



Jimma University

Jimma Institute of Technology

School of Graduate Studies

Department of Information Technology

Interpretable Semantic Textual Similarity

By

Abdo Ababor

**A Thesis Submitted to the Faculty of Computing of Jimma University in
Partial Fulfillment of the Requirements for Degree of Master of Science
in Information Technology**

Principal Advisor: Mr. Debela Tesfaye (Ass Prof)

Co-Advisor: Mr. Tefari Kibebew (MSc)

November, 2017

Jimma, Ethiopia

Jimma University
Jimma Institute of Technology
School of Graduate Studies
Department of Information Technology

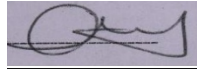
Interpretable Semantic Textual Similarity

By

Abdo Ababor Abafogi

Approved By Board of Examiner:

Mr. Debela Tesfaye



Advisor

Signature

Date

External Examiner

Signature

Date

Internal Examiner

Signature

Date

Faculty Dean

Signature

Date

Declaration

I, Abdo Ababor declare that this thesis is my own work and this work has not been submitted before for a degree at any other institution.

Declared by:

Name: Abdo Ababor

Signature: _____

Date: _____

This research has been submitted for Examination with my approval as university advisor.

Advisor name: Debela Tefaye (Ass Prof)

Signature: 

Date: _____

This research has been submitted for Examination with my approval as university co-advisor.

Co-advisor name: Tefari Kibebew (MSc)

Signature: _____

Date: _____

Jimma, Ethiopia

November, 2017

ACKNOWLEDGEMENTS

First of all, I would like to thank the almighty Allah for giving me the strength to complete this work. I want to express my deepest gratitude to Mr. Debela Tesfaye for supervising this thesis. Furthermore, I appreciate the useful advises and guidance provided by Mr. Tefari Kebebew. In addition, I want to thank all my friends for their valuable information and support. My special thanks to my family for their moral support and encouragement during my study. I thank especially, my spouse Fatuma Nasir and all my family.

Dedication

This work is dedicated to my daughter, Yasmin Abdo

Table of contents

Contents	page number
ACKNOWLEDGEMENTS	iv
List of Tables	x
List of figures	xi
List of Abbreviations and Acronym	xii
Abstract	xiii
CHAPTER ONE	1
INTRODUCTION	1
1.1. Background	1
1.2. Statement of problem	2
1.3. Objective	3
1.3.1. General Objective	3
1.3.2. Specific objective.....	3
1.4. Significance of the Study	3
1.5. Scope of the Study.....	4
1.6. Methodology	4
1.7. Organization of the Thesis	6
CHAPTER TWO	8
LITERATURE OVERVIEW	8
2.1. Introduction.....	8
2.2. Possible Approaches to semantic textual similarity.....	9
2.2.1. String-Based Similarity	9
2.2.2. Corpus-Based Similarity.....	11
2.2.2.1. <i>Vector space model</i>	11

2.2.2.2. <i>Latent Semantic Analysis</i>	12
2.2.2.3. <i>Generalized Latent Semantic Analysis</i>	14
2.2.2.4. <i>Explicit Semantic Analysis</i>	14
2.2.2.5. <i>Hyperspace Analogue to Language</i>	15
2.2.2.6. <i>Latent Dirichlet Allocation</i>	15
2.2.2.7. <i>Word2Vec</i>	16
2.2.2.8. <i>Point wise Mutual Information</i>	17
2.2.2.9. <i>Extracting DIStributionally</i>	17
2.2.3. Knowledge-Based Similarity.....	17
2.3.3.1. <i>Path-based Measures</i>	18
2.3.3.2. <i>Information Content-based Measure</i>	19
2.3.3.3. <i>Feature-based Measure</i>	22
2.4. Measuring Semantic textual similarity	22
2.4.3. Preprocessing.....	22
2.4.4. Alignment	25
2.4.4.1. Identifying Chunks	25
2.4.4.2. Aligning Chunks.....	25
2.4.4.3. Scoring Aligned Chunk	27
2.4.4.4. Labeling Aligned Chunks.....	28
2.5. Related work	29
2.5.1. Rule base approach	29
2.5.2. Machine learning approach with linguistic and rule blended	31
CHAPTER THREE	37
INTERPRETABLE SEMANTIC TEXTUAL SIMILARITY (ISTS).....	37
3.1. Introduction	37

3.2.	Architecture of ISTS	37
3.2.1.	Preprocessing.....	39
3.2.2.	Chunks	39
3.2.3.	Post processing	40
3.3.	Similarity calculator	41
3.3.2.	Wikipedia Corpus	42
3.3.2.1.	<i>Corpus Selection</i>	42
3.3.2.2.	<i>LSA Word Similarity</i>	43
3.3.2.3.	<i>Word Co-Occurrence Generation</i>	43
3.3.2.4.	<i>SVD Transformation</i>	45
3.3.3.1.	<i>WordNet based similarity</i>	46
3.3.4.	<i>Feature Extraction for Chunk-to-Chunk Similarity Measures</i>	49
3.4.	Alignment.....	55
3.5.	Type prediction	57
3.6.	Score Classification.....	60
3.7.	Evaluation of ISTS	61
3.8.	Tools.....	61
	CHAPTER FOUR.....	62
	EXPERIMENTATION AND DISCUSSION.....	62
4.1.	Introduction.....	62
4.2.	Datasets	62
4.3.	Limitation and Challenges	70
4.4.	Evaluation	63
4.5.	ISTS results	68

CHAPTER FIVE	71
CONCLUSION AND RECOMMENDATIONS	71
5.1. Introduction.....	71
5.2. Conclusion	71
5.3. Recommendations.....	72
References.....	73

List of Tables

<i>Table – 3.1 hypernym score and type</i>	48
<i>Table – 3.2 example of chunk alignment with score and type</i>	55
<i>Table – 4.1 POSim comparison with human annotated, dice – coefficient and cosine</i>	65
<i>Table – 4.2 images dataset result</i>	69
<i>Table – 4.3 headline dataset result</i>	69

List of figures

<i>Figure – 3.1 Interpretable semantic textual similarity achitecture.....</i>	<i>38</i>
<i>Figure – 4.1 sample output of images dataset</i>	<i>67</i>
<i>Figure – 4.2 sample output of Headline dataset.....</i>	<i>68</i>

List of Abbreviations and Acronym

ADJP	Adjective Phrase
ADVP	Adverb Phrase
EQUI	Equivalent
ESA	Explicit Semantic Analysis
GLSA	Generalized Latent Semantic Analysis
HAL	Hyperspace Analogue to Language
IDF	Invers Document Frequency
ISTS	interpretable Semantic Textual Similarity
NER	Named Entity Relationship
NLP	Natural Language Processor
NP	Noun Phrase
OPPO	Opposite
POS	Part of Speech
PP	Preposition Phrase
REL	Related
SIMI	Similar
SPBC	Stanford Parser Based Chunk
SPE1	Specific
SPE2	General
S1	Sentence One
S2	Sentence Two
STS	Semantic Textual Similarity
SVD	Singular Value decomposition
VP	Verb Phrase

Abstract

This thesis focuses on the problem of interpretable semantic textual similarity in English language. The system takes pair of sentence then it identifies the chunks in each sentence according to standard gold chunks, align corresponding chunk, assign degree of similarity score as well as predict reason of similarity/dissimilarity for each aligned chunks. To do this computation distributional hypothesis approach blend with knowledge based was selected. Latent semantic analysis (LSA) is a purely statistical technique, which leverages word co-occurrence information from a large unlabeled large corpus of text relies on the distributional hypothesis that the words occurring in similar contexts tend to have similar meanings. To do so LSA word similarity computed from a statistical analysis of preprocessed Wikipedia corpus as well as it boosted by WordNet and string similarity.

Furthermore semantic similarity measures between corresponding chunks are introduced in the theoretical part. We selected and implemented 10 similarity measures. In the experimentation part we proposes five chunk similarity measures inspired by state-of-the-art measures described in the chapter three. The evaluation is conducted two results (Run1 and Run2) on two data sets (Images and Headlines).

We can be concluded that the performance of the system obtained was promising and gives a best result on Run1 which depends on *POSim*.

CHAPTER ONE

INTRODUCTION

1.1. Background

This chapter gives readers a general insight about the background of the study, the problems that motivated the study. The chapter also gives highlight of the method and approaches followed in come up with solutions to the problems. The objective, significance, scope and limitation of the study is also included in this chapter.

Similarity is a complex concept which has been widely discussed in the linguistic, philosophical, and information theory communities [1]. Standard semantic relations such as synonymy, paraphrase, redundancy, and entailment all result from judgments of likeness whereas antonym, contradiction, and inconsistency derive from judgments of difference [2]. Semantic related tasks have been a noticed trend in NLP community. One needs to come up with a consistent computational model to assess this type of relation. When a word level semantic relation requires exploration, there are many potential types of relations that can be considered: hierarchical (e.g. IS-A or hypernym-hyponym, part-whole, etc.), associative (e.g. cause-effect), equivalence (synonymy), etc. [3].

Semantic similarity can be broadly construed as being assessed between any two texts of any size. Depending on the granularity of the texts, such as: word-to-word similarity, phrase-to-phrase similarity, sentence-to-sentence similarity, paragraph-to-paragraph similarity, or document-to-document similarity are measured in different ways. Mixed combinations are also possible such as assessing the similarity of a word to a sentence or a sentence to a paragraph. For instance, in text summarization it might be useful to assess how well a sentence summarizes an entire paragraph [4]. Particularly, the task Semantic Textual Similarity (STS) has captured a huge attention in the NLP community despite being recently introduced since SemEval 2012 [5] and ongoing up to date.

Semantic Textual Similarity (STS) measures the degree of semantic equivalence between two sentences. STS captures the notion that some texts are more similar than others, measuring their degree of semantic equivalence.

Textual similarity can range from complete un-relatedness to exact semantic equivalence, and a graded similarity intuitively captures the notion of intermediate shades of similarity, as pairs of text may differ

from some minor nuanced aspects of meaning, to relatively important semantic differences, to sharing only some details, or to simply being related to the same topic.

Although STS systems measure the degree of semantic equivalence in terms of a score which is useful in many tasks, they stop short of explaining why the texts are similar, related, or unrelated. They do not indicate what kind of semantic relations exist among the constituents (words or chunks) of the target texts. Interpretable STS (ISTS) adds an explanatory layer. Given the input pairs of sentences participants need first to identify the chunks in each sentence, and then, align chunks across the two sentences, indicating the relation and similarity score of each alignment. ISTS for each sentence pair, participating systems had to identify the chunks in each sentence, align corresponding chunks and assign a similarity/relatedness score and type of the alignment for each alignment. The alignment types were semantically equivalent, opposite, similar, related and unrelated in meaning [6].

1.2. Motivation

The core motivation behind interpretable semantic textual similarity is that for a local language there are no work has been done on interpretation of text similarity. We tried to do the thesis for a local language on this title. Unfortunately it needs many things to do which we cannot cop and need linguistic profession. First and for most to propose new algorithm or even to use existing algorithm it need train and test datasets that annotated by linguistic. On other hand it need human annotated train and test dataset which have pair sentence, split of sentences into chunks, alignment of similar chunks together, assign similarity score for aligned pair chunks and assign type for aligned chunks (why it is aligned). This annotation needs the effort of exerted linguistic, as well as lexical database toolkit like WordNet is not available for local language. For these reason we do on foreign (English) language.

1.3. Statement of problem

Methods to assess the semantic similarity of larger texts, in particular sentences, have been proposed for paraphrasing and entailment semantic relation identification at sentence level [7], [8], [9], and [10]. Semantic similarity toolkit SEMILAR has been proposed that work at different levels of text granularity (word-to-word, sentence-to-sentence, paragraph-to-paragraph, document-to-document, or a combination) [11] regardless of interpretation at which point similarity is there and why?

STS systems measure the degree of semantic equivalence in terms of a score which is useful in many tasks, did not explain why the texts are similar, related, or unrelated. They do not indicate what kind of

semantic relations exist among the constituents (words or chunks) of the target texts. Similarities and difference may be clearly identified as a sentence but, on what words or chunks their similarity or difference recognized, how much the degrees.

However a few researchers tried to interpret semantic textual similarity among chunks but limited one to one alignment only. That is the main problem seen so far.

Finding explicit relations among constituents in the paired texts would enable a meaningful interpretation of the similarity scores that attract me as topic of this thesis.

1.4. Objective

1.4.1. General Objective

The general objective is to compute whether two sentences are related or unrelated, by supplementing the similarity score with an explanatory layer.

1.4.2. Specific objective

- ✓ Align the chunks across both sentences
- ✓ For each chunks alignment, provide the corresponding similarity score.
- ✓ For each score classify the type of relation.
- ✓ Determining the similarity between chunks.

1.5. Significance of the Study

Significance of scientific study is multi-dimensional; Academic and personal. Semantic text similarity is more directly applicable in a number of NLP tasks such as Machine Translation and evaluation, Summarization, Machine Reading, Deep Question Answering etc. on other hand interpretable semantic text similarity shows clearly the interpretation how much the text is similar or not similar at chunk level. For instance in deep Question Answering by cross checking expressed answers to conceptual questions are similar to ideal answers and displays points at which two answers are related or unrelated is an interesting part of ISTS. Also, the system contributes to future researchers in the area of Semantic text similarity especially in developing interpretable semantic textual similarity. Generally the research outcome contributes benefit to individuals and future researchers.

1.6. Scope of the Study

Interpretable semantic text similarity methods work at sentence-to-sentence levels of text but not paragraph-to-paragraph and/or document-to-document similarity. The text (sentences) divided into chunks and aligned in concept, gives score and reason of similarity. Our focus is determining the similarity between chunks of two sentences. Due to limitation of time we focused only chunk level alignments for short statements whenever possible.

1.7. Methodology

The combination of distributional approach and knowledge base is the method of this research work. Many literatures related to interpretable STS (ISTS) are reviewed to understand different possible way of measure and interpret STS. Semantic textual similarity model is based on a combination of latent semantic analysis (LSA) and knowledge from WordNet. It manages the different inputs of the system, texts in English and with varying length, and uses the interpretable semantic textual similarity model to compute the similarity between the given pieces of text interprets the text. The ISTS system composes several modules design to handle the computation of a similarity score among pieces of text of different lengths.

LSA word similarity relies on the distributional hypothesis that the words occurring in similar contexts tend to have similar meanings. Thus, evidence for word similarity can be computed from a statistical analysis of a large text corpus. LSA does not rely on any human-organized knowledge; rather, it “learns” from corpus. Statistical word similarity measures have limitations. Related words can have similarity scores only as high as their context overlap. Also, word similarity is typically low for synonyms having many word senses since information about different senses are mixed together. To reduce the limitation of statistical word similarity additional information is needed; which is solved by using WordNet.

1.6.1 Literature review

To accomplish the objectives of this research mentioned above several articles and literatures review. Materials concerning semantic textual similarity and its interpretation were also reviewed. Since there are several approaches used in computing semantic textual similarity, literature review was also carried out on approaches used for interpretation of semantic textual similarity.

1.6.2 Corpus selection

For this research corpus selection and processing is done on Wikicorpus. Wikicorpus is a corpus contains 600 million words which collected from Wikipedia and has been automatically enriched with linguistic

information. Additionally knowledge based (WordNet) is another source of information for computing interpretable semantic textual similarity.

1.6.3 Preprocessing

On other hand many preprocessing perform to reduce inflectional forms of words to a common base form and increase performance of the system. This can be done by many preprocessing techniques such as: Tokenization, stop-word removal, lemmatization, part of speech tagging, parsing, named entity (name of location, organization, date, money, person, time and percent) recognizing are performed by Stanford CoreNLP toolkit [50].

1.6.4 Scoring pair chunks

Scoring each pair chunks are computed by combining different features taken from lexical, semantic and syntactic features are computed for the texts using a variety of resources and supplied to a classifier, which then assigns weights to the features. It takes a set of input and predicts the scores for a set of input.

1.6.5 Alignment

Monolingual alignment is the task of discovering and aligning similar semantic units in a pair of sentences expressed in a natural language. Such alignments provide valuable information regarding how and to what extent the two sentences are related [62]. A chunk is a textual unit of adjacent words grouped on the basis of linguistic properties which display the relations between their internal words [72].

At chunking process two sentences split into gold standard chunks. Once sentences are chunked, and similarity score is calculated between chunks of all possible chunk-pairs next task is aligning pair chunks, based on similarity score. Where, chunk pairs with a high similarity are aligned first, followed by pairs with lower similarity [51].

1.6.6 Labeling aligned chunks

Labeling aligned chunks is another important part of the thesis. ISTS focus on explaining two sentences that may be related/unrelated, by supplementing the similarity score with an explanatory layer. For each alignment of chunks c_1 and c_2 , the alignment type is determined according to the following rules: If the similarity score between c_1 and c_2 is 5, the type is equivalency. If all word senses of c_1 matched the word senses in c_2 , the type is general for c_1 and specific for c_2 similarly. If both c_1 and c_2 contain a single word sense, and are directly connected by an antonym relation then the type is opposite. Also, all unaligned chunks are labeled with not aligned.

The rule based approach is used for chunk alignments and scoring [12]. A number of conditions are defined for a chunk pair that might be checked before applying a rule. First identify the chunks in each sentence separately, regardless of the corresponding sentence in the pair. Align chunks in order, using the interface from the clearest and strongest correspondences to the most unclear or weakest ones. For each alignment, provide a similarity/relatedness from 5 (maximum similarity/relatedness) to 0 (no relation at all) score. A given pair of sentences, participating systems will need to align the chunks in sentence1 to the chunks in sentence2, adding a score for the similarity/relatedness between each pair of chunks and describing what kind of relation exists between them. The implementation of the system coding, testing and maintenance will be done. Finally the performance of the developed system is measured by examining the correlation between human judgment and machine calculations.

1.6.7. Experimentation and evaluation

The experimentation for evaluating the effectiveness of the system was done by using selected test dataset. Once the necessary data has been used for training, another dataset were used for testing the performance of the system. 750 headlines were prepared for training purpose and 756 image pair sentence selected for training set. Moreover a total 700 pair sentence that is 350 from image and 350 from headline prepared for testing the system. On the other hand, since the correctness of the chunking has direct effect on the performance of the system. So training and testing of chunking, aligning, scoring and labeling done on the selected datasets. The correctness of Alignment, score and type techniques are selected for effectiveness measure as it is the most popular and most widely used measure of interpretable semantic textual similarity.

1.7. Organization of the Thesis

The thesis is divided into five chapters and their organization is described as follows. This chapter, Chapter One, is the introductory part of the study. It contains background of the study, statement of the problem, objective and significance of the study. It also discusses the methodology used and the evaluation techniques. Chapter two discusses review of literature which comprises two parts, conceptual review and review of related works. The conceptual review involves review of basics of semantic textual similarity including its three well-known method of measuring textual similarity such as: string based for lexical similarity, corpus based and knowledge based for semantic similarity. It also discusses an overview of interpretable semantic textual similarity and basic topics like preprocessing, chunk alignment, scoring and type of semantic textual similarity interpretation. As well as the approaches in semantic textual similarity

are discussed in detail. Review of related works tries to discuss related works done in the area of interpretable semantic textual similarity.

Chapter three, Interpretable Semantic Textual Similarity, method and technique focuses on designing and developing the model of the system. It describes in detail about document pre-processing, alignment tool to be used, architecture of the system along with description of its components, system performance evaluation model. The fourth chapter focuses experimentation and discussion of test dataset as well as system evaluation. The chapter also discusses analysis of results obtained from experiments. The last chapter, Conclusion and Recommendation, concludes what has been done and achieved in the research and forwards direction for future work.

CHAPTER TWO

LITERATURE OVERVIEW

2.1. Introduction

Text similarity measures play an increasingly important role in text related applications in many tasks such as information retrieval, text classification, document clustering, topic detection, topic tracking, questions generation, question answering, essay scoring, short answer scoring, machine translation, text summarization and others. Finding similarity between words is a fundamental part of text similarity which is then used as a primary stage for sentence, paragraph and document similarities. Words can be similar in two ways lexically and semantically. Words are similar lexically if they have a similar character sequence. Words are similar semantically if they have the same thing, are opposite of each other, used in the same way, used in the same context and one is a type of another [8]. Different methods may operate at different levels of representation of the input expressions; for example, they may treat the input expressions simply as surface strings (i.e. string sequences and character composition), they may operate on syntactic (i.e. an arrangement of words in a sentence) or semantic (i.e. meaning of word based on co-occurrence or Knowledge-Based) representations of the input expressions, or on combining information from different levels [21].

In semantic textual similarity (STS), systems rate the degree of semantic equivalence between two text snippets. In literature there are many papers published on Paraphrase and Textual Entailment. Paraphrase defined as to repeat something written or spoken using different words. According to this definition paraphrase introduces two important aspects: same meaning and different words. Paraphrasing can be seen as bidirectional Textual Entailment (TE) and methods from the two areas are often similar [21]. Paraphrase and TE recognizers judge whether or not two given language expressions constitute paraphrases or a correct TE pair. STS is related to both TE and Paraphrasing, but differs in a number of ways and it applicable to many NLP tasks. STS is different from TE because it is bidirectional graded equivalence between the pair of textual snippets. In the case of TE the equivalence is directional, e.g. a car is a vehicle, but a vehicle is not necessarily a car. STS also differs from both TE and Paraphrasing, both tasks have been defined to date in the literature rather than being a binary yes/no decision (e.g. a vehicle is not a car), STS to be a graded similarity notion for a vehicle and a car. So a vehicle and a car are more similar than a wave and a car [20].

2.2. Possible Approaches to semantic textual similarity

In order to measure STS word similarity has a great role. Words similarity can be measured in two ways lexically and semantically. Lexical similarity is introduced through different String-Based algorithms, Semantic similarity are introduced through Corpus-Based and Knowledge-Based algorithms. String-Based measures operate on string sequences and character composition. A string metric is a metric that measures similarity or dissimilarity between two text strings for approximate string matching or comparison. Corpus-Based similarity is a semantic similarity measure that determines the similarity between words based on information gained from large corpora. Knowledge-Based similarity is a semantic similarity measure that determines the degree of similarity between words using information derived from semantic networks [21]. Some of the most popular for each type presented briefly.

2.2.1. String-Based Similarity

String similarity measures work on string sequences and character composition. String similarity measures similarity or dissimilarity (distance) between two text strings for approximate string matching or comparison. A basic way to compare two texts take texts representations at character level and compare them without any semantic processing solely based on their string sequences. Among string based similarity measures some of them will be presented as follow.

One possibility is to compare the texts' *longest common substring* [22] algorithm considers the similarity between two strings is based on the length of contiguous chain of characters that exist in both strings. Thereby, the length l of the longest contiguous character sequence longest common substring shared between the two texts t_1 and t_2 is compared with the text length:

$$sim(t_1, t_2) = 1 - \frac{l(t_1) + l(t_2) - 2 \cdot l(lcs(t_1, t_2))}{l(t_1) + l(t_2)}$$

However, this measure has limitations, e.g. in cases of word insertions/deletions or typographical errors which break the common substring.

To overcome the limitation of longest common substring the *longest common subsequence* measure proposed [23] by dropping the contiguity requirement. Similarity is then computed the longest common substring function now refers to the non-contiguous shared subsequence. Greedy String Tiling [24] is a method which further allows dealing with shared substrings which do not appear in the same order in both texts. The measure determines the set of shared contiguous substrings; each substring is a match of

maximal length. Similarity is then computed as the number of marked characters those participating in any shared substring divided by the text length.

Furthermore, the popular *Jaro distance* [8, 9] another string based measure which based on the number and order of the common characters between two strings. It takes into account typical spelling deviations and mainly used in the area of record linkage. It is especially suitable for short strings such as person or place names:

$$sim_{jaro}(s1, s2) = \frac{1}{3} \left(\frac{m}{|s1|} + \frac{m}{|t2|} + \frac{m - s}{m} \right), \text{ if } m > 0, \text{ and } 0 \text{ else}$$

Where m refers to the number of matching characters between $s1$ and $s2$. Matching characters which do not appear in the same order in both texts are called transpositions, and t is defined as a half of the number of transpositions. The *Jaro-Winkler distance* [28] is an extension of Jaro distance; it uses a prefix scale which gives more favorable ratings to strings that match from the beginning for a set prefix length. It is a variation of the original metric which assigns a higher similarity score to texts with a matching prefix, i.e. texts which match from the beginning rather than any position within the string sequence:

$$sim_{winkler}(s1, s2) = sim_{jaro}(s1, s2) + l.p. (1 - sim_{jaro}(s1, s2))$$

Where $0 \leq l \leq 4$ the number of characters in a common prefix for $s1$ and $s2$ p is a scaling factor for assigning higher weights to a *longer common prefix*, and is originally set to 0 to 1 for no similarity and exact match respectively[28].

Moreover *Edit distance* is another way of computing the dissimilarity between two strings. Conventionally, the distance is computed for a set of characters with three kinds of operations like substitution, insertion, and deletion. At this point, the distance between two string $s1$ and $s2$ is the minimum number of edit operations that transform $s1$ into $s2$. The following measures are often referred to as edit-distance metrics. The *Levenshtein distance* [29] is a simple metric that assigns uniform costs to all edit operations insertion, deletion, and substitution. The Monge Elkan distance is an edit-distance metric that uses an affine gap model. The intuition behind this model is that particular sequences of alignments and misalignments between character sequences are more likely to occur than others. *Damerau-Levenshtein* defines distance between two strings by counting the minimum number of operations needed to transform one string into the other, where an operation is defined as an insertion, deletion, or substitution of a single character, or a transposition of two adjacent characters [30],[31].

Another well-known string similarity measure is N-gram. N-gram is a sub-sequence of n items from a given sequence of text. Distance is computed by dividing the number of similar n-grams by maximal number of n-grams. N-gram models Text similarity measures based on n-gram text representations exist for words and characters. Character n-gram profiles [32] as implemented by [33], discard all characters (case insensitive) which are not in the alphabet. All n-grams on character level are then generated and weighted by a $tf*idf$ scheme. While in the original implementation only $n = 3$ is used, other values for n may also be considered. Finally, the feature vectors of both string sequences are compared by computing the cosine between them. A method for comparing texts by means of word n-grams has been proposed [34]. Two sets of n-grams are generated for both texts, and may then be compared using the *Jaccard coefficient*.

2.2.2. Corpus-Based Similarity

Corpus-Based similarity is a semantic similarity measure that determines the similarity between words based on information gained from large corpora. A corpus is a large collection of written or spoken texts that is used for language research. Semantic relatedness is based on co-occurrence statistics, typically over a large corpus. In order to produce a reliable word co-occurrence statistics, a very large and balanced text corpus is required [35].

R. Mihalcea et al, [36] Proposed to use a single word similarity measure at a time out of a rich set of measures in combination with a bidirectional aggregation strategy. He proposed aggregation strategy computes a directional similarity score from a text t_1 to a second text t_2 and vice-versa, whereas for each word a counterpart in the other text is sought which maximizes the pairwise similarity. The similarity scores are weighted by a term frequency (tf) and a term's inverse document frequency (idf) [37] on a corpus then normalized. The final text similarity score is the average of applying this strategy in both directions. Corpus based similarity measure approach has several similarity metrics like: vector space model, latent semantic analysis, generalized latent semantic analysis, explicit semantic analysis, hyperspace analogue to language, word2vec, and etc.

2.2.2.1. Vector space model

This model is an algebraic model in which the texts are represented as a vector. The information retrieval's vector space model [89] in which each text is modeled as a "bag of words" and represented using a vector. For example $doc = (doc_1, doc_2, doc_3 \dots doc_n)$ and $query = (query_1, query_2, query_3 \dots query_n)$ where, document is represented by doc . Each value in the vector is a non-zero value which weight of the term in

the document. This weight can be calculated by $tf * idf$, i.e., term frequency-inverse document frequency. The similarity between two texts is then computed as the cosine similarity of the vectors. This can be calculated by comparing the deviation of angles between the query vector and the document vector [90]. The technique is the simplest, but performs not enough when the texts to be compared share few words, for instance, when the texts use synonyms to express similar messages.

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: [91]

- a. 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.
- b. The word set in IR systems usually exclude 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.
- c. 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.

This technique is also trivially inappropriate for comparing individual words; using lexical resources, and using Latent Semantic Analysis (LSA) techniques attempt to overcome this limitation.

2.2.2.2. Latent Semantic Analysis

Mostly LSA is commonly used that is a short form of Latent Semantic Analysis [44] presents a technique for representing a text T in a semantic space based on corpus statistics. LSA is a purely statistical technique, which leverages word co-occurrence information from a large unlabeled large corpus of text.

In LSA, a set of representative words needs to be identified from a large number of contexts. A word by context matrix is formed based on the presence of words in contexts.

LSA does not rely on any human-organized knowledge; rather, it “learns” its representation by applying Singular Value Decomposition (SVD) to the words-by-documents co-occurrence matrix. LSA is essentially a dimensionality reduction technique that identifies a number of most prominent dimensions in the data, which are assumed to correspond to “latent concepts”. Meanings of words and documents are then compared in the space defined by these concepts.

The matrix is decomposed by SVD into the product of three other matrices including the diagonal matrix of singular values [91]. 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 [92].

- I. Because of the computational limit of SVD, the dimension size of the word by context matrix is limited to the 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.
- II. Secondly, 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.
- III. 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 [92].

LSA convert the term-document matrix which describes the occurrences of terms in document into three smaller matrixes like follows: $X = U\Sigma V^T$ [93]

Where, U could be preserved as the semantic space of words. Each word could be represented as a row vector in U . When measuring semantic similarity of two sentences, all word vectors appear in the sentence were summed and then averaged with the length of the sentences. Vector of the two sentences represented by $V1$ and $V2$. With $V1$ and $V2$, the similarity of the two sentences can be measured with cosine similarity. Cosine similarity defined as follows: [93]

$$\text{Cos}(V1, V2) = \frac{V1 \cdot V2}{\|V1\| \|V2\|}$$

In experiment, [93] directly used the LSA model provided by SEMILAR [29]. A word is represented as a row vector in the LSA model and the model was decomposed from the whole Wikipedia articles [93]. Also developed two weighted LSA features to further use semantic information, they are IDF weighted-LSA and Freq-weighted-LSA. IDF weighted-LSA weighted the words; one word is represented as a 200-dimension vector generated from LSA using inverse document frequency and then summed up all the weighted vectors of words which appeared in the sentence to be the representation of the sentence. The cosine distance of two sentence representations is the value of this feature. Freq-weighted-LSA used word frequency to weight the words and following the same steps mentioned in IDF weighted-LSA. LSA are very difficult to interpret, since the computed concepts cannot be readily mapped into natural concepts manipulated by humans.

2.2.2.3. Generalized Latent Semantic Analysis

This semantic analysis is a framework for computing semantically motivated term and document vectors which represent as GLSA or General LSA [46]. It extends the LSA approach by focusing on term vectors instead of the dual document-term representation. GLSA requires a measure of semantic association between terms and a method of dimensionality reduction. The GLSA approach can combine any kind of similarity measure on the space of terms with any suitable method of dimensionality reduction. The traditional term document matrix is used in the last step to provide the weights in the linear combination of term vectors.

2.2.2.4. Explicit Semantic Analysis

Gabrilovich and Markovitch introduced the concept of Explicit Semantic Analysis (ESA) in 2007 [47]. The idea underlying ESA is to represent and compare texts (from single terms to entire documents) as vectors in a high dimensional concept space. ESA was introduced as an approach to compute the semantic relatedness of terms or short phrases. ESA is a vectorial reorientation of entire document that uses a document as knowledge base. The ESA representation of a real-world document d is a vector d_{ESA} whose elements are the cosine similarities between d and all documents in a collection DI , called index collection here. The supposed rationale of the ESA retrieval model is that each document in DI functions as a semantic concept to which the original document d is compared: d_{ESA} is understood as a projection of d into the concept space spanned by DI . The semantic relatedness between two documents $d1$ and $d2$ is computed by the cosine similarity between the ESA vectors of $d1$ and $d2$.

In ESA a word is represented as a column vector in the $tf * idf$ matrix of the text corpus and a document is represented as the centroid of the vectors representing its words. In short ESA uses prior knowledge of relationship between words and concepts makes it possible to assign, readable labels to the concept that make up the vector space. Whereas in LSA the concepts are latent (they are undefined and need to be discovered) [47].

2.2.2.5. Hyperspace Analogue to Language

This measure represented as HAL creates a semantic space from word co-occurrences [45]. A word-by-word matrix is formed with each matrix element is the strength of association between the word represented by the row and the word represented by the column. The user of the algorithm then has the option to drop out low entropy columns from the matrix. As the text is analyzed, a focus word is placed at the beginning of a ten word window that records which neighboring words are counted as co-occurring. Matrix values are accumulated by weighting the co-occurrence inversely proportional to the distance from the focus word; closer neighboring words are thought to reflect more of the focus word's semantics and so are weighted higher. HAL also records word-ordering information by treating the co-occurrence differently based on whether the neighboring word appeared before or after the focus word.

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 that 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 pre-defined width [45]. The window 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 author experimental results show that HAL was not as promising as LSA in the computation of similarity for short texts [45]. 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 weakened as large number of words are added to it.

2.2.2.6. Latent Dirichlet Allocation (LDA)

LDA define the contexts across which words are distributed, and each component of the semantic representation corresponds to a particular topic. However, unlike content words they cannot be observed directly in the data. Instead they are hidden variables which arise in a generative model of the distribution of words across documents. LDA models the relationship between words and documents in terms of topics, with each document being a mixture of topics and each topic being a unigram distribution over words. Moreover LDA becomes computationally very expensive on large data sets [98].

2.2.2.7. Word2Vec

Word2vec is a two-layer neural net that processes text. Its input is a text corpus and its output is a set of vectors: feature vectors for words in that corpus. While Word2vec is not a deep neural network, it turns text into a numerical form that deep nets can understand. The purpose and usefulness of Word2vec is to group the vectors of similar words together in vector space. That is, it detects similarities mathematically [80].

Word2vec creates vectors that are distributed numerical representations of word features, features such as the context of individual words. It does so without human intervention [80]. Given enough data, usage and contexts, Word2vec make highly accurate guesses about a word's meaning based on past appearances. Those guesses can be used to establish a word's association with other words (e.g. "man" is to "boy", "woman" is to "girl"). The output of the Word2vec neural net is a vocabulary in which each item has a vector attached to it, which can be fed into a deep-learning net or simply queried to detect relationships between words. Measuring cosine similarity, no similarity is expressed as a 90 degree angle, while total similarity of 1 is a 0 degree angle, complete overlap.

Continuous Bag-of-Words Model (CBOW), as unlike standard bag-of-words model, it uses continuous distributed representation of the context. Note that the weight matrix between the input and the projection layer is shared for all word positions [99]. Mikolov et al. [99] introduced the Skip-gram model, an efficient method for learning high quality vector representations of words from large amounts of unstructured text data. Unlike most of the previously used neural network architectures for learning word vectors, training of the Skip-gram model does not involve dense matrix multiplications. This makes the training really efficient: an optimized single-machine implementation can train on more than 100 billion words in one day. The training of the Skip-gram model is to find word representations that are useful for predicting the surrounding words in a sentence or a document. More formally, given a sequence of training words w_1 ,

w_2, w_3, \dots, w_T , the objective of the Skip-gram model is to maximize the average log probability [81]. Skip-gram is similar to CBOW, but instead of predicting the current word based on the context, it tries to maximize classification of a word based on another word in the same sentence.

2.2.2.8. Point wise Mutual Information

Point wise Mutual Information - Information Retrieval (PMI-IR) [49] is a method for computing the similarity between pairs of words, it uses AltaVista's Advanced Search query syntax to calculate probabilities. The more often two words co-occur near each other on a web page, the higher is their PMI-IR similarity score.

2.2.2.9. Extracting DIStributionally

Extracting DIStributionally similar words using COoccurrences (DISCO) [50]. Distributional similarity between words assumes that words with similar meaning occur in similar context. Large text collections are statistically analyzed to get the distributional similarity. DISCO is a method that computes distributional similarity between words by using context words for counting co-occurrences. When two words are subjected for exact similarity DISCO simply retrieves their word vectors from the indexed data, and computes the similarity according to Lin measure [51]. If the most distributionally similar word is required; DISCO returns the second order word vector for the given word.

DISCO has two main similarity measures DISCO1 and DISCO2; DISCO1 computes the first order similarity between two input words based on their collocation sets. DISCO2 computes the second order similarity between two input words based on their sets of distributional similar words.

2.2.3. Knowledge-Based Similarity

Knowledge-based measures operate on lexical-semantic resources that express human knowledge about words. The knowledge based similarity includes: dictionaries, thesauri, or wordnets etc. For instance dictionaries and wordnets encode knowledge about words and their definitions, as well as the relations between words encoded in thesauri and wordnets in a machine-readable format [53]. Among a well-known knowledge based lexical semantic the most one is probably WordNet [54]. WordNet is a semantic network database which developed by University Princeton for English language. Some versions of WordNet have been developed in many languages. WordNet was designed in four type of word depends on their Parts of Speech (POS) often known as content word (noun, adjective, verb, and adverb).

In WordNet the smallest unit is synset, it represents a specific (single) meaning of a word. Synset includes the word, its synonyms and its explanation. The specific meaning of one word under one type of Part of speech 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. For example, the words nighttime, night and dark constitute a single synset that has the following gloss: the time after sunset and before sunrise while it is dark outside. Synsets are connected to one another through the explicit semantic relations. The similarity between two words can be determined using their relative positions in the knowledge base hierarchy. The two words can have high similarity score if the words are in the same WordNet synset or if one word is a hypernym of another word [55].

As mentioned above WordNet is a lexical database hierarchically organized and groups words into synsets. It provides semantic relations between synonym sets based on the grammatical rules, and it categorizes the words as nouns, verbs, adjectives and adverbs. There Morphologic functions are used in order to realize the root form of the word stored in the database. If you want to check sentence similarity first a sentence is parsed into a list of tokens and these tokens are stemmed by WordNet to find the root of the token.

In order to know similarity between two words, many similarity metrics have been proposed. Their similarity can be estimated by seeing their relative positions within the knowledge base hierarchy. Let see some of the concept of metrics one by one.

2.3.3.1. Path-based Measures

In Path-based similarity measures two concepts determined by the length of the path connecting between the concepts and its position in the hierarchy.

The Shortest Path based Measure: This measure takes length of concept *co1* and concept *co2* into considerate. The measure assumes that the similarity (*sim_path*) between *co1* and *co2* depend on how close of the *co1* and *co2* are in the hierarchy [112].

$$\text{sim_path}(co1,co2) = 2 * \text{deep}_{\text{max}} - \text{length}(co1,co2)$$

The similarity between *co1* and *co2* is the shortest path $\text{length}(co1,co2)$ from *co1* to *co2*. If $\text{length}(co1,co2)$ is 0, $\text{sim_path}(co1,co2)$ becomes $2 * \text{deep}_{\text{max}}$ (the maximum value). If $\text{length}(co1,co2)$ is $2 * \text{deep}_{\text{max}}$, $\text{sim_path}(co1,co2)$ becomes 0 (the minimum value). Thus, the values of $\text{sim_path}(co1,co2)$ are propagate between 0 and $2 * \text{deep}_{\text{max}}$.

Where, $\text{length}(co1,co2)$: the length of the shortest path from synset coi to synset coj in WordNet.

deep_{\max} : the max depth(coi) of the taxonomy

$\text{sim}(coi,coj)$: semantic similarity between concept coi and concept coj .

Wu & Palmer's Measure: it introduced a measure [63] takes the position of concepts $co1$ and $co2$ in the hierarchy into account relatively to the position of the immediate common concept $\text{lsc}(co1,co2)$. It assumes that the similarity between $co1$ and $co2$ is the path length and depth in path-based measures.

$$\text{Sim}_{wp}(co1,co2) = \frac{2 * \text{depth}(\text{lsc}(co1,co2))}{\text{length}(co1,co2) + 2 * \text{depth}(\text{lsc}(co1,co2))}$$

Where, $\text{lsc}(co1,co2)$: the lowest common subsumer of $co1$ and $co2$

The similarity between two concepts ($co1, co2$) is the function of their distance and the $\text{lsc}(co1,co2)$.

If the $\text{lsc}(co1,co2)$ is root, $\text{depth}(\text{lsc}(co1,co2))=1$, $\text{sim}_{wp}(co1,co2) > 0$; if the two concepts have the same sense, the concept $co1$, concept $co2$ and $\text{lsc}(co1,co2)$ are the same node.

$\text{sim}_{wp}(co1,co2) = 1$; otherwise $0 < \text{depth}(\text{lsc}(co1,co2)) < \text{deep}_{\max}$, $0 < \text{length}(co1,co2) < 2 * \text{deep}_{\max}$, $0 < \text{sim}_{wp}(co1,co2) < 1$. Thus, the values of $\text{sim}_{wp}(co1,co2)$ are in $(0, 1]$.

Leacock&Chodorow's Measure: according to Leacock and Chodorow the maximum depth of hierarchy into account and proposed measure as the next:

$$\text{simLC}(co1, co2) = -\log \frac{\text{length}(co1,co2)}{2 * \text{deep}_{\max}}$$

The similarity between two concepts ($co1, co2$) is the shortest path $\text{length}(co1,co2)$ from $co1$ to $co2$. When $co1$ and $co2$ have the same sense, $\text{length}(co1,co2) = 0$. To avoid $\log(0)$ 1 added to both $\text{length}(co1,co2)$ and $2 * \text{deep}_{\max}$. Thus the values of $\text{simLC}(co1,co2)$ are in $(0, \log(2 * \text{deep}_{\max} + 1)]$

2.3.3.2. Information Content-based Measure

Information Content-based Measure is another key WordNet based measure. According to this measure each concept includes much information found in WordNet. Likewise the assumptions are based on the Information content of each concept. The more common information share, the more similar the concepts are.

Resnik's Measure: Resnik proposed this similarity measure [41]. It assumes that for couple of concepts, similarity is depended on the information content that incorporates them in the hierarchy.
 $simResnik(co1, co2) = -\log p(lsc(co1, co2)) IC(lsc(co1, co2))$

The values of $simResnik(co1, co2)$ only rely on concept pairs' lsc in the hierarchy.

Lin's Measure: Lin proposed method that measure similarity [113].

$$simLin(co1, co2) = \frac{2*IC(lsc(co1, co2))}{IC(co1)+IC(cos2)}$$

Lin's Measure uses both the amount of information which is common for both concepts and all the information looked-for fully describe these terms. The $simLin$ has taken the information content of both concepts into account respectively. As $IC(lsc(co1, co2)) \leq IC(co1)$ and $IC(lsc(co1, co2)) \leq IC(co2)$, therefore the values of this measure becomes between 1 and 0.

Jiang's Measure: Jiang computed semantic distance to get semantic similarity [114].

$$disJiang(co1, co2) = (IC(co1) + IC(co2) - 2IC(lsc(co1, co2)))$$

To put it another way semantic similarity is the opposite of the semantic distance. The measure has taken the IC of both concepts into account respectively.

The First step of this method is obtaining Information Content (IC) through statistical analysis of corpora [14]. It assumes that, for a concept co in the hierarchy, let $p(co)$ be the probability of chance upon an instance of concept co . $IC(co)$ can be calculated as negative the log: $-\log p(co)$, which means that it is opposite as probability increases, IC decreases.

$IC(co) = -\log p(co)$ Probability of a concept was estimated as: $p(co) = freq(co)/N$ Where N is represent total number of nouns, and $freq(co)$ is represent the frequency of instance of concept co taking place in the taxonomy. When calculating $freq(co)$, each noun or any of its hierarchical hyponyms that occurred in the given corpora is included, which indicates that if $co1$ is-a $co2$, then $p(co1) < p(co2)$. Thus if the concept is abstract, it is higher associated probability and the lower its information content.

$$freq(co) = \sum_{w \in W(co)} count(w)$$

The measure is simple, unluckily, it relies on corpora analysis, and additionally sparse data problem is inevitable. In order to solve this problem, Nuno proposed a method hyponyms-based IC. Accordingly WordNet is used as a statistical resource to employ IC values. It regards value of a concept of the hyponyms it has [116].

For a concept co , the more hyponyms it has, it determines the more abstract it is. That is to say concepts with many hyponyms (abstract concept) express less information than concepts that are leaves. From this point of view Root node is the least informative than leaf nodes which is the most informative in the taxonomy. To put it another way for root IC is 0 and for leaf IC is 1. When you traverse from the leaf to the root node, IC will declines from 1 to 0. It indicates that the method is corpora independent. Nevertheless two concepts with the same number of hyponyms will have the same IC and all the leaves will have the same IC too.

$$IC(co) = \frac{\log(\frac{\text{hypo}(co)+1}{\text{node_max}})}{\log(\frac{1}{\text{node_max}})} = 1 - \log(\frac{\text{hypo}(co)+1}{\text{node_max}})$$

The next one is based on the assumption that taxonomical leaves denote the semantic of the most specific concepts of a domain in WordNet, a concept has less, information it expresses more leaves [117].

$$IC(co) = -\log\left(\frac{\frac{|\text{leaves}(co)|}{|\text{subsumers}(co)|} + 1}{\text{max_leaves} + 1}\right)$$

Where, let co be the set of concepts, Max_leaves represents the number of leaves matching to the root node of the hierarchy.

The fourth states that every concept is defined with sufficient semantic embedding with the organization, property restrictions, property functions, and other logical assertions [118]. The IC value is the relation and hyponyms

$$IC(co) = \rho \cdot IC_{rel}(co) + (1 - \rho) \cdot IC_{Nmo}(co)$$

$$IC_{rel}(co) = \frac{\log(\text{rel}(co) + 1)}{\log(\text{rel_max} + 1)}$$

$$\rho = \frac{\log(\text{total_rel} + 1)}{\log(\text{rel_max}) + \log(\text{node_max})}$$

Where, $rel(co)$: Denotes the number of relations of concept co and rel_max : The total number of relations.

The last one assumes that each concept is unique in the hierarchy and IC value is the function of concept's, which can separate different concepts effectively and gives more accurate value [64]. It was defined as:

$$IC(co) = \frac{\log(\text{depth}(co))}{\log(\text{deep_max})} * \left(1 - \frac{\log\left(\sum_{a \in \text{hypo}(c)} \frac{1}{\text{depth}(a)} + 1\right)}{\log(\text{node_max})}\right)$$

Where for a given concept co , a is a concept of the taxonomy hierarchy, which satisfies $a \in \text{hypo}(co)$.

If co is root, $\text{deep}(\text{root})$ is 1 and $\log(\text{depth}(co))$ is 0. If co is a leaf, $\text{hypo}(co)$ is 0. Then

$$\sum_{a \in \text{hypo}(co)} \frac{1}{\text{depth}(a)} = 0$$

$$IC(co) = \frac{\log(\text{depth}(co))}{\log(\text{deep_max})}$$

2.3.3.3. Feature-based Measure

This measure is independent on the taxonomy as well as independent on the subsumers of the concepts, and attempts to exploit the properties of the ontology to obtain the similarity values. It take concept's feature into considerate, which is based on the assumption that each notion is described by a set of words representing its features or properties, of their WordNet definitions or glosses. The less non-common characteristics two concepts have and the more common characteristics they have indicate the more similar the concepts are [115].

2.4. Measuring Semantic textual similarity

Recently, STS has attracted an attention of many researchers. For example the Semantic Textual Similarity competitions in Semantic Evaluation Exercises have been held from 2012 to current year (2017). In STS, systems rate the degree of semantic equivalence between two text snippets. Snippets of text, STS capture the notion that some texts are more similar than others, measuring their degree of semantic equivalence. The similarities are identified by score that ranges over a continuous scale [0, 5], where 5 represent semantically equivalent sentences and 0 represents unrelated sentences [20]. The similarity of both sentences is measured by summation of each score. Interpretable STS (ISTS) adds an explanatory layer to STS in order to clarify more. To do so first identify the chunks in each sentence, and then, align chunk based on relatedness across the two sentences, indicating the relation and similarity score of each alignment.

2.4.3. Preprocessing

Different languages use specific preprocessing techniques mostly because of grammatical and morphological reasons. The goal of preprocessing is to reduce inflectional forms of words to a common

base form and increase performance of the system. This can be done by many preprocessing techniques such as: Tokenization, stop-word removal, lemmatization, stemming, part of speech tagging, parsing, named entity recognizing, chunking etc. Some of the most popular for preprocessing will be presented briefly.

2.4.3.1. Tokenization

Tokenization is the task of chopping up documents into tokens and throwing away punctuation and other unwanted characters. The same process must be applied to document and query to assure that a sequence of characters in text will match the same sequence typed in the query [65]. Sentences are decomposed after applying well known Natural Language Processing by OpenNLP [66]; Stanford CoreNLP [67], [68]; Tree Tagger [70]; Asiya toolkit [71].

2.4.3.2. Stop Words removal

Some words that occur in most documents have a small impact in the text similarity. The NLTK stop word list used for stop word removal [72] filter out punctuations and stop-words by using a pre-compiled stop-words list. The stop word list was augmented with adverbs that occurs more than 500, 000 times in the corpus [73]. The function words such as prepositions, conjunctions, and articles carry less semantics than content words (i.e. nouns, verbs, adjectives, and adverbs) and thus removing them might eliminate the noise and provide a more accurate estimate of semantic similarity [75]. To determine a list of stop-words the terms in the document collection are sorted by collection frequency occurrences of terms, and the most frequent terms with little or none semantic value relative to the domain of the documents are then discarded. Semantic content of documents must be taken into account when selecting the stop words [40].

2.4.3.3. Lemmatization

Lemmatization is a technique from Natural Language Processing which does full morphological analysis and identifies the base or dictionary form of a word, which is known as the lemma. Lemmatization performed by the WS4J library [66], Tree Tagger [70], Stanford CoreNLP Toolkit [67], [68], [73], Asiya toolkit [71]. Then the WordNet-based Lemmatizer implemented in Natural Language Toolkit (NLTK) was used to lemmatize all words to their nearest base forms in WordNet, for example, was is lemmatized to be [76]. The similarity is calculated as follows: first of all, words in sentences p and q are lemmatized and mapped to the related WordNet synsets [62].

2.4.3.4. Stemming

Stemming usually refers to a crude heuristic process that chops off the ends of words in the hope of retrieving the stem of the word correctly most of the time. It often includes the removal of derivational affixes [65]. Stemming is a language-dependent process in a similar way to other NLP techniques. It transforms inflated words into their most basic form. Word2Vec handles the stem variations to some extent when it learns the vector representation from the raw input data. Thus for the domain-specific models, only remove stop words and do not need stem [76].

2.4.3.5. Part-of-speech (POS)

POS-tagging is the main process of making up the chunks in a sentence as corresponding to a particular part of speech. POS tagging is the process of assigning a POS tag such as noun, verb, pronoun, preposition, and adverb, adjective or other tags to each word in a sentence. Nouns can be further divided into singular and plural nouns, verbs can be divided into past tense verbs and present tense verbs and so on [85]. The input data undergoes the data preprocessing in which it uses Tree Tagger [70] to perform POS tagging. Also POS tagging is carried out using Stanford CoreNLP [67], [68], [69], [75] as well as tagged by OpenNLP [66]. The MT metrics for each text pair were computed with the Asiya toolkit [71]. Words are POS-tagged using Penn Treebank compatible POS-taggers: NLTK [80] for simple and OpenNLP for syntax [75].

2.4.3.6. Parsing

Syntactic parsing with the Stanford CoreNLP Toolkit uses the Stanford Parser to obtain the dependency parsing from given sentences [67], [70], [69]. The MT metrics for each text pair were computed with the Asiya toolkit [71].

2.4.3.7. Named Entity Recognition

Seven types of named entities such as: location, organization, date, money, person, time and percent, recognized by the Stanford CoreNLP toolkit [67] for English [69], [75], [76] were considered [62]. For English, all the pre-trained NER models made available by the Apache OpenNLP library were used [95]. In addition to the overlap of capitalized words, the syntax system uses the OpenNLP named entity recognizer and classifier to compute the overlap of entities for each entity class separately. In interpretable semantic textual similarity additional work is done as described next.

2.4.4. Alignment

Monolingual alignment is the task of discovering and aligning similar semantic units in a pair of sentences expressed in a natural language. Such alignments provide valuable information regarding how and to what extent the two sentences are related [78]. Word alignment is direction-dependent and not restricted to one-to-one alignments [77].

2.4.4.1. Identifying Chunks

Chunking is a process to parse the sentence into a form that is a chunk based sentence structure. A chunk is a textual unit of adjacent POS tags which display the relations between their internal words [85]. A sequence of adjacent words grouped on the basis of linguistic properties [83]. OpenNLP chunker was used to chunk the input sentences and some post processing was done [66]. For the post processing [66] was based on a few rules observed from gold standard chunks. Those rules include combining chunks of specific chunk tags given by OpenNLP chunker. A large number of rules were discovered but the following were the rules, which maximized accuracy [79].

2.4.4.2. Aligning Chunks

Two sentences spliced into gold standard chunks. Firstly, the similarity between chunks of all possible chunk-pairs is calculated, upon which chunks are aligned. Where, chunk pairs with a high similarity score are aligned first, followed by pairs with lower similarity [68]. To do so there are many post processing left which affect the alignment of chunks as well as similarity scores. Those things will be present in detail.

Punctuation characters were removed from the tokens [77]; [80] except for the decimal point in numbers [80] also ignores case information, and symbols. This enables to match expressions like long term and long-term [77]. All numbers written as words were converted into numerals, e.g., “2.2 million” was replaced by “2200000” and “fifty six” by “56”. All mentions of time were converted into military time, e.g., “5:40pm” was replaced by “1740” and “1h30am” by “0130” [80]. Normalized temporal expressions are aligned if they denote the same point in time or the same time interval (e. g. 14:03 and 2.03 pm). On other hand measurement expressions are aligned if they express the same absolute value (e. g. \$100k and 100.000\$) [77]. Abbreviations were expanded using a compiled list of commonly used abbreviations then aligned [80].

If one of both sequences consists of exactly one all-caps-token then the system test if it is the acronym of the other sequence (e. g. US and United States). Additionally a small database which containing high-

frequency synonyms (e. g. does and do), antonyms (e. g. doesn't and does) and negations (e. g. don't, never, no) created for lookups [77]. Remaining content words are aligned using cosine similarity on word2vec vectors [81].

According to [30] there were three lookups relations for synonym, antonym and hypernym. A strict synonym lookup file was created using WordNet. As well as, an antonym lookup file was created by building an antonym set for a given word from its direct antonyms and their synsets. Also another lookup file for strict hypernym was constructed [30].

Analogously to align each content word to the content word of the other sentence with the same POS tag that yields the highest similarity score. In this way weak alignments problem can happen. The solution is rejecting alignments with a similarity less than $1 / 3$ [35].

The presence of a common word sequence in sentence 1(S1) and sentence 2 (S2) is indicative of an identical, and contextual similar word in the other sentence for each word in the sequence. To aligns identical words in a sequences of length n containing at least one content word. The system align all identical word sequence pairs in S1 and S2 containing at least one content word [78]. Note, afterward up to the end S1 represent sentence 1 and S2 represent S2.

Named entities separately align to enable the alignment of full and partial mentions (and acronyms) of the same entity. The Stanford Named Entity Recognizer [67] used to identify named entities [78]. Aligning named entities performed in two steps separately. First aligned the exact term matches. Second any unmatched term of a partial mention named entity is aligned to all terms in the full mention named entity. This was done based on only first letter acronyms and aligns an acronym to all terms in the full mention of the corresponding name. Since named entities are instances of nouns, named entity alignment is also informed by contextual evidence, but happens before alignment of other generic content words. Many stop words (e.g. determiners, modals) typically demonstrate little variation in the dependencies they engage in, for this reason ignored type equivalences for stop words and implemented only exact matching of dependencies. [78]

Another work was done previously aligning start by constructing the token to token link matrix in which each element at position (i, j) determines that there exists a link between token i (from S1) and token j (from S2). A link exists in the matrix if and only if the monolingual word aligner has determined that both tokens are related. Then, the system uses token regions to group individual tokens into segments, and calculates the weight between every segment in the sentence pair. The weight among two segments is

proportional to the number of links that interconnect tokens inside those segments. Chunk to chunk alignment calculated by summing regions collapsed the token to token link matrix onto a chunk to chunk link matrix. After that, to detect chunk to chunk similarity algorithm used to discover which segments (x, y) are which score the highest weight. The algorithm extended to discover which are the segments that are linked to either segment x or segment y, but not with a maximum alignment ratio. This processing to find not maximal weights is essential to effectively assign the context alignment tag for 1: N relations. In addition, the system is also aware of chunks that have been left unaligned [82].

The naive approach directly assigns the tag as a majority classifier would do, that is: for the segments with highest weight it always assigns the equivalence tag, for the segments that are linked with lower weights it always assigns the context alignment tag, and for the not aligned segments it always assigns the not aligned tag. The machine learning approach used to improve the tag assignment by overcoming the naive approach limitation that employs mostly only for segment pairs tagged as equivalent. In this approach many related features were used such as: Jaccard overlap, segment length, WordNet similarity among segment heads, WordNet depth, and etc. features obtained. To induce the model the Support Vector Machine (SVM) implemented [83] under the latest version of Weka [84].

2.4.4.3. Scoring Aligned Chunk

To assign segment pair similarity scores the system can also use two distinct approaches: the naive approach and the cube based regression approach. The naive scorer directly assigns a certain score to each one of the tags, which has been previously assigned using the naive tagger: for equivalence tags it assigns a score of 5 and for not aligned and context aligned tags it assigns 'NIL'. The regression approach uses the cube to improve the score given to segment pairs tagged by the machine learning tagger. Its returning value is used directly as the value for the pair similarity score [82].

The word similarity score less than fixed threshold (0.3) was reset to 0 in order to avoid noisy alignments. Then the chunk similarity normalized by the number of tokens in the shorter chunk such that it assigned higher scores to pairs of chunks for example physician and general physician. Finally optimal alignment at chunk level is done in order to calculate the sentence level similarity. The chunk-to-chunk similarity threshold is 0.4 to prevent noisy alignments. However, the similarity score was normalized by the average number of chunks in the given texts pair. All threshold values were set empirically based on the performance on the training set [30].

2.4.4.4. Labeling Aligned Chunks

Given two sentences split into gold standard chunks, a system carries out the task requirements using sense-based string kernel by considering each chunk as a text snippet. After that, for each alignment of chunks c_1 and c_2 , the alignment type is determined according to the following rules: If the similarity score between c_1 and c_2 is 5, the type is EQUI. If all word senses of c_1 matched the word senses in c_2 , the type is SPEC2; similarly for SPEC1. If both c_1 and c_2 contain a single word sense, and are directly connected by an antonym relation in BabelNet, then the type is OPPO. If the similarity score between c_1 and c_2 is in range $[3, 5[$, the type is SIM; while if it is in range $]0, 3[$, the type is REL. If any chunk has no corresponding chunk in the other sentence, then the type is either NOALI or ALIC based on the alignment restriction in the subtask [68].

2.5. Related work

Very limited works have been done in the past in the areas of Interpretable STS. Interpretable STS focus to explain why two sentences may be related or unrelated, by supplementing the similarity score with an explanatory layer. As a first step in this direction, given a pair of sentences, systems needed to align the chunks across both sentences, and for each alignment, classify the type of relation, and provide the corresponding similarity score [20].

For interpretable STS the similarity scores range from 0 to 5. With respect to the relation between the aligned chunks, the present pilot only allowed 1:1 alignments. As a consequence, to include a special alignment context tag (ALIC) to simulate those chunks which had some semantic similarity or relatedness in the other sentence, but could not have been aligned because of the 1:1 restriction. In the case of the aligned chunks, the following relatedness tags were defined: EQUI denotes semantically equivalent chunks, oppositional meaning is labeled with OPPO, SPE1/2 denote similar meaning of the chunks, but the chunk in S1/S2 is more specific than the other one. SIM and REL denote similar and related meanings, respectively. ALIC is not used, because our algorithm is not restricted to one-to-one alignments. Finally, all unaligned chunks are labeled with NOALI [20]. According to extensive feature extraction from word alignments for semantic textual similarity do not differentiate between SIMI and REL; all REL alignments are considered as SIMI alignments. From previous work done on interpretable STS some of them which have relatively better result will explained in this section.

2.5.1. Rule base approach

In this approach the consideration is given to NeRoSim and Inspire system. The system takes given sentence pair and maps chunks of the first sentence to those from the second by assigning different relations and scores based on a set of rules.

For the interpretable similarity NeRoSim used a rule-based approach blended with chunk alignment labeling and scoring based on semantic similarity features based on regression models by combining a wide array of features including semantic similarity scores obtained from various methods (e.g. sim-Mikolov). The Inspire system made use of a rule-based approach using Answer Set Programming for determining chunk boundaries and for aligning chunks and assigning alignment type and score.

Both NeRoSim and Inspire system performs similar preprocessing steps like: stop word removal, POS tagging, lemmatization, and named-entity recognition by Stanford CoreNLP [67]. Similarly in both system lookup for synonym, antonym and hypernym relations from WordNet.

The Inspire system extends the basic ideas from NeRoSim [12], however the researchers realize the rules in logic programming and obtain the result with an Answer Set Solver. For chunking the Inspire system is based on a joint POS-tagger and dependency parser and an Answer Set Program (ASP)¹ [108], [109] that determines chunk boundaries. The Inspire system realizes chunking as a preprocessing step.

Inspire system alignment is based on ideas of NeRoSim entry, however re-implemented the system and realizes the rule engine in ASP [108], [109] which gives flexibility for reordering rules or applying them in parallel. ASP alignment architecture is present as follows: input sentence pairs then preprocess with POS, NER, WordNet and Word2Vec represented as a set of ASP facts. A generic set of rules represents how alignments can be defined and changed. Lookup of relate word in WordNet and distributional similarity with the Word2Vec tool [99] with SkipGram context representation, window size 10, vector dimension 200, and pruning below frequency 50. Word-to-word similarity $\text{sim}(w_1, w_2)$ is computed using cosine similarity between vectors of words w_1 and w_2 .

NeRoSim experiments was applied on rules in the training data set by varying thresholds for sim-Mikolov scores and selected the thresholds that produced the best results in the training data set. Since three runs named R1, R2 and R3 were submitted. R1 applied full set of rules with 375 stop-words. However EQUI4 was modified such that it would apply when unmatched content words of the bigger chunk were of noun rather than proper noun type. R2 Same as R1 but with extended stop-words from 375 to 686. R3 Applied full set of rules with extended stop words.

Results F1-measures Baseline is 0.555 for Headlines whereas on Headlines test data, R3 performed well 0.642. F1-measures Baseline of Images is 0.555 for Images test data, R1 was the best in alignment, F1-measures metrics result found was 0.584. R3 performed better among all runs in case of Headlines data in overall. This was chiefly due to modified EQ4 rule which reduced the number of incorrect EQUI alignments. Researcher observed that performance of the system was least affected by size of stop-word list for Headlines data as both R1 and R2 recorded similar F1-measures for all evaluation metrics. However, R1 performed relatively better than R2 (0.561) in Images data-particularly in correctly aligning

¹ <https://bitbucket.org/snippets/knowlp/yrjqr>

chunk relations. It could be that images are described mostly using common words and thus were filtered by R2 as stop words.

Similarly Inspire system adds some features and rearrangement of NeRoSim alignment order. The Inspire implements three run named Run1, Run2, and Run3. In their Run1 they optimized alignment, type and score used implementation of NeRoSim rules in the same order, as it is however SIMI4, SIMI5, and REL1 which were excluded. Run2 is optimized for prediction of alignment this is done by using all NeRoSim rules in their original order for both dataset including SIMI4, SIMI5, and REL1. In Run 2 system scorer tool does not strictly punish overlapping alignments. Run 3 solve the limitation of Run 2 by rearranging of alignment.

Inspire system tested and optimized it on the training data for Headlines, Images as criteria for accuracy the competition used the F1 full consideration of alignment, type, and score. Run 1 performs top in all categories on H and I. The configuration of Run 3 performs least on both datasets. For Gold-Standard Chunks 0.48, 0.55 and 0.56 are the baseline given for datasets image, Headlines and Answer-Students 0.61, 0.70 and 0.51 respectively for top Run. The mean of baseline is 0.53 and mean of result found from the best output of three runs is 0.61. The future directions stated in the paper are: represent semantic knowledge in ASP externals and use ASP guesses, constraints, and optimization.

2.5.2. Machine learning approach with linguistic and rule blended

SVCSTS is one of the research work on ISTS uses Monolingual word aligner [78] and supervised machine learning techniques for interpretable STS. In ISTS the main challenge is to find the semantic relationships between the chunks of S1 and S2. Chunks from the input sentence pair are to be aligned, labeled with the type of alignment and are to be scored on a scale of 0-5 based on their semantic similarity.

OpenNLP chunker was used to chunk the input sentences and some post processing was done. For the post processing we observed a few rules from gold standard chunks. Those rules include combining chunks of specific chunk tags given by OpenNLP chunker. A large number of rules were discovered but the following were the rules, which maximized accuracy. PP + NP + PP + NP, PP + NP, VP + PRT, NP + O + NP, VP + ADVP, and VP + PP + NP + O, etc. Applying these rules they increased accuracy from 86.58% to 90.16% against the gold standard chunks.

The following features were used for each chunk alignment to assign a type for the alignment. Length of S1 and S2 chunks are the first two features. Number of nouns, verbs, adjectives, prepositions in S1 and in

S2 chunk. The path similarity between words of S1 and S2 chunks another features. Unigram and Bigram overlap between S1 and S2 chunks are four features from N-gram.

Labeling aligned chunks was performed by supervised machine learning using Scikit-Learn tool [94]. Average score for each alignment type was calculated from the gold standard data. The average scores that were used to score chunk alignment are 5, 3.75, 3.55, NIL, 0, 2.94, 2.82 and 4 for EQUI, SPE1, SPE2, ALIC, NOALI, SIMI, REL and OPPO type respectively the average score calculated from gold standard chunks.

SVCSTS was experimented the classification of labels using 3 classifiers LinearSVC, SVC with RBF (Radial Basis Function) Kernel and SVC with Polynomial Kernel. But the classifier SVC with RBF (with parameters $C = 1.0$, $\gamma=0.7$) demonstrated to give better results. The experimentation results (F1 Type+Score) of the best of both data are present as: baseline and SVCSTS for Headlines are 0.5556 and 0.5887 respectively, which is Run 1. For Images data Run 2 gives best result which is 0.5964 whereas baseline is 0.4326.

UWB is another system explores machine learning and rule-based approaches to the ISTS task. More focus on machine learning and experiment with a wide variety of machine learning algorithms as well as with several types of features. The core of the system consists in exploiting distributional semantics to compare similarity of sentence chunks.

As a first step of the approach perform the following text preprocessing: 32 stop-words predefined in list to remove from input text, remove special characters (E.g. dots, commas, quotation marks other punctuation) that violate the tokenization. Lowercasing all words as well as lemmatization performed with the Stanford CoreNLP tool [67].

Chunk Semantic Similarity attempts to estimate the similarity function are based upon estimating semantic similarity of individual words and compiling them into one number for a given chunk pair. The experiment was done with Word2Vec [99] and GloVe [110] for estimating similarity of words. Then compile all the word similarities in one number that reflects semantic similarity of whole chunks via the following methods: 1) the vector composition method and 2) an adapted method for constructing vectors called lexical semantic vectors. In this case do not weight words with their information content use methods for distributional semantics (Word2Vec and GloVe) rather than semantic networks. Then maximal similarities with words from chunks a and b, respectively was taken. Also to identify important word weight the vectors with invers document frequency weighting used.

UWB system was done on ISTS with machine learning approach task divided into to three classification / regression tasks: Alignment binary classifications that decide whether two given chunks should be aligned with each other. Score classification / regression which experiment with both classification and regression of the chunks similarity score. Type classifications classify all aligned pairs of chunks into a predefined set of types. The classification experiment takes place with Weka [84].

Four categories of features (lexical, syntactic, semantic, and external) employed. Lexical features consist of the following features: word base form overlap, word lemma overlap, chunk length difference, word sentence positions difference. POS tagging and syntactic parsing are performed with Stanford CoreNLP [67] for syntactic features. Post-processing were done if one chunk is aligned with multiple chunks in the other sentence, these chunks should be merged into one chunk. Rule-based Approach was used for the chunk alignment to iterate over all chunks from sentence S^a and find the chunk with maximal similarity from sentence S^b .

Experimental setup Machine learning approach employs the following classifiers and classification frameworks: Alignment binary classification – Voted perceptron (Weka). Score classification – Maximum entropy (Brainy). Type classification – Support vector machines (Brainy). These classifiers perform best on the evaluation datasets. We achieved the best results for estimating chunk similarity with Word2Vec and the modified lexical semantic vectors

Run 1 experimented with reduced feature set (word overlap, word positions difference, POS tags difference, semantic similarity, global semantic similarity, paraphrase database) and Run 3 with all features. In rule-based approach UWB have achieved the best results for estimating chunk similarity with Word2Vec and the modified lexical semantic vectors also, set the threshold for the similarity score to 2.5. All lower values are set to 0. This is the run 2. The results clearly show that the unsupervised runs 2 perform much worse than the supervised runs 1 and 3.

The experimentation results of the three official runs are as follow: On gold-standard chunks the result found from Images, Headlines and Answer students dataset is 0.6708 (run 3), 0.6296 (run 3), and 0.6248(run 1) respectively. Generally overall results (mean) of F1 full consideration of alignment, type, and score is Run1 0.638, Run3 0.637, and Run2 0.566.

VRep system was another research done on ISTS area. According to this work VRep system uses many features that extracted to create a learned rule-based classifier to assign a label. It uses semantic and syntactic (form of the chunks) relationship features. VRep makes extensive use of WordNet for both STS,

where the Vector relatedness measure is used, and for iSTS, where features are extracted to create a learned rule-based classifier. Preprocessing in the first step, consist tokenization, lowercase all characters, punctuation removal, POS tagging Lingua::EN::Tagger² is used. POS tags are used for stop word removal and for alignment reasoning. Stop word removal remove any words that are not tagged as a noun, verb, adjective, or adverb. This reduces chunks and sentences to content words.

VRep's STS computation is similar to the method described by NeRoSim [12]. Chunk similarity takes two chunks (c_1 , c_2) as input and computes the weighted sum of maximum word to word similarities, $\text{sim}(w_i, w_j)$. To do this, the $\text{sim}(w_i, w_j)$ is found for each word in c_2 against c_1 , and the maximum is added to a running sum. For words in WordNet, $\text{sim}(w_i, w_j)$ is the Vector relatedness measure with a threshold applied. The Vector measure was chosen for several reasons.

Firstly it returns values scaled between 0 and 1 which is beneficial for applying thresholds in both chunk alignment and alignment reasoning. Secondly the Vector measure works well when w_i and w_j are different parts of speech because it does not rely on WordNet hierarchies. When calculating $\text{sim}(w_i, w_j)$ all possible senses of both w_i and w_j are used, and $\text{sim}(w_i, w_j)$ is chosen as the maximum value. This eliminates the need for word sense disambiguation (WSD). After computing the measure, a threshold is applied that reduces any value less than 0.9 to 0.0. The threshold prevents dissimilar terms from impacting the STS which improves the accuracy and prevents noisy chunk alignments.

In chunk alignment chunkSim is computed between each chunk of two aligned sentences and the chunk with the highest chunkSim is selected for alignment. Multiple alignments are allowed for a single chunk. If all chunks have a similarity of 0, no alignment (NOALI) is assigned. Due to the high $\text{sim}(w_i, w_j)$ threshold, no threshold is required for chunkSim as with NeRoSim [12].

Alignment Reasoning takes as input a chunk pair and provides a reason why that chunk pair is aligned. VRep's alignment reasoning is inspired by NeRoSim [12] features (antonyms, synonyms, etc.) and SVCTSTS [79] features (number of words or counts of parts of speech in a chunk pair). Both these systems classify a chunk pair using features extracted from the chunk pair itself. VRep combines the two approaches and extracts a total of 72 syntactic and semantic features for each chunk pair. The classifier uses only 24 of original 72 features and a series of 10 rules. All classifiers are WEKA [84] implementations.

² <http://search.cpan.org/acoburn/Lingua-EN-Tagger/>

Alignment scores are assigned as the required scores, 0 for NOALI and 5 or EQUI, or the average alignment score for each class. The best performing set of scores came for all topics, came from the images data set alone. Scores used for each class are as follows: EQUI = 5.00, OPPO = 4.00, SPE1 = 3.24, SPE2 = 3.69, SIMI = 2.975, REL = 3.00, NOALI = 0.00.

The experimentation results of the three official runs are as follow: On gold-standard chunks the result found from Images, Headlines and Answer students dataset is 0.597 (run 3), 0.547 (run 3), and 0.580 (run 3) respectively. Generally overall results (mean) of F1 full consideration of alignment, type, and score is Run3 0.575, Run2 0.573, and Run1 0.556.

The future directions stated in the paper are: more analysis (JRIP and other analysis criteria) should be done to refine the features used in classification. , additional metrics, such as word2vec measure for words outside of WordNet could be incorporated. Additional data should be added for training classifiers and also to reduce the class imbalance and will likely result in a set of rules for the REL class.

FBK-HLT-NLP system is built combining different linguistic features in a classification model for predicting chunk-to-chunk alignment, relation type and STS score. The input data undergo a data pre-processing in which we use a Python implementation of MBSP [111] a library providing tools for tokenization, sentence splitting, part of speech tagging, chunking, lemmatization and prepositional phrase attachment. To compute the chunk-to-chunk alignment, the relation type and the STS score we use a total of 245 features.

Chunk tags. A total of 18 features (9 for chunk1 and 9 for chunk2) are related to chunk tags (e.g.noun phrase, prepositional phrase, verb phrase). For each chunk in the SYS datasets -chunked with MBSP- the system takes into consideration the chunk tags as identified by that library. For the GS datasets -already chunked datasets- the system first re-chunks the datasets with MBSP and then evaluates if chunks in the god standard correspond to chunks as identified in MBSP. If this is the case, chunk tag is extracted; otherwise the systems does the same operation (i.e. re-chunking and tag extraction) using a regular expressions-based shallow parser for English that uses a part-of-speech tagger extended with a tokenizer, lemmatizer and chunker. If no corresponding chunk is found, no chunk tag is assigned.

Four further features are related to tokens and lemmas overlap between a pair of chunks. In particular, the system considers the percentage of tokens and lemmas in chunks1 that are present also in chunk2 and vice versa. It consider if the chunks are evaluated as aligned, if *chunk1* is not aligned, if *chunk2* is not aligned (3 features).

Twelve WordNet based features evaluate the type of relation between chunks by considering all the lemmas in the two chunks and checking whether a lemma in chunk1 is a synonym, antonym, hyponym, hypernym, meronym or holonym of a lemma in chunk2.

Word embedding use a distributional representation of the chunk for a total of 200 features (100 for chunk1 and 100 for chunk2) by first calculating word embedding and then combining the vectors of the words in the chunk (i.e. by calculating the element wise mean of each vector). Mikolov word2vec [99] with 100 dimensions using ukWaC, GigaWords (NYT), Europarl V.7, Training Set (JRC) corpora were used. Neural Network used multitask MLP to classify chunk pairs. The system uses three classifiers: one for the chunk alignment, one for alignment type, one for STS score.

FBK-HLT-NLP combines the output of the three classifiers organized in a pipeline. First, it label as “not aligned” all the punctuation chunks; then we label as “aligned” all the chunks aligned by the first classifier, allowing multiple alignments for each chunk. For every aligned chunk pair it adds the type label and the STS score. FBK-HLT-NLP do not take into consideration chunk pairs classified as “not aligned” by the first classifier even if they are classified with a label different from NOTALI or with an STS score higher than 0.

The experimentation results of the three official runs are as follow: On gold-standard chunks the result found from Images, Headlines and Answer students dataset is 0.574 (run 1), 0.562 (run 3), and 0.589 (run 3) respectively. Generally overall results (mean) of F1 full consideration of alignment, type, and score is Run3 0.572, Run2 0.571, and Run1 0.551.

The future direction stated by the author is adding additional datasets should use as well as, deep analysis of the distribution of the type labels and of the STS scores can improve significantly the performance of the system.

Among the above study discussed in section related work Run1 of the Inspire system was the best result with 0.696 of F1 on headline dataset. Similarity run3 of UWB system gave F1 0.6708 result that was the best result on Image data.

CHAPTER THREE

INTERPRETABLE SEMANTIC TEXTUAL SIMILARITY (ISTS)

3.1. Introduction

Interpretable semantic textual similarity (ISTS) is a semantic task adds a vital explanatory layer to pair semantic sentence similarity. The task was introduced for the first time as a pilot task in 2015 semeval semantic textual similarity (STS) tasks. Several approaches were proposed including NeRoSim [12], UBC-Cubes [82] and Exb-Thermis [77]. For each sentence pair, the systems identify the chunks in each sentence according to standard gold chunks, align corresponding chunks and assign a similarity/relatedness score and type of the alignment for each aligned chunks.

The alignment types EQUI (semantically equivalent), OPPO (opposite in meaning), SPE1 (one chunk is more specific than other), SPE2 (one chunk is more general than other), SIM (similar meanings, but no EQUI, OPPO, SPE1, SPE2), REL (related meanings, but no SIM, EQUI, OPPO, SPE1, SPE2), ALIC (does not have any corresponding chunk in the other sentence because of the 1:1 alignment restriction), and NOALI (has no corresponding chunk in the other sentence) [82]. On the 2015 pilot subtask only one-to-one (1:1) alignments were allowed. Because of 1:1 chunk alignment 1 chunk aligned only with 1 chunk, even if that chunk is similar with two or more chunks. So many-to-many (N:M) alignments are solution to this limitation and ALIC indicate N:M alignments relation in this research.

3.2. Architecture of ISTS

The ISTS system is composed of several modules designed to handle the computation of similarity score of pieces of text of different lengths. Figure 3.1 shows its model which has four main modules, (1) for chunk input pair of sentence, (2) for calculating the similarity between each chunk of S1 with each chunk of S2, (3) align similar chunks based on similarity score, and (4) finally predict type of each alignment.

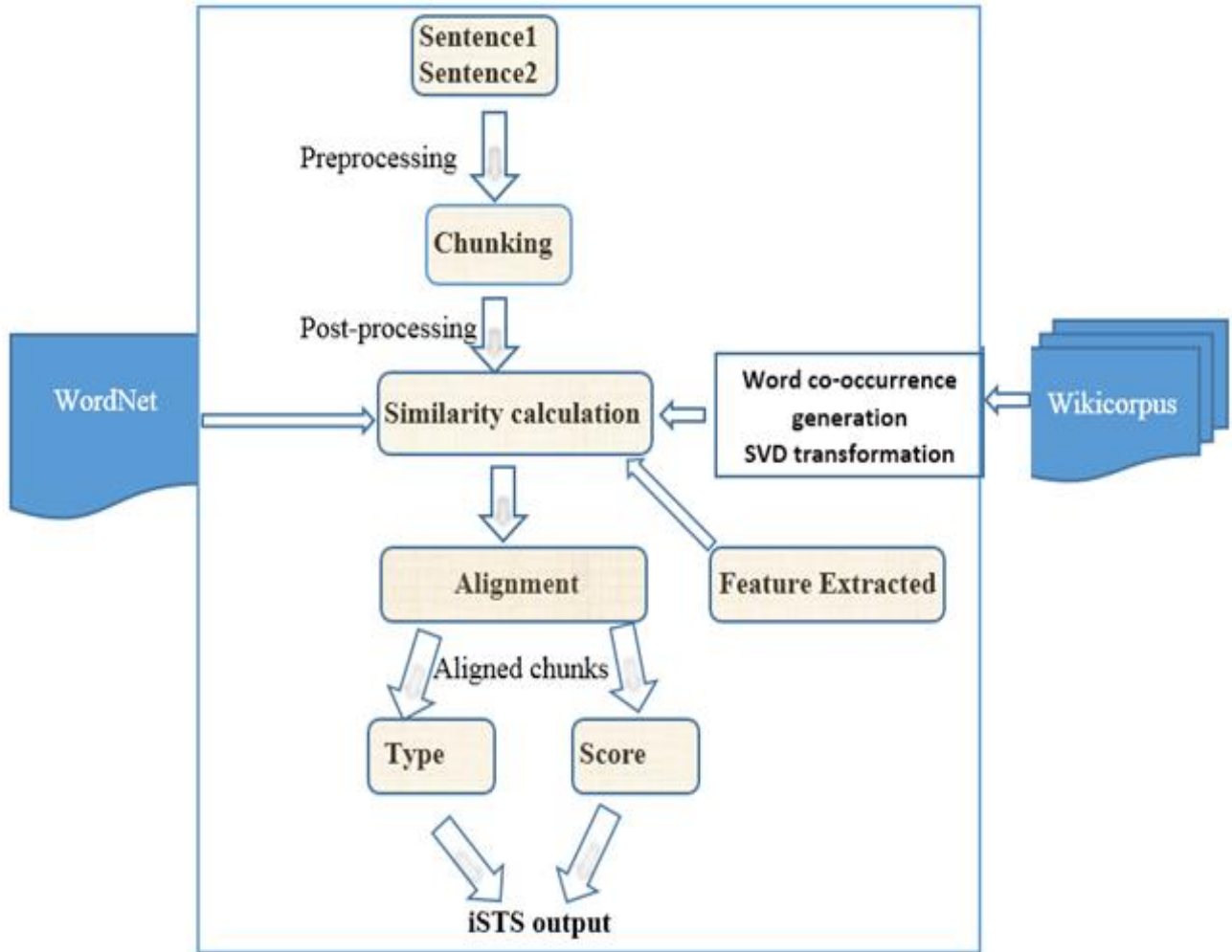


Figure – 3.1 Interpretable semantic textual similarity achitecture

The core of the system is the interpretable semantic textual similarity architecture, which is based on a combination of corpus base (i.e. LSA) similarity score and knowledge base (i.e. WordNet). From the WordNet we used three features such as: antonym, synonym and hypernym. In addition LSA similarity measure based on Wikicorpus and also few string base similarity measure used that will discuss in section 3.3.4. Some rules were discovered to predict the reason why it could be aligned.

The ISTS system taking different length of inputs and many preprocessing are used before chunk pairs of text into segments. After a chunk is identified post process is taking place then semantic similarity calculation is perform depend on surface similarity, WordNet, corpus and additional features. The next step is aligning chunks of S1 with chunks of S2 based on similarity score of pair chunk. Lastly ISTS result

is computed in the form of Type and score of similarity for all chunks. Detail of the system is presented in this chapter as next.

3.2.1. Preprocessing

The system takes two texts (S1 and S2) as input then segment paired sentence into chunks. Before computing similarity many preprocessing are done to simplify the chunking and similarity calculation. Among preprocessing we used tokenization, part of speech tagging, and parsing is presented briefly in this section as follows.

Tokenization is the task of chopping up texts into tokens and throwing away punctuation and other unwanted characters. A sentence is decomposed after applying well known Natural Language Processing by Stanford CoreNLP [67]. All punctuations removed except some punctuation explicitly described as next:

There is a punctuation mark that has different uses when used in English. Apostrophe mark (‘), for example, in English shows possession and contraction. In English there are also a number of tricky cases of the use of apostrophe as it is used for the contractions. For example, it is used for the contractions purpose with the words like *aren’t*, *didn’t*, *it’s*. Therefore, an exception list is created to preserve apostrophe mark when it is used as contraction with English words.

There are also punctuation markers which are used for various purposes in English language. For example, full stop (.) is used to indicate the end of a sentence besides to serve as abbreviation. For instance, the dot in “*B.C*,” “*T.V*.” serves as abbreviation. So an exception list is created for such kind of punctuation markers if existed in the text. On another way colon (:) is removed unless between number which use to separate hour from minute and minute from second.

Another key thing done in preprocessing step is POS-tagging. It is the process of assigning a POS tag such as verb, noun, adverb, adjective, pronoun, and preposition or other tags to each word in a sentence. Verbs can be divided into past tense and present tense and nouns can be divided into plural and singular nouns, and so on. POS tagging are carried out using Stanford CoreNLP Toolkit [67]. Syntactic parsing with the Stanford CoreNLP uses Stanford Parser to obtain the dependency parsing from given sentences [67].

3.2.2. Chunks

I used gold chunks to perform chunking for input text. In order to make chunks according to standard gold chunks we used Stanford coreNLP [67] API for generating parse trees. Before chunking, we do preprocess

like tokenization, part of speech tagging, parsing and removed most of the punctuations except some of them listed above under preprocessing. Output of parser is further post-processed to combine each single preposition phrase with the preceding phrase. In the case of chunking based on Stanford coreNLP parser, a conjunction such as ‘and’ was consistently being separated into an independent chunk in most cases and therefore improved chunking can be realized by potentially combining chunks around a conjunction. These processing heuristics are based on observations from gold chunks data. We observed that quality of chunk has a huge impact on the overall score of system chunks track. A large number of rules were discovered such as: PP+NP+PP+NP, PP+NP, PP+ADJP+VP, PP+ADJP, PP+ADV, NP+PP+ADJP, VP+VP, VP+NP, ADJP+ADJP, ADJP+NP, and many other rules were discovered

3.2.3. Post processing

Several text posts processing operations are performed after a sentence is chunked. For example replace all hyphens with white spaces. Also remove all non-alphanumeric symbols like slashes angular brackets etc. from the chunks. A predefined vocabulary, POS tags, and regular expressions are used to recognize multi-word terms including noun and verb phrases, proper nouns, numbers and time [73] lowercase the words. The two consecutive words in one sentence that appears as compound in the other sentence is replaced by the supposed compound if only appear in the other sentence.

Some words that occur in most documents have a small impact in the text similarity. The NLTK stop word list used for stop-word removal [80] filter out punctuations and stop-words by using a pre-compiled stop-words list. The function words such as prepositions, conjunctions, and articles carry less semantics than content words such as: nouns, verbs, adjectives, and adverbs and thus removing them might eliminate the noise and provide a more accurate estimate of semantic similarity [75].

Some expressions need normalized to simplify similarity calculation and alignment. Temporal expressions are normalized into common form e. g. 4:00 am and 6:20 pm are normalized into 4:00 and 18:20 respectively.

Now a day seven types of named entities such as: location, organization, date, money, person, time and percent, are recognized by Stanford CoreNLP toolkit [67] for English. In addition the overlap between capitalized words, the syntax system uses the Stanford CoreNLP toolkit NER and classifier to compute the overlap between entities for each entity class separately.

Lemmatization is a technique from Natural Language Processing which does full morphological analysis and identifies the dictionary or base form of a word, which is known as the lemma. We perform lemmatization by Stanford CoreNLP Toolkit [67] for content words (nouns, verbs, adjectives, and adverbs) but not for functional words articles, pronouns, prepositions, conjunctions, auxiliary verbs, modal verbs, and punctuations from the chunks because they do not carry semantic content, but keep the cardinal numbers.

Acronym is expanded by using a list of commonly used Acronym compiled for countries, capital cities, units of measurement etc. All number written in the form of text replaced by digit number, e.g. sixty six, two hundred nineteen, 2 million replaced by 66, 219, and 2000000 respectively.

3.3. Similarity calculator

The interpretable semantic textual similarity module manages inputs of text in English with different length, and uses the semantic chunk similarity modeled to compute the similarity between the given pieces of sentence. Before calculating similarity a pair of sentence should be preprocessed, chunked, and also post-process performed. For similarity calculation the system performs calculation based on string, corpus and linguistic (i.e. Named entity and WordNet) similarity. Corpus similarity measured by LSA in order to avoid weak alignment 0.2 threshold values taken. Then similarity calculator uses a feature extracted from observation of gold chunks datasets. Finally aligning chunks uses an alignment algorithm. Generally any word-to-word similarity measure includes simple word match, WordNet based similarity measures, and LSA based similarity measures.

3.3.1. String similarity

Content word similarity- we developed a new algorithm calculates content word (i.e. verb, noun, adjective and adverb) similarity. The algorithm matches words based on their string similarity and part of speech tag similarity (*POSim*). Being same POS tag is not enough to determine pair words are similar or not. Additionally word N-gram similarity which calculated by jaccard coefficient between lemmas lowercased of word in *ChiS1* and word in *ChjS2* without stop-words for $n = 2$ taken into consideration. If all words in pair chunks are identical N-grams similarity score is = 1; otherwise, N-gram similarity is between 0 and 1. It's used to calculate string similarity mostly where there is slight spelling error which is not handled by content word overlap. For example if we compare two string *motorcycle* $\langle == \rangle$ *amotorcycle* N-gram only considered as it is similar. Because ISTS model is not automatically correct

spelling. To clarify more we used N-gram similarity to count word overlap of content word in *ChiS1* and *ChjS2* are identical, then Content word similarity = 1. To that end N-gram similarity considers word overlap if it is ≥ 0.9 and *POSim* measure chunk-to-chunk similarity even though it depends on word-to-word similarity discussed in this section. The detail will discuss under chunk similarity section.

Number similarity- is not handed by content word similarity even if the both numbers are equivalent; because of number is CD type given by POS tagger. Numbers overlap between *ChiS1* and *ChjS2* no matter if it's written in text form or digit form. The similarity computed after converting number in text form into digit form. If number is exactly the same value the similarity score is then Number similarity (*NumSim*) = 1. Else number similarity less than 1; the explanation will discuss under feature extraction for chunk-to-chunk similarity measures section.

3.3.2. Wikipedia Corpus

3.3.2.1. *Corpus Selection and Preprocess*

Wikipedia has involved much interest from researchers in different fields [96], and it is especially attractive for NLP: It contains many millions of words of text, as well as reasonably edited, which can be used for NLP purposes. Besides, its license allows the texts for the use and redistribution³. The Wikicorpus contains more than 750 million words is a trilingual corpus of English (600 million of words), Spanish (120 million of words) and Catalan (50 million of words) and that is freely available for download⁴.

A very large and balanced text corpus is required to produce reliable word co-occurrence statistics. We selected the Wikicorpus of English that contained 600 million words collected from Wikipedia and have been enriched with linguistic information. In Wikicorpus to differentiate one document from another <doc... tag was added at begin and </doc> added at the end of a document. See sample presented as next:

```
<doc id="33980" title="Waterfall model" dbindex="164">
```

```
The waterfall model be a sequential software development model ...
```

```
</doc>
```

³ <http://www.fsf.org/licensing/licenses/fdl.html>

⁴ <http://www.cs.upc.edu/~nlp/wikicorpus/>

The first line (i.e. header) contains information about the document like: doc id, title, database index etc. The remaining lines (from second line to one more line) contain the text and to that end terminated by end tag.

Using large corpus is expensive for a reason like computer memory size to process term to document matrices. Due to the reason we used subdocument of Wikicorpus and lemmatization performed on selected document.

3.3.2.2. *LSA Word Similarity*

LSA word similarity relies on the distributional hypothesis that the words occurring in similar contexts tend to have similar meanings [25]. Thus, evidence for word similarity can be computed from a statistical analysis of a large text corpus. LSA is a fully mathematical/statistical technique for extracting and assuming relations of expected contextual usage of words in passages of discourse. It is not a traditional 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 from meaningful passages or samples such as sentences or paragraphs [44].

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 document. Each cell contains the frequency with which the word of its row appears in the passage denoted by its column. Next, the cell entries are subjected to a preliminary transformation, whose details we will describe later, in which each cell frequency is weighted by a function that expresses both the word's importance in the particular document and the degree to which the word type carries information in the domain of discourse in general [44].

3.3.2.3. *Word Co-occurrence Generation.*

When using the values of the document-term matrix, $tf * idf$ is used to ignore word present in almost every document. The model is comprised of two different elements. First, the term frequency (tf) part that is the number of times a term is represented in a document, while and inverse document frequency (idf), is the number of documents in the corpus divided by the document frequency of a word, but inverted. $tf * idf$ is a composite weight which combines tf and idf is calculated by the following formula [107]

$$w_{i,j} = t_{f_{i,j}} * idf_i$$

But tf and idf are calculated by the formula

$$t_{f_{i,j}} = (freq_{i,j}/t_j)$$

$$idf_i = \log_2(N/df_i)$$

The $w_{i,j}$ is given by the formula $w_{i,j} = t_{f_{i,j}} * \log_2(N/df_i)$

Where,

i : a term

j : a document

tf : a frequency of a term i in document j

t_j : total number of terms in document j

idf_i : the inverse document frequency of a term i

df_i : the document frequency of a term i (total number of documents containing term i)

$w_{i,j}$: weight of term i in document j

N : total number of documents

Because $tf * idf$ uses the logarithmic scale of its calculations idf , the result cannot be negative. This means that the cosine similarity cannot be negative either. The range of a $tf * idf$ cosine similarity is therefore 0 to 1, where 0 it is not similar and 1 it is exactly the same. The selected corpus from Wikipedia corpus is 1830 documents contained more than 2,525,357 words. We enforced to take 1830 documents because number of row (terms) greater than or equal to column (document). The reason why we limited number of documents in this corpus when SDV matrixes computed the Machine memory process if number of document is less than 1830. We generate word co-occurrence model based on a vocabulary of about 25,000 English words (noun, verb, adjective and adverb). The 25,000 common English words and noun phrases are extracted from Wikipedia corpus. We manually exclude proper nouns from the corpus because there are not many of them and they are all ranked among the top places since proper nouns start with an uppercase letter.

The final dimensions of our word co-occurrence matrices are 25,000 \rightarrow 25,000 when words. Our vocabulary includes only content words (i.e., nouns, verbs, adjectives and adverbs). There are no proper nouns in the vocabulary of the only exception of a list of country names [51].

3.3.2.4. SVD Transformation.

LSA applies singular value decomposition (SVD) to the matrix. SVD has been found to be effective in improving word similarity measures [31]. This is a form of factor analysis, or more properly the mathematical generalization of which factor analysis is a special case. So if document D1 contains (w1, w2) and document D2 contains (w2, w3), we can conclude that there is something common between documents D1 and D2. LSA does this by decomposing the input raw *tf * idf* matrix (*A*, see below) into three different matrices (*U*, *S* and *V*) that are decomposition of the original one (*A*) using SVD. Once that is done, the three vectors are reduced and the original vector rebuilt from the reduced vectors. Because of the reduction, noisy relationships are discarded and relations become very clearly visible. In pseudo-code:

$$A = U \times S \times V^t$$

$$A_k = U_k \times S_k \times V_k^t$$

Where:

A = the original matrix

U = the word vector

S = the sigma vector

V = the document vector

A_k = the reduced matrix

U_k = the reduced word sub-matrix consisting of 0...*k*-1 columns

S_k = the reduced sigma sub-matrix consisting of 0...*k*-1 columns, 0...*k*-1 rows

V_k = the reduced document sub-matrix consisting of 0...*k*-1 columns.

k= where represent dimension used

There is a mathematical proof that any matrix can be so decomposed perfectly, using no more factors than the smallest dimension of the original matrix. One can reduce the dimension of the solution simply by deleting coefficients in the diagonal matrix, ordinarily starting with the smallest. In practice, for computational reasons, for very large corpora only a limited number of dimensions are used [44]. In

general, using the full dimension in SVD hurts the performances and dimension reduction indeed helps discarding noise.

On an experiment we trained as 50, 100, 150, 200, 250, 300, 350, 400, 450 and 500 largest SVD dimension. Meanwhile 250 dimensions gave a good result; so we selected the 250 SVD and reduce the word vectors to 250 dimensions. The LSA similarity between two words is defined as the cosine similarity with their corresponding word vectors after the SVD transformation. To calculate term-to-term similarity two vectors are involved (U_k and S_k). Now, the similarity calculated by using the cosine of the angle between two vectors. When it calculates word similarity it takes word x of chunk i in $S1$ ($ChiS1$) and it propagates calculating similarity with word y of chunk j in $S2$ ($ChjS2$), this circulation it takes the best result among words of $S2$. Up to end the process is continuing until all words with pair sentence cross checked in matrix form.

In order to calculating word-to-word similarity in LSA the system taken into account a word unmatched in any of the other similarity feature listed under section feature extraction. That is to say a word in $S1$ or $S2$ matched in one of synonym, antonym, hypernym, Date/time similarity and location similarity are excluded from LSA similarity calculation.

3.3.3. Linguistic Measures

Linguistic measures use syntactic composition of the sentence or semantic information contained in sentence to determine semantic similarity [8].

Named entity in $ChiS1$ and $ChjS2$ taken into consideration are location and date/time entity. If location entity is overlapped but not identical, location entity similarity score is then $LNESim = 1$. If date/time entity is overlapped but not identical, date/time entity similarity score is then $DNESim = 1$.

Here it is important to understand the reason why I'm not employing for identical named entity; because, it is handled by content word similarity.

3.3.3.1. WordNet based similarity

Statistical word similarity measures have limitations. Related words can have similarity scores only as high as their context overlap (e.g. "doctor" and "hospital"). Also, word similarity is typically low for synonyms having many word senses since information about different senses is mixed together. We reduced the above issues by using additional information from WordNet.

Many WordNet features like path depth, synonym and antonym are used to measure semantic information of a text. We increase the similarity between two words if any relation in the following categories. All of these categories 1-3 are based on linguistic properties on English.

1. One word is the hypernym for the other.
2. One word is the Antonym for the other.
3. One word is the synonym for the other.

Path depth is a function to tell whether a path passes through more general or more specific concepts. The Path depth similarity is calculated between each content word of each chunk in S1 (*ChiS1*) and each content word of each chunk in S2 (*ChjS2*).

To calculate path depth hypernym is one of the hierarchies found in the WordNet to check whether a word is general or specific as well as related. Similarity of two words determined by the depth a word from common parent indexes where two terms join. In addition to calculate similarity hypernym use for labeling pair chunks. Hypernym indicates the relation of two words if *A* is a kind of word *B*, that means *A* is more specific than *B* and *B* is more general than *A*. The strength of similarity depends on their depth that is less depth more similar large depth is less similar. This can be done in two ways one is *A* hypernym *B* and the other is *B* hypernym *A*. Both way has no impact on the depth it is always equal between two words however, the main thing to look two ways Common parent index (*CPI*) can be different and must be known before calculation of hypernym similarity. *CPI* indicates the distance between two words where it joins each other. The summation of *CPI* of *A* hypernym *B* and *B* hypernym *A* give the depth. Based on the relation of depth and *CPI*, five conditions take into consideration in hypernyms similarity for score calculation and type prediction. All condition highlighted in Table-3.1

	<i>A</i> hypernym <i>B</i>		<i>B</i> hypernym <i>A</i>			
Difference	<i>CPI</i>	Depth	<i>CPI</i>	Depth	Type	Score
C1	0	X	X	X	SPE2	$1/\sqrt{x}$
C2	X	X	0	X	SPE1	$1/\sqrt{x}$
C3	$x/2$	X	$x/2$	X	SIMI	$1/\sqrt{(x/2)}$
C4	$< x/2$	X	$> \mathbf{x/2}$	X	-	$\frac{1}{\sqrt{x/2}}$, where $x/2$ is the bold one

C5	$> x/2$	X	$< x/2$	X	-	$\frac{1}{\sqrt{x/2}}$ where $x/2$ is the bold one
-----------	------------------------------	---	---------	---	---	--

Table – 3.1 hypernym score and type

Different conditions compare word A from chunk $ChiS1$ and word B from chunk $ChjS2$ described as follow: In condition $C1$ word A is the parent of word B so the distance of word A from CPI is zero, so word A is parent and the distance of word B from CPI is equal with the depth of their hypernym so word B is child of word A . $C1$ used if word A is more general and word B , type $SPE2$ is given to a chunk. The score calculated by dividing one for word depth.

Condition $C2$ is the inverse of $C1$, word B is the parent of word A so the distance of word B from CPI is zero, so word B is parent and the distance of word A from CPI is equal with the depth of their hypernym so word A is child of Word B . $C2$ taken into account if word A is more specific and type $SPE1$ is given to a chunk. The score calculated by dividing one for word depth

Condition $C3$ is used if and only if the distance of word A from CPI is equal with the distance of word B from CPI . The summation of word A from CPI and word B from CPI gives depth. Considering one is as parent and others as child is not possible and type $SIMI$ is given to a chunk. The score calculated by dividing one for half of the word depth. Condition $C4$ and $C5$ take place when $C1$, $C2$, and $C3$ have not occurred. These conditions fail to predict type and later the type will detected by other feature. Meanwhile the hypernym similarity calculated by dividing 1 for larger CPI . Here we took threshold value for hypernmy at depth 10; meaning if the depth greater than 10 no more involves in this similarity and the similarity can detected by LSA similarity measure.

Another WordNet feature is synonyms; for a word in $ChiS1$, if their synonyms found in $ChjS2$ the synonym similarity ($SynoSim$) is then $SynoSim = 1$. The $SynoSim$ used for handling similarity that cannot handle by string similarity (e.g. N-gram and content word overlap) and named entity similarity.

On other hand antonym is very important to identify a word that is opposite in meaning. For words in $ChiS1$, if their antonyms found in $ChjS2$ the antonyms similarity ($AntoSim$) is then $AntoSim = 1$. The $AntoSim$ mostly used for handling similarity that cannot be handled by (e.g. $POSim$ and Named entity similarity). Also $AntoSim$ used for labeling opposite type.

3.3.4. Feature Extraction for Chunk-to-Chunk Similarity Measures

We can determine similarity between word with word, chunk with chunk and, sentence with sentence etc. But, complete sentence contains more information than just its words or chunks. So to clarify text (sentence) similarity if it is parsed into chunk, score is calculated and relation type is assigned then it became very attractive. This section, describes the features used for calculating chunk similarity. Chunk similarity measured by many features that listed in feature extraction section. Each and every feature depends on word -to-word described above in word and string similarity, corpus based similarity and linguistic similarity. Chunk similarity measured is a combinatorial similarity measure that computes similarity score based on the number of words shared by pair chunks of pair sentence.

The system takes a number of features for type prediction as well as for score calculation. These features employ at chunk level similarity, but to determine chunk similarity it need summation of *F1, F2, F4, F5, F6, F7, F8, F9* and *F10* similarly, feature extraction is used not only for score calculation but also to align chunks and to predict a reason why the chunk is aligned. Likewise, *F0* is also participated in score similarity calculation instead of *F1* in different run.

F0: Dice-coefficient Similarity (lemma of content word)

F1: Common word similarity (*POSim*)

F2: Number similarity

F3: has negation

F4: WordNet synonym

F5: WordNet antonym

F6: WordNet hypernym

F7: Date/time similarity

F8: Location similarity

F9: LSA similarity

Feature extractions are used in various place of ISTS system and the detail is present as follows: *F0* used for calculation of lemmatized content word similarity regardless of the word POS tag. *F1* computed by equation 1 that is to calculate content word overlap the system use the new algorithm we proposed POS

tagged base similarity (*POSim*) for computing of common content word similarity which described in section 3.5 If all content words are common for pair chunk $POSim = 1$, else if some content words are common $POSim$ is between 0 and 1. Finally If there are no word overlap $POSim = 0$, the equation i.

$$POSim = \sqrt{\frac{2(n+v)}{w}} + \sqrt{\frac{adv+adj}{(advt+adjt)}} \log(2(adv + adj)) \dots \dots \dots (1)$$

The most used method to calculate common word similarity up to date is dice-coefficient, cosine similarity, Euclidean distance etc. But, in all case all words given the same weight. However there is a situation in which noun has weighted than adjective and verb than adverb.

n – count of matched noun in pair chunk

v – count of matched verb in pair chunk

adj – count of matched adjective in pair chunk

adv – count of matched adverb in pair chunk

$adjt$ – count of adjective in pair chunk

$advt$ – count of adverb in pair chunk

w – count of all content words in pair chunk

Well, the best way to describe *POSim* would be it gives high weight for noun and verb. To realize term significance of weighting let take simple pair chunk example from Images data. *with a baby in his lap <==> holding a baby*, moreover before computing *POSim* the chunks should tag by Stanford CoreNLP as:

with_IN a_DT baby_NN in_IN his_PRP\$ lap_NN <==> holding_VBG a_DT baby_NN Then content word only select for computation based on their POS tag.

baby_NN lap_NN <==> holding_VBG baby_NN) There is one common noun and no common verb, adjective and adverb. Afterward it is straightforward to compute with (*POSim*) use equation 1 as

$\sqrt{\frac{2(1+0)}{4}} + \sqrt{\frac{0+0}{0}} \log(2(0))$ in deed the adjective and adverb parts become 0 to do so if neither adjective nor adverb matched threshold value is necessary taken the logarithm part substitute by $\log(1)$. The

weighted part that is $\sqrt{\frac{2(1+0)}{4}}$ gives 0.707 to calculate chunk similarity unfortunately this feature (*POSim*)

is enough because all features from $F2$ to $F9$ give zero. Finally it multiplied by 5 then $3.536 \approx 4$ that is equal with the human annotated score given in training dataset.

$F2$ - is regarding a cardinal number whether it is in text form or in digit form. If in text form it is converted into digit. Experimentally from the training dataset the similarity between cardinal number in $ChiS1$ and $ChjS2$ computed as follows:

$$NumSim = 1 - \frac{|n1-n2|}{|n1+n2|} \dots\dots\dots (2)$$

Where $n1$ and $n2$ are number in $ChiS1$ and $ChjS2$ respectively and $NumSim$ is the one minus the difference between $n1$ and $n2$.

If $NumSim$ score is 1, assign similarity score 5 and EQUI type if and only if pair chunks matched by all content words or if the pair chunks contain number only. Otherwise, chunks with digits are aligned to produce alignment label of SIMI. The similarity score is depending on the differences between the digits extracted from the chunks. We have explained the following heuristics:

- If the $NumSim$ is between 0.9 and 0.99 alignment will produce similarity score 4 (e.g., 12 <=> 10).
- If the $NumSim$ is between 0.89 and 0.55, assign similarity 3 (e.g., 18 <=> Ten).
- If the $NumSim$ is between 0.54 and 0.11, assign similarity 2 (e.g., 10 dead <=> 1 dead).
- If the $NumSim$ is between 0.0 and 0.10, assign similarity score 1 (e.g., to 35 years <=> to 1,000 years).
- Else $NumSim$ is 0, which is when either chunk has number or none of chunk has number.

Furthermore the above similarity score multiplied by 5 if pair chunks contain number only. However if pair chunk contains some extra words before multiplying by 5 it should compute other available similarity features then the summation is done. Finally the result multiplied by 5 to get the actual score. On other hand string similarity measures have a limitation to detect semantic similarity computed by $F4$, $F5$ and $F6$ which is depend on WordNet in addition to word and string similarity measure. Furthermore, $F6$ is one of these methods which explained in section linguistic measure. Word-to-word similarity scores calculation formula found in Table 3.1

$$SynoSim = \frac{2*(Schis1 n Wchjs2)}{(Wchis1+ Wchjs2)} \dots\dots\dots (3)$$

$$AntoSim = \left(\frac{2*(A_{chis1} \cap W_{chjs2})}{(W_{chis1} + W_{chjs2})} \right) * 0.8 \dots \dots \dots (4)$$

$$HypeSim = \frac{2*(h_{chis1} \cap h_{chjs2})}{(W_{chis1} + W_{chjs2})} \dots \dots \dots (5)$$

F7 is regarding date/time entity recognized by Stanford NER, if both chunks have date or time, the score is more than 2. Based on their similarity strength the score is given between 2 to 5.

No way to differentiate date similarity even in hypernym all days has the same depth and the month as well. Providing that we used a rule base similarity based on the experiment on training dataset a date in chis1 is aligned with date of chjs2. To determine similarity scores and types we extracts new rules. For day if the day is exactly the same no doubt date similarity (*dtSim*) score is 1, if the day is subsequent the *dtSim* score is 0.8 but if not subsequent the *dtSim* score is 0.6. For a month if the month and day is exactly the same *dtSim* score is 1, when same in month and differ in day similarity determined by day rule. If the month is subsequent the *dtSim* score is 0.6 but if not subsequent the *tSim* score is 0.4. What discussed yet is regarding word to word similarity at word level. At chunk level if some extra words there date similarity calculated by the next equation 6.

$$DateSim = \frac{2*(dtSim)}{W_{chis1} + W_{chjs2}} \dots \dots \dots (6)$$

To put it another way if all content words of pair chunks are different and both chunks contain location F8 not zero; then similarity score given 2. Additionally if some extra words are matched the similarity score is computed with other feature. So at word level location similarity (*Lsim*) = 0.4 to make at chunk level it calculated as the next equation.

$$LocSim = \frac{2*(LSim)}{W_{chis1} + W_{chjs2}} \dots \dots \dots (7)$$

Yet much of similarity has focused on string and linguistic similarity measure. LSA is another important method of the situation when linguistic and string similarity fails to capture similarity. It is the only measure the system used based on corpus. To put it another way if all content words of pair chunks are different and both chunks contain related words F9 result between 0 and 1. Importantly before calculate chunk (*ChiS1* and *Chjs2*) similarity among each word of the pair chunk (*WxChiS1* and *WyChjs2*) if and only if the words are not giving similarity in any feature of F1 to F8 used for feature F9. Where *Wx* represents each word in chunk of S1 and *Wy* represent each word in chunk of S2. That is to say for every word in the chunk it does matrix, then it takes the best score among matrices done.

The step is continued to end until all of these word in *ChiS1* assigns score of term-to-term similarity. The sum of all pairs term of the chunk calculated. Finally the summation is divided by a minimum number of words found in pair chunk. If another feature has similarity score more than 0 F10 result could not determine chunk scores and enforced to add together thus features; then it converted to human similarity score multiplying by 5.

$$LSAsim = \frac{\sum(bestmatch(cosine_similarity(Wx,Wy)))}{\min W} \dots\dots\dots (8)$$

Wx – a word found in chunk *ChiS1* of S1; *Wy* – a word found in chunk *ChjS2* of S2; *cosine_similarity(Wx, Wy)* computed on vector of word *Wx* and *Wy*) that for a single word *Wx* it calculate *cosine_similarity* with all word in chunk *ChjS2*. For a word *Wx* we need one word which is most similar among words in *ChjS2*. To do that *bestmatch(cosine_similarity(Wx, Wy))* select one word for each word in *ChiS1*. That is to say for each word *Wx* in *ChiS1* there is one best match word *Wy* in *ChjS2* and the inverse is true. $\sum bestmatch$ Compute the finally the summation divided by $\min W$ minimum number of words in either chunk.

Before calculating chunk (*ChiS1* and *ChjS2*) similarity we calculate the similarity among each word in the chunk (*WxChiS1* and *WyChjS2*. Where *Wx* represents each word in chunk of S1 and *Wy* represents each word in chunk of S2. Yet we described the computation of each feature involve in chunk similarity measure. Finally the accumulation provides the chunk similarity.

$$ChSim = 5 * (POSim + AntoSim + SynoSim + hypeSim + Datesim + NumSim + LocSim + LSAsim) \dots (9)$$

Where:

ChSim: Chunk similarity

POSim: Content word match

AntoSim: Antonym similarity

SynoSim: Synonym similarity

hypeSim: hypernym similarity

Datesim: date similarity

NumSim: Number similarity

LocSim: Location similarity

LSAsim: LSA similarity

Meanwhile the final score that is *ChSim* have been normalized multiplying by 5, so the score will be between 0 (minimum) and 5(maximum). However in the above equation (9) dice- coefficient is not used. Moreover it taken into account in the second way of chunk similarity calculation depends on lemma of content word. It calculated two times of common word for both *chis1* and *chjs2* divided by count of words in both chunks.

$$DiceSim: = \frac{2*(chis1 \cap chjs2)}{chis1+ chjs2} \dots\dots\dots(10)$$

Algorithm for score calculation

- Input pair of sentence
- Select *chiS1*
- Takes word *x* of chunk *chiS1* and recursively try to match with word *y* of chunk *chjS2*
 - Start looking for *DiceSim / POSim*
 - Succeed calculate similarity score & move to next word in chunk *chiS1*
 - if the chunk has no extra word it move to *ChiS1 ← i=i+1*
 - Fail
 - move looking for *SynoSim/AntoSim*
 - Succeed calculate similarity score & move to next word in chunk *chiS1*
 - if the chunk has no extra word it move to *ChiS1 ← i=i+1*
 - Fail
 - move looking for hypernymy
 - Succeed calculate similarity score & move to next word in chunk *chiS1*
 - if the chunk has no extra word it move to *ChiS1 ← i=i+1*
 - Fail
 - move looking for *DateSim/LocSim*
 - Succeed calculate similarity score & move to next word in chunk *chiS1*
 - if the chunk has no extra word it move to *ChiS1 ← i=i+1*
 - Fail
 - move looking for *NumSim*
 - Succeed calculate similarity score & move to next word in chunk *chiS1*
 - if the chunk has no extra word it move to *ChiS1 ← i=i+1*

- Fail
 - move looking for *LSA_{sim}*
 - Succeed calculate similarity score & move to next word in chunk *chiS1*
 - if the chunk has no extra word it move to $ChiS1 \leftarrow i=i+1$
 - Fail - assign score 0 to word x then it moves to $ChiS1 \leftarrow i=i+1$
 - Reaped until all word of S1 checked for similarity score & calculate at chunk level
- End

3.4. Alignment

The ISTS manages the different inputs of the system, texts in English and with varying length, and uses the chunk similarity model to compute the similarity between the given pieces of text. In order to calculate similarity with different feature the system, align chunks using an *alignment module*. A module depends on several rules of chunk similarity to align pair chunk. Many-to-many (N:M) alignments without restrictions able to explicitly represent all interactions between chunks of pair sentence. For instance, let consider the following pair sentence taken from Images data (#104):

[A yellow and blue airplane]¹ [is flying]² [in the sky]³

[The white airplane]¹ [is flying]² [in the blue sky]³

Alignment of chunks: 1-1 (SIMI 3), 2 - 2 (EQUI 5), 3- 3 (SPE2 4)

Alignment of Chunk	Score	Type
1-1	3	SIMI
2-2	5	EQUI
3-3	4	SPE2

Table – 3.2 example of chunk alignment with score and type

In above example fortunately all of pair chunks are aligned in parallel that is *Ch1S1* with *Ch1S2*, *Ch2S1* with *Ch2S2* and *Ch3S1* with *Ch3S2*. Both *Ch1S1* and *Ch1S2* talking about the same object that is airplane but their difference is on extra information of the object. Furthermore the description indicate that both chunks pointing different airplane one is white another is yellow and blue. *Ch2S1* and *Ch2S2* are pointing about flying and no more information so it is equal. The last pair chunks indicates about sky in which airplane is flying one has extra information of sky color as it is blue; but, from S1 we have no information

whether sky is blue or white. Based on the information we can conclude that *Ch3S1* and is more general than *Ch3S2*. In addition *Ch3S1 and Ch3S2* has very strong similarity.

For the system chunks, the chunk module, converts sentences *S1* and *S2* to chunks *ChiS1* and *ChjS2* respectively. The goal of alignment module is to determine the decision whether chunks of *S1* are similar with chunks of *S2*, (which are non-zero score) or not similar (which is zero). Chunks of *S1* and chunks of *S2* can have more than one chunk (multiple alignment), that is not necessarily contiguous. Aligned chunks are further classified using type prediction and score classifier.

Type prediction module identifies a pair of aligned chunks *ChiS1* and *ChjS2* concatenate with a relation type like EQUI (equivalent), OPPO (opposite), SPE1 (specific), SPE2 (general), SIMI (similar), and REL (related). Score classifier module assigns a similarity score ranging between 0-5 for a pair of chunks.

The alignment module takes the chunks of given pair sentence, then it select first chunk in *S1* (*ChiS1*) and recursively try to match chunk one to each of the chunks in *S2* (*ChjS2*). Alignment contains multiple steps and has precedence to apply for high accuracy. For correct alignment, a sequence has a vital role as Sultan [78].

To overcome the problem of weak alignment it starts matching procedures from EQUI type if fail the procedures continue checking for OPPO type. If OPPO type not matched it moves to matching procedures for SPE1 or SPE2. Unfortunately chunk pair may not similar, or even if similar when the condition is difficult to decide one chunk has more information than other. Then the system moves to matching procedures of SIMI. If the chunk is related but not SIMI it checks for REL. If REL no matched aligns NOALI label to chunk *ChiS1*. The next step is moving to the second chunk of *S2*. It follows same process of first chunk of *S2* until all chunks in *S2* done matrices. At the end if chunk of *S1* not matched NOAL will align with score zero. Similarly a process will continue cross check of each and every chunks of pair sentence. This is exactly what our ISTS system does, as specified in the algorithm below:

- Input one sentence pair
- Select *ChiS1*
 - Recursively try to match *ChiS1* to each of *ChjS2* if Score ≥ 1
 - Start looking for EQUI/OPPO
 - Succeed, move to $ChiS1 \leftarrow i=i+1$
 - Fail
 - Move to looking for SPE1/SPE2
 - Succeed, move to $ChiS1 \leftarrow i=i+1$
 - Fail

- Move to looking for SIMI
 - Succeed, move to $ChiS1 \leftarrow i=i+1$
- Fail
- Move to looking for REL
 - Succeed, move to $ChiS1 \leftarrow i=i+1$
- Fail
- Assign NOALI label to chunk
 - Move to $ChiS1 \leftarrow i=i+1$
 - Repeat until end of $ChiS1 \leftarrow i=n$
- Recursively check if $ChjS2$ aligned with any $ChiS1$
 - Succeed, Move to $ChjS2 \leftarrow j=j+1$
- Fail
 - Assign NOAL
- Repeat until all $ChjS2$ aligned
- End

3.5. Type prediction

Alignment inputs a chunk pair and provides a reason why that chunk pair is aligned. There are seven types of alignment used to provide a reason for a chunk pair and one type (NOALI) for a chunk has no corresponding semantically similar chunk. Their difference is described as next:

- EQUI: semantically both chunks are equivalent.
- OPPO: semantically both chunks are opposite.
- SPE1: semantically both chunks are similar but $ChiS1$ has more information.
- SPE2: semantically both chunks are similar but $ChjS2$ has more information.
- SIMI: similar chunks but no EQUI, OPPO, SPE1 or SPE2.
- REL: related chunks but no SIMI, EQUI, OPPO, SPE1, SPE2.

In this section, we describe the feature and rules used for chunk type. The type given to a pair chunk is defined. We used an alignment reasoning inspired by NeRoSim [12] features (antonyms, synonyms, etc.) and SVCTSTS [79] features (number of words or counts of parts of speech in a chunk pair). Both these systems classify a chunk pair using features extracted from the chunk pair itself. We combine the two approaches and extract a total of 9 syntactic and semantic features for each chunk pair in addition to the proposed *POSim*.

This system uses gold chunks of a given sentence pair and maps chunks of the first sentence to those from the second by assigning different relations and scores based on a set of rules. The system performs stop

word removing, POS tagging, lemmatization, and named-entity recognition in the post processing steps. Lemmatization performed on noun, verb, adjective, and adverb only. It also uses lookups for synonym, antonym and hypernym relations from WordNet.

Type prediction features (F2, F3, F5, F6, F7, F8 and F9) listed under similarity calculator section that predict type of paired chunk in addition to calculate score. Additionally the difference count of each content word of the chunk can determine chunk label.

- Count of noun in chunk *ChiS1*
- Count of noun in chunk *ChjS2*
- Count of adjective in chunk *ChiS1*
- Count of adjective in chunk *ChjS2*
- Count of verb in chunk *ChiS1*
- Count of verb in chunk *ChjS2*
- Count of adverb in chunk *ChiS1*
- Count of adverb in chunk *ChjS2*

Next, we define a set of rules for each relation type. What we get from gold standard datasets indicates that having adjective in either chunks (*ChiS1* or *ChjS2*) or different number of adjectives in pair chunks decide type of alignment. Particularly for SPE1 and SPE2 type adjective has vital role for labeling.

E.g. – a bus <==> red double decker bus. The first chunk has no adjective but, the second chunk have adjective. So the first chunk is general because has no adjective and the second chunk have adjective that indicates specific.

In addition a binary feature was designed to indicate whether two chunks in a given pair have the same polarity (i.e., affirmative or negative) by looking up a manually-collected negation list with 20 negation words (e.g. no, never, etc.). Negation list is mostly used for labeling opposite type.

3.5.1. EQUI Rules

EQUI Rules are applied when score of a chunk is 5.

EQUE1 - Both chunks have same tokens - e.g. Resigns <==> resigns

EQUE2 - Both chunks have same content words- e.g. in Bethlehem <==> of Bethlehem

EQUE3 - A content words match using synonym lookup - e.g. Syrian troops <==> Syrian army

EQUE4 - All content words of a chunk match but proper noun type un-match the other chunk - e.g. Boeing 787 Dream-liner \Leftrightarrow on 787 Dreamliner

EQUE5 - Both chunks have equal number of content words ($ChiS1, ChjS2$) > 0.9 - e.g. in Indonesia boat sinking \Leftrightarrow in Indonesia boat capsized

EQUI6 - Both chunks has *LSASim* score > 0.8 - e.g. restored \Leftrightarrow resumes

3.5.2. OPPO Rules

OPPO rules are applied only when none of chunk has number and conjunction.

OPPO1: A content word in a chunk has an antonym in the other chunk - e.g. S. Korea \Leftrightarrow North Korea.

OPPO2: if either chunk has negation - e.g. Ethiopian \Leftrightarrow not Ethiopian

3.5.3. SPE1, SPE2 Rules

SP1: If chunk $ChiS1$ but $ChjS2$ has a conjunction and $ChiS1$ contains all the content words of $ChjS2$ then $ChiS1$ is SPE2 of $ChjS2$ or vice-versa. (4) - e.g. A motorcycle \Leftrightarrow A silver and blue motorcycle

SP2: If chunk $ChiS1$ contains all content words of chunk $ChjS2$ and some extra content words that are not verbs, $ChjS2$ is a SPE2 of $ChiS1$ or vice-versa. (4) - e.g. A white Apple computer \Leftrightarrow An Apple computer

SP3: If chunks $ChiS1$ and $ChjS2$ contain only one noun each say $n1$ and $n2$, $n1$ is hypernym of $n2$, $ChjS2$ is SPE1 of $ChiS1$ or vice versa (4) - e.g. A white dog \Leftrightarrow A white animal

3.5.4. SIMI Rules

According to SIMI relation type the most similar chunks has scoring 4 and the least one is scoring 1. A SIMI rule applied on all chunks pair contains a token of DATE/TIME type only or of LOCATION type only. Additionally when both chunks share at least one noun similarity type is determined by LSA score.

SIMI1: Only the un-matched word in each chunk is a CD type the similarity is determined by CD type number difference (4)-e.g. 6 March 2013 \Leftrightarrow 12 March 2013

SIMI2: Each chunk has a token of DATE/TIME type and it contains day without month similarity chunk score is 4 or 3. If both days are consecutive score given to a chunk is 4 - e.g. for Monday \Leftrightarrow for Tuesday
But if the days are not consecutive score given to a chunk is 3 - e.g. Monday \Leftrightarrow Thursday

SIMI3: Each chunk has a token of LOCATION type (2) - e.g. in Iraq <==> in Syria

SIMI4: When both chunks share at least one noun then assign 3 if $ChSim(ChiS1, ChS2) \geq 0.5$ and 2 otherwise if $ChSim(ChiS1, ChjS2) \geq 0.4$. -e.g. Nato troops \Leftrightarrow NATO strike

SIMI5: Each chunk contains a CD type only the similarity is determined by number difference and Type is SIMI (4)-e.g. 6 <==> 12

SIMI6: Each chunk has a token of DATE/TIME type and both contains month, chunk similarity score is 3 or 2. If both months are consecutive score given to a chunk is 3 - e.g. November 25, 2013 <==> October 8, 2013. But if a months are not consecutive score given to a chunk is 2 - e.g. 7 August 2013 <==> 11 April 2013.

3.5.5. REL Rules

RE1: If both chunks share at least one content word then assign REL relation. However scores are assigned based on LSAsim similarity as follow:

(i) 4 if $LSAsim(ChiS1, ChjS2) \Leftrightarrow [0.7, 0.9]$

(ii) 3 if $LSAsim(ChiS1, ChjS2) \Leftrightarrow [0.5, 0.7]$

(iii) 2 if $LSAsim(ChiS1, ChjS2) \Leftrightarrow [0.40, 0.5]$

REL // 4 // in Afghanistan <==> in Afghan attack

REL // 3 // in front of graffiti <==> in front of the building

REL // 2 // Chinese general <==> of Chinese army singers

3.5.6. NOALIC Rules

NOALIC: If a chunk does not get any relation after applying all the rules and similarity calculation is equal zero, the chunk to be mapped, NOALIC.

NOALIC2: If a chunk in a S1 $ChiS1$ is aligned with more than one chunk in a S2 $ChjS2$. that is already aligned and has $ChSim(ChiS1, ChS2) \leq 2$, assign NOALIC relation to $ChiS1$ with NIL.

3.6. Score Classification

It is important to understand the variety of options available when using an algorithm, as they can make a significant difference in the quality of results. For each chunk aligned similarity score was calculated. The scoring module uses a variety of features listed under section feature extracted such as: F1, F2, F4, F5, F6, F7, F8, and F9. These entire features added together to decide pair chunk whether

similar or not. To that end, score for pair chunk is already described in section 3.8 beside to type classification which is calculated by ChSim (equation 9)

3.7. Evaluation of ISTS

The system evaluated using the official scorer provided by task organizers (SemEval) for evaluation of ISTS, which computes four distinct metrics: *ALI* (segment pair alignment correctness), *Type* (segment pair alignment correctness taking type into account), *Score* (segment pair alignment correctness taking score into account) and *Type + Score* (segment pair alignment correctness taking type and score into account).

3.8. Tools

3.8.1. Java

We used Java as programming language because of its modularity, robustness, scalability and high availability of libraries and tools for development. Java is an object oriented programming language we used for preprocessing training datasets and Wiki-corpus, chunking, post processing, extracting similarity from WordNet, calculating similarity, aligning chunk, scoring and labeling as well as final result output. Generally all algorithm used in this ISTS system were written in java on NetBeans 8.1

3.8.2. JAMA

JAMA is a basic linear algebra package for Java. It provides user-level classes for constructing and manipulating matrices. It is meant to provide sufficient functionality for routine problems, packaged in a way that is natural and understandable. In this thesis we used the JAMA library, and works fine as is, although we used NetBeans for adding the path to the JAMA library `Jama-1.0.3.jar` and running code. JAMA library uses to do the SVD decomposition matrix for computing corpus based similarity with LSA.

3.8.3. WordNet API

WordNet is a lexical database which is freely available to download and provides a large repository for English lexical items. WordNet was designed to establish the connections between four types of Parts of Speech (POS) - noun, verb, adjective, and adverb. For this purpose window platform WordNet version 2.1 is used.

CHAPTER FOUR

EXPERIMENTATION AND DISCUSSION

4.1. Introduction

This chapter is devoted to the experimentation and method used for the evaluation of the study. It discusses train and test dataset used for the experimentation. The test result of the findings of the study is also discussed in this chapter. A brief analysis of the experimentation result is also presented.

The ISTS system needs to perform chunk, align the chunks, label and assign score to the alignments. To access WordNet dictionary we used an application programming interface (API). We chose Java WordNet Library because of its easy configuration through properties file and its speed. *MapBackedDictionary* that requires a map representation of WordNet dictionary was used.

net.didion.jwnl.utilities.DictionaryToMaptargetFolder properties.xml

we use *MapBackedDictionary* representation and the properties file look like following truncated example.

```
<?xml version="1.0" encoding="UTF-8"?>
<jwnl_properties language="en">
<version publisher="Princeton" number="2.1" language="en"/>
<dictionary class="net.didion.jwnl.dictionary.MapBackedDictionary">
...
<param name="file_type" value="net.didion.jwnl.princeton.file.PrincetonObjectDictionaryFile"/>
<param name="dictionary_path" value="c:/program files/WordNet/2.1Map"/>
</dictionary>
<resource class="PrincetonResource"/>
</jwnl_properties>
```

4.2. Datasets

This section presents dataset used for this study. We first introduce the annotation procedure, followed by the source of the sentence pairs, the evaluation method, and finally inter-tagger annotation data. In this work dataset first, a pair of sentences is given. Second, the annotator identifies the chunk in each sentence, despite of the corresponding sentence in the pair. Third, for each alignment, the annotator calculates a

score which indicate similarity of chunks pair. Fourth, the annotator aligns the chunks if score is greater than one or equal one. The sequence is from the strongest correspondences to weakest ones. Finally, it chooses the type of label for each alignment which shows a reason why it aligned.

Source of the dataset

The dataset comprises pairs of sentences from Headlines news and image descriptions. We have mentioned a sample pair from Images in table 4.1 with their Alignment, Score and Type. The Headlines corpus is composed of naturally occurring news headlines whereas the Images dataset consist of images with description. The dataset comprised 756 and 750 sentence pairs from Headlines and Images, respectively. Headlines contain slightly less chunks and few tokens per chunk than image descriptions. More than half of the pairs aligned in both data have a score of 5 and Type EQUI. In other way EQUI is the most used Type, followed by SIMI, SPE2 and SPE1, REL and OPPO respectively. The analysis in scores and types is very similar in both datasets. Additionally there are a large number of unaligned chunks, especially in the Images dataset. Finally, FACT and POL are infrequently used in the headlines dataset, and never in the Images dataset. We excluded FACT and POL in our work.

Training dataset and test dataset

Two publicly-available pair sentence datasets (*Images* and *Headlines*) used to training and to evaluate the performance of the interpretable sentence similarity measures. Headlines dataset consist of 756 pairs of sentences used for training and 375 sentence pairs used for test. On the other hand Images dataset consist of 750 pairs of sentences used for training and 375 sentence pairs used for test.

Our experiments show that there are significant differences in annotations between datasets. Particularly in image datasets sometimes the parser parse article as a phrase especially when a sentence begins by article and the next phrase is noun phrase. So it need merge with next phrase to make a correct noun phrase. In headline datasets space separated punctuation like colon (:), hyphen (-) and double hyphen (--) are where one chunk ends and another chunk starts. Verbs are sometimes as chunk, and 'to' and 's' often start a new chunk in headline.

4.3. Evaluation

We have been prepared two system named Run1 and Run2. Both runs are used the same training data that is Headlines and Images as well as the same algorithm for score calculation except one algorithm which makes vary. However in both run exactly the same type prediction algorithm with the same precedence is

used. The algorithm is explained in section 4.5 with detail of alignment module. Run1 calculates score by equation 9 in contrast Run2 calculated by equation 11. The main difference is in Run1 *POSim* used for content word similarity. In comparison in Run2 *POSim* substituted by *diceSim* but, all other similarity measures are taken as it is. So Run2 calculated as the next equation (11).

$$\text{ChSim} = 5 * (\text{diceSim} + \text{AntoSim} + \text{SynoSim} + \text{hypeSim} + \text{Datesim} + \text{NumSim} + \text{LocSim} + \text{LSAsim}) \dots (11)$$

Furthermore, both runs consider the same similarity measures between two words, but chunk level similarity measure is different. To make very clear we took sample train data from the Images that annotated by human along cosine, dice-coefficient and the new proposed (*POSim*) chunk similarity score. The comparison of sample chunk pair are highlighted in table 4.1, to clarify more in the table all three similarity measures calculated from one. To put it in another way if chunk pair exactly match the result is one also if nothing is match the result is zero. To conclude the similarity result will be between zero and one. In table 4.1 except human annotated similarity (from training Dataset) all have three similarity scores. The first one is normal (non-*italic* and non-**bold**) result which indicates the similarity score directly taken from the output of an algorithm before normalization (multiplying by five) done. The other is the bold score which indicates multiplication of normal result by five as it is. The final one is an *italic* that is an approximate of the bold result. The human annotated similarity measure was represented by single digit, so we have taken approximate value to make a single digit.

Sample pair chunks from sentence1 and sentence2	Data set	cosine	Dice	POSim
lush green field <==> green field <i>in_IN a_DT lush_JJ green_JJ field_NN</i> <i>in_IN a_DT green_JJ field_NN</i>	4	0.82	0.80	0.84
		4.1	4.00	4.15
		4	4	4
legitimate representative <==> sole representative <i>as_IN legitimate_JJ representative_NN</i> <i>as_IN sole_JJ representative_NN</i>	4	0.50	0.50	0.70
		2.5	2.5	3.5
		3	3	4
A dark brown horse <==> A brown horse <i>dark_JJ brown_JJ horse_NN</i> <i>a_DT brown_JJ horse_NN</i>	4	0.82	0.8	0.84
		4.1	4.0	4.15
		4	4	4
bitter immigration debate <==> immigration debate <i>bitter_JJ immigration_NN debate_NN</i> <i>immigration_NN debate_NN</i>	4	0.82	0.86	0.89
		4.1	4.3	4.45
		4	4	4
A cat <==> A black and white cat <i>a_DT cat_NN</i> <i>a_DT black_JJ and_CC white_JJ cat_NN</i>	4	0.58	0.5	0.70
		2.9	2.5	3.5
		3	3	4
with a big necklace <==> with a black top and a necklace <i>with_IN a_DT big_JJ necklace_NN</i> <i>with_IN a_DT black_JJ top_NN and_CC a_DT necklace_NN</i>	3	0.61	0.60	0.63
		3.05	3.0	3.15
		3	3	3
A passenger train <==> A passenger train <i>a_DT passenger_NN train_NN</i> <i>a_DT passenger_NN train_NN</i>	5	1.0	1.0	1.0
		5	5	5
		5	5	5
a brown horse <==> a brown horse <i>a_DT brown_JJ horse_NN</i> <i>a_DT brown_JJ horse_NN</i>	5	1.0	1.0	1.0
		5	5	5
		5	5	5
with a baby in his lap <==> holding a baby <i>with_IN a_DT baby_NN in_IN his_PRP\$ lap_NN</i> <i>holding_VBG a_DT baby_NN</i>	4	0.47	0.5	0.70
		3.33	2.5	3.5
		3	3	4
A yellow and blue airplane <==> The white airplane <i>A_DT yellow_JJ and_CC blue_JJ airplane_NN</i> <i>The_DT white_JJ airplane_NN</i>	3	0.41	0.4	0.63
		2.05	2.0	3.15
		2	2	3
a bus <==> Red double decker bus <i>a_DT bus_NN</i> <i>Red_JJ double_JJ decker_JJ bus_NN</i>	3	0.45	0.33	0.63
		2.25	1.66	3.15
		2	2	3

Table – 4.1 POSim caparison with human annotated, dice – coefficient and cosine

If number of content words with POS tagged and without tag is equal the similarity measured by POS but if not equal POS tag measurement fails to compute. At the worst case it is calculated by dice-coefficient. Let take two chunks from training dataset “*A yellow and blue airplane <==> The white airplane*” the chunk is taken from sentence one and sentence two respectively. According to human annotation of training data the chunks labeled and scored as:

SIMI // 3 // *A yellow and blue airplane <==> The white airplane*. To boost this similarity calculation we used POS tagged by Stanford POS tagger.

A_DT yellow_JJ and_CC blue_JJ airplane_NN <==> The_DT white_JJ airplane_NN. Any words except noun, verb, adjective and adverb should remove from the chunks. Hence determine the calculation before the words tagged gives 0.26 and 0.25 by cosine and Dice-coefficient respectively similarity score. The result multiplied by 5 as a reason the dataset was give score between 0 and 5. Finally cosine and Dice-coefficient score is 1.3 and 1.25 respectively so it is far from annotated similarity score. To boost this similarity calculation we used POS tagged by Stanford tagger.

A_DT yellow_JJ and_CC blue_JJ airplane_NN <==> The_DT white_JJ airplane_NN. any words except noun, verb, adjective and adverb should remove from the chunks.

Hence a determiner (*A, The*) and conjunction (*and*) removed from the chunks, five words remain in the chunk (*yellow_JJ blue_JJ airplane_NN <==> white_JJ airplane_NN*). Therefore POS tag content word based similarity gives high weight to noun and verb inversely gives low weight to adjective and adverb. When it computed with

$\sqrt{\frac{2(n+v)}{w}} + \sqrt{\frac{adv+adj}{(adv+adj)}} \log(2(adv + adj))$ we gave 1 threshold if there is no common adjective or an adverb with a chunk *adv + adj* gives zero result, $\log(0)$ gives an error. To overcome this problem we take 1 instead of 0, it does no effect on the final results because $\log(0 + 1)$ gives 0. There is no an unmatched noun or verbs $\sqrt{0.4}$ that is 0.632. If all content words are just matched similarity score is 1 (maximum) and when there is no matched word in a chunk similarity score is 0 (minimum). The result propagates between 1 and 0. According to ISTS the score must be between 0 (un-matched) and five (equivalent matched). Finally the result gained from *POSim* similarity multiplied by five gives 3.16 so it approximate to 3 that is exactly equal to human annotation.

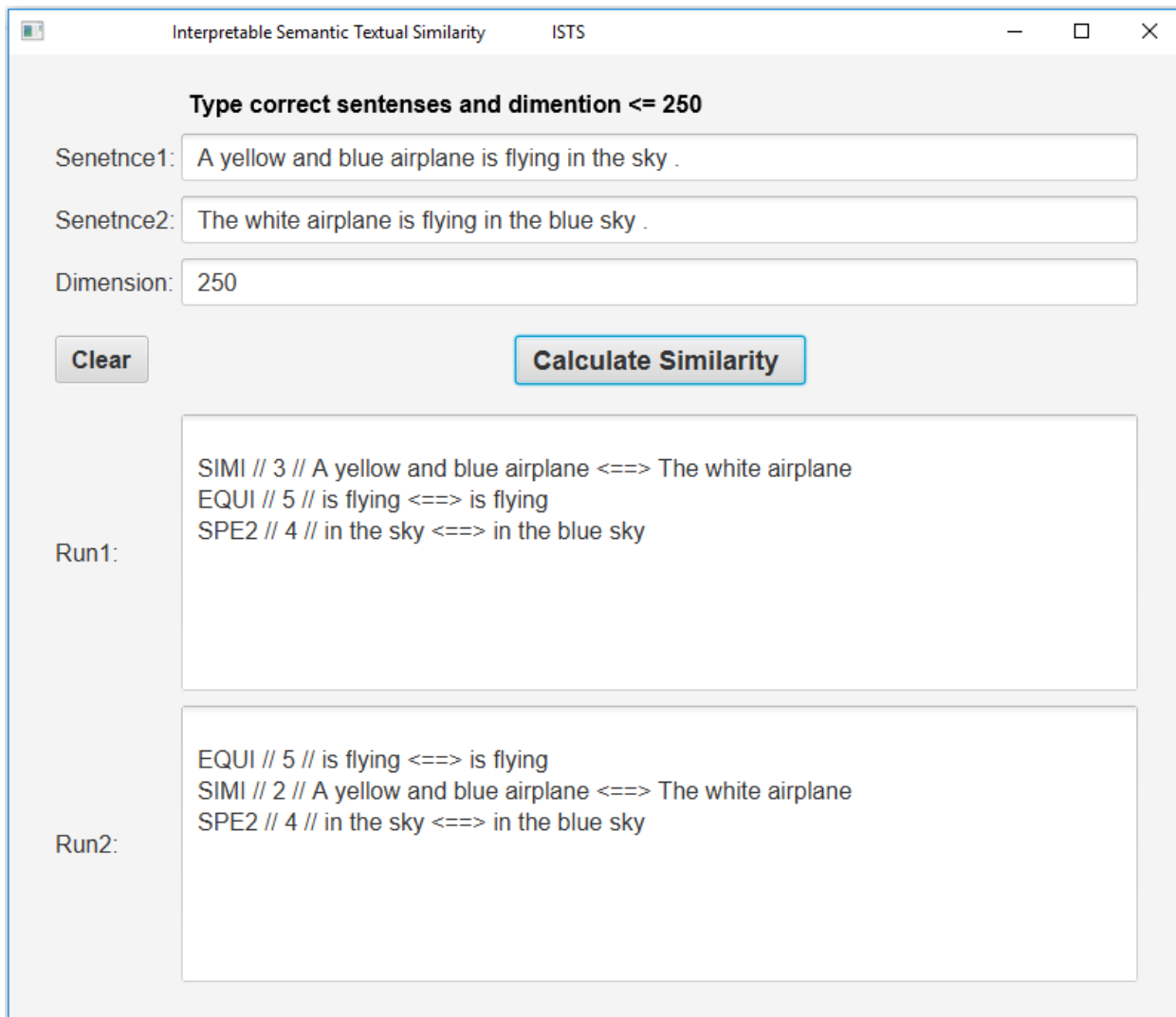


Figure – 4.1 sample output of images dataset

As you see on Figure 4.2 sample output taken from Headlines data first pair chunk contain numbers only, so alignment reasoning must be SIMI also score calculated by NumSim which is 2. The second pair chunk score is 5 it shows as alignment label is EQUI, because it is exactly equal. The third pair chunk is detected by location similarity and given rule based score 2 alignment reason is SIMI. The last pair explaining about the same thing but, the degree of similarity is not equivalent crash more specific than accident.

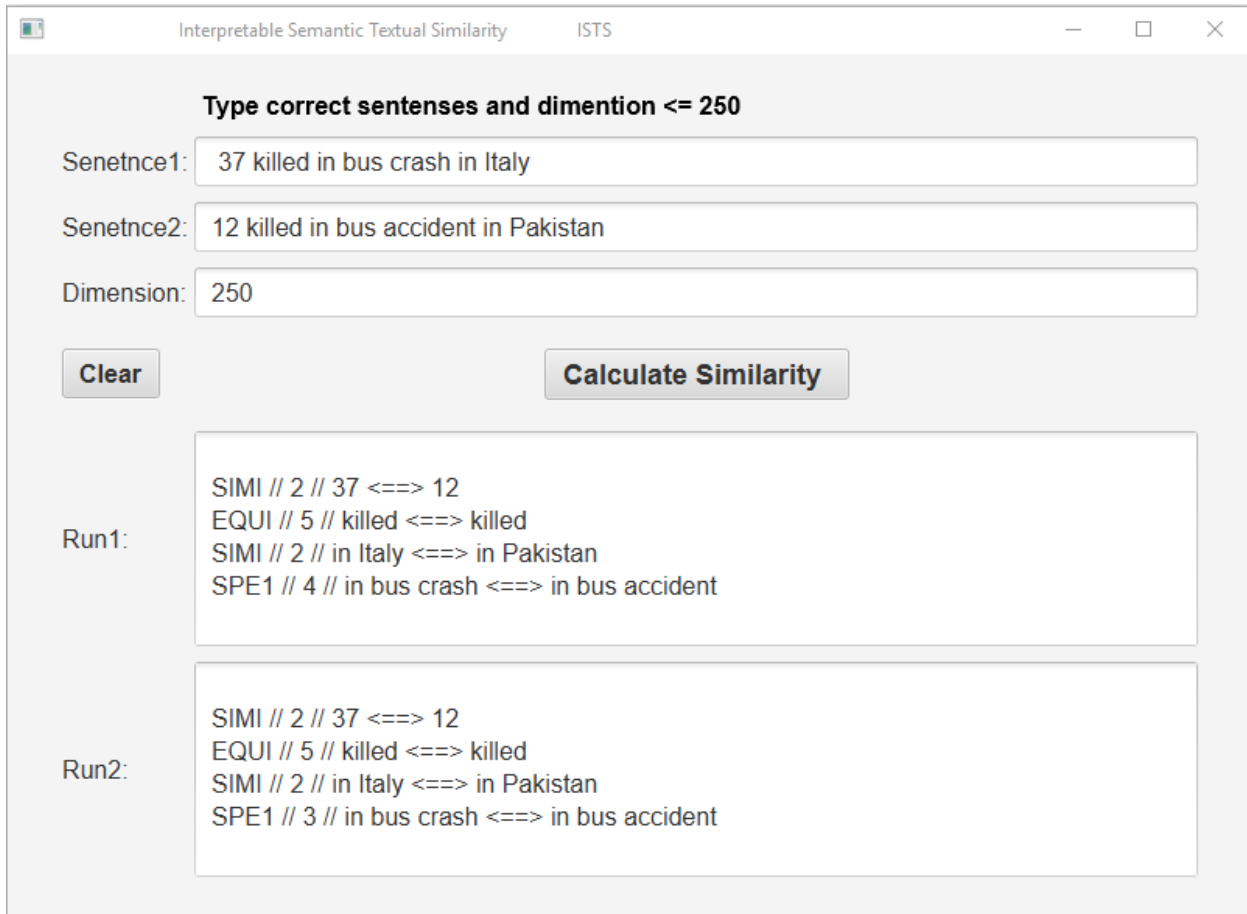


Figure – 4.2 sample output of Headline dataset

4.4. ISTS results

The system evaluation were computed by F1 score based on alignments *ALI* (chunk pair alignment correctness), *F1 Type* (chunk pair alignment correctness of type), *F1 Score* (chunk pair alignment correctness of score) and *F1* full consideration of alignment, type, and score are represented as *Type + Score* (pair chunk alignment correctness of type and score). In order to evaluate systems which perform ISTS, the segment align is mapped into chunk alignment, whereas all chunk pairs in the aligned pairs are with some weight.

We have evaluated the chunking accuracy of SPBC chunker by comparing its output against the gold chunks of ISTS 2016 training data: the training and test data sets each consist of 375 pairs of Images annotation data and 375 pairs of Headlines texts. The chunker yielded the highest average accuracies on both the training and test datasets compared to other chunkers which are described next. The accuracies on the training dataset were 89.20% and 87.34% at chunk level and sentence level respectively. For the

test dataset, the accuracies were 87.81% and 87% at chunk and sentence level, respectively. The result of ourRun1 and Run2 of the evaluation is shown in Table-4.2 and Table-4.3 for Images and Headlines dataset respectively with given baseline.

Image					
	ALI	Type	Score	Type+Score	Rank
Baseline	0.8556	0.4799	0.7456	0.4799	4
Run1	0.8734	0.7723	0.8512	0.7514	1
Run 2	0.8424	0.7262	0.7533	0.6611	3
UWB_run3	0.8922	0.6867	0.8408	0.6708	2

Table – 4.2 images dataset result

Headlines					
	ALI	Type	Score	Type+Score	Rank
Baseline	0.8462	0.5462	0.761	0.5461	4
Run1	0.8512	0.7232	0.7612	0.7120	1
Run 2	0.7942	0.7232	0.7365	0.660	3
Inspire_run1	0.8194	0.7031	0.7865	0.696	2

Table – 4.3 headline dataset result

The ISTS task published results briefly on the organization’s website⁵. They evaluated 9 systems including baseline.

On Table 4.2 and Table 4.3 are shown the results of the ISTS task in comparison to our best result (POSim Overlap Measure with dice-coefficient). The rank was awarded according to correlation for Interpretable STS. In comparison to the baseline our results are better on both Headlines and Images data sets. On

⁵ <http://alt.qcri.org/semEval2016/task2/index.php?id=results>

Images data set our best (Run1) results is 0.7514. Similarly on Headlines data our best (Run1) result is 0.7120 and it is better than the baseline.

We computed the percentage difference of our best result in comparison to the winner and baseline. As you can see our result for Images is 8.02% better than the result of the winner. Likewise, our result for Headlines is only 1.6% better than the result of the winner. Our similarity measure is better than the baseline by 27.15% on the Images data and by 16.59% on the Headlines data.

4.5. Limitation and Challenges

Limitation of this research depends on the limitation of chunking algorithm. At the back of a chunking algorithm there is Stanford CoreNLP parser. That is to say, the output of parses is accepted by chunker as input. The accuracy of parser is well but not perfect. For insistence sometimes the parser split article as a phrase especially when a sentence begins by article and the next phrase is noun phrase. So it directly affects the accuracy of chunk as a result if it is not segmented correctly the final output also not accurate.

In addition, there are some limitations we realized in training dataset that degrade the result of ISTS. With this in mind let take one sentence *A young woman with a bracelet is wearing a bikini top and jeans* (#103 in image data), human annotated chunk parsed the sentence into four as: [*A young woman*]¹ [*with a bracelet*]² [*is wearing*]³ [*a bikini top and jeans*]⁴. Yet it has no problem however the human annotated label alignment concatenates two non-consecutive chunks (2 and 4) as: *with a bracelet a bikini top and jeans* which is illogical and degrade score as well as label.

In addition to chunking problem the difficulty of the ISTS task of aligning the chunks and allocating relation types, we found some differences in annotation of human which made some errors. In image data (#65), for example, *on a sofa* <=> *on a blue sofa* the annotation type is SIMI but the SPE2 type best describes the relation of the chunks. Likewise, in the same data (#693) *in a field* <=> *in a green field*, the SPE1 type has been labeled in the training set but it should be SPE2. Similarly, in image data (#193) *A young boy* <=> *A young blonde girl* has been given a label SPE2 in the training set. Indeed the second chunk has more information; however these two chunks are referring to different object and actually difficult to decide which one is more general. Therefore it should be SIMI because it is like comparing mango and apple fruit. This isn't the only evidence that supports the challenges of the training dataset.

CHAPTER FIVE

CONCLUSION AND RECOMMENDATIONS

5.1. Introduction

The results of the study are summarized in this chapter. Moreover, issues that should be done in the future to enhance the ISTS result which also improves the ISTS are presented.

5.2. Conclusion

ISTS system helps the users to take pair sentence and give the output in chunk with similarity score and reason of it aligned. In this study, ISTS that is based on similarity like: string/word, WordNet and corpus-based approach were developed for English users to specify their degree and type of similarity. In addition some rule were used for type prediction and score calculation. All chunk alignment, similarity score and interpretation of similarity score depend on the 10 features. Chunk alignment depends on similarity score, it hasn't its own feature. Totally we implemented 10 similarity measures including a novel similarity measures, for two systems named as Run1 and Run2. However we used 9 measures at a time for Run1 also for Run2 to calculate score. Both systems has 8 common features and one difference feature that is Run1 used *POSim* and Run2 used *DiceSim*. So in score calculation all features are participated except *has negation* feature. For type prediction (interpretation) we used 7 feature (*NumSim*, *has negation*, *AntoSim*, *hypeSim*, *DateSim*, *LocSim*, and *LSAsim*). But, *DiceSim*, *POSim*, *SynoSim* features are not participated in type prediction. Moreover there are five novel algorithms proposed for similarity score calculation and interpretation such as: *POSim*, *HypeSim*, *DateSim*, *NumSim*, and *AntoSim*

Among an introduced novel algorithm *POSim* has great role for chunk-to-chunk similarity calculation based on POS tagged content word gives high and equal weight for verb and noun In contrast it gives less and equal weight for adverb and adjective. Additionally *POSim* algorithm can use for any language as long as an adjective help to modifies (limits or describes) a noun or a pronoun as well as an adverb help to modifies a verb, an adjective. So giving a weight for noun and verb is very significant rather than considering as the same weight with adjective and adverb. The algorithm only not enough to determine chunk similarity because, it focus is surface similarity (word overlap). So it boosts the ISTS by using three similarity features such as: synonym, antonym as well as hypernym from WordNet and LSA similarity demonstrated in chapter 4.

Hypernym similarity (*HypeSim*) also boosted the performance the ISTS system. This similarity measure has a great potential to predict alignment reason as well as compute similarity score. To avoid incorrect weak alignment the threshold value 10 taken as the highest distance between indexes of both word.

Another important algorithm proposed for date entity similarity (*DateSim*) is very useful, because all days are on the same hierarchy in WordNet and the months are similarly on same from their parent, so difficult to differentiate. Named entity recognizer recognize as it is date only, like WordNet difficult to differentiate. As described in section 3.4.4 *DateSim* used to differentiate similarity score.

Number similarity (*NumSim*) has a great role on score calculation and type prediction because it detects similarity between pair chunk that couldn't handle by any other features like *LSAsim*, string similarity and WordNet based similarity feature.

Finally *AntoSim* is another significant algorithm to identify an oppositeness of aligned chunks. It's a kind of dice coefficient works for antonym only.

Performance of systems using corpus based approach is highly affected by the size, reliability and correctness of the corpus used for the study [106]. The ISTS used Wikipedia corpus for LSA similarity. Experimentation shows that LSA similarity was relatively less reliable than WordNet and String/word similarity score.

However, the size of the documents used for this research was limited which affected the level of performance to be achieved. Despite of the limitation, it can be concluded that the performance of the system obtained was promising and gives a best result.

5.3. Recommendations

It is believed that there is much room for improvement of the performance of ISTS system developed in this research. Therefore, the following recommendations should be looked at in the future so that effective ISTS system can be developed to help users in their similarity need:

- The size of the documents used for this research was limited. This limitation affected the accuracy of chunk alignment as well as degree of similarity; because if resources used are small, we may not be able to generate all word co-occurrence. Therefore, some work should be done with large and high quality corpora to minimize these problems.

- This thesis work focuses on chunk to chunk similarity as well as *POSim* algorithm proposed for this purpose. Moreover we recommend that *POSim* algorithm if somebody who interest to do sentence to sentence similarity without chunking a sentences.
- For local language yet there are no work done on semantic textual similarity and its interpretation, unfortunately no one brave to do this. They are many challenges, the first one is developing training and testing dataset need linguistic. Second for a local language we haven't knowledge based lexical database like wordNet. The door is still open for someone who want to contribute for user of local language.

References

- [1] V. Hatzivassiloglou et al. “Detecting text similarity over short pas-sages: Exploring linguistic feature combinations via machine learning”. *In Proceedings of the Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, 1999, pp.203–212
- [2] R. Jackendoff. “Semantics and Cognition”, MIT Press, Cambridge, MA, 1983
- [3] R. Rada, et al. “Development and Application of a Metric on Semantic Nets”, *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 19, No. 1, 17-30.
- [4] V. Rus. “Opportunities and Challenges in Semantic Similarity”, *Proceedings of the Twenty-Seventh International Florida Artificial Intelligence Research Society Conference*, pp. 208–213, 2014.
- [5] E. Agirre et al. “SemEval-2012 Task 6: A Pilot on Semantic Textual Similarity”, *First Joint Conference on Lexical and Computational Semantics (*SEM)*, pp 385–393, Montreal, Canada, June 7-8, 2012.
- [6] E. Agirre et al. “SemEval-2015 Task 2: Semantic Textual Similarity, English, Spanish and Pilot on Interpretability”, *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, Denver, Colorado, pp 252–263, June 4-5, 2015
- [7] V. Rus et al. “Assessing Student Paraphrases Using Lexical Semantics and Word Weighting”, *In Proceedings of the 14th International Conference on Artificial Intelligence in Education*, Brighton, UK. 2009.
- [8] I. Androutsopoulos & P. Malakasiotis. “A survey of paraphrasing and textual entailment methods”, *Journal of Artificial Intelligence Research*, 38:135-187, 2010

- [9] S. Fernando & M. Stevenson, “A semantic similarity approach to paraphrase detection”, Computational Linguistics UK (CLUK 2008) 11th Annual Research Colloquium, 2008
- [10] C. Corley & R. Mihalcea, “Measuring the Semantic Similarity of Texts”, *In Proceedings of the ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment*, Ann Arbor, MI, 2005.
- [11] V. Rus et al. “Semilar: The semantic similarity toolkit”, *In Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, 2013.
- [12] R. Banjade et al., “NeRoSim: A System for Measuring and Interpreting Semantic Textual Similarity”, *In Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pp. 164–171, Denver, Colorado, June 4-5, 2015.
- [13] E. Agirre et al. “SEM 2013 shared task: Semantic Textual Similarity”, *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task*, pp.32–43, Atlanta, Georgia, June 13-14, 2013
- [14] M. R. Aliguliyev. “A new sentence similarity measure and sentence based extractive technique for automatic text summarization”, *Expert Systems with Applications*, 36(4):7764–7772, 2009.
- [15] V. Rus & C. Graesser, “Deeper Natural Language Processing for Evaluating Student Answers in Intelligent Tutoring Systems”, *Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI-06)*, (2006).
- [16] P.M. McCarthy & D.S. McNamara, “User-Language Paraphrase Corpus Challenge”, online, 2008.

- [17] V. Rus & M. Lintea, “A comparison of greedy and optimal assessment of natural language student input using word-to-word similarity metrics”, *In Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*, pp. 157-162, 2012
- [18] http://trac.research.cc.gatech.edu/ccl/export/158/SecondMindProject/SM/SM.WordNet/Paper/WordNetDotNet_Semantic_Similarity.pdf
- [19] E. Agirre et al. “SemEval-2014 Task 10: Multilingual semantic textual similarity”, *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pp. 81–91, Dublin, Ireland, August 23-24, 2014
- [20] E. Agirre, “SemEval-2015 Task 2: Semantic Textual Similarity, English, Spanish and Pilot on Interpretability”, *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pp. 252–263, Denver, Colorado, June 4-5, 2015. ©2015 Association for Computational Linguistics
- [21] W. H. Gomaa, A. A. Fahmy, “A Survey of Text Similarity Approaches”, *International Journal of Computer Applications (0975 – 8887)*, Volume 68– No.13, April 2013
- [22] D. Gusfield, “Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology”, Cambridge University Press, 1997.
- [23] L. Allison, & T. I. Dix, “A bit-string longest-common-subsequence algorithm”, *Information Processing Letters*, 23:305–310, 1986.
- [24] M. J. Wise, “YAP3: Improved detection of similarities in computer program and other texts”, *In Proceedings of the 27th SIGCSE technical symposium on computer science education*, pp. 130–134, Philadelphia, PA, USA, 1996.

- [25] W. W. Cohen, P. Ravikumar, & S. E. Fienberg, “A Comparison of String Distance Metrics for Name-Matching Tasks”, *In Proceedings of the IJCAI Workshop on Information Integration on the Web*, pp. 73–78, Acapulco, Mexico, 2003b.
- [27] M. A. Jaro, “Probabilistic linkage of large public health data file”, *Statistics in Medicine*, 491-8, 1995.
- [28] W. E. Winkler, “String Comparator Metrics and Enhanced Decision Rules in the Fellegi-Sunter Model of Record Linkage”, *In Proceedings of the Survey Research Methods Section*, pp. 354–359, 1990.
- [29] V. I. Levenshtein, “Binary codes capable of correcting deletions, insertions, and reversals”, *Soviet Physics Doklady*, 10(8):707–710, 1966.
- [30] P. A. Hall, & G. R. Dowling, “Approximate string matching”, *Comput. Surveys*, 12:381-402, 1980.
- [32] V. Keselj, et al. “N-gram-based author profiles for authorship attribution”, *In Proceedings of the Conference of the Pacific Association for Computational Linguistics*, pp. 255–264, Halifax, Canada, 2003.
- [33] E. Agirre, et al. “Plagiarism Detection across Distant Language Pairs”, *In Proceedings of the 23rd International Conference on Computational Linguistics*, pp. 37–45, Beijing, China, 2010.
- [34] C. Lyon, et al. “Detecting short passages of similar text in large document collections”, *In Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 118–125, Pittsburgh, PA USA, 2001.

- [35] L. Han, et al. “UMBC_EBIQUITY-CORE: Semantic Textual Similarity Systems”, *In Proceedings of the 2nd Joint Conference on Lexical and Computational Semantics*, pp. 44–52, Atlanta, GA, USA, 2013.
- [36] R. Mihalcea, C. Corley, & C. Strapparava, “Corpus-based and Knowledge based Measures of Text Semantic Similarity”, *In Proceedings of the 21st National Conference on Artificial Intelligence*, pp. 775–780, Boston, MA, USA, 2006.
- [37] G. Salton, & M. J. McGill, “Introduction to Modern Information Retrieval”, McGraw-Hill, 1983.
- [39] Y. Li, et al, “Sentence Similarity Based on Semantic Nets and Corpus Statistics”, *IEEE Transactions on Knowledge and Data Engineering*, 18(8):1138–1150, 2006.
- [40] R. Rada, et al, “Development and Application of a Metric on Semantic Nets”, *IEEE Transactions on Systems, Man, and Cybernetics*, 19(1):17–30, 1989.
- [41] P. Resnik, “Using Information Content to Evaluate Semantic Similarity in a Taxonomy”, *In Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pp. 448–453, Montreal, Canada, 1995.
- [43] C. D. Manning, P. Raghavan, & H. Schütze, “Introduction to Information Retrieval”, Cambridge University Press, 2008.
- [44] T. K. Landauer, P. W. Foltz, & D. Laham, “An Introduction to Latent Semantic Analysis”, *Discourse Processes*, 25(2):259–284, 1998.
- [45] C. Burgess, K. Livesay, & K. Lund, “Explorations in context space: Words, sentences, discourse,” *Discourse Processes*, vol. 25, no. 2-3, pp. 211-257, 1998.

- [46] I. Matveeva, et al, “Generalized latent semantic analysis for term representation”, *In Proc. of RANLP*, 2005.
- [47] E. Gabrilovich & S. Markovitch, “Computing Semantic Relatedness using Wikipedia-based Explicit”, 2007.
- [49] P. Turney, “Mining the web for synonyms: PMIIR versus LSA on TOEFL”, *In Proceedings of the Twelfth European Conference on Machine Learning (ECML)*, 2001
- [50] K. Peter, “Experiments on the difference between semantic similarity and relatedness”, *In Proceedings of the 17th Nordic Conference on Computational Linguistics - NODALIDA '09*, Odense, Denmark, 2009.
- [51] D. Lin, “Extracting Collocations from Text Corpora”, *In Workshop on Computational Terminology*, Montreal, Kanada, 57–63, 1998,
- [53] A Composite Model for Computing Similarity between Texts. Dissertation August 2013
- [54] C. Fellbaum, “WordNet: An Electronic Lexical Database”, MIT Press, 1998.
- [55] B. Alberto, et al. “Plagiarism Detection across Distant Language Pairs, In Proceedings of the 23rd International Conference on Computational Linguistics, pp. 37–45, 2010.
- [56] G. Tsatsaronis, I. Varlamis, & M. Vazirgiannis, “Text Relatedness Based on a Word Thesaurus”, *Journal of Artificial Intelligence Research*, 37:1–39, 2010.
- [57] A. Kennedy, & S. Szpakowicz, “Evaluating Roget’s Thesauri”, *In Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 416–424, Columbus, OH, USA, 2008.

- [58] D. Ramage, A. N. Rafferty, & C. D. Manning, “Random Walks for Text Semantic Similarity”, *In Proceedings of the Workshop on Graph-based Methods for Natural Language Processing*, pp. 23–31, Singapore., 2009.
- [59] E. Yeh, D. Ramage, C. D. Manning, E. Agirre, & A. Soroa, “ WikiWalk: Random walks on Wikipedia for Semantic Relatedness”, *In Proceedings of the Workshop on Graph-based Methods for Natural Language Processing*, pp. 41–49, Singapore, 2009.
- [62] B. Davide et al., “SOPA: Random Forests Regression for the Semantic Textual Similarity task”, *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pp. 132–137, Denver, Colorado, June 4-5, 2015, ©2015 Association for Computational Linguistics
- [63] Z. Wu, & M. Palmer, “Verb semantics and lexical selection”, *In Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, pp. 133–138, Las Cruces, NM, USA, 1994.
- [64] L. Meng, et al. “A New Model of Information Content Based on Concept's Topology for measuring Semantic Similarity in WordNet”, *International Journal of Grid and Distributed Computing*, vol. 5, no. 3, (2012) September, pp. 81-94.
- [65] R. L. Cilibrasi, and P. M. Vitanyi, “The Google Similarity Distance”, *IEEE Trans. Knowledge and Data Engineering*,19:3, pp. 370-383, 2007.
- [66] O. Ana, et al. “ASAP-II: From the Alignment of Phrases to Text Similarity”, *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pp. 184–189, Denver, Colorado, June 4-5, 2015.©2015 Association for Computational Linguistic.

- [67] D. Manning, et al. “The Stanford CoreNLP natural language processing toolkit”, *In Proceedings of ACL: System Demonstrations*, 2014.
- [68] H. Basma, et al. “FCICU: The Integration between Sense-Based Kernel and SurfaceBased Methods to Measure Semantic Textual Similarity”, *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pp. 154–158, Denver, Colorado, June 4-5, 2015. ©2015 Association for Computational Linguistic.
- [69] T. T. Vu, et al. “TATO: Leveraging on Multiple Strategies for Semantic Textual Similarity”, *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pp. 190–195, Denver, Colorado, June 4-5, 2015. ©2015 Association for Computational Linguistic.
- [70] P. A. Ngoc, et al. “FBK-HLT: A New Framework for Semantic Textual Similarity”, *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pp. 102–106, Denver, Colorado, June 4-5, 2015, ©2015 Association for Computational Linguistic.
- [71] T. Liling, et al. “USAAR-SHEFFIELD Semantic Textual Similarity with Deep Regression and Machine Translation Evaluation Metrics”, *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pp. 85–89, Denver, Colorado, June 4-5, 2015. ©2015 Association for Computational Linguistics
- [72] A. Piyush, et al. “DCU: Using Distributional Semantics and Domain Adaptation for the Semantic Textual Similarity SemEval-2015 Task 2”, *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pp. 143–147, Denver, Colorado, June 4-5, 2015. ©2015 Association for Computational Linguistics

- [73] H. Lushan, et al. “Samsung Align-and-Differentiate Approach to Semantic Textual Similarity”, *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pp. 172–177, Denver, Colorado, June 4-5, 2015, ©2015 Association for Computational Linguistics.
- [75] F. Saric, et al. “TakeLab: systems for measuring semantic text similarity”, *In: 1st Joint Conf. on Lexical and Computational Semantics*, pp.441–448 (2012)
- [76] Z. Jiang, et al. “ECNU: Using Traditional Similarity Measurements and Word Embedding for Semantic Textual Similarity Estimation”, *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pp. 117–122, Denver, Colorado, June 4-5, 2015. ©2015 Association for Computational Linguistics
- [77] H. Christian, et al. “ExB Themis: Extensive Feature Extraction from Word Alignments for Semantic Textual Similarity”, *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pp. 264–268, Denver, Colorado, June 4-5, 2015 ©2015 Association for Computational Linguistics
- [78] Md. S. Arafat, et al. “Dls©cu: Sentence similarity from word alignment”, *In Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pp. 241–246, Dublin, Ireland, 2014, Association for Computational Linguistics and Dublin City University.
- [79] K. Sakethram, et al. “UMDuluth-BlueTeam: SVCSTS - A Multilingual and Chunk Level Semantic Similarity System”, *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pp. 107–110, Denver, Colorado, June 4-5, 2015 ©2015 Association for Computational Linguistics.

- [80] B. Steven, “NLTK: The Natural Language Toolkit”, *Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions*, pp. 69–72, Sydney, July 2006.© 2006 Association for Computational Linguistics
- [81] M. Tomas, “Distributed representations of words and phrases and their compositionality”, *In Advances in Neural Information Processing Systems*, 2013, pp. 3111–3119.
- [82] E. Agirre, et al. “UBC: Cubes for English Semantic Textual Similarity and Supervised Approaches for Interpretable STS”, *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pp. 178–183, Denver, Colorado, June 4-5, 2015.©2015 Association for Computational Linguistics
- [83] C. C. Chang, and C. J. Lin, “LIBSVM: A library for support vector machines” *ACM Transactions on Intelligent Systems and Technology*, 2(3):27:1–27, 2011.
- [84] M. Hall, “The WEKA Data Mining Software: An Update. SIGKDD Explorations”, 11(1):10–18, 2009.
- [85] Y. L. Nay, et al. “Developing a Chunk-based Grammar Checker for Translated English Sentences”, *25th Pacific Asia Conference on Language, Information and Computation*, pp. 245–254
- [86] S. Abney, “Tagging and Partial Parsing, In: Ken Church, Steve Young, and Gerrit Bloothoof (eds.), *Corpus-Based Methods in Language and Speech*”, Kluwer Academic Publishers, Dordrecht, 1996.
- [87] Landauer and S. Dumais, “A solution to plato’s problem: The latent semantic analysis theory of the acquisition, induction, and representation of knowledge”, *In Psychological Review*, 104, pp. 211–240, 1997.

- [89] C. T. Meadow, “Text Information Retrieval Systems”, Academic Press, Inc. 1992.
- [90] Corpus-based and Knowledge-based Measures of Text Semantic Similarity
<http://www.cse.unt.edu/~rada/papers/mihalcea.aaai06.pdf>
- [91] N. Okazaki, et al. “Sentence Extraction by Spreading Activation through Sentence Similarity,”
IEICE Trans. Information and Systems, vol. E86D, no. 9, pp. 1686-1694, 2003.
- [92] P. W. Foltz, et al. “The measurement of textual coherence with latent semantic analysis”,
Discourse Processes, vol. 25, no. 2-3, pp. 285-307,1998.
- [93] L. Yang, et al. “yiGou A Semantic Text Similarity Computing System Based on SVM”,
Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015), pp. 80–
84, Denver, Colorado, June 4-5, 2015.©2015 Association for Computational Linguistics
- [94] F. Pedregosa, “Scikit-learn: Machine learning in Python”, *The Journal of Machine Learning
Research*, vol. 12, pp. 2825-2830, (2011).
- [95] B. Hanna, et al. “MiniExperts An SVM Approach for Measuring Semantic Textual Similarity”,
Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015), pp. 96–
101, Denver, Colorado, June 4-5, 2015.© 2015 Association for Computational Linguistics
- [96] S. Reese, et al. “Wikicorpus: A Word-Sense Disambiguated Multilingual Wikipedia Corpus”,
Paper presented at *the 7th Language Resources and Evaluation Conference*, LaValleta, Malta.
(2010). (<http://www.cs.upc.edu/~nlp/papers/reese10.pdf>)
- [98] D. Blei, et al. “Latent dirichlet allocation”, *Journal of Machine Learning Research*, 3:993–1022,
2003.

- [99] T. Mikolov, et al. “Efficient Estimation of Word Representations in Vector Space”, *In Proceedings of Workshop at ICLR*, 2013.
- [100] Y. H. Li, et al. “An Approach for Measuring Semantic Similarity Using Multiple Information Sources”, *IEEE Trans. Knowledge and Data Eng.*, vol. 15, no. 4, pp. 871-882, July/Aug, 2003.
- [101] M. Anderka, B. Stein, “The ESA Retrieval Model Revisited SIGIR”, 09, July 19–23, 2009, Boston, Massachusetts, USA. ACM 978-1-60558-483-6/09/07.
- [102] N. Okazaki, et al. “Sentence Extraction by Spreading Activation through Sentence Similarity”, *IEICE Trans. Information and Systems*, vol. E86D, no. 9, pp.1686-1694, 2003.
- [103] A. Budanitsky, and G. Hirst, “Evaluating WordNet-based measures of lexical semantic relatedness”, *Comp. Linguistics*, 32(1), 13–47, 2006.
- [104] M. A. Rodriguez and M. J. Egenhofer, “Determining Semantic Similarity among Entity Classes from Different Ontologies”, *IEEE Trans. Knowledge and Data Eng.*, vol. 15, no. 2, pp. 442-456, Mar./Apr. 2003
- [105] E. Agirre et al. “Why do you say they are similar? Interpretable Semantic Textual Similarity”, *Proceedings of SemEval-2016*, pp. 524–536, San Diego, California, June 16-17, 2016.©2016 Association for Computational Linguistics
- [106] L. Ballesteros, and B. Croft, “Dictionary-based methods for cross-lingual information retrieval”, *In Proceedings of the 7th International DEXA Conference on Database and Expert Systems Applications*, pp. 791-801
- [107] R. Baeza-Yates, and Ribeiro-Neto, B. (1999). *Modern information retrieval*. England: ACM Press.

- [108] V. Lifschitz. 2008. What is answer set programming?. In *AAAI*, volume 8, pp. 1594–1597.
- [109] Gerhard Brewka, Thomas Eiter, and Mirosław Truszczyński, (2011), Answer set programming at a glance, *Communications of the ACM*, 54(12):92–103.
- [110] R. S. Jeffrey Pennington, and Christopher Manning, “Glove: Global vectors for word representation”, In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing(EMNLP)*, pp. 1532–1543, Association for Computational Linguistics. 2014.
- [111] D. Walter & V. Antal, “*Memory-based language processing*”, Cambridge University Press. 2005.
- [112] G. Varelas, et al. “Semantic similarity methods in WordNet and their application to information retrieval on the web”, *Proceedings of the 7th annual ACM international workshop on Web information and data management*, (2005) October 31- November 05, Bremen, Germany.
- [113] D. Lin, “An information-theoretic definition of similarity”, *Proceedings of the 15th International Conference on Machine Learning*, Madison, Wisconsin, USA (1998).
- [114] J. J. Jiang and D. W. Conrath, “Semantic similarity based on corpus statistics and lexical taxonomy”, *Proceedings of International Conference on Research in Computational Linguistics*, (1997) August 22-24; Taipei, Taiwan.
- [115] A. Tversky, “Features of Similarity”, *Psychological Review*, vol. 84, no. 4, (1977).
- [116] N. Seco, et al. “An intrinsic information content metric for semantic similarity in WordNet”, *Proceedings of the 16th European Conference on Artificial Intelligence*, (2004) August 22-27, Valencia, Spain.
- [117] D. Sánchez, et al. “Ontology-based information content computation”, *Knowl.-Based Syst.*, vol. 24, no. 2, (2011).

- [118] Md. H. Seddiqui and M. Aono, “Metric of intrinsic information content for measuring semantic similarity in an ontology”, *Proceedings of 7th Asia-Pacific Conference on Conceptual Modeling*, (2010) January 18-21; Brisbane, Australia.