International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10
DOI: https://doi.org/10.17762/ijritcc.v11i10.8491
***Article Received:** 26 July 2023 **Revised:** 16 September 2023 **Accepted:** 05 October 2023*

Sentence Semantic Similarity based Complex Network approach for Word Sense Disambiguation

[1]Gopal Mohadikar, [2]Sonu Khapekar, [3]Dr. M. K. Kodmelwar, [4]Supriya Bhosale, [5]Sopan Bapu Kshirsagar, [6]Ganesh Chandrabhan Shelke

[1]Sr. Assistant Professor, Department of Mechanical Engineering, Tolani Maritime Institute, Induri, Pune, India(MS).ORCID ID 0009-0004-5593-7607
Email: gmohadikar@gmail.com
[2]Affiliation: Nutan Maharashtra Institute of Engineering & Technology, Talegaon(D), Pune, India(MS).
ORCID ID: 0009-0007-9677-8931
E-mail: sonukhapekar999@gmail.com
[3]Affiliation: Vishwakarma Institute of Information Technology, Pune
ORCID:0000-0001-5248-528X
Email: manohar.kodmelwar@viit.ac.in
[4]Affiliation: Nutan Maharashtra Institute of Engineering & Technology, Talegaon(D), Pune, India(MS).
ORCID ID:0000-0002-7847-0681
Email: supriyabhosale9@gmail.com
[5]Affiliation: Nutan Maharashtra Institute of Engineering & Technology, Talegaon(D), Pune, India(MS).
ORCID ID:0009-0006-0683-5369
Email:sopankshirsagar02@gmail.com
[6]Affiliation: Vishwakarma Institute of Information Technology, Pune.
ORCID ID: 0009-0009-2297-7748
Email: ganesh.shelke@viit.ac.in

Abstract
Word Sense Disambiguation is a branch of Natural Language Processing(NLP) that deals with multi-sense words. The multi-sense words are referred to as the polysemous words. The term lexical ambiguity is introduced by the multi-sense words. The existing sense disambiguation module works effectively for single sentences with available context information. The word embedding plays a vital role in the process of disambiguation. The context-dependent word embedding model is used for disambiguation. The main goal of this research paper is to disambiguate the polysemous words by considering available context information. The main identified challenge of disambiguation is the ambiguous word without context information. The discussed complex network approach is disambiguating ambiguous sentences by considering the semantic similarities. The sentence semantic similarity-based network is constructed for disambiguating ambiguous sentences. The proposed methodology is trained with SemCor, Adaptive-Lex, and OMSTI standard lexical resources. The findings state that the discussed methodology is working fine for disambiguating large documents where the sense of ambiguous sentences is on the adjacent sentences.

Keywords: word sense disambiguation, semantic similarity, word vector, complex network

1. Introduction

Figure 1.1 Complex Network for WSD

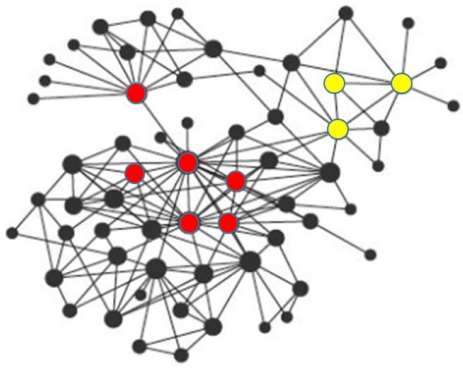

Figure 1.2 S-3 based WSD for large documents

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10
DOI: https://doi.org/10.17762/ijritcc.v11i10.8491
***Article Received:** 26 July 2023 **Revised:** 16 September 2023 **Accepted:** 05 October 2023*

Semantic similarity is a fundamental concept in NLP and computational linguistics shown in Figure 1.1. It revolves around gauging the closeness of meaning between words, phrases, or entire texts. This closeness is crucial in various applications, such as information retrieval, text summarization, question-answering systems, and machine translation [1]. In assessing semantic similarity, we consider how similar two linguistic units are in meaning, context, or usage. It's not just about the words themselves but their meaning and how they interact within a given context [2]. For instance, "strong" and "powerful" are semantically similar because they share a related meaning in most contexts shown in Figure 1.2. Measuring semantic similarity often involves mathematical or computational models that process linguistic features, word embeddings, or other representations to quantify the likeness in meaning. Techniques like cosine similarity on word vectors, knowledge graph-based methods, and deep learning approaches play vital roles in this analysis. One of the popular approaches is Word2Vec, which maps words into continuous vector spaces based on their co-occurrence patterns in a corpus. Words with similar meanings are mapped close to each other in this vector space, indicating their semantic similarity. Understanding semantic similarity is crucial in many NLP tasks [2][3]. For example, in a search engine, if a user searches for "apple fruit," the search engine should retrieve documents containing "apple fruit" but also documents related to "fruit" more broadly. This requires a nuanced understanding of semantic similarity. In conclusion, semantic similarity is a core concept in the field of NLP, enabling computers to comprehend and process language more effectively [4]. By measuring the closeness of meanings between words and texts, we can enhance various language-driven applications, ultimately improving how we interact with and extract information from vast amounts of textual data. Table 1.1 shows the semantic similarity of words with respect to semantic score.

Table 1.1 Semantic Similarity of Words

Term 1	Term 2	Semantic Similarity
Apple	Banana	High
Car	Bicycle	Moderate
Dog	Cat	High
Book	Pen	Low
Planet	Galaxy	High

In this table, we're comparing different terms and assigning a level of semantic similarity. The values (High, Moderate, Low) are based on the perceived similarity of the terms in meaning. Table 1.2 provides a simple representation of semantic similarity.

Table 1.2 Summary of word semantic similarity techniques

Technique	Description	Advantages	Disadvantages	Applications
Jaccard Similarity	Compares sets of words or tokens	Simple and easy to implement	Ignores word order	Text clustering, document similarity
Cosine Similarity	Compares word frequency vectors	Handles high-dimensional data	Sensitive to document length	Information retrieval, text classification
Edit Distance	Measures edit operations needed	Captures spelling similarities	Sensitive to length differences	Spell checking, bioinformatics
WordNet-Based Similarity	Utilizes WordNet for similarity	Considers semantic relationships	Limited to WordNet's coverage	Semantic search, ontology alignment
Distributional Similarity	Measures similarity based on word co-occurrence	Captures contextual relationships	Sensitive to corpus choice	Sentiment analysis, recommendation systems
Word Embeddings	Represents words as vectors in a space	Captures semantic relationships	Data-intensive training	Language translation, sentiment analysis

2. Literature Review

The review of semantic similarity techniques is done with respect to the techniques as,

2.1. Cosine Similarity

Cosine similarity is a widely used metric to measure the similarity between two vectors, often applied in NLP for assessing semantic similarity. It's particularly valuable in tasks like text analysis, document similarity, and recommendation systems [5].

As shown in Figure 2.1, In the context of semantic similarity, cosine similarity quantifies the cosine of the angle between two non-zero vectors, representing the textual content of interest. Let's break down the key components and the working principle of cosine similarity.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10
DOI: https://doi.org/10.17762/ijritcc.v11i10.8491
Article Received: 26 July 2023 *Revised:* 16 September 2023 *Accepted:* 05 October 2023

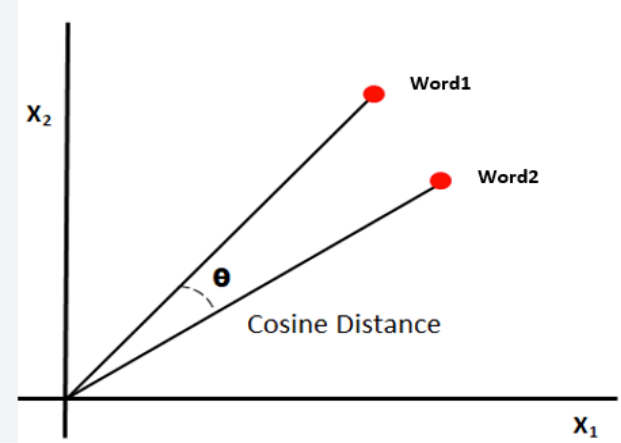

Figure 2.1 Cosine Distance of words

Given two vectors, A and B, cosine similarity is calculated using the following formula-1:

$$\text{Cosine Similarity (A, B)} = \frac{\text{A} * \text{B}}{||A|| * ||B||} \quad (1)$$

Where A*B represents the dot product of vectors A and B.
‖A‖ * ‖B‖ denotes the Euclidean norms of vectors A and B, respectively.
The cosine similarity ranges from -1 to 1. 1 indicates perfect similarity, where the vectors align completely. 0 means no similarity, implying the vectors are orthogonal. -1 represents perfect dissimilarity, with the vectors pointing in opposite directions.

2.2. Jaccard Similarity

Jaccard Similarity is shown in Figure 2.2, It is a metric used to determine the similarity and diversity of sample sets. It measures the similarity between two sets by calculating the ratio of the intersection of the sets to the union of the sets [6]. In the context of semantic similarity, Jaccard Similarity is often used to compare the similarity of documents or texts based on the words or terms they contain. The Jaccard Similarity coefficient (J) is calculated using the following formula-2:

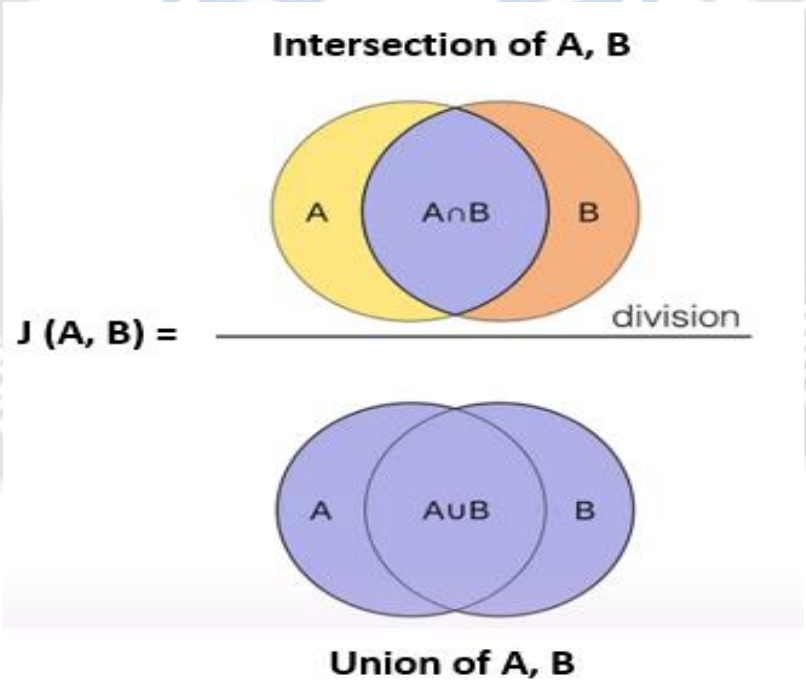

Figure 2.2 Jaccard Similarity based on set theory

$$J(A,B) = \frac{\text{Number of common elements in sets } A \text{ and } B}{\text{Number of distinct elements in sets } A \text{ and } B} \quad (2)$$

Where:
A and *B* are the sets being compared, typically representing the words or terms in two documents.
The numerator represents the count of common elements (words or terms) in both sets.
The denominator represents the count of distinct elements (words or terms) across both sets.
A higher Jaccard Similarity coefficient indicates a higher level of similarity between the sets being compared. Researchers and practitioners often use Jaccard Similarity in NLP tasks such as text clustering, document similarity, and information retrieval. It helps in identifying relationships and similarities between different pieces of text, which is valuable in various

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10
DOI: https://doi.org/10.17762/ijritcc.v11i10.8491
Article Received: *26 July 2023* ***Revised:*** *16 September 2023* ***Accepted:*** *05 October 2023*

applications like recommendation systems, text summarization, and more.

It's essential to preprocess the text appropriately before calculating the Jaccard Similarity. This preprocessing may include removing stop words, stemming or lemmatizing words, and converting text to lowercase to ensure accurate similarity measurements.

2.3. Word2Vec

As shown in figure 2.3, Word2Vec is based on the idea that words with similar meanings tend to appear in similar contexts within a large corpus of text. It learns word embeddings, which are vector representations of words, in such a way that semantically similar words are closer to each other in the vector space [7]. The two main algorithms used for Word2Vec are Continuous Bag of Words (CBOW) and Skip-gram. CBOW predicts a target word based on its context, while Skip-gram predicts the context words given a target word. Both approaches result in word embeddings that encode semantic information.

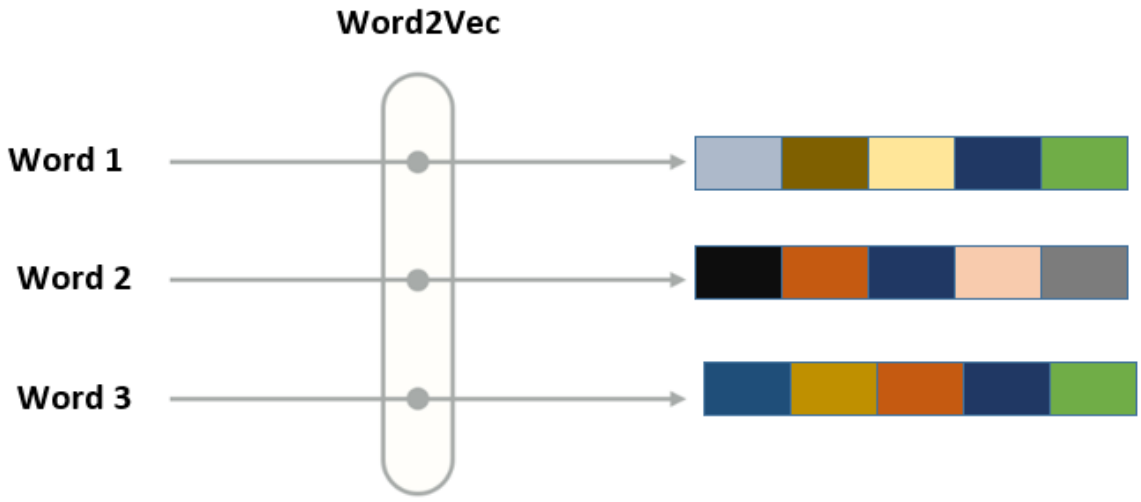

Figure 2.3 Word2Vector technique

2.4. GloVe

As shown in Figure 2.4, GloVe, short for Global Vectors for Word Representation, is an unsupervised learning algorithm designed to capture semantic similarities between words. It utilizes co-occurrence statistics to represent words in a continuous vector space.

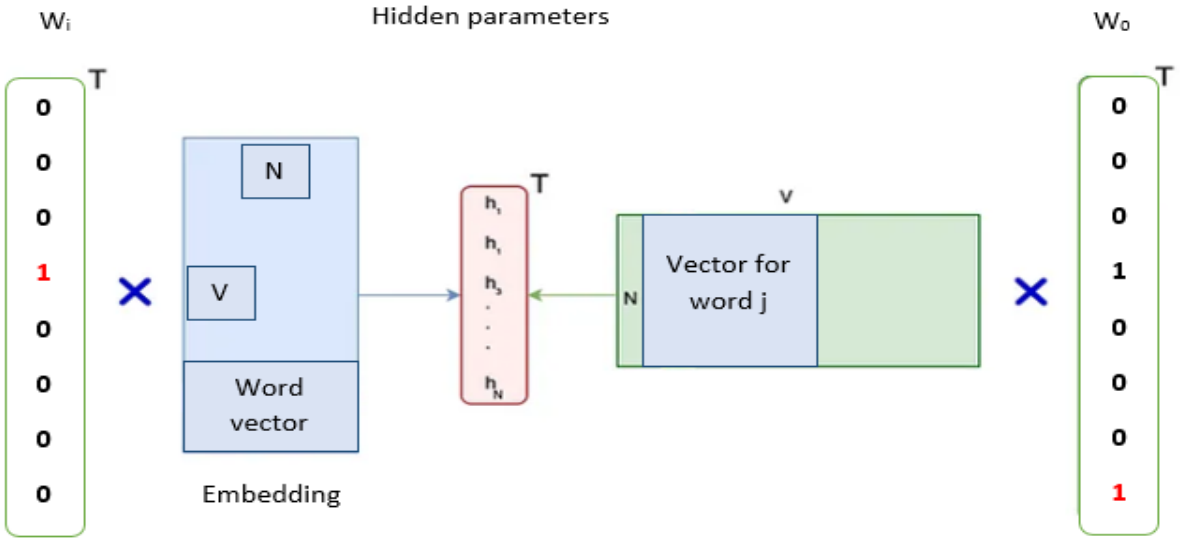

Figure 2.4 GloVe Technique

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10
DOI: https://doi.org/10.17762/ijritcc.v11i10.8491
Article Received: 26 July 2023 Revised: 16 September 2023 Accepted: 05 October 2023

The fundamental idea behind GloVe is to establish relationships between words based on their co-occurrences within a large corpus of text. Co-occurrence is a powerful indicator of semantic similarity: words that often appear together are likely to have related meanings. GloVe calculates word embeddings by constructing a co-occurrence matrix, where each element represents how often two words co-occur [8]. The algorithm then optimizes these embeddings to minimize the difference between the dot product of word vectors and the logarithm of their co-occurrence probabilities. Once the training is complete, each word is represented as a high-dimensional vector in the embedding space. Importantly, words with similar meanings or usage patterns are positioned close to each other in this vector space. Semantic similarity in GloVe is quantified through vector operations. Words with similar meanings will have vectors that are close in distance, often measured using cosine similarity. The closer the vectors, the more similar the words in terms of meaning. Researchers and practitioners use these vector representations to solve various NLP tasks, such as sentiment analysis, named entity recognition, and machine translation. In conclusion, GloVe is a powerful tool for capturing semantic similarity in word representations, offering a robust foundation for many NLP applications.

2.5. FastText

FastText is an extension of the Word2Vec model. Instead of considering words as the smallest units like Word2Vec, FastText views words as bags of character n-grams [9]. This approach allows it to generate embeddings for out-of-vocabulary words and capture subword information.

2.6. BERT

BERT (Bidirectional Encoder Representations from Transformers) is a revolutionary NLP model that has significantly advanced the field of semantic similarity analysis. Developed by Google AI in 2018, BERT is designed to understand the context and meaning of words in a sentence by considering both the left and right context, hence the term "bidirectional".

At its core, BERT is a deep learning model based on the Transformer architecture. Unlike previous NLP models, BERT is pre-trained on an extensive corpus of text from the internet, allowing it to capture a vast amount of linguistic knowledge [10]. This pre-training phase equips BERT with a rich understanding of word relationships, sentence structures, and the nuances of language.

2.7. Doc2Vec

The Doc2Vec algorithm involves training a neural network to predict words in a similar context, while simultaneously learning the document vectors. This allows the model to understand the semantic representation of documents in a continuous vector space [11]. To measure semantic similarity between documents using Doc2Vec, you can use various similarity metrics such as cosine similarity, Euclidean distance, or Manhattan distance on the document vectors. Table 2.1 shows the summary of semantic similarity techniques with key features.

Table 2.1 Summary of Semantic Similarity Models

Technique	Description	Key Features	References
Cosine Similarity	Measures cosine of the angle between vectors	Vector-based approach, widely used	[Salton et al., 1975]
Jaccard Similarity	Compares set intersection and union of terms	Good for text data, set-based approach	[Jaccard, 1901]
Word2Vec	Neural network-based word embeddings	Captures semantic relationships	[Mikolov et al., 2013]
GloVe	Global vectors for word representation	Considers global word co-occurrences	[Pennington et al., 2014]
FastText	Word embeddings with subword information	Handles out-of-vocabulary words	[Bojanowski et al., 2017]
BERT Embeddings	Bidirectional Encoder Representations from Transformers	Deep contextual embeddings	[Devlin et al., 2018]
Doc2Vec	Document embeddings using neural networks	Captures document-level semantics	[Le and Mikolov, 2014]

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10
DOI: https://doi.org/10.17762/ijritcc.v11i10.8491
Article Received: 26 July 2023 Revised: 16 September 2023 Accepted: 05 October 2023

3. Methodology

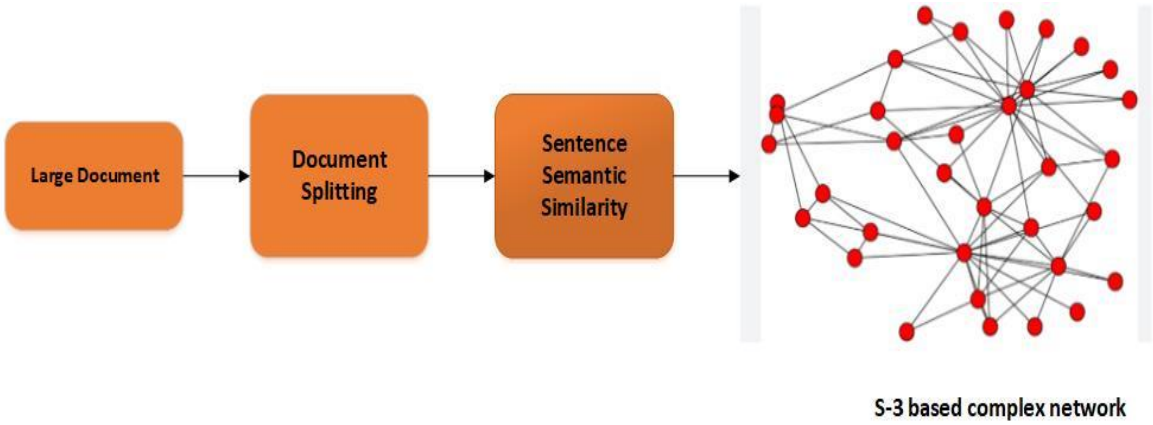

Figure 3.1 Complex network approach

The discussed model for WSD in large documents works concerning the semantic similarity of the sentences. The WSD for polysemous words is concerned with semantic similarity and context information. The proposed methodology accepts large documents as input. The sentence splitting of the document is done by assigning a unique identifier to every sentence. This sentence is further used as a vertex id. All the sentences of the document are represented in vertex space by considering individual sentences as the vertex with a unique id. The challenge here is how to draw the edges between the vertex and what pair of the vertex is selected for the edges. The proposed methodology calculates the semantic similarities of the vertex of semantic similarity sentences and based on the semantic similarity value, the weighted edge is drawn. The importance of semantic similarity is in the range of '0' to '1'. The '1' is very close to the meaning of semantic similarity and the value '0' indicates a significantly less semantic similar value

3.1. Sentence Semantic Similarity Model

The above section describes the semantic similarity of the words based on the HyperTrees. This section elaborated on the calculation of the semantic similarity of the sentences. The hypothesis is tested here that we can use the HyperTrees for calculating the semantic similarities of the sentences. Based on the HyperTrees, the semantic similarity of the sentences is calculated. As shown in figure 3.2, the semantic similarity of sentence-1, "Let's go to the bank", and sentence-2, "Said by Anay to Ram for deposit", is calculated. The semantic similarity of every word of sentence 1 to every other word of sentence 2 is calculated. It is observed that the semantic similarity of the phrase pair 'bank' and 'deposit' is different. The semantic similarity of the word pair 'bank' and 'deposit' is 0.5882 when calculated individually. The semantic similarity of the words 'bank' and 'deposit' in the sentences is 0.3. The error value 0.12356 needs to be addressed. The S-3 model generates the semantic similarity of the words based on the semantic similarity score; the weighted edge is drawn and the same semantic similarity score is used to disambiguate the vertex without context information. In the above example, the semantic similarity score is 0.34 for vertex-1, and vertex-2, and based on the same, the edge is drawn and the weight value is represented as a weighted edge, as shown in Figure 3.1.

wup(bank_deposit$n$1 , deposit_account$n$1) = 0.3

T1= HyperTrees(bank_deposit$n$1)=[1]*ROOT*n1 <entity$n$1 <abstraction$n$6<measure$n$2<system_of_measurement$n$1<standard$n$1<medium_of_exchange$n$1<money$n$1<fund$n$1<bank_deposit$n$1

T2= HyperTrees(deposit_account$n$1)=[1]*ROOT*n1 <entity$n$1<abstraction$n$6<attribute$n$2 <state$n$2 <relationship$n$3 <account$n$3<bank_account$n$1<savings_account$n$1<deposit_account$n$1

Lowest_Common_Subsumer(s) = arg_max(depth(subsumer(T1,T2)))={ subsumer(T1[1], T2[1]) }= { abstraction$n$6 }

Depth_LCS = depth(abstraction$n$6) = 3

Depth_1 = min(depth({tree in T1 | tree contains LCS })) = 10

Depth_2 = min(depth({tree in T2 | tree contains LCS })) = 10

*Min_score= 2 * Depth_LCS / (Depth_1 + Depth_2) = 2 * 3 / (10 + 10) = 0.3*

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10
DOI: https://doi.org/10.17762/ijritcc.v11i10.8491
Article Received: 26 July 2023 Revised: 16 September 2023 Accepted: 05 October 2023

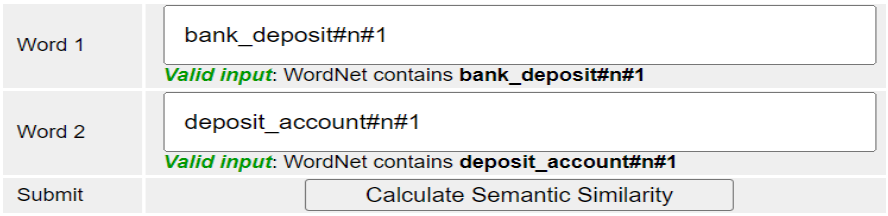

Figure 3.2 Calculation of Semantic Similarity

4. Result and Discussion

The adaptive methodology for WSD in the large documents is tested with the RNN-LSTM model. The complex network approach for the WSD in the large document plays playing important role. The construction of a complex network of large documents is done by calculating the semantic similarity of the sentences. This methodology is the major contributor to the disambiguation of ambiguous sentences with a lack of context information. As shown in Figure 4.1.1, The proposed methodology generates 83% and 80% accuracy with OMSTI for disambiguating single sentences and large documents respectively.

This methodology generates 81.3% and 81.00% accuracy while disambiguating single sentences and large documents for SemCor shown in Figure 4.1.2. This methodology generates 85.39% and 83% accuracy with Adaptive-Lex for disambiguating single sentences and large documents respectively shown in Figure 4.1.3. As shown in Figure 4.1.4, the percentage of improvement is 4.09 for single-sentence disambiguation and 3.00% for disambiguating large documents.

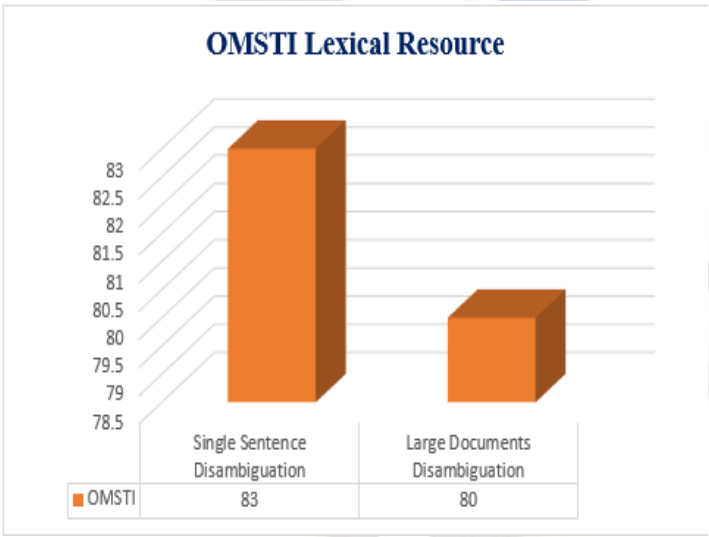

Figure 4.1.1 Result of WSD for OMSTI

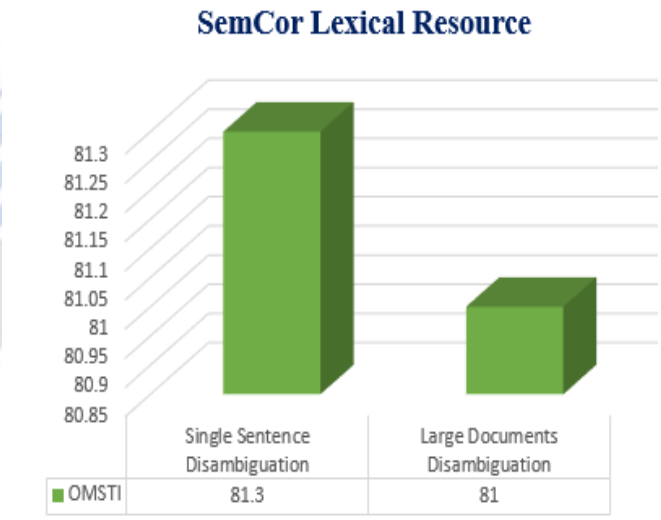

Figure 4.1.2 Result of WSD for SemCor

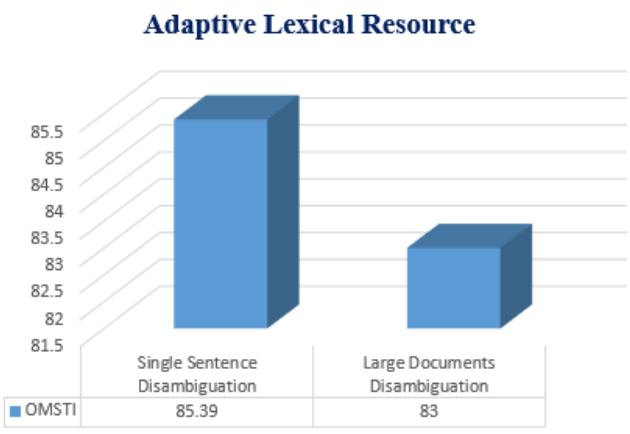

Figure 4.1.2 Result of WSD for Adaptive-Lex

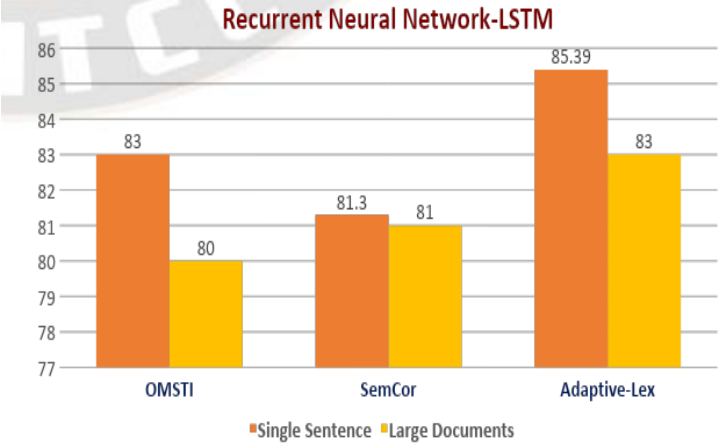

Figure 4.1.1 Comparative Result of WSD

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10
DOI: https://doi.org/10.17762/ijritcc.v11i10.8491
Article Received: 26 July 2023 Revised: 16 September 2023 Accepted: 05 October 2023

5. Conclusion and future work

WSD is a live problem of NLP, most of the WSD research is on a single-sentence disambiguation. The complex network approach is a revolutionary methodology for disambiguating large documents. In the complex network approach, the document is represented with the vertices and based on the sentence semantic similarities the semantic score of the sentences is calculated. The semantic similarity of the sentences is the distance between the vertices. The sentences with a lack of context information are disambiguated by considering the closest semantic similar sentence. This methodology is working fine with the freely available lexical resource WordNet. The training of the module is done with Adaptive-Lex SemCor and OMSTI. This methodology is tested for 1200 sentences in the future maximum sentences will be added with the context-dependent dataset.

References

[1] Correa Jr, Edilson A., Alneu A. Lopes, and Diego R. Amancio. "Word sense disambiguation: A complex network approach." Information Sciences 442 (2018): 103-113.

[2] Veronis, Jean, and Nancy Ide. "Word sense disambiguation with very large neural networks extracted from machine readable dictionaries." COLING 1990 Volume 2: Papers presented to the 13th International Conference on Computational Linguistics. 1990.

[3] Kokane, Chandrakant, et al. "Word Sense Disambiguation: A Supervised Semantic Similarity based Complex Network Approach." International Journal of Intelligent Systems and Applications in Engineering 10.1s (2022): 90-94.

[4] Kokane, Chandrakant D., Sachin D. Babar, and Parikshit N. Mahalle. "Word Sense Disambiguation for Large Documents Using Neural Network Model." 2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT). IEEE, 2021.

[5] Salton, Gerard, Anita Wong, and Chung-Shu Yang. "A vector space model for automatic indexing." Communications of the ACM 18.11 (1975): 613-620.

[6] Verma, Vijay, and Rajesh Kumar Aggarwal. "A comparative analysis of similarity measures akin to the Jaccard index in collaborative recommendations: empirical and theoretical perspective." Social Network Analysis and Mining 10 (2020): 1-16.

[7] Mikolov, Tomas, et al. "Efficient estimation of word representations in vector space." arXiv preprint arXiv:1301.3781 (2013).

[8] Pennington, Jeffrey, Richard Socher, and Christopher D. Manning. "Glove: Global vectors for word representation." Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). 2014.

[9] Bojanowski, Piotr, et al. "Enriching word vectors with subword information." Transactions of the association for computational linguistics 5 (2017): 135-146.

[10] Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018).

[11] Le, Quoc, and Tomas Mikolov. "Distributed representations of sentences and documents." International conference on machine learning. PMLR, 2014.