorporation of corpus statistics method to be adaptable to different domains. The proposed

method can be used in a variety of applications that involve text knowledge representation and discovery.Hybrid based model for semantic similarity of sentences is shown in figure 1.1.

Frequency estimates are generated from a terabyte-sized corpus of Web data, and the impact of corpus size on the effectiveness of the measures is studied in [15].The evaluation is based on one TOEFL question set and two practice questions sets, each consisting of a number of multiple choice questions seeking the best synonym for a given target word.

1.4.4 Vector Based Document Model

In vector based document model [16], the input query are matched against the number of documents.Such documents are represented as word vectors.The most relevant documents are which matched with the input query are determined.This method relies on the assumption that more similar documents have more words in common. But it is not always the case that texts with similar meaning necessarily share many words. Again, the sentence representation is not very efficient as the vector dimension is very large compared to the number of words in a short text or sentence, thus, the resulting vectors would have many null components.

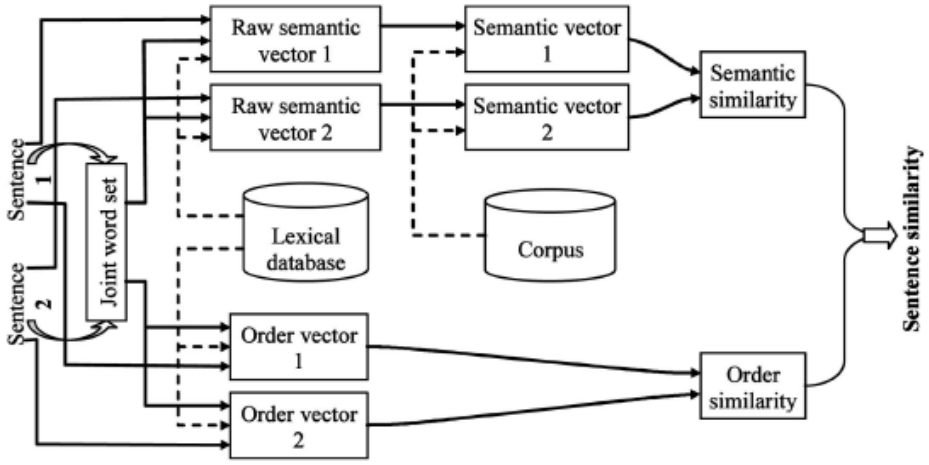

Figure 1.1 Sentence similarity computation (hybrid based measure).

CHAPTER 2

Text Similarity using Corpus Based Word Similarity and String Similarity

The consideration of string similarity measures and Word to Word semantic similarity helps to find the textual semantic similarity more efficiently. The approaches for string similarity and corpus based word to word semantic similarity are discussed in this section.

2.1 Approaches for String Similarity Measures

2.1.1 String Edit Distance

String edit distance is the minimum number of insertions, deletions, and substitutions required to transform one string into the other [17].

The edit distance D (i, j) between two strings S_1 [1...i] and S_2 [1...j] is defined recursively as:

$$D[i,j] = min\begin{Bmatrix} \mathrm{D[i-1,j]+1}, \\ \mathrm{D[i,j-1]+1}, \\ \mathrm{D[i-1,j-1]+t(i,j)} \end{Bmatrix} \quad (2.1)$$

Where,

D (i, 0) = i, D (0, j) = j and

$$t(i,j) = \begin{Bmatrix} 0 & if\ S_1[i] = S_2[j] \\ 1 & if S_1\ [i] \neq S_2[j] \end{Bmatrix}$$

The edit distance D (m, n) for two strings of length m and n can be computed in O (mn) time using dynamic programming. The obtained edit distance can be normalized to get the similarity score within the range 0 to 1.

2.1.2 Longest Common Subsequence

A common subsequence of two strings is a subsequence that appears in both strings. A longest common subsequence [18] is a common subsequence of maximal length. The longest common subsequence C (i, j) between two strings S_1 [1…i] and S_2 [1…j] can be recursively defined as:

$$c[i,j] = \begin{cases} 0 & \text{if i} = 0 \text{ or j} = 0, \\ \text{c[i} - 1, \text{j} - 1] + 1 & \text{if i, j} > 0 \; and \; \text{x}_\text{i} = y_j, \\ \max(\text{c[i, j} - 1], \text{c[i} - 1, \text{j]}) & \text{if i, j} > 0 \; and \; \text{x}_\text{i} \neq \text{y}_\text{j} \end{cases} \quad (2.2)$$

2.1.3 Normalized and Modified Version of Longest Common Subsequence

[19] uses three different modified and normalized versions of longest common subsequence (LCS) algorithm and their weighted sum for measuring string similarity between words. Let p_i and r_j be the two words from the sentences P and R respectively, then the string similarity between these two words is computed using following three modified form of LCS algorithms. Algorithm 1 and Algorithm 2 are defined for Maximal Consecutive LCS starting at first character and at any character n respectively.

- Normalized longest common subsequence(NLCS)

$$v_1 = NLCS(p_i, r_j) = \frac{Length(LCS(p_i, r_j))^2}{Length(p_i) * Length(r_j)} \quad (2.3)$$

- Normalized maximal consecutive longest common subsequence starting at first character($NMCLCS_1$)

$$v_2 = NMCLCS_1(p_i, r_j) = \frac{Length(MCLCS_1(p_i, r_j))^2}{Length(p_i) * Length(r_j)} \quad (2.4)$$

- Normalized maximal consecutive longest common subsequence starting at any character n (NMCLCSn)

$$v_3 = NMCLCS_n(p_i, r_j) = \frac{Length(MCLCS_n(p_i, r_j))^2}{Length(p_i) * Length(r_j)} \quad (2.5)$$

Weigths w_1, w_2, w_3 are choosen such that $w_1 + w_2 + w_3 = 1$ then the string similarity is computed as:

$$\alpha = w_1 v_1 + w_2 v_2 + w_3 v_3 \quad (2.6)$$

Algorithm 1. $MCLCS_1$ (Maximal Consecutive LCS starting at first character)

Input: r_i, sj /*r_i and s_j are two input strings where $|r_i| = \tau$, $|s_j|$ =ŋ and $\tau \leq$ ŋ*/
Output: r_i /*r_i is the maximal consecutive LCS starting at character 1 */

1. $\tau \leftarrow |r_i|$, ŋ$\leftarrow |s_j|$
2. while $|r_i| \geq 0$ do
3. If $r_i \cap s_j$ then /* i.e., r_i c $s_j = r_i$ */
4. return r_i
5. else
6. $r_i \leftarrow r_i \backslash c_r$ /* i.e ., remove the right most character fom r_i*/
7. end
8. end

Algorithm 2. $MCLCS_n$ (Maximal Consecutive LCS starting at character n)

Input: r_i, s_j /*r_i and s_j are two input strings where $|r_i| = \tau, |s_j| =$ ŋ and r ≤ ŋ */
Output: x /*x is the maximal consecutive LCS starting at character n */

1 $\tau \leftarrow |r_i|$, ŋ $\leftarrow |s_j|$
2. while $|r_i| \geq 0$ do
3. determine all n-grams from r_i where n=1….$|r_i|$ and
4. $\overline{r_i}$ is the set of n-grams
5. If $x \in s_j$ where $\{\{x|x \in \overline{r_i}, x = Max(\overline{r_i})\}$ then /* i is the number if n-grams and $Max(\overline{r_i})$ returns the maximum length n-gram from $\overline{r_i}$*/
6. return x
7. else
8. $\overline{r_i} \leftarrow \overline{r_i} \backslash x$ /*remove x from $\overline{r_i}$*/
9. end
10. end

2.2 Corpus based Word to Word similarity

2.2.1 Second Order Co-occurrence PMI Method

The Second Order Co-occurrence PMI (SOC-PMI) [25] method uses Pointwise Mutual Information to sort lists of important neighbor words of the two target words from a large corpus. The main advantage of using SOC-PMI is that it can calculate the similarity between two words that do not co-occur frequently, but because they mostly co-occur with the same neighboring words. Let W_1 and W_2 be the two words to determine the semantic similarity and $C = \{c_1, c_2, \ldots, c_m\}$ denotes a large corpus of text (after some preprocessing e.g., stop words elimination and lemmatization) containing m words (tokens). Also, let $T = \{t_1, t_2, \ldots, t_n\}$ be the set of all unique words (types) which occur in the corpus C. Unlike the corpus C, which is

an ordered list containing many occurrences of the same words, *T is* a set containing no repeated words. The parameter α is set, which determines how many words before and after the target word *W*, will be included in the context window. The window also contains the target word *W itself*, resulting in a window size of $2\alpha + 1$ words. The steps in determining the semantic similarity involve scanning the corpus and then extracting some functions related to frequency counts. The *type frequency* function, $f^t(t_i) = |\{k: c_k = t_i\}|$, where $i = 1, 2, ., n$ which tells us how many times the type t_i appeared in the entire corpus.

Let $f^b(t_i, W) = |\{k: t_k = W \text{ and } t_k \neq t_i\}|$,

where $i = 1, 2, ., n$ and $-\alpha \leq j \leq \alpha$, be the *bigram frequency* function $f^b(t_i, W)$ denotes how many times word t_i appeared with word W in a window of size $2\alpha + 1$ words.

Then the *pointwise mutual information* function is defined for only those words having $f^b(t_i, W) > 0$ as:

$$f^{pmi}(t_i, W) = log_2 \frac{f^b(t_i, W) * m}{f^t(t_i) f^t(W)}, \tag{2.7}$$

Where $f^t(t_i) f^t(W) > 0$ and m is total number of tokens in corpus C as mentioned earlier. Now, for word W_1, a set of words, X_w sorted in descending order by their PMI values with word w is constructed and taken the top most β_1 words having $f^{pmi}(t_i, W) > 0$.

X = {X_i}, where i=1, 2 …β_1 and

$f^{pmi}(t_1, W_1) \geq f^{pmi}(t_2, W_1) \geq \cdots f^{pmi}(t_{\beta_1}, W_1)$

Similarly, for word W_2, a set of words, Y is defined as sorted in descending order by their PMI values with W_2 and taken the top-most β_2 words having $f^{pmi}(t_i, W_2) > 0$

$Y = \{Y_i\}$, where $i = 1, 2 \ldots \beta_2$ and

$f^{pmi}(t_1, W_2) \geq f^{pmi}(t_2, W_2) \geq \cdots f^{pmi}(t_{\beta_2}, W_2)$

The value for βs (β_1 & β_2) depends on the word W and the number of types in the corpus (this will be discussed in the next section).

We define the *β-PMI summation* function. For word W_1, the *β-PMI* summation function as:

$$f^{\beta}(W_1) = \sum_{i=1}^{\beta_1} (f^{pmi}(X_i, W_2))^{\gamma} \tag{2.8}$$

Where, $f^{pmi}(X_i, W_2) > 0$ and $f^{pmi}(X_i, W_1) > 0$,

which sums all the positive PMI values of words in the set Y also common to the words in the set X. In other words, this function actually aggregates the positive PMI values of all the semantically close words of W_2 which are also common in W_1. Note that we call it semantically-close because all these words have high PMI values with W_2 and this doesn't ensure the closeness with respect to the distance within the window size.

Similarly, for word $W2$, the *β-PMI summation* function is:

$$f^{\beta}(W_2) = \sum_{i=1}^{\beta_2} (f^{pmi}(Y_i, W_1))^{\gamma} \tag{2.9}$$

Where, $f^{pmi}(Y_i, W_2) > 0$ and $f^{pmi}(Y_i, W_1) > 0$

which sums all the positive PMI values of words in the set X also common to the words in the set Y. In other words, this function aggregates the positive PMI values of all the semantically-close words of W_1 which are also common in W_2. The criteria for choosing the exponential parameter γ will be discussed in the next subsection.

Finally, we define the *semantic PMI similarity* function between two words, W_1 and W_2,

$$Sim(W_1, W_2) = \frac{f^{\beta}(W_1)}{\beta_1} + \frac{f^{\beta}(W_2)}{\beta_2} \tag{2.10}$$

2.2.1.1 Choosing the Values of β and γ

The value of β is related to how many times the word, W appears in the corpus, i.e., the frequency of W as well as the number of types in the corpus. β *is* defined as:

$$\beta = \left(\log\left(f^{t}(W_i)\right)\right)^2 \frac{(log_2(n))}{\delta} \tag{2.11}$$

where i = 1, 2 and δ is a constant. The value of δ depends on the size of the corpus. The smaller the corpus, the smaller the value of δ is to be choosen. If we lower the value of β we lose some important / interesting words, and if we increase it we consider more words common to both W_1 and W_2 and this significantly degrades the result.

γ should have a value greater than 1. The higher we choose the value of γ, the greater emphasis on words having very high PMI values with W.

2.3 Integration of String similarity and Corpus based Word to Word similarity measure

The short text similarity based on string similarity and word similarity is first proposed by Islama D. [19]. The modified and normalized version of Longest Common Subsequence (as described in section 2.1) is used as string similarity measure and Second Order Co-occurrence PMI Method is used as corpus based measure for word similarity measure. The main aim is to derive a semantic text similarity score within range of 0 and 1.

Let,

m = number of tokens in P

n = number of tokens in R such that $n \geq m$

δ = number of tokens that are matches exactly (from P and R)

$P = [p_1, p_2 \ldots . p_{m-\delta}]$

$R = [r_1, r_2 \ldots . r_{n-\delta}]$

$v_1 = NLCS(p_i, r_j)$

$v_2 = NMCLCS_1(p_i, r_j)$

$v_3 = NMCLCS_n(p_i, r_j)$

Now, the String Similarity matrix M_1 of size $(m - \delta) \times (n - \delta)$ is constructed whose each element α_{ij} is defined as:

$\alpha_{ij} = w_1 v_1 + w_2 v_2 + w_3 v_3$

$$M_1 = \begin{bmatrix} \alpha_{11} & \alpha_{12} & \ldots & \alpha_{1j} & \ldots & \alpha_{1(n-\delta)} \\ \alpha_{21} & \alpha_{22} & \ldots & \alpha_{2j} & \ldots & \alpha_{2(n-\delta)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \alpha_{i1} & \alpha_{i2} & \ldots & \alpha_{ij} & \ldots & \alpha_{i(n-\delta)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \alpha_{(m-\delta)1} & \alpha_{(m-\delta)2} & \ldots & \alpha_{(m-\delta)j} & \ldots & \alpha_{(m-\delta)(n-\delta)} \end{bmatrix} \quad (2.12)$$

Semantic similarity matrix M_2 of size $(m - \delta) \times (n - \delta)$ is constructed as:

$$M_2 = \begin{bmatrix} \beta_{12} & \beta_{12} & \ldots & \beta_{1j} & \ldots & \beta_{1(n-\delta)} \\ \beta_{21} & \beta_{22} & \ldots & \beta_{2j} & \ldots & \beta_{2(n-\delta)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \beta_{i1} & \beta_{i2} & \ldots & \beta_{ij} & \ldots & \beta_{i(n-\delta)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \beta_{(m-\delta)1} & \beta_{(m-\delta)2} & \ldots & \beta_{(m-\delta)j} & \ldots & \beta_{(m-\delta)(n-\delta)} \end{bmatrix} \quad (2.13)$$

Where, $\beta_{ij} = Sim(p_i, r_j)$

Here the integration of matrix is based on equal weights [19].

$M \leftarrow \psi M_1 + \phi M_2$

Where,

ψ=string matching matrix weight factor

ϕ =semantic similarity matrix weight factor

$$M = \begin{bmatrix} \gamma_{11} & \gamma_{12} & \cdots & \gamma_{1j} & \cdots & \gamma_{1(n-\delta)} \\ \gamma_{21} & \gamma_{22} & \cdots & \gamma_{2j} & \cdots & \gamma_{2(n-\delta)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \gamma_{i1} & \gamma_{i2} & \cdots & \gamma_{ij} & \cdots & \gamma_{i(n-\delta)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \gamma_{(m-\delta)1} & \gamma_{(m-\delta)2} & \cdots & \gamma_{(m-\delta)j} & \cdots & \gamma_{(m-\delta)(n-\delta)} \end{bmatrix} \quad (2.14)$$

Where, $\gamma_{ij} = \alpha_{ij} + \beta_{ij}$

Then, maximum valued element γ_{ij} is added to a list ρ and the corresponding rows and column are removed. This step is repeated until either $\gamma_{ij} = 0$ and/or (m – δ - $|\rho|$) = 0 or both.

Finally the similarity between P and R is computed as:

$$S(P,R) = \frac{(\delta + \sum_{i=1}^{|\rho|} \rho_i) * (m+n)}{2mn} \quad (2.15)$$

CHAPTER 3

Problem Definition

Similarity measure for long text, documents have been in research from long time but similarity measure for short text has not been given much emphasis. Short Texts and sentences similarity measures are now considered to be important research topic due to its many applications in the field of Natural language processing and information retrieval. The existing methods for computing sentence similarity have been adopted from approaches used for long text documents which results in a very high dimensional space processing, inefficient for measuring semantics and are not adaptable to all application domains[16], [20], [23].

Nepali text similarity has been studied less so far in comparison to other languages. Nepali Texts are based on Devanagari Script. Though there are other languages too based on Devanagari Script, Nepali language have different literature. Semantic similarity is the very complex topic. The semantics of the sentence may be affected by the words used within the context exihibiting different types of semantic relationship as defined in sub-section [1.2.2].

3.1 Why move to Semantic Text Similarity?

Text similarity based on lexical common terms matching does not cover the semantics of the sentence. Semantic text similarity measure is the challenging problem that should dealt with the human judgment of likeness and differences. The short text semantic similarity measure has many applications in natural language processing and information retrieval , automated answers checking, image retrieval on the web (using short text surrounding the image), semantic routing, word sense disambiguation , conversational agent, automatic text summarization based on semantics, machine translation and documents classification.

3.2 Why String Similarity Measures for Semantic Text Similarities?

String similarity measures for considering semantic similarity between Nepali texts have not been studied so far. The two strings with few differences in string (lexically) may refer to be same on the semantic basis. For such search queries, web retrieval is not so efficient rather they deal only with search based on exact keyword matching i.e lexical term similarity for Nepali words. People with different background, knowledge, and expectation organize the

information in web, users query may not be not adequate to represent the information they want to retrieve. The spellings may not be exact but may be more similar to what they mean. Keywords matching technique fails to retrieve semantically thus retrieve more irrelevant results. Such techniques are constrained by attempting to match the user keyword to the source document and present information to the user with documents that matched the user keyword [23].

For example the following two sentences, S_1 and S_2 can be considered as carrying same semantics (when the context words are considered) but the two words" प्रचन्ड "and"प्रचन्डा "are lexically different.

S_1:"**प्रचण्ड** नेपालका नेता हुन् |"

S_2:"**प्रचाण्ड** नेपालका राजनैतिक ब्यक्ति हुन् |"

Dictionary-based similarity measure cannot provide any similarity value between these two proper names. And the chance to obtain a similarity value using corpus-based similarity measures is very low. So, for such cases good similarity score can be obtained using string similarity measures [19].

In the next example given below the sentence S_3 has word "सगरमाथा" and S_4 has word "सागरमाथा" which are lexically different words but semantically they refers to same meaning.

S_3= "**सगरमाथा** हिमाल नेपालको सान हो |"

S_4= "**सागरमाथा** हिमाल नेपालको शिर हो |"

However the word "**सगरमाथा**" is the correct spelling that is found in corpus. So, in such cases the other word "**सागरमाथा**" is not found in corpus and thus frequently co-occurring words (neighbour words) cannot be found in corpus which is the significant while employing word to word similarity based on corpus or large texts. The web search results for these sentences are different which means that semantic measure is not considered for Nepali texts. Figure 3.1 shows the comparision between searches with correct spelling and incorrect spelling for Nepali language which shows that search with minor spelling mistake did not give desired result for Nepali language where as in figure 3.2, the search for English language with similar types of search returned the expected result. The Normalized and modified version of string similarity measures can be used to detect similarity between such words.

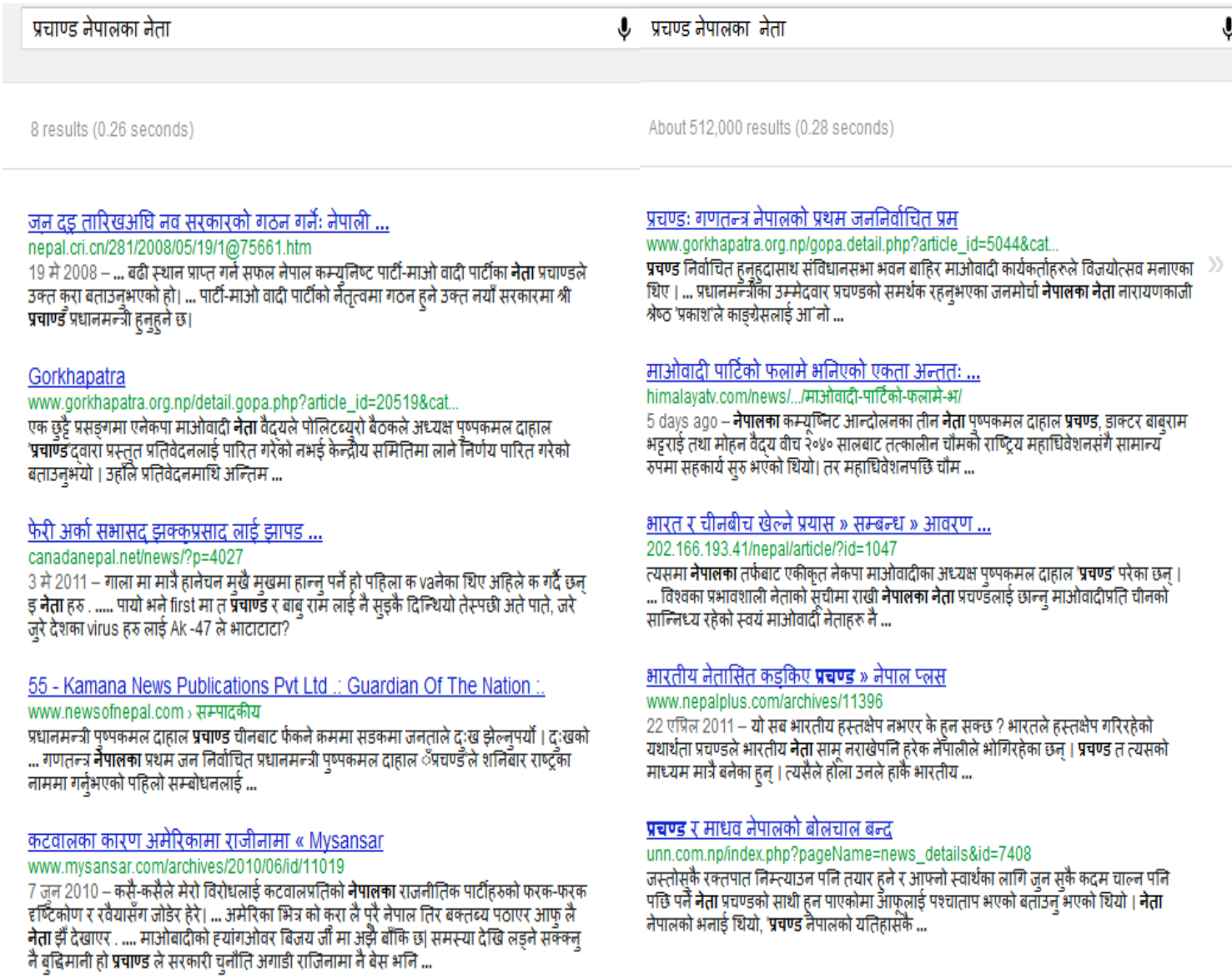

Figure 3.1 Comparision search using the minor spelling mistake and using correct spelling.

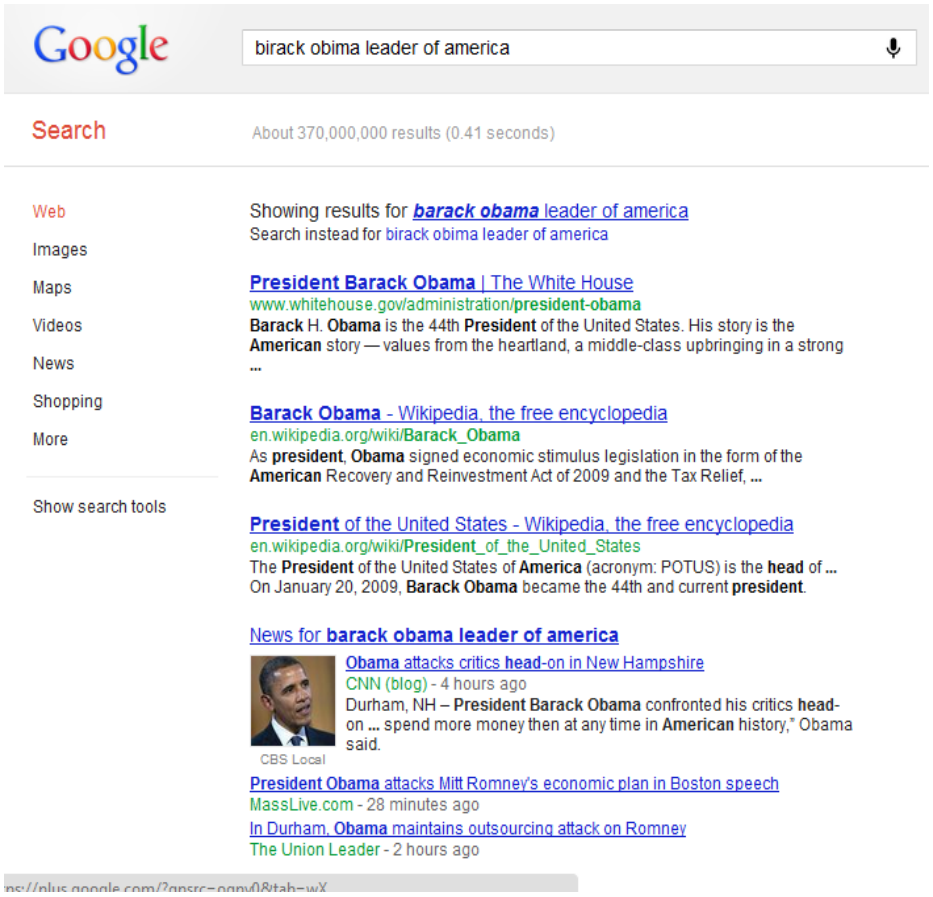

Figure 3.2 Google search with spelling mistake for English language

3.3 Why Corpus Based Word Similarity Measure for Semantic Similarities?

There is relatively large number of word-to-word similarity metrics in the literature, ranging from distance-oriented measures computed on semantic networks or knowledge-based (dictionary/thesaurus-based) measures, to metrics based on models of information theory (or corpus-based measures) learned from large text collections.

Knowledge based measures based on Word-Net or Synsets can be effective measure for semantic similarity. But many languages like Nepali do not have the resources like WordNet or Synsets. So, for such languages it is not possible to determine semantic similarity based on lexical database. And also for the sentences to be semantically similar, it is not always the case that they contain exactly same words or their synonyms. The sentences can exhibit same meaning even though they do not share more common words or their synonyms. For the language like Nepali, the corpus based word to word similarity is the solution measure for

semantic similarity. The context words for the given word can be considered to detect the semantic relatedness of the words. Such context based measures can also be helpful in determining the best synonyms from the given words options [12]. The corpus based measures are proven to be more time efficient than other WordNet based measures or hybrid measures. Moreover the real texts that are found in large corpus are often not found in lexical database [19].

For Example here two sentences S_1 and S_2 here have different lexical terms.

S_1: " **देश विकास**को लागी शिक्षा **महत्वपूर्ण** भूमिका हुन्छ |"

S_2: "**मुलुक**को **उन्नति**को लागि शिक्षा **आवश्यक** कुरा हो |"

They refer to same meaning on semantic basis even though they have different lexical terms. For the overall semantic similarity measure each word in the sentence contributes a meaning to the sentence. Word to word similarity measures based on corpus is the effective measure to find the semantic. The word "देश" and "मुलुक" are synonyms; the similarity measure should be higher for such cases. The words "भूमिका" and "कुरा" are not exact synonyms but they represent the semantic relatedness for the neighbor words like "आवश्यक" and "महत्वपूर्ण" and thus put weights for semantic similarity for wholesentence. Such relatedness is considered in corpus based measures.

CHAPTER 4

Implementation

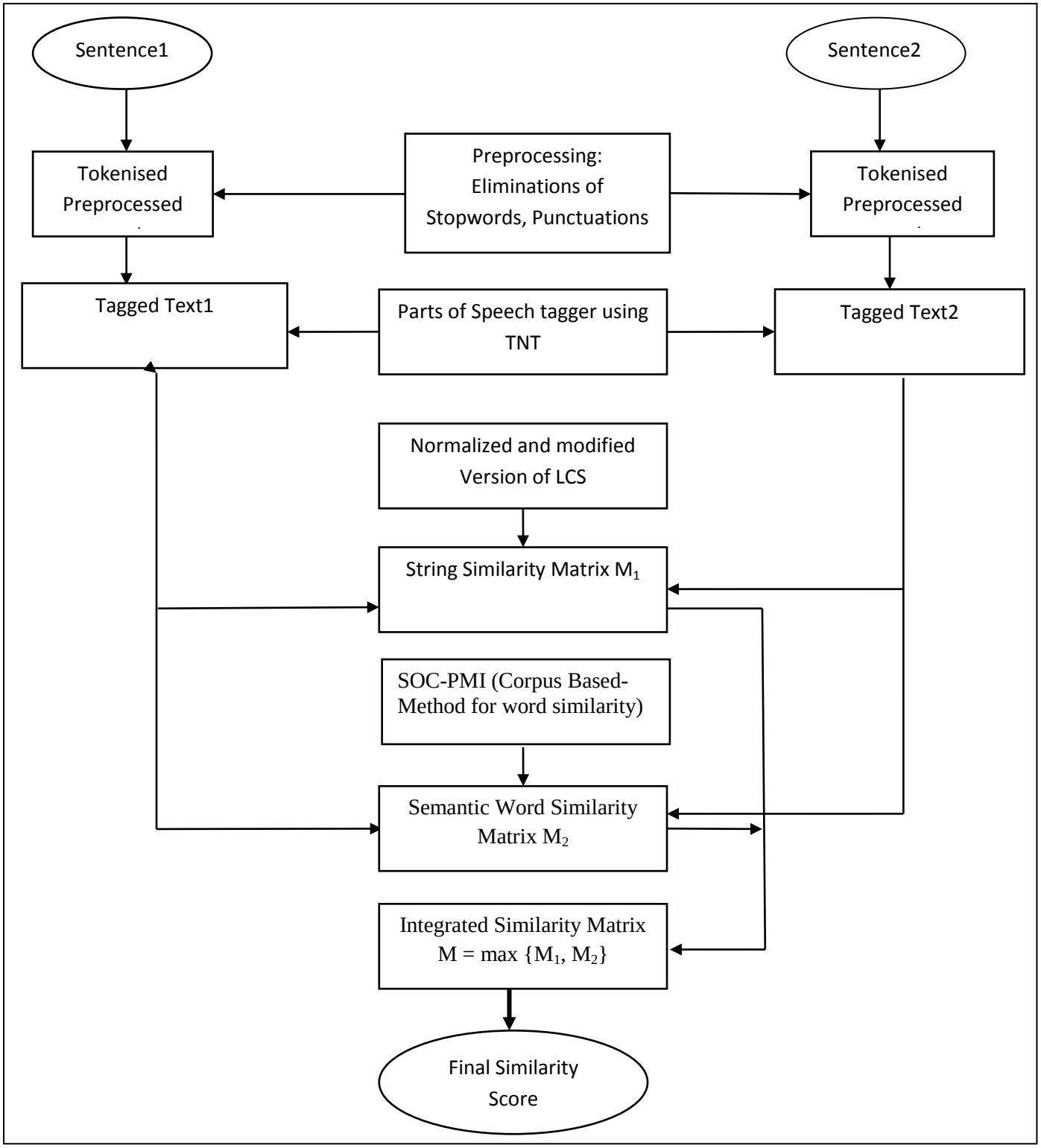

Figure 4.1 Model for the semantic Text Similarity computation

4.1 Description of the model.

At first two source sentences (Nepali language) are taken as input. The sentences go through the preprocessing stage for removal of stop words and punctuations. Stop words are words which are filtered out prior to, or after, processing of natural language data (text). There is not one definite list of stop words which all tools use, if even used. Some tools specifically avoid removing them to support phrase search. Then the tokenized sentences are tagged using the part of speech tagger. The tagged word sequences are now input for the string similarity computation and semantic word similarity computation methods. String Similarity between each pair of words from two input word sequences is computed based on modified and normalized version of Longest Common Subsequence (LCS) Algorithm. The integration of three modified and normalized version of the LCS are used in this work. They are Normalised longest common subsequence, Maximal Consecutive Normalized Longest Common Subsequence starting at first character and Maximal consecutive Normalized Longest Commin Subsequence starting at any character. The String edit distance measure and the modified versions of LCS are compared. The modified versions of LCS gave more accurate and reasonable score where as String edit distance gave more false score given a threshold of 0.5. The comparision is described in more details in chapter 5. Word similarity between each pair of words in the input word sequences is computed using Corpus based method –Second Order Co-Occurences Pointwise Mutual Information (SOC-PMI). The main advantage of using SOC-PMI is that it can calculate the similarity between two words that do not co-occur frequently, but because they mostly co-occur with the same neighboring words.

The String Similarity matrix and Semantic Similarity matrix is integrated to get the final similarity matrix. The integration method used here is different from the one that is used in in [19] which was based on the equal weigths for the string similarity score and semantic word level smiliarity score. The approach for integration used here is by chosing the maximum element comparing the both string similaritry matrix and semantic similarity matrix. From the integrated matrix the maximum valued element is extracted and the corresponding row and column of the maximum valued element is removed from the matrix thus reducing the size of matrix. This process is of extracting the maximum valued element and doing their summation is continued until the value of the element is zero or until the matrix column size

or row size is reduced to 1. Then the final similarity is computed within the range 0-1 by multiplying the summation with the harmonic mean of the lengths of the two input sentences.

4.2 Preprocessing of the Corpus

The Nepali National Corpus – General Corpus (NNC-GC) consists of around 14 million words. The sample from the Nepali National Corpus is taken for this experimentation. The sample consists of total number of words *341173* with repetition and total number of words *50940* without repetition. Preprocessing of the sample Corpus includes the following steps.

1. Removal of Punctuations, symbols and special characters from the corpus.
2. Removal of Stop words from the copus.
3. Parts of Speech (POS) tagging: The texts are tagged using TNT POS tagger.

4.3 Choice of Programming language : Java

Java is a programming language and computing platform first released by Sun Microsystems in 1995. It is the underlying technology that powers state-of-the-art programs including utilities, games, and business applications. Nowadays it is difficult to find the electronic appliances that does not support Java including mobile and TV devices.

A Java virtual machine (JVM) is a virtual machine that can execute Java bytecode. It is the code execution component of the Java software platform. The Java Virtual Machine provides a platform-independent way of executing code; programmers can concentrate on writing software, without having to be concerned with how or where it will run. It is responsible for all the things like garbage collection, array bounds checking, etc. JVM is platform dependent. Oracle Corporation is the current owner of the official implementation of the Java SE platform. This implementation is based on the original implementation of Java by Sun. The Oracle implementations are packaged into two different distributions. The Java Runtime Environment (JRE) which contains the parts of the Java SE platform required to run Java programs. This package is intended for end-users. The Java Development Kit (JDK), is intended for software developers and includes development tools such as the Java compiler, Javadoc, Jar, and a debugger.

4.4 Improvement for the integrating the String similarity and Word similarity score

Let, M_1 = String similarity matrix of size $(m - \delta) \times (n - \delta)$

M_2 = Semantic similarity matrix of size $(m - \delta) \times (n - \delta)$

Where m and n is the length of the sentences P and R respectively and δ is count for exactly same words.

$$M_1 = \begin{bmatrix} \alpha_{11} & \alpha_{12} & \dots & \alpha_{1j} & \dots & \alpha_{1(n-\delta)} \\ \alpha_{21} & \alpha_{22} & \dots & \alpha_{2j} & \dots & \alpha_{2(n-\delta)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \alpha_{i1} & \alpha_{i2} & \dots & \alpha_{ij} & \dots & \alpha_{i(n-\delta)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \alpha_{(m-\delta)1} & \alpha_{(m-\delta)2} & \dots & \alpha_{(m-\delta)j} & \dots & \alpha_{(m-\delta)(n-\delta)} \end{bmatrix} \quad (4.1)$$

Where $\alpha_{ij} = StringSimilarity(p_i, r_j)$

$$M_2 = \begin{bmatrix} \beta_{12} & \beta_{12} & \dots & \beta_{1j} & \dots & \beta_{1(n-\delta)} \\ \beta_{21} & \beta_{22} & \dots & \beta_{2j} & \dots & \beta_{2(n-\delta)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \beta_{i1} & \beta_{i2} & \dots & \beta_{ij} & \dots & \beta_{i(n-\delta)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \beta_{(m-\delta)1} & \beta_{(m-\delta)2} & \dots & \beta_{(m-\delta)j} & \dots & \beta_{(m-\delta)(n-\delta)} \end{bmatrix} \quad (4.2)$$

Where, $\beta_{ij} = WordSimilarity(p_i, r_j)$

In [19] the integration of matrix is based on equal weights.

$$M \leftarrow \psi M_1 + \phi M_2 \quad (4.3)$$

Where, ψ=string matching matrix weight factor.

ϕ =semantic similarity matrix weight factor.

Two words with few spelling differences would have more string similarity then the word level semantic similarity. Moreover misspelled words are often not found in the corpus so obviously the word level similarity returns 0 for such cases. In this case integration by equal weights method reduces the similarity score. So instead the maximum value returned by comparing the both measures could be used which also enhance the similarity scores for many cases which have slightly misspelled words which would have 0 similarity based on corpus. For such case considering only the string similarity measures gives nice score.

So, $M \leftarrow \psi M_1 + \phi M_2$, is changed to $M \leftarrow \{m: m \in M_1 \text{ if } M_1 > M_2 \text{ else } m \in M_2\}$,

$$M = \begin{bmatrix} \gamma_{11} & \gamma_{12} & \cdots & \gamma_{1j} & \cdots & \gamma_{1(n-\delta)} \\ \gamma_{21} & \gamma_{22} & \cdots & \gamma_{2j} & \cdots & \gamma_{2(n-\delta)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \gamma_{i1} & \gamma_{i2} & \cdots & \gamma_{ij} & \cdots & \gamma_{i(n-\delta)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \gamma_{(m-\delta)1} & \gamma_{(m-\delta)2} & \cdots & \gamma_{(m-\delta)j} & \cdots & \gamma_{(m-\delta)(n-\delta)} \end{bmatrix} \quad (4.4)$$

For example, for the word pairs "सगरमाथा" and "सागरमाथा", the word level semantic similarity gives 0 score but the string similarity measure gives the score of 0.60 So, if we take equal weights of the word similarity and string similarity the obtained similarity is halved (0.30) which degrades the overall similarity score.

4.5 A Walk through Example.

4.5.1 Example considering Corpus Based Word Similarity

The sample from the large corpus Nepali National corpus-General corpus (NNC-GC) is used for experiments.

Let P= "देश विकासको लागी शिक्षा महत्वपूर्ण भूमिका हुन्छ ।"

R= "मुलुकको उन्नतिको लागि शिक्षा आवश्यक कुरा हो । "

Step 1: Part of speech tagging for the input sentences

देश\NN विकास\NN को\PKO लागी\VBO शिक्षा\NN महत्वपूर्ण \JJ भूमिका\NN हुन्छ\VB |\YF

मुलुक\NN को\PKO उन्नति\NN को\PKO लागि\VBO शिक्षा\NN आवश्यक\JJ कुरा\NN हो\VB |\YF

Step 2: Elimination of all special characters, punctuations and stop words.

P = {देश विकास लागी शिक्षा महत्वपूर्ण भूमिका} and

R = {मुलुक उन्नति लागी शिक्षा आवश्यक कुरा} where m = 6 and n = 6.

Step 3: Removal of exactly matched tokens.Here, only two tokens (i.e., शिक्षा, लागी) in P exactly matches with R therefore we set δ (exactly matched token) to 2. We remove. शिक्षा, लागी from both P and R.

P = {देश विकास महत्वपूर्ण भूमिका} and

R = {मुलुक उन्नति आवश्यक कुरा}.

As m− δ≠ 0, we proceed to next step.

Step 4: Constrution of a 4×4 string similarity matrix, M_1. Normalized and modified Version of LCS is used for string similarity computation.

$$M_1 = \begin{bmatrix} & \text{देश} & \text{विकास} & \text{महत्वपूर्ण} & \text{भूमिका} \\ \text{मुलुक} & 0 & 0 & 0.015 & 0.075 \\ \text{उन्नति} & 0 & 0.017 & 0 & 0.014 \\ \text{आवश्यक} & 0.028 & 0.067 & 0 & 0.014 \\ \text{कुरा} & 0 & 0.112 & 0 & 0.094 \end{bmatrix} \quad (4.5)$$

Step 5: Constrution of a 4×4 semantic similarity matrix, M_2. SOCPMI method is used for corpus based word semantic similarity measure.

$$M_2 = \begin{bmatrix} & \text{देश} & \text{विकास} & \text{महत्वपूर्ण} & \text{भूमिका} \\ \text{मुलुक} & 0.976 & 0.660 & 0.621 & 0.490 \\ \text{उन्नति} & 0.326 & 0.297 & 0.253 & 0.288 \\ \text{आवश्यक} & 0.640 & 0.539 & 0.638 & 0.439 \\ \text{कुरा} & 0.595 & 0.533 & 0.540 & 0.557 \end{bmatrix} \quad (4.6)$$

Step 6: A 4 × 4 joint matrix is constructed: M, Here the integration is different than in the method described in [19]. The string similarity matrix M_1 is already computed before M_2 so the maximum valued element can be choosen as value for element of the matrix M.

$$\mathrm{M} = \begin{bmatrix} & \text{देश} & \text{विकास} & \text{महत्वपूर्ण} & \text{भूमिका} \\ \text{मुलुक} & \mathbf{0.976} & 0.660 & 0.621 & 0.490 \\ \text{उन्नति} & 0.326 & 0.297 & 0.253 & 0.288 \\ \text{आवश्यक} & 0.640 & 0.539 & 0.638 & 0.439 \\ \text{कुरा} & 0.595 & 0.533 & 0.540 & 0.557 \end{bmatrix} \quad (4.7)$$

Step 7: The maximum-valued matrix-element, γ_{ij} is extracted from the matrix and is added this matrix element to a list (say, ρ and $\rho \leftarrow \rho \cup \gamma_{ij}$) if $\gamma_{ij} \geq 0$. All the matrix elements of i^{th} row and j^{th} column from M are removed.

Step 8: The process of the finding of the maximum-valued matrix-element, γ_{ij} adding it to ρ and removing all the matrix elements of the corresponding row and column is repeated until either $\gamma_{ij} = 0$, or $m - \delta - |\rho| = 0$, or both.

Maximum valued element is [देश] [मुलुक] =0.976

After removing all the matrix elements of the corresponding row and column, we get matrix:

$$M = \begin{bmatrix} & \text{विकास} & \text{महत्वपूर्ण} & \text{भूमिका} \\ \text{उन्नति} & 0.297 & 0.253 & 0.288 \\ \text{आवश्यक} & 0.539 & \mathbf{0.638} & 0.439 \\ \text{कुरा} & 0.533 & 0.540 & 0.557 \end{bmatrix} \quad (4.8)$$

Next Maximum valued element is [महत्वपूर्ण] [आवश्यक] =0.638

After removing all the matrix elements of the corresponding row and column, we get matrix:

$$M = \begin{bmatrix} & \text{विकास} & \text{भूमिका} \\ \text{उन्नति} & 0.297 & 0.288 \\ \text{कुरा} & 0.533 & \mathbf{0.557} \end{bmatrix} \quad (4.9)$$

Next Maximum valued element is [भूमिका] [कुरा] = 0.557

Next Maximum valued element is [महत्वपूर्ण] [आवश्यक] =0.638

After removing all the matrix elements of the corresponding row and column, we get matrix:

$$M = \begin{bmatrix} & \text{विकास} \\ \text{उन्नति} & \mathbf{0.297} \end{bmatrix} \quad (4.10)$$

Next Maximum valued element is [विकास] [उन्नति] =0.297

Now, $m - \delta - |\rho| = 0$ so we stop.

This total score is multiplied by the reciprocal harmonic mean of m and n to obtain a similarity score between 0 and 1, inclusively. The following formula is used for the overall similarity..

$$S(P,R) = \frac{(\delta + \sum_{i=1}^{|\rho|} \rho_i) * (m+n)}{2mn}$$

$$S(P,R) = \frac{(2 + 2.55) * (6+6)}{2 * 6 * 6}$$

$$= 0.692$$

Overall sentence similarity is 0.692

4.3.2 Example Considering String Similarity and Corpus Based Word Similarity

Let, P= "**प्रचण्ड** नेपालका नेता हुन् |"

R= "**प्रचाण्ड** नेपालका राजनैतिक ब्यक्ति हुन् |"

Step 1: Part of speech tagging for the input sentences

प्रचण्ड\JJ नेपाल\NNP का\PKO नेता\NN हुन्\VB |\YF

प्रचाण्ड\NNP नेपाल\NNP का\PKO राजनैतिक\JJ ब्यक्ति\NN हुन्\VB |\YF

Step 2: Elimination of all special characters, punctuations and stop words

P= "प्रचण्ड नेपाल नेता"

R= " प्रचाण्ड नेपाल राजनैतिक ब्यक्ति" where m = 4 and n = 5.

Step 3: Removal of exactly matched tokens.Here, only one tokens (i.e., नेपाल) in P exactly matches with R therefore we set δ (exactly matched token) to 1. We remove नेपाल from both P and R.

P= "प्रचण्ड नेता"

R= " प्रचाण्ड राजनैतिक ब्यक्ति"

As m− δ≠ 0, we proceed to next step.

Step 4: Constrution of a 4×2 String similarity matrix, M_1. Normalized and modified Version of LCS is used for string similalrity computation.

$$M_1 = \begin{bmatrix} & \text{प्रचाण्ड} & \text{राजनैतिक} & \text{ब्यक्ति} \\ \text{प्रचण्ड} & 0.58 & 0.013 & 0.040 \\ \text{नेता} & 0.015 & 0.062 & 0.017 \end{bmatrix} \quad (4.11)$$

Step 5: Constrution of a 4×2 Semantic similarity matrix, M_2. SOCPMI method is used for corpus based word semantic similarity measure.

$$M_2 = \begin{bmatrix} & \text{प्रचाण्ड} & \text{राजनैतिक} & \text{ब्यक्ति} \\ \text{प्रचण्ड} & 0 & 0.403 & 0.171 \\ \text{नेता} & 0 & 0.329 & 0.292 \end{bmatrix} \quad (4.12)$$

Step 6: Constrution of a 4×2 integrated similarity matrix, M

$$M = \begin{bmatrix} & \text{प्रचाण्ड} & \text{राजनैतिक} & \text{ब्यक्ति} \\ \text{प्रचण्ड} & \mathbf{0.58} & 0.403 & 0.171 \\ \text{नेता} & 0.015 & 0.329 & 0.292 \end{bmatrix} \quad (4.13)$$

Step 7: Maximum valued elements are extracted from the matrix.

Maximum valued element is [प्रचण्ड] [प्रचाण्ड] =0.58

After removing all the matrix elements of the corresponding row and column, we get matrix

$$M = \begin{bmatrix} & \text{राजनैतिक} & \text{ब्यक्ति} \\ \text{नेता} & \mathbf{0.329} & 0.292 \end{bmatrix} \quad (4.14)$$

Next Maximum valued element is [नेता] [राजनैतिक] =0.329

Now, m− δ − |ρ| = 0 so we stop.

The final similarity is computed using the formula:

$$S(P,R) = \frac{(\delta + \sum_{i=1}^{|\rho|} \rho_i) * (m+n)}{2mn}$$

$$S(P,R) = \frac{(1 + 1.20) * (3+4)}{2 * 3 * 4}$$

$$= 0.55$$

Therefore the overall sentence similarity score is `0.55`

CHAPTER 5

Testing and Analysis

5.1 Some Tested Pair of Sentences

The sample from the Nepali National Corpus is taken for this experimentation. The sample consists of total number of words *341173* with repetition and total number of words *50940* without repetition. Here in the table 5.1, for the first example all the string similarity values and word semantic similarity values are listed and among them the maximum valued similarity score between words are listed which are used for overall similarity. For the all other examples only the maximum valued elements are listed. The other column contains the overall sentence similarity score computed.

Sentence Pairs and word similarity scores	Sentence Similarity Score
P = "देश विकासको लागी शिक्षा महत्वपूर्ण भूमिका हुन्छ ।" R = "मुलुकको उन्नतिको लागी शिक्षा आवश्यक कुरा हो ।" StringSimilarity[देश][मुलुक]=0 StringSimilarity[देश][उन्नति]=0 StringSimilarity[देश][आवश्यक]=0.028 StringSimilarity[देश][कुरा]=0 StringSimilarity[विकास][मुलुक]=0.02 StringSimilarity[विकास][उन्नति]=0.017 StringSimilarity[विकास][आवश्यक]=0.067 StringSimilarity[विकास][कुरा]=0.112 StringSimilarity[महत्वपूर्ण][मुलुक]=0.015 StringSimilarity[महत्वपूर्ण][उन्नति]=0.008 StringSimilarity[महत्वपूर्ण][आवश्यक]=0.033 StringSimilarity[महत्वपूर्ण][कुरा]=0.012 StringSimilarity[भूमिका][मुलुक]=0.075 StringSimilarity[भूमिका][उन्नति]=0.014 StringSimilarity[भूमिका][आवश्यक]=0.014 StringSimilarity[भूमिका][कुरा]=0.094 SemanticSimilarity[देश][मुलुक]=0.976 SemanticSimilarity[देश][उन्नति]=0.326 SemanticSimilarity[देश][आवश्यक]=0.64	0.692

SemanticSimilarity[देश][कुरा]=0.595
SemanticSimilarity[विकास][मुलुक]=0.66
SemanticSimilarity[विकास][उन्नति]=0.297
SemanticSimilarity[विकास][आवश्यक]=0.539
SemanticSimilarity[विकास][कुरा]=0.533
SemanticSimilarity[महत्वपूर्ण][मुलुक]=0.621
SemanticSimilarity[महत्वपूर्ण][उन्नति]=0.253
SemanticSimilarity[महत्वपूर्ण][आवश्यक]=0.638
SemanticSimilarity[महत्वपूर्ण][कुरा]=0.54
SemanticSimilarity[भूमिका][मुलुक]=0.49
SemanticSimilarity[भूमिका][उन्नति]=0.288
SemanticSimilarity[भूमिका][आवश्यक]=0.439
SemanticSimilarity[भूमिका][कुरा]=0.557

IntegratedSimilarity[देश][मुलुक]=0.976
IntegratedSimilarity[देश][उन्नति]=0.326
IntegratedSimilarity[देश][आवश्यक]=0.64
IntegratedSimilarity[देश][कुरा]=0.595
IntegratedSimilarity[विकास][मुलुक]=0.66
IntegratedSimilarity[विकास][उन्नति]=0.297
IntegratedSimilarity[विकास][आवश्यक]=0.539
IntegratedSimilarity[विकास][कुरा]=0.533
IntegratedSimilarity[महत्वपूर्ण][मुलुक]=0.621
IntegratedSimilarity[महत्वपूर्ण][उन्नति]=0.253
IntegratedSimilarity[महत्वपूर्ण][आवश्यक]=0.638
IntegratedSimilarity[महत्वपूर्ण][कुरा]=0.54
IntegratedSimilarity[भूमिका][मुलुक]=0.49
IntegratedSimilarity[भूमिका][उन्नति]=0.288
IntegratedSimilarity[भूमिका][आवश्यक]=0.439
IntegratedSimilarity[भूमिका][कुरा]=0.557

Here the maximum valued elements are extracted for overall sentence similarity score are as follows:

1. SemanticSimilarity[देश][मुलुक]=0.976
2. SemanticSimilarity[महत्वपूर्ण][आवश्यक]=0.638
3. SemanticSimilarity[भूमिका][कुरा]=0.557
4. SemanticSimilarity[विकास][उन्नति]=0.297

P= "प्रचाण्ड नेपालका नेता हुन् ।" R= "प्रचण्ड नेपालका राजनैतिक व्यक्ति हुन् ।" 1. StringSimilarity[प्रचाण्ड][प्रचण्ड]=0.58 2. SemanticSimilarity[नेता][व्यक्ति]=0.306	0.55
P= "कविहरु सृजनशील हुन्छन ।" R= "साहित्यकारहरु राम्रा कृति सृजना गर्छन ।" 1. SemanticSimilarity[कवि][कृति]=0.761 2. SemanticSimilarity[सृजनशील][सृजना]=0.537	0.379
P= "अमेरिकाले विकासको क्षेत्रमा धेरै प्रगति गरेको छ ।" R= "अमेरिका विकसित मुलुक हो ।" 1. SemanticSimilarity[मुलुक][क्षेत्र]=0.759 2. SemanticSimilarity[विकास][विकसित]=0.713	0.66
P= "सगरमाथा हिमाल नेपालको सान हो ।" R= "सागरमाथा हेमाल नेपालको शिर हो ।" 1. StringSimlarity[सगरमाथा][सागरमाथा]=0.603 2. StringSimlarity[हिमाल][हेमाल]=0.42	0.506
P= “उनी सुन्दर छे ।” R="उनी मनमोहक छे ।” 1. SemanticSimilarity[सुन्दर][मनमोहक]=0.507	0.836
P= "कर तिर्नु नागरिकको दायित्व हो ।" R= "कर तिर्नु हरेकको कर्तव्य हो ।" 1 SemanticSimilarity[दायित्व][कर्तव्य]=0.52 2 SemanticSimilarity[नागरिक][हरेक]=0.351	0.718
P= "शिक्षा हाम्रो हक हो ।" R= "शीक्षा हाम्रो अधिकार हो ।" 1. SemanticSimilarity[हक][अधिकार]=0.57 2. StringSimlarity[शिक्षा][शीक्षा]=0.465	0.678
P= "शान्ति हरेक नेपालीको चाहना हो ।" R= "सबैको इच्छा अहिंसा हो ।" 1. SemanticSimilarity[हरेक][सबै]=0.541 2. SemanticSimilarity[चाहना][इच्छा]=0.441 3. SemanticSimilarity[नेपाली][सबै]=0.439	0.363

4. SemanticSimilarity[शान्ति][अहिंसा]=0.26	
P="आफ्नो कला संस्कृतिलाइ संरक्षण गर्नु सबैको कर्तव्य हो ।" R="हाम्रो परम्परा रीतिरिवाज सुरक्षा गर्नु दायित्व हो ।" 1. SemanticSimilarity[परम्परा][संस्कृति]=0.683 2. SemanticSimilarity[रीतिरिवाज][संस्कृति]=0.569 3. SemanticSimilarity[परम्परा][कला]=0.536 4. SemanticSimilarity[दायित्व][कर्तव्य]=0.52 5. SemanticSimilarity[सुरक्षा][संरक्षण]=0.515	0.51
P="गुणस्तरीय शिक्षा राज्यको दायित्व हो ।" R="राम्रो शीक्षा सरकारको जिम्मेवारी हो ।" 1. SemanticSimilarity[सरकार][राज्य]=0.830 2. SemanticSimilarity[दायित्व][जिम्मेवारी]=0.832 3. SemanticSimilarity[गुण][राम्रो]=0.352	0.754
P="जीवन संघर्ष हो ।“ R= "जिन्दगीमा चुनौतीको सामना गर्नुपर्छ ।" 1. SemanticSimilarity[संघर्ष][चुनौती]=0.467 2. SemanticSimilarity[जीवन][जिन्दगी]=0.28	0.28
P="नेपालमा पर्यटन ब्यबसायको राम्रो सम्भावना छ ।" R="नेपालमा पर्यटक धेरै आउने हुँदा त्यसबाट राम्रो आर्थिक कारोबार हुनसक्छ ।" 1. SemanticSimilarity[पर्यटन][पर्यटक]=0.95 2. SemanticSimilarity[व्यवसाय][कारोबार]=0.667 3. SemanticSimilarity[व्यवसाय][आर्थिक]=0.645	0.673
P="कडा परिश्रम गरेर काम गरे सफल भईन्छ ।" R="मिहिनेत,निरन्तरता दिएर काम गरे सफल हुन्छौ ।" 1. SemanticSimilarity[परिश्रम][मिहिनेत]=0.663 2. SemanticSimilarity[परिश्रम][काम]=0.362	0.636
P="जीवनमा सफल हुन सकारात्मक सोच राख्न जरूरी छ ।" R="राम्रो सोच सफल जीवनको संकेत हो ।" 1. SemanticSimilarity[सकारात्मक][राम्रो]=0.432 2. SemanticSimilarity[जरूरी][राम्रो]=0.68	0.686
P="गरिब असहायको सहयोग गर्नु राम्रो काम हो ।" R="गरिब सेवा गर्नु धर्म हो ।" 1. SemanticSimilarity[सेवा][सहयोग]=0.586	0.665

2. SemanticSimilarity[सेवा][राम्रो]=0.503 3. SemanticSimilarity[धर्म][राम्रो]=0.282	
P="गौतम बुद्धा संसार शान्ति प्रचार हुन् ।" R="बुद्ध विश्व शान्ति संदेश फैलाए ।" 1. StringSimlarity[बुद्धा][बुद्ध]=0.833 2. StringSimlarity[गौतम][गौतमा]=0.80 3. SemanticSimilarity[प्रचार][सन्देश]=0.199 4. SemanticSimilarity[संसार][विश्व]=0.332	0.519
P="देशको लागी त्याग गर्नु सक्नुपर्छ ।" R="मुलुकको लागी बलिदान दिन सक्नुपर्छ ।" 1. SemanticSimilarity[देश][मुलुक]=0.976 2. SemanticSimilarity[त्याग][बलिदान]=0.209	0.637
P="नेपालमा रोजगारीको अभाव छ ।" R="नेपालमा जागिरको समस्या छ ।" 1. SemanticSimilarity[अभाव][समस्या]=0.663 2. SemanticSimilarity[रोजगारी][जागिर]=0.177	0.613
P="स्वदेशको माया सबलाई हुन्छ ।" R="आफ्नो देश सबलाई प्यारो लाग्छ ।" 1. StringSimlarity[स्वदेश][देश]=0.5 2. SemanticSimilarity[माया][प्यारो]=0.12	0.432
P="मेरोडीना एक प्रसिद्ध खेलाडी हुन् ।" R="मेरोडोना एक लोकप्रिय खेलाडी हुन् ।" 1. StringSimlarity[मेरोडीना][मेरोडोना]=0.578 2. SemanticSimilarity[प्रसिद्ध][लोकप्रिय]=0.586	0.722
P="जीवनमा समयको महत्व धेरै हुन्छ ।" R="जिन्दगीमा हरेक पल महत्वपूर्ण हुन्छ ।" 1. SemanticSimilarity[महत्व][महत्वपूर्ण]=0.659 2. SemanticSimilarity[समय][हरेक]=0.692 3. SemanticSimilarity[जीवन][जिन्दगी]=0.28	0.476
P="देशको प्रगतिको लागि राम्रा नेताहरु आवश्यक ।" R="मुलुकको उन्नतिका लागि मन्त्रीहरु राम्रा जरूरी ।" 1. SemanticSimilarity[देश][मुलुक]=0.976 2. SemanticSimilarity[मन्त्री][नेता]=0.381 3. SemanticSimilarity[प्रगति][उन्नति]=0.418 4. SemanticSimilarity[आवश्यक][जरूरी]=0.29	0.678
P="सबैले नियम पालना गर्नुपर्छ ।"	0.75

R="हरेकले कानून पालना गर्नुपर्छ ।" 1. SemanticSimilarity[नियम][कानून]=0.708 2. SemanticSimilarity[सबै][हरेक]=0.544	
P="कम्प्युटरले जीवन धेरै सहज बनाएको छ ।" R="कम्प्युटारले आधुनिक जीवनमा धेरै सहयोग गरेको छ ।" 1. StringSimlarity[कम्प्युटर][कम्प्युटार]=0.806 2. SemanticSimilarity[सहज][आधुनिक]=0.379	0.546
P="मुनामदन निकै लोकप्रिय कथा मानिन्छ ।" R="मूनामदन नेपाली साहित्यमा प्रसिद्ध कृति हो ।" 1. SemanticSimilarity[कथा][साहित्य]=0.45 2. SemanticSimilarity[लोकप्रिय][प्रसिद्ध]=0.586 3. StringSimlarity[मुनामदन][मूनामदन]=0.50 4. SemanticSimilarity[कथा][कृति]=0.48	0.441
P = "भगवानमा विश्वास राख्नुपर्छ ।" R = "भगवानमा श्रद्धा गर्नुपर्छ ।" SemanticSimilarity[विश्वास][श्रद्धा]=0.187	0.396
P="शिक्षा महत्व बुझ्नुपर्छ ।" R="ज्ञान महत्वपूर्ण छ ।" 1. SemanticSimilarity[शिक्षा][ज्ञान]=0.486 2. SemanticSimilarity[महत्व][महत्वपूर्ण]=0.659	0.572
P="उसंग शक्ति छ ।" R="उ बलियो छ ।" SemanticSimilarity[शक्ति][बलियो]=0.458	0.729
P= "प्रचण्ड नेपालका एक नेता हुन् ।" P= "कविहरु सृजनशील हुन्छन ।" 1. SemanticSimilarity[नेपाल][सृजनशील]=0.048 2. SemanticSimilarity[प्रचन्ड][कवि]=0	0.101
P= "शिक्षा हाम्रो हक हो ।" R= "बुद्ध शान्तिका दुत हुन ।" 1. SemanticSimilarity[हक][बुद्ध]=0.111 2. SemanticSimilarity[शिक्षा][शान्ति]=0.262	0.124
P="नेपालमा पर्यटन ब्यबसायको राम्रो सम्भावना छ ।" R= "मुनामदन निकै लोकप्रिय कथा हो ।" 1. SemanticSimilarity[ब्यबसाय][मुनामदन]=0 2. SemanticSimilarity[राम्रो][निकै]=0.576	0.205

3. SemanticSimilarity[नेपाल][कथा]=0.218 4. SemanticSimilarity[पर्यटन][मुनामदन]=0	
P = "मेरो नाम श्याम हो ।" R= "मेरो नाम राम हो ।" SemanticSimilarity[श्याम][राम]=0.215	0.738
P=" देश विकासको लागी शिक्षा महत्वोपूर्ण भूमिका हुन्छ ।" R= "मुलुकको उन्नतिको लागि आर्थिक विकास आवश्यक कुरा हो ।" 1. SemanticSimilarity[देश][मुलुक]=0.976 2. SemanticSimilarity[विकास][उन्नति]=0.297 3. SemanticSimilarity[महत्वपूर्ण][आवश्यक]=0.638 4. SemanticSimilarity[भूमिका][कुरा]=0.557 5. SemanticSimilarity[विकास][आर्थिक]=0.823	0.654
P="विज्ञानको बिकासले मान्छेको जीवनमा धेरै परिवर्तन ल्याएको छ ।" R="वैज्ञानिक उपलब्धिले मानिसको जीवन नयाँ मोडमा ल्याएको छ ।" 1. SemanticSimilarity[विज्ञान][वैज्ञानिक]=0.738 2. SemanticSimilarity[परिवर्तन][नयाँ]=0.619 3. SemanticSimilarity[विकास][वैज्ञानिक]=0.572	0.611

Table 5.1 Semantic Text similarity measures for the pair of sentences

5.2 Testing Accuracy

Number of sentences analyzed = 35

Number of correct predicted sentences = 26

Accuracy = (26/35) ×100 = 74%

The results are evaluated in terms of accuracy, the number of pairs predicted correctly divided by the total number of pairs. Recall is defined as the percentage of pairs in the manually annotated pairs set identified by the method and precision is defined as the percentage of pairs returned by the method that also occurred in the manually annotated pairs set. In general, it is easy to obtain high performance for one of the two measures but relatively difficult to obtain high performance for both.

F-measure (*F*) is the geometric mean of precision (*P*) and recall (*R*) and expresses a trade-off between those two measures. These performance measures are defined as follows:

$$P = TP/(TP + FP) \quad (5.1)$$

$$R = TP/(TP + FN) \quad (5.2)$$

$$F = (1 + \beta)\, PR/(\beta P + R)$$

$$= 2PR / (P + R), \quad (5.3)$$

With $\beta = 1$ such that precision and recall weighted equally.

where, *TP* = True Positive (how many pairs of sentences were similar and they were indeed labeled as similar in the data set), FP = False Positive (how many were classified as non similar while they truly are similar), and FN = False Negative (how many were labeled as similar when they should not have been), respectively.

So,

P = 26/ (26+9) = 0.74

R = 26/ (26+3) = 0.89

F = (2*0.74*0.89)/ (0.74+0.89)

= 0.80

5.4 Comparison of String Edit Distance and Normalized and Modified Version of LCS for some pair of Words

Tested Word pairs	**Similarity score using String Edit Distance measure**	**Similarity score using normalized and modified version of LCS**
प्रचण्ड प्रचण्डा	0.875	0.875
हेमालय हिमालय	0.833	0.465
नेपाल जेपाल	0.80	0.480

राम्रो नराम्रो	0.857	0.642
कम्प्युटर कम्प्युटार	0.900	0.805
मुनामदन मूनामदन	0.857	0.500
मेरोडीना मेरोडोना	0.875	0.578
शीक्षा शिक्षा	0.833	0.465
सगरमाथा सागरमाथा	0.875	0.602
हिमाल हिलमा	0.600	0.360

Table 5.2 String similarity measures comparision

Analysis:

The string edit distance gives higher similarity values than the normalized and modified version of LCS but it does not consider the maximal consequetive sequences starting from the first character and /or any other character within the word. The result is that it gives higher similairy value even for those which are not similar.

Example for set of words pairs

1. "हिमाल" and " हिलमा" : string edit distance measure= 0.60 and LCS measure = 0.36

2. "जेपाल" and " नेपाल" : string edit distance measure= 0.80 and LCS measure = 0.48

Given the threshold value of 0.5, string edit distance measures would give more false scores for similarity measure but considering the modified and normalized version of LCS gives more accurate and reasonable score.

5.5 Complexity Analysis

(i) Three string similarity functions are used to determine a combined string similarity score. The time complexity calculation is straightforward. Assuming that the maximum length of the two strings is m, the time complexity of LCS, $MCLCS_1$ and $MCLCS_n$ are O (m^2), O (m^2) and O (m^2), respectively. So, the total complexity of the string matching is O (m^2).

(ii) The semantic word similarity method (SOCPMI) used has a linear time complexity with the size of the corpus (the size of the NNC). The quadratic time complexity of the window size (10 words) can be ignoresd as it is much smaller than the size N of the corpus. So the total complexity of the corpus-based similarity matching is O (N).

(iii) Finally, the two measures are combined; this has a quadratic complexity of the size of the matrices. If s is the length of the longest sentence, then the time complexity is O (s^2).

The total complexity is O (N) + O (m^2) + O (s^2).

CHAPTER 6

Conclusion and Recommendation

6.1 Conclusion

As the information technology is depending more on semantic information, the need for semantic level text processing is growing higher. The strict lexical text similarity measures do not consider the semantics of the text, moving towards the semantic similarity approach is the ultimate goal. The semantic similarity measure used here is the unsupervised model (there is no training data available). The consideration of string similarity contributed for finding similarity between sentences even though they have little differences like the inclusion of misspelled words. The word to word similarity based on corpus contributed for finding semantic similarity between words.

The time complexity of the algorithms is given mainly by the number of searches in the corpus and in wordnet. Since only one corpus base method has been used and no word net based approach used, it has lower time complexity than the other system based on wordnet and multiple corpus based approaches.The improvement has been done when integrating the similarity values obtained from string similarity meaure and corpus based measure. Mispelled words are often not found in the corpus, so the word level semantic similarity returns 0 but they could have nice string similarity score with the correctly spelled words being compared. The maximum value returned by comparing the both measures is used instead of using integration by equal weights.

6.2 Recommendation

In future the work for recognizing the negative sense sentence would be done.The handling of opposite meaning texts requires deeper reasoning. The similarity score is very much influenced by the stemming, lemmatization and tagging methods used.The availability of better approaches in future would obviously enhance the semantic text similarity measure. Further research has to emphasize primarily the variable of window size. Additionally, a larger number of lexemes have to be included. Chunking and parsing the corpus should also improve the results.

Appendices

Appendix 1

Code for Computing LCS

```
/**
* Function to compute Longest Common Subsequence algorithm. That is, given
* two strings A and B,this program will find longest common sequence.
* @param String, String
* @return String length of LCS
* @author Laxman
**/

public static String LCSAlgorithm(String a, String b) {
// These are "constants" which indicate a direction in the backtracking
array.
private static final int NEITHER = 0;
private static final int UP = 1;
private static final int LEFT = 2;
private static final int UP_AND_LEFT = 3;
int n = a.length();
int m = b.length();
int S[][] = new int[n + 1][m + 1];
int R[][] = new int[n + 1][m + 1];
int ii, jj;

// It is important to use <=, not <. The next two for-loops are
// initialization
for (ii = 0; ii <= n; ++ii) {
  S[ii][0] = 0;
  R[ii][0] = UP;
}
for (jj = 0; jj <= m; ++jj) {
  S[0][jj] = 0;
  R[0][jj] = LEFT;
}

// This is the main dynamic programming loop that computes the score and
// backtracking arrays.
for (ii = 1; ii <= n; ++ii) {
  for (jj = 1; jj <= m; ++jj) {

    if (a.charAt(ii - 1) == b.charAt(jj - 1)) {
      S[ii][jj] = S[ii - 1][jj - 1] + 1;
      R[ii][jj] = UP_AND_LEFT;
    }

    else {
      S[ii][jj] = S[ii - 1][jj - 1] + 0;
      R[ii][jj] = NEITHER;
    }

    if (S[ii - 1][jj] >= S[ii][jj]) {
      S[ii][jj] = S[ii - 1][jj];
      R[ii][jj] = UP;
```

```
    }

    if (S[ii][jj - 1] >= S[ii][jj]) {
      S[ii][jj] = S[ii][jj - 1];
      R[ii][jj] = LEFT;
    }
  }
}

// The length of the longest substring is S[n][m]
ii = n;
jj = m;
int pos = S[ii][jj] - 1;
char lcs[] = new char[pos + 1];

// Trace the backtracking matrix.
while (ii > 0 || jj > 0) {
  if (R[ii][jj] == UP_AND_LEFT) {
    ii--;
    jj--;
    lcs[pos--] = a.charAt(ii);
  }

  else if (R[ii][jj] == UP) {
    ii--;
  }

  else if (R[ii][jj] == LEFT) {
    jj--;
  }
}

return new String(lcs);
}
```

Code for Computing String edit distance measure

```
/**
* Function to compute LevenshteinDistance . That is, given
* two strings A and B,it computes the string edit distance
* @param String, String
* @return integer: String edit distance metric
* @author Laxman
**/

public static int computeLevenshteinDistance(String str1,
  String str2) {
    int[][] distance = new int[str1.length() + 1][str2.length() + 1];

    for (int i = 0; i <= str1.length(); i++)
      distance[i][0] = i;
    for (int j = 1; j <= str2.length(); j++)
      distance[0][j] = j;

    for (int i = 1; i <= str1.length(); i++)
      for (int j = 1; j <= str2.length(); j++)
        distance[i][j] = minimum(distance[i - 1][j] + 1,
            distance[i][j - 1] + 1,
```

```
            distance[i - 1][j - 1]
                + ((str1.charAt(i - 1) == str2.charAt(j - 1)) ? 0 : 1));

    return distance[str1.length()][str2.length()];
}
```

Code for Computing PMI summation for word similarity

```
/**
* Function to calculate the pmi summation for SemanticWordSimilarity matrix
* and returns the matrix
* @param Hashtable for word coroccurenece count, word frequency count
* @return double array: Word similarity matrix
* @author Laxman Manandhar
* */
public static Double[][] PmiSummationForWordSimilarity(
  Hashtable<String, Double> BetaWordsHashtable[],
  Hashtable<String, Integer> WordCoOccurenceCountHash[],
  Hashtable<String, Integer> WordFrequencyCountHash[],
  Hashtable<String, Integer> MainWordCountHash, String AllWords[],
  int lengthP, int lengthR, long totalWordsRepeat) {
  double pmi1 = 0.0, pmi2 = 0.0;
  Double[][] WordSimilarityMatrix = new Double[lengthP][lengthR];
  for (int i = 0; i < lengthP; i++) {
   int k = 0;
   for (int j = lengthP; j < AllWords.length; j++) {
    if (BetaWordsHashtable[i].size() == 0
        || BetaWordsHashtable[j].size() == 0) {
      WordSimilarityMatrix[i][k] = 0.0;
      k++;
      for (String word : BetaWordsHashtable[i].keySet()) {
        if (WordCoOccurenceCountHash[j].containsKey(word)) {
          double value = (WordCoOccurenceCountHash[j].get(word) *
totalWordsRepeat)
              / (MainWordCountHash.get(AllWords[j]) *
WordFrequencyCountHash[j]
                  .get(word));
          pmi1 = pmi1 + Math.pow(log2(value), 2);
        }
      }
      double betaSummation1 = pmi1 / BetaWordsHashtable[i].size();
      for (String word : BetaWordsHashtable[j].keySet()) {
        if (WordCoOccurenceCountHash[i].containsKey(word)) {
          double value = (WordCoOccurenceCountHash[i].get(word) *
totalWordsRepeat)
              / (MainWordCountHash.get(AllWords[i]) *
WordFrequencyCountHash[i]
                  .get(word));
          pmi2 = pmi2 + Math.pow(log2(value), 2);
        }
      }
      double betaSummation2 = pmi2 / BetaWordsHashtable[j].size();
      WordSimilarityMatrix[i][k] = (betaSummation1 + betaSummation2) / 25;
      if (WordSimilarityMatrix[i][k] > 1) {
        WordSimilarityMatrix[i][k] = 0.95;
      }
      k++;
      pmi1 = 0.0;
      pmi2 = 0.0;
```

```
    }
  }
}
return WordSimilarityMatrix;
}
```

Code for computing word to word similarity based on Second order PMI

```
/**
* Function to compute word to word similarity
* and returns the matrix
* @param
* @return
* @author Laxman
* */
public static void main(String args[]) {
Scanner input;
String P = "देश विकास लागी शिक्षा महत्वपूर्ण भूमिका";
String R = "मुलुकको उन्नतिको लागी आर्थिक विकास आवश्यक कुरा";
String PWords = "देश शिक्षा महत्वपूर्ण भूमिका";
String RWords = "मुलुक उन्नति आर्थिक आवश्यक कुरा";
String[] PsplitBeforeProcess = P.split(" ");
String[] RsplitBeforeProcess = R.split(" ");
// same words count
int countSameWords = 0;
for (int c = 0; c < PsplitBeforeProcess.length; c++) {
  for (int d = 0; d < RsplitBeforeProcess.length; d++) {
    if (PsplitBeforeProcess[c].equalsIgnoreCase(RsplitBeforeProcess[d])) {
      countSameWords++;
    }
  }
}
int m = PsplitBeforeProcess.length;
int n = RsplitBeforeProcess.length;
String[] PWordsArray = PWords.split(" ");
String[] RWordsArray = RWords.split(" ");
String[] AllWords = (PWords + " " + RWords).split(" ");
int x = 0, y = 0, z = 0, i = 0, j = 0, k = 0, l = 0;
String[][] afterMainWordArray = new String[AllWords.length][5];
String[][] beforeMainWordArray = new String[AllWords.length][5];
Double[][] StringSimilarityMatrix = new
Double[PWordsArray.length][RWordsArray.length];
boolean[] last = new boolean[AllWords.length];
int[] reachedIndexForLastCase = new int[AllWords.length];
int[] limitIndexForLastCase = new int[AllWords.length];
String previousLine[] = null;
int count;
// Hash tables initializations
Hashtable<String, Integer> WordCoOccurenceCountHash[] = new
Hashtable[AllWords.length];
Hashtable<String, Integer> WordFrequencyCountHash[] = new
Hashtable[AllWords.length];
Hashtable<String, Double> PmiValuesHashCoOccuredWords[] = new
Hashtable[AllWords.length];
Hashtable<String, Double> Max5PmiHashForMainWord[] = new
Hashtable[AllWords.length];
```

```
Hashtable<String, Integer> BetaCoOccuredWordsForMainWord[] = new
Hashtable[AllWords.length];
Hashtable<String, Double> ReturnedPMIvalues[] = new
Hashtable[AllWords.length];
Hashtable<String, Integer> MainWordCountFrequcyHash = new Hashtable<String,
Integer>();
for (int c = 0; c < WordCoOccurenceCountHash.length; c++) {
  WordCoOccurenceCountHash[c] = new Hashtable<String, Integer>();
  WordFrequencyCountHash[c] = new Hashtable<String, Integer>();
  PmiValuesHashCoOccuredWords[c] = new Hashtable<String, Double>();
  Max5PmiHashForMainWord[c] = new Hashtable<String, Double>();
  BetaCoOccuredWordsForMainWord[c] = new Hashtable<String, Integer>();
  MainWordCountFrequcyHash.put(AllWords[c], 1);
}

// String Similarity matrix computation
LCS stringSimilarityObj = new LCS();
StringSimilarityMatrix = stringSimilarityObj.computeStringSimilarityMatrix(
    PWordsArray, RWordsArray);
// Reading from processed corpus file for computing the wordSimilarity
try {
  input = new Scanner(new File("processedCorpus.txt"));
  while (input.hasNextLine()) {
    i = 0;
    String strLine = input.nextLine();
    String strWords[] = strLine.split("[\\s]+");
    for (int c = 0; c < AllWords.length; c++) {
      if (last[c] == true) {
        for (k = 0; k < limitIndexForLastCase[c]; k++) {
          afterMainWordArray[c][reachedIndexForLastCase[c]] = strWords[k];
          reachedIndexForLastCase[c]++;
        }
        checkWordInHashTable(c, afterMainWordArray,
            WordCoOccurenceCountHash[c], AllWords[c]);
        x = 0;
        last[c] = false;
        y = 0;
      }
    }
// looping through the main words and then scanned lines from corpus
// and incrementing the counts for each encountered words with their
// neigbour words three cases to handle: last range case, first range case
// normal range case.
for (int c = 0; c < AllWords.length; c++) {
  for (i = 0; i < strWords.length; i++) {
    String[] strWordsTaggedsplit = strWords[i].split("\\/");
    if (strWordsTaggedsplit[0].equalsIgnoreCase(AllWords[c])) {
      count = MainWordCountFrequcyHash.get(AllWords[c]);
      count++;
      MainWordCountFrequcyHash.put(AllWords[c], count);
      x = strWords.length - 5;
      if (i >= x) {
        last[c] = true;
        reachedIndexForLastCase[c] = 0;
        for (k = i + 1; k < strWords.length; k++) {
          afterMainWordArray[c][reachedIndexForLastCase[c]] = strWords[k];
          reachedIndexForLastCase[c]++;
        }
        limitIndexForLastCase[c] = 5 - reachedIndexForLastCase[c];
```

```
        int startIndexForBeforeWords = i - 5;
        for (k = 0; k < 5; k++) {
          beforeMainWordArray[c][k] = strWords[startIndexForBeforeWords];
          startIndexForBeforeWords++;
        }
        checkWordInHashTable(c, beforeMainWordArray,
            WordCoOccurenceCountHash[c], AllWords[c]);
      }
      else if (i < 6) {
        for (k = 0; k < i; k++) {
          beforeMainWordArray[c][k] = strWords[k];
        }
        int startFromLastpreviousLine = previousLine.length - 1;
        while (k <= 4) {
          beforeMainWordArray[c][k] =
previousLine[startFromLastpreviousLine];
          startFromLastpreviousLine--;
          k++;
        }
        for (k = 0; k < 5; k++) {
          afterMainWordArray[c][k] = strWords[i + 1];
          i++;
        }
        checkWordInHashTable(c, beforeMainWordArray,
            WordCoOccurenceCountHash[c], AllWords[c]);
        checkWordInHashTable(c, afterMainWordArray,
            WordCoOccurenceCountHash[c], AllWords[c]);
      } else {
        l = i;
        for (k = 0; k < 5; k++) {
          beforeMainWordArray[c][k] = strWords[l - 5];
          l++;
        }
        for (k = 0; k < 5; k++) {
          afterMainWordArray[c][k] = strWords[i + 1];
          i++;
        }
        checkWordInHashTable(c, beforeMainWordArray,
            WordCoOccurenceCountHash[c], AllWords[c]);
        checkWordInHashTable(c, afterMainWordArray,
            WordCoOccurenceCountHash[c], AllWords[c]);

      }

    }
    previousLine = strWords;
  }
 } // for each words end
} // while end
count = 0;
  findBetaCoOccuredWord(WordCoOccurenceCountHash, WordFrequencyCountHash,
      MainWordCountFrequcyHash, BetaCoOccuredWordsForMainWord, AllWords,
      totalWordsNonRepeat);
  countFrequency(WordFrequencyCountHash, BetaCoOccuredWordsForMainWord);
  ReturnedPMIvalues = findPMIForHashTableWords(
      BetaCoOccuredWordsForMainWord, WordFrequencyCountHash,
      MainWordCountFrequcyHash, PmiValuesHashCoOccuredWords, AllWords,
      totalWordsRepeat);
  Double[][] semanticSimilarityMatrix = PmiSummationForWordSimilarity(
```

```
        ReturnedPMIvalues, BetaCoOccuredWordsForMainWord,
        WordFrequencyCountHash, MainWordCountFrequcyHash, AllWords,
        PWordsArray.length, RWordsArray.length, totalWordsRepeat);
    for (int a = 0; a < PWordsArray.length; a++) {
      for (int b = 0; b < RWordsArray.length; b++) {
        String StringSimilarityval = new DecimalFormat("##.###")
            .format(StringSimilarityMatrix[a][b]);
      }
    }
    for (int a = 0; a < PWordsArray.length; a++) {
      for (int b = 0; b < RWordsArray.length; b++) {
        String semanticSimilarityval = new DecimalFormat("##.###")
            .format(semanticSimilarityMatrix[a][b]);
        System.out.printf("SemanticSimilarity[%s][%s]=%s\n", PWordsArray[a],
            RWordsArray[b], semanticSimilarityval);
      }
    }
    integrateMatricesFinalSimilarity(StringSimilarityMatrix,
        semanticSimilarityMatrix, PWordsArray.length, RWordsArray.length, m,
        n, countSameWords, PWordsArray, RWordsArray);

    } catch (FileNotFoundException e) {
    e.printStackTrace();
   }

}
```

References

[1] E. Agirre, D. Cer, M. Diab, and A. G. Agirre, “Semeval-2012 task 6: A pilot on semantic textual similarity,” First Joint Conference on Lexical and Computational Semantics (*SEM), vol. 1, p. 385–393, June 2012.

[2] M. Mohler and R. Mihalcea, “Text-to-text semantic similarity for automatic short answer gradings,” in 12th Conference of the European Chapter of the ACL, (Athens Greece,), p. 567–575, ACL, 2009.

[3] J. O’Shea, Z. Bandar, K. Crockett, and D. McLean, “A comparative study of two short text semantic similarity measures,” Springer-Verlag Berlin Heidelberg N.T. Nguyen (Eds.): KES-AMSTA , LNAI 4953, p. 172–181, 2008.

[4] G. Erkan and D. Radev, “Graph-based lexical centrality as salience in text summarization,” Journal of Artificial Intelligence Research 22, vol. 41, no. 5, p. 457–479, 2004.

[5] V. Gupta and G. S. Lehal, “A survey of text summarization extractive techniques,” Journal of Emerging Technologies in Web Intelligence, vol. 2, August 2010.

[6] I. Donevska, “Measuring word similarity using natural text descriptions,” Mater’s Theses, Indiana University , Purdue University Fort Wayne, August 2011.

[7] C. Leacock and M. Chodorow, “Combining local context and wordnet sense similarity for word sense identification. in wordnet, an electronic lexical database.,” 1998.

[8] M. Lesk, “Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone,” in In Proceedings of the SIGDOC Conference., 1986.

[9] Z. Wu and M. Palmer, “Verb semantics and lexical selection.,” in In Proceedings of the Annual Meeting of the Association for Computational Linguistics., 1994.

[10] P. Resnik, “Using information content to evaluate semantic similarity,” in In Proceedings of the 14th International Joint Conference on Artificial Intelligence., 1995.

[11] E. Park, D. Ra, and M. Jang, “Techniques for improving web retrieval effectiveness,” Information Processing and Management, vol. 41, no. 5, p. 1207–1223, 2005.

[12] J. Jiang and D. Conrath, “Semantic similarity based on corpus statistics and lexical taxonomy,” in In Proceedings of the International Conference on Research in Computational Linguistics., 1997.

[13] R. Mihalcea, C. Corley, and C. Strapparava, “Corpus-based and knowledge-based measures of text semantic similarity,” American Association for Artificial Intelligence, 2006.
[14] Y. Li, D. McLean, Z. A. Bandar, J. D. O’Shea, and K. Crockett, “Sentence similarity based on semantic nets and corpus statistics,” IEEE Transaction on Knowledge and Data Engineering, vol. 18, no. 8, 2006.
[15] E. Terra and C. L. A. Clarke, “Frequency estimates for statistical word similarity measures,” in Proceedings of HLT-NAACL 2003 Main Papers, pp. 165–172 Edmonton, May-June 2003.
[16] C. Meadow, B. Boyce, and D. Kraft, Text Information Retrieval Systems(2nd Edition). Academic Press, 2000.
[17] E. Ristad and P. N. Yianilos, “Learning string edit distance,” IEEE Transactions on Pattern Analysis and Machinge Intelligence, vol. 20, May 1998.
[18] L. Allison and T. I. Dix, “A bit-string longest-common-subsequence algorithm,” in Inf. Proc. Lett., vol. 23, p. 305–310, 1986.
[19] A.Islam and D.Inkpen, “Semantic text similarity using corpus-based word similarity and string similarity,” ACM Trans. Knowl.Discov.Data, vol. 10, p. 25, Nov. 2008.
[20] T. K. Landauer, P. W. Foltz, and D. Lahamj, “Introduction to latent semantic analysis,” in Discourse Processes, vol. 25, pp. 259–284, 1998.
[21] C. Burgess, K. Livesay, and K. Lundj, “Explorations in context space: Words, sentences,” in Discourse Processes, vol. 25, pp. 211–257, 1998.
[22] P. Turney, “Mining the web for synonyms: Pmi-ir versus lsa on toefl.,” in In Proceedings of the Twelfth European Conference on Machine Learning, ECML, 2001.
[23] A.Islam and D.Inkpen, “Second order co-occurrence pmi for determining the semantic similarity of words,” in In Proceedings of the International Conference on Language Resources and Evaluation. (Genoa, Italy), p. 1033–1038, 2006.
[24] K. Parsons, A.McCormac, M. Butavicius, S. Dennis, and L.Ferguson, “The use of a context-based information retrievaltechnique,” Command, Control, Communications and Intelligence Division Defence Science and Technology Organisation, july 2009.
[25] J. O’Shea, Z. Bandar, K. Crockett, and D. McLean, “A semantic similarity approach to paraphrase detection,” 2008. of the Twelfth European Conference on Machine Learning, ECML, 2001.