Performance of Supervised Machine Algorithm in Textual Similarity in Data Mining

Bollikonda Lalitha[1] Konda Adilakshmi[2]
[1,2]Assistant Professor
[1,2]Department of Computer Science & Engineering
[1,2]Gokaraju Rangaraju Institute of Engineering & Technology

Abstract— *The semantic similarity of the sentences is calculated using information from a structured lexical database and from corpus statistics. The use of lexical database enables the method to model human common sense knowledge and the incorporation of corpus statistics allows the method to be adaptable to different domains. Semantic Textual Similarity (STS) is a measure of the degree of semantic equivalence between two pieces of text. Semantic Textual Similarity is a measure of how close the meanings of two text sequences are. Given two input text segments, the Semantic Textual Similarity of the two sentences is determined by automatically deriving a score that indicates the given sentences similarity at semantic level, thus going behind the simple lexical matching methods traditionally used for this task.*

Keywords— Semantic similarity, Textual similarity, Lexical Database

I. INTRODUCTION

Traditionally, techniques for detecting similarity between long texts (documents) have centered on analyzing shared words. Such methods are usually effective when dealing with long texts because similar long texts will usually contain a degree of co-occurring words. However, in short texts, word co-occurrence may be rare or even null. This is mainly due to the inherent flexibility of natural language enabling people to express similar meanings using quite different sentences in terms of structure and word content. Since such surface information in short texts is very limited, this problem poses a difficult computational challenge. The focus of this project is on computing the similarity between very short texts, primarily of sentence length. Although sentence similarity is increasingly in demand from a variety of applications, as described earlier, the adaptation of available measures to computing sentence similarity has three major drawbacks. First, a sentence is represented in a very high-dimensional space with hundreds or thousands of dimensions. This results in a very sparse sentence vector which is consequently computationally inefficient. High dimensionality and high sparsity can also lead to unacceptable performance in similarity computation. Second, some methods require the user's intensive involvement to manually preprocess sentence information. Third, once the similarity method is designed for an application domain, it cannot be adapted easily to other domains. This lack of adaptability does not correspond to human language usage as sentence meaning may change, to varying extents, from domain to domain. To address these drawbacks, this project aims to develop a method that can be used generally in applications requiring sentence similarity computation. An effective method is expected to be dynamic in only focusing on the sentences of concern, fully automatic without requiring the users' manual work, and readily adaptable across the range of potential application domains.

In Web page retrieval, sentence similarity has proven to be one of the best techniques for improving retrieval effectiveness, where titles are used to represent documents in the named page finding task. In image retrieval from the Web, the use of short text surrounding the images can achieve a higher retrieval precision than the use of the whole document in which the image is embedded. In text mining, sentence similarity is used as a criterion to discover unseen knowledge from textual databases. In addition, the incorporation of short-text similarity is beneficial to applications such as text summarization, text categorization, and machine translation. These exemplar applications show that the computing of sentence similarity has become a generic component for the research community involved in text-related knowledge representation and discovery.

II. LITERATURE SURVEY

In general, there is extensive literature on measuring the similarity between documents or long texts, but there are very few methods relating to the measurement of similarity between very short texts or sentences. This section reviews some related work in order to explore the strengths and limitations of previous methods, and to identify the particular difficulties in computing sentence similarity. Related works can roughly be classified into three major categories: word co-occurrence methods, corpus-based methods, descriptive features based methods.

A. Word co- occurrence methods

The word co-occurrence methods are often known as the "bag of words" method. They are commonly used in Information Retrieval (IR) systems. The systems have a precompiled word list with n words. The value of n is generally in the thousands or hundreds of thousands in order to include all meaningful words in a natural language. Each document is represented using these words as a vector in n-dimensional space. A query is also considered as a document. The relevant documents are then retrieved based on the similarity between the query vector and the document vector. This technique relies on the assumption that more similar documents share more of the same words. If this technique were applied to sentence similarity, it would have three obvious drawbacks:

[1] The sentence representation is not very efficient. The vector dimension n is very large compared to the number of words in a sentence, thus the resulting vectors would have many null components.

[2] The word set in Information Retrieval systems usually excludes function words such as the, of, an, etc. Function words are not very helpful for computing

document similarity, but cannot be ignored for sentence similarity because they carry structural information, which is useful in interpreting sentence meaning. If function words were included, the value for n would be greater still.

[3] Sentences with similar meaning do not necessarily share many words.

One extension of word co-occurrence methods is the use of a lexical dictionary to compute the similarity of a pair of words taken from the two sentences that are being compared (where one word is taken from each sentence to form a pair). Sentence similarity is simply obtained by aggregating similarity values of all word pairs. Another extension of word co-occurrence techniques leads to the pattern matching methods which are commonly used in conversational agents and text mining. Pattern matching differs from pure word co-occurrence methods by incorporating local structural information about words in the predicated sentences. A meaning is conveyed in a limited set of patterns, where each is represented using a regular expression (generally consisting of parts of words and various wildcards) to provide generalization. Similarity is calculated using a simple pattern matching algorithm. This technique requires a complete pattern set for each meaning in order to avoid ambiguity and mismatches. Manual compilation is an immensely arduous and tedious task. At present, it is not possible to prove that a pattern set is complete and, thus, there is no automatic method for compiling such a pattern set. Finally, once the pattern sets are defined, the algorithm is unable to cope with unplanned novel utterances from human users.

B. Corpus Based Methods

One recent active field of research that contributes to sentence similarity computation is the methods based on statistical information of words in a huge corpus. Well known methods in corpus-based similarity are the latent semantic analysis (LSA) and the Hyperspace Analogues to Language (HAL) model. In LSA, a set of representative words needs to be identified from a large number of contexts (each described by a corpus). A word by context matrix is formed based on the presence of words in contexts. The matrix is decomposed by singular value decomposition (SVD) into the product of three other matrices, including the diagonal matrix of singular values. The diagonal singular matrix is truncated by deleting small singular values. In this way, the dimensionality is reduced. The original word by context matrix is then reconstructed from the reduced dimensional space. Through the process of decomposition and reconstruction, LSA acquires word knowledge that spreads in contexts. When LSA is used to compute sentence similarity, a vector for each sentence is formed in the reduced dimension space; similarity is then measured by computing the similarity of these two vectors. Because of the computational limit of singular value decomposition, the dimension size of the word by context matrix is limited to several hundred. As the input sentences may be from an unconstrained domain (and thus not represented in the contexts), some important words from the input sentences may not be included in the LSA dimension space. Second, the dimension is fixed and, so, the vector is fixed and is thus likely to be a very sparse representation of a short text such as a sentence. Like other methods, LSA ignores any syntactic information from the two sentences being compared and is understood to be more appropriate for larger texts than the sentences dealt with in this work.

Another important work in corpus-based methods is Hyperspace Analogues to Language (HAL). Indeed, HAL is closely related to LSA and they both capture the meaning of a word or text using lexical co-occurrence information. Unlike LSA, which builds an information matrix of words by text units of paragraphs or documents, HAL builds a word-by-word matrix based on word co-occurrences within a moving window of a predefined width. The window (typically with a width of 10 words) moves over the entire text of the corpus. An N _ N matrix is formed for a given vocabulary of N words. Each entry of the matrix records the (weighted) word co-occurrences within the window moving through the entire corpus. The meaning of a word is then represented as a 2N-dimensional vector by combining the corresponding row and column in the matrix. 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 experimental results showed that HAL was not as promising as LSA in the computation of similarity for short texts.

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.

C. Descriptive Features Based Methods

The third category of related work is the descriptive features-based methods. The feature vector method tries to represent a sentence using a set of predefined features. Basically, a word in a sentence is represented using semantic features, for example, nouns may have features such as HUMAN (with value of human or nonhuman), SOFTNESS (soft or hard), and POINTNESS (pointed or rounded). A variation of feature vector methods is the introduction of primary features and composite features. Primary features are those primitive features that compare single items from each text unit. Composite features are the combination of pairs of primitive features. A text is then represented in a vector consisting of values of primary features and composite features. Similarity between two texts is obtained through a trained classifier. The difficulties for this method lie in the definition of effective features and in automatically obtaining values for features from a sentence. The preparation of a training vector set could be an impractical, tedious, and time-consuming task. Moreover, features can be well-defined for concrete concepts; however, it still is problematic to define features for abstract concepts.

Overall, the aforementioned methods compute similarity according to the co-occurring words in the texts and ignore syntactic information. They work well for long texts because long texts have adequate information (i.e., they have a sufficient number of co-occurring words) for manipulation by a computational method. This project addresses the limitations of these existing methods by forming the word vector dynamically based entirely on the words in the compared sentences. The dimension of the

vector is not fixed but varies with the sentence pair and, so, it is far more computationally efficient than existing methods. The proposed algorithm also considers word order, which is a further aspect of primary syntactic information.

III. Designing & Implementation

A. Proposed System

The proposed method derives text similarity from semantic and syntactic information contained in the compared texts. A text is considered to be a sequence of words each of which carries useful information. The words, along with their combination structure, make a text convey a specific meaning. Texts considered in this project are assumed to be of sentence length.

Unlike existing methods that use a fixed set of vocabulary, the proposed method dynamically forms a joint word set only using all the distinct words in the pair of sentences. For each sentence, a raw semantic vector is derived with the assistance of a lexical database. A word order vector is formed for each sentence, again using information from the lexical database. Since each word in a sentence contributes differently to the meaning of the whole sentence, the significance of a word is weighted by using information content derived from a corpus. By combining the raw semantic vector with information content from the corpus, a semantic vector is obtained for each of the two sentences. Semantic similarity is computed based on the two semantic vectors. An order similarity is calculated using the two order vectors. Finally, the sentence similarity is derived by combining semantic similarity and order similarity.

This project takes a data set as an input and provides the similarity between all the sentences in the data set and writes the output to a file. The preprocessing steps involved in computing the semantic textual similarity are:

1) Tokenizing: Tokenization is the process of breaking a stream of text up into words, phrases, symbols, or other meaningful elements called tokens.
2) POS Tagging: In corpus linguistics, part-of-speech tagging (POS tagging or POST), also called grammatical tagging or word-category disambiguation, is the process of marking up a word in a text (corpus) as corresponding to a particular part of speech, based on both its definition and its context—i.e., its relationship with adjacent and related words in a phrase, sentence, or paragraph.
3) Stemming: Stemming is the process of reducing inflected(or derived) words to their stem, base or root form - generally a written word form. Thus the words of the sentences are represented by their stems in calculating word similarity rather than the original words.

The following sections present a detailed description of each of the above steps. Since semantic similarity between words is used both in deriving sentence semantic similarity and word order similarity, description of the method for measuring word semantic similarity is as follows:

B. Semantic Similarity between Wordds

A number of semantic similarity methods have been developed and are proven to be useful in some specific applications of computational intelligence. Generally, these methods can be categorized into two groups: edge counting-based (or dictionary/thesaurus-based) methods and information theory-based (or corpus-based) methods. In this project we use a word similarity measure which provides the best correlation to human values.

There are semantic knowledge bases readily available, some examples are WordNet, Spatial Date Transfer Standard, and Gene Ontology. The knowledge bases tend to consist of a hierarchical structure modeling human common sense knowledge for a particular domain or, general English Language usage. The hierarchical structure of the knowledge base is important in determining the semantic distance between words.

In this project, WordNet lexical database is used which has general English language words organized in hierarchical structure.

Given two words, w1 and w2, the need is to find the semantic similarity s(w1,w2). This can be done by analysis of the lexical knowledge base (in this project, WordNet is used) as follows: Words are organized into synonym sets (synsets) in the knowledge base, with semantics and relation pointers to other synsets. Therefore, the first class in the hierarchical semantic network that subsumes the compared words can be found. One direct method for similarity calculation is to find the minimum length of path connecting the two words.

C. Word similarity between sentences

Let us consider a pair of sentences, T1 and T2, that contain exactly the same words in the same order with the exception of two words from T1 which occur in the reverse order in T2. For example:

– T1: A quick brown dog jumps over the lazy fox.
– T2: A quick brown fox jumps over the lazy dog.

Since these two sentences contain the same words, any methods based on ”bag of words” will give a decision that T1 and T2 are exactly the same. However, it is clear for a human interpreter that T1 and T2 are only similar to some extent. The dissimilarity between T1 and T2 is the result of the different word order. Therefore, a computational method for sentence similarity should take into account the impact of word order. For the example pair of sentences T1 and T2, the joint word set is:

T = {A quick brown dog jumps over the lazy fox}

Assign a unique index number for each word in T1 and T2. The index number is simply the order number in which the word appears in the sentence. For example, the index number is 4 for dog and 6 for over in T1. In computing the word order similarity, a word order vector, r, is formed for T1 and T2, respectively, based on the joint word set T. Taking T1 as an example, for each word wi in T, find the same or the most similar word in T1 as follows:

1) If the same word is present in T1, fill the entry for this word in r1 with the corresponding index number from T1. Otherwise, try to find the most similar word $\tilde{w}_i$ in T1.
2) If the similarity between wi and $\tilde{w}_i$ is greater than a preset threshold, the entry of wi in r1 is filled with the index number of $\tilde{w}_i$ in T1.

3. If the above two searches fail, the entry of wi in r1 is 0. Having applied the procedure on the previous page, the word order vectors for T1 and T2 are r1 and r2, respectively. For the example sentence pair, we have:

r1 = {1 2 3 4 5 6 7 8 9}
r2 = {1 2 3 9 5 6 7 8 4}

Thus, a word order vector is the basic structural information carried by a sentence. The task of dealing with word order is then to measure how similar the word order in two sentences is. The proposed measure for measuring the word order similarity of two sentences as:

$$S_r = 1 - \frac{\| \mathbf{r}_1 - \mathbf{r}_2 \|}{\| \mathbf{r}_1 + \mathbf{r}_2 \|}.$$

That is, word order similarity is determined by the normalized difference of word order. The following analysis will demonstrate that Sr is an efficient metric for indicating word order similarity. To simplify the analysis, consider only a single word order difference, as in sentences T1 and T2. Given two sentences, T1 and T2, where both sentences contain exactly the same words and the only difference is that a pair of words in T1 appears in the reverse order in T2. The word order vectors are:

$$\mathbf{r}_1 = \{a_1 \dots a_j \dots a_{j+k} \dots a_m\} \text{ for } T_1,$$
$$\mathbf{r}_2 = \{b_1 \dots b_j \dots b_{j+k} \dots b_m\} \text{ for } T_2.$$

a_j and a_{j+k} are the entries for the considered word pair in T_1, b_j and b_{j+k} are the corresponding entries for the word pair in T_2, and k is the number of words from w_j to w_{j+k}. From the above assumptions, we have:

$$a_i = b_i = i \text{ for } i = 1, 2, \dots, m \text{ except } i \neq j, j+k,$$
$$a_j = b_{j+k} = j,$$
$$a_{j+k} = b_j = j+k,$$
$$\|\mathbf{r}_1\| = \|\mathbf{r}_2\| \equiv \|\mathbf{r}\|,$$

then:

$$S_r = 1 - \frac{k}{\sqrt{2\,\| \mathbf{r} \|^2 - k^2}}. \quad (9)$$

The same formula for a sentence pair with only one different word at the kth entry can also be derived. For the more general case with a more significant difference in word order or a larger number of different words, the analytical form of the proposed metric becomes more complicated (which we do not intend to present in this paper). The above analysis shows that Sr is a suitable indication of word order information. Sr equals 1 if there is no word order difference. Sr is greater than or equal to 0 if word order difference is present. Since Sr is a function of k, it can reflect the word order difference and the compactness of a word pair. The following features of the proposed word order metric can also be observed:

1) Sr can reflect the words shared by two sentences.
2) Sr can reflect the order of a pair of the same words in two sentences. It only indicates the word order, while it is invariant regardless of the location of the word pair in an individual sentence.
3) Sr is sensitive to the distance between the two words of the word pair. Its value decreases as the distance increase.
4) For the same number of different words or the same number of word pairs in a different order, Sr is proportional to the sentence length (number of words); its value increases as the sentence length increases. This coincides with intuitive knowledge, that is, two sentences would share more of the same words for a certain number of different words or different word order if the sentence length is longer. Therefore, the proposed metric is a good one for indicating the word order in terms of word sequence and location in a sentence.

D. Overall sentence similarity

Semantic similarity represents the lexical similarity. On the other hand, word order similarity provides information about the relationship between words: which words appear in the sentence and which words come before or after other words. Both semantic and syntactic information (in terms of word order) play a role in conveying the meaning of sentences. Thus, the overall sentence similarity is defined as a combination of semantic similarity and word order similarity:

$$S(T_1, T_2) = \delta S_s + (1-\delta) S_r$$
$$= \delta \frac{\mathbf{s}_1 \cdot \mathbf{s}_2}{\|\mathbf{s}_1\| \cdot \|\mathbf{s}_2\|} + (1-\delta) \frac{\|\mathbf{r}_1 - \mathbf{r}_2\|}{\|\mathbf{r}_1 + \mathbf{r}_2\|}, \quad (10)$$

where $\delta \leq 1$ decides the relative contributions of semantic and word order information to the overall similarity computation. Since syntax plays a subordinate role for semantic processing of text, δ should be a value greater than 0.5, i.e., $\delta \in (0.5, 1]$.

E. Pearson's Correlation

In an intrinsic evaluation, the system output is usually compared with human judgments by computing Pearson correlation.

The correlation coefficient is typically denoted by r and measures the strength of linear dependence between two similarity score vectors, i.e. how well the system reflects human judgments. The value of r is in the interval [-1,1] where r = 0 can be interpreted as not similar at all, and r = -1 and r = 1 are perfect linear relationships (completely similar). The relationship between $\vec{x}$ and $\vec{y}$ can be visualized in a scatter plot: For perfect linear relationships, all data points are on a regression line that has a positive (r = 1) or negative (r = -1) slope. Pearson's r between two score vectors $\vec{x}$ and $\vec{y}$ is computed as follows, with $n = |\vec{x}| = |\vec{y}|$:

$$r = \frac{n \sum x_i y_i - \sum x_i \sum y_i}{\sqrt{n \sum x_i^2 - (\sum x_i)^2} \sqrt{n \sum y_i^2 - (\sum y_i)^2}}$$

F. Statistics from the Corpus

The probability of a word w in the corpus is computed simply as the relative frequency:

$$\hat{p}(w) = \frac{n+1}{N+1}, \quad (11)$$

where N is the total number of words in the corpus, n is the frequency of the word w in the corpus (increased by 1 to avoid presenting an undefined value to the logarithm). Information content of w in the corpus is defined as:

$$I(w) = -\frac{\log p(w)}{\log(N+1)} = 1 - \frac{\log(n+1)}{\log(N+1)}, \qquad (12)$$
$$\text{so } I \in [0, 1].$$

The next step is to convert the high dimensional FV into low dimensional Reduced Feature Vector (RFV) to reduce the storage and execution time complexities. So, a counter loop is invoked to remove the duplicate or redundant entries of a feature. Therefore, only one instance of each different feature occurred is stored in RFV. It highly reduces the FV dimension and increases the efficiency of the system with good performance. The complete sentence level processing is shown in the Fig. 3.1.

Fig. 3.1: Steps involved in computing Semantic Textual Similarity

The proposed model is divided into the two phases: Text Learning Phase (TLP) and Text Classification Phase (TCP). TLP performs the learning function on a given set of text documents. It performs the steps of first stage; i. e., Text Document Training Processor (TDTP) and then the steps of second stage; i. e., Feature Category Similarity Analyzer (FCSA). The TDTP is used to process the text document by converting it into its small and constituent parts or chunks by using NLP concepts at the Sentence, Document and Integrated Corpora Levels. Then, it searches and stores the desired, important and non-redundant concepts by removing stop words, invalid words and extra words. In the next step, it performs word stemming and feature reduction. The result of sentence level preparation is low dimensional Raw Semantic Vector (RSV). Each RSV of a document is sent for document level preparation, so that Integrated Raw Semantic Vector (IRSV) is obtained. To obtain IRFV, all the RFVs are integrated into one. Now, Reduced Feature Frequency Calculator (RFFC) is used to calculate the total frequency of each different word occurred in the document. Finally, all redundant entries of each exclusive word are removed and all the words with their associated frequencies are stored in decreasing order of their frequencies. At the integrated corpora level, the low dimension Integrated Corpora Feature Vector (ICFV) is resulted. In such a way, feature vectors at each level are made low dimensional by processing and updating step by step. Such functionality helps a lot to search the appropriate concepts with reduced vector length to improve system performance and accuracy. FCSA performs similarity measure based analysis for feature pattern (TD – ICFV) using the enriched fuzzy logic base. The membership degree of each feature is associated with it. Therefore, an analysis is performed between every feature of a text document and class. SVMC is used for the categorization of the text documents. It uses the concept of hyper planes to identify the suitable category. Furthermore, SVMC accuracy is checked by providing some new and unknown text documents to be classified into the respective Category Group (CG). This task is performed in TCP. The proposed Similarity Based Concept Mining Model (SCMM) is shown in Fig.3.2. In the next sections, this model and its components are discussed in detail.

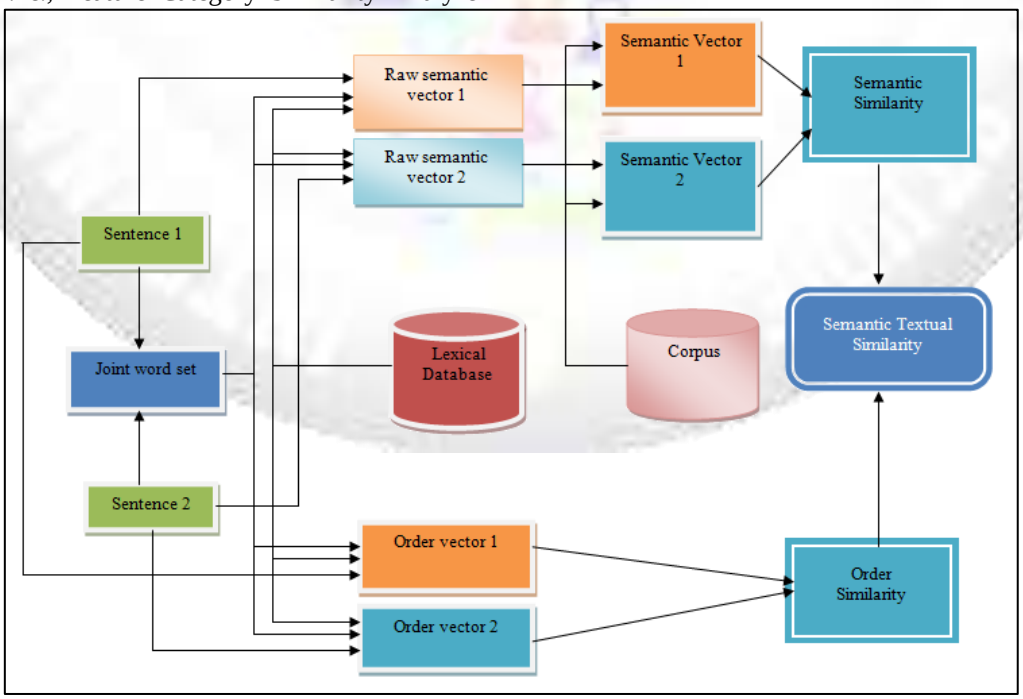

Fig. 3.2: Similarity Based Concept Mining Model

IV. EXPERIMENTAL RESULTS

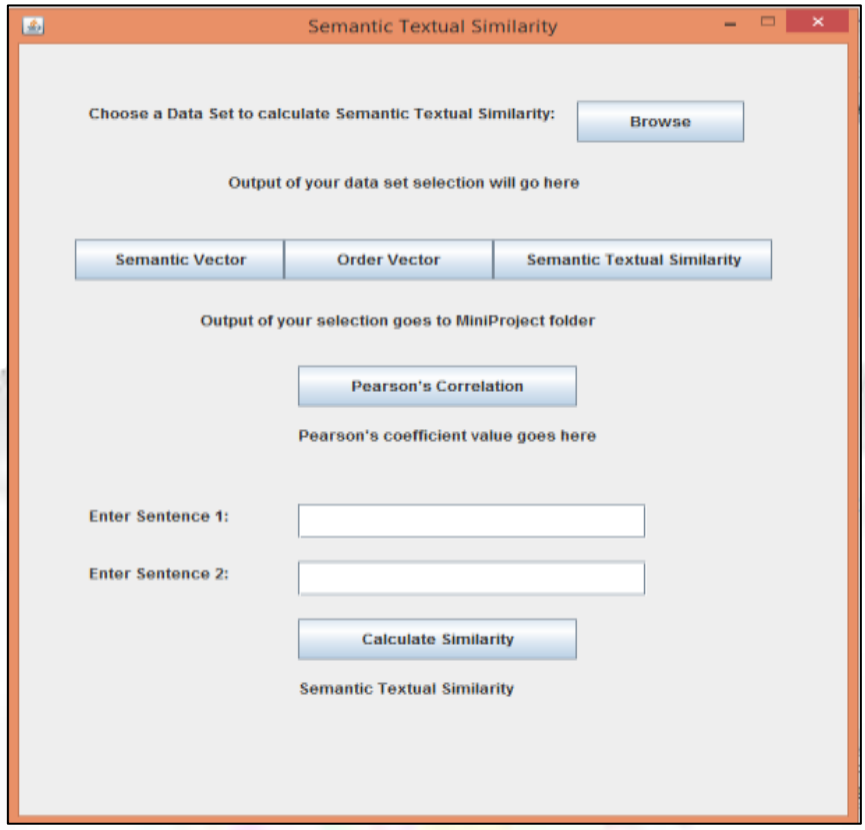

Fig. 4.1: Semantic similarity checking

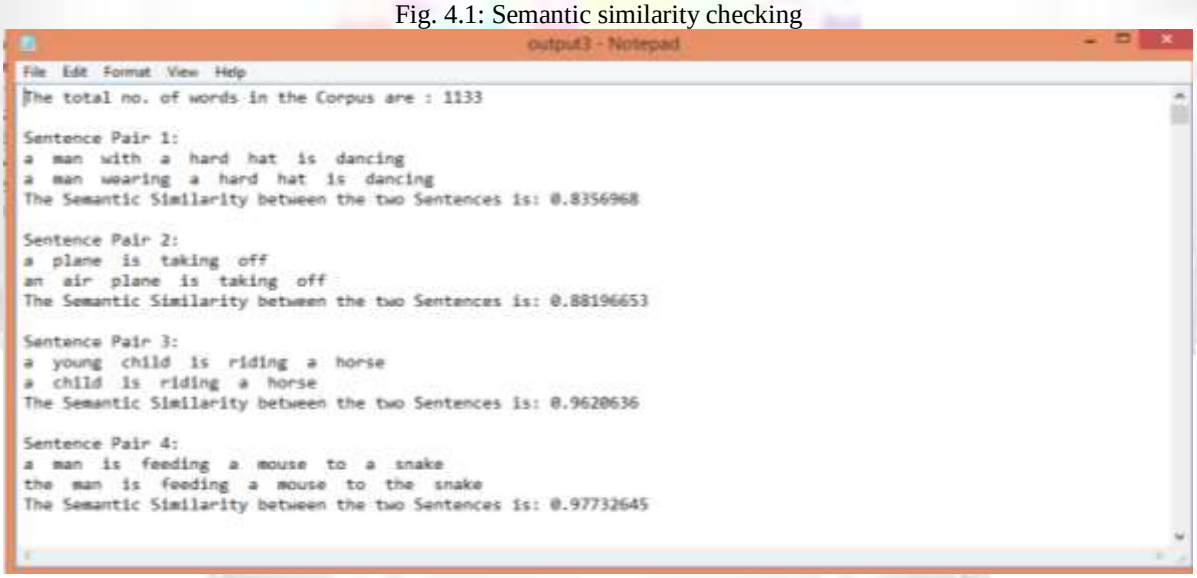

Fig. 4.2: Word counting

From the above fig 4.1 and 4.2, the data set was taken from The Semantic Evaluation (SemEval) 2017. The data set contains the data as sentence pairs. The Datasets English – English, Arabic – English, Arabic – Arabic, Spanish – Spanish, Turkish – English has 250 sentence pairs. And Turkish – English have 500 sentence pairs. Each pair was assigned similarity scores in the range [0, 5] by multiple human annotators (0: no similarity, 5: identicality) and the average of the annotations was taken as the final similarity score. Each dataset is divided at 80% of data is taken as training data and 20% of data is taken as testing pairs. All data sets were translated into English using Google Translator.

A. Applications:

1) Natural Language Processing tasks

1) text categorization
2) text summarization

2) Internet-related applications

1) web-page retrieval
2) image retrieval
3) tweets search

3) *Artificial Intelligence*
4) *Text Mining*
5) *Machine learning*
6) *Machine translation*
7) *Dialogue systems*
8) *Short answer grading*
9) *Spam filtering*
10) *Recommendations/Suggestions in e-commerce websites*

V. CONCLUSION & FUTURE SCOPE

This system presented an efficient method for measuring the semantic similarity between sentences or very short texts, based on semantic and word order information. First, semantic similarity is derived from a lexical knowledge base and a corpus. Second, the proposed method considers the impact of word order on sentence meaning. The derived word order similarity measures the number of different words as well as the number of word pairs in a different order. The overall sentence similarity is then defined as a combination of semantic similarity and word order similarity.

The lexical knowledge base models common human knowledge about words in a natural language; this knowledge is usually stable across a wide range of language application areas. A corpus reflects the actual usage of language and words. Thus, this project not only captures common human knowledge, but it is also able to adapt to an application area using a corpus specific to that application.

A. Future Scope

Increase the efficient scoring by making the similarity range lie between 0 to 5. An improvement to the algorithm to disambiguate word sense using the surrounding words to give a little contextual information.

REFERENCES

[1] Budanitsky and G. Hirst, "Semantic Distance in WordNet: An Experimental, Application-Oriented Evaluation of Five Measures," Proc. Workshop WordNet and Other Lexical Resources, Second Meeting of the North Am. Chapter of the Assoc. for Computational Linguistics, 2001.
[2] E.K. Park, D.Y. Ra, and M.G. Jang, "Techniques for Improving Web Retrieval Effectiveness," Information Processing and Management, vol. 41, no. 5, pp. 1207-1223, 2005.
[3] R. Rada, H. Mili, E. Bichnell, and M. Blettner, "Development and Application of a Metric on Semantic Nets," IEEE Trans. System, Man, and Cybernetics, vol. 9, no. 1, pp. 17-30, 1989.
[4] P. Resnik, "Using Information Content to Evaluate Semantic Similarity in a Taxonomy," Proc. 14th Int'l Joint Conf. AI, 1995.
[5] F.J. Ribadas, M. Vilares, and J. Vilares, "Semantic Similarity between Sentences through Approximate Tree Matching," Lecture Notes in Computer Science, vol. 3523, pp. 638-646, 2005.
[6] G. Salton, Automatic Text Processing: the Transformation, Analysis, and Retrieval of Information by Computer. Wokingham, Mass.: Addison-Wesley, 1989.
[7] Hatzivassiloglou, J. Klavans, and E. Eskin, "Detecting Text Similarity over Short Passages: Exploring Linguistic Feature Combinations via Machine Learning," Proc. Joint SIGDAT Conf. Empirical Methods in NLP and Very Large Corpora, 1999.
[8] D. Jurafsky and J.H. Martin, Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. Prentice Hall, 2000.
[9] Y. Ko, J. Park, and J. Seo, "Improving Text Categorization Using the Importance of Sentences," Information Processing and Management, vol. 40, pp. 65-79, 2004.