Efficient Textual Similarity using Semantic MinHashing

1st Waqas Nawaz
Department of Information Systems, Islamic University of Madinah,
Madinah, Saudi Arabia
wnawaz@iu.edu.sa

2nd Maryam Baig
Department of Computer Science, NUCES-FAST University,
Islamabad, Pakistan
maryam-baig@hotmail.com

3rd Kifayat Ullah Khan
College of Accountancy, Finance and Economics, Birmingham City Business School,
Birmingham City University, UK
kifayat.khan@bcu.ac.uk

***Abstract*—Quantifying the likeness between words, sentences, paragraphs, and documents plays a crucial role in various applications of natural language processing (NLP). As Bert, Elmo, and Roberta exemplified, contemporary methodologies leverage neural networks to generate embeddings, necessitating substantial data and training time for cutting-edge performance. Alternatively, semantic similarity metrics are based on knowledge bases like WordNet, using approaches such as the shortest path between words. MinHashing, a nimble technique, quickly approximates Jaccard similarity scores for document pairs. In this study, we propose employing MinHashing to gauge semantic scores by enhancing original documents with information from semantic networks, incorporating relationships such as synonyms, antonyms, hyponyms, and hypernyms. This augmentation improves lexical similarity based on semantic insights. The MinHash algorithm calculates compact signatures for extended vectors, mitigating dimensionality concerns. The similarity of these signatures reflects the semantic score between the documents. Our method achieves approximately 64% accuracy in the MRPC and SICK data sets.**

***Index Terms*—MinHashing, Semantic similarity, WordNet, Natural Language Processing (NLP), Jaccard similarity, Algorithm**

I. Introduction

Semantic text similarity is one of the fundamental challenges in many natural language processing (NLP) tasks such as question answering in biomedical [1], automatic machine translations [2], and automatic text summarization [3] to name a few. Natural language is a vibrant resource where different words with non-identical lexical structures are said to be semantically similar if their meaning is the same. For instance, the words *destination* and *last stop* technically have the same meaning with different linguistic structures. Similarly, the meaning of a word depends on the context; for example, the word *crash* has different meanings in automobiles, the stock market, and parties in our daily lives. To understand the context, researchers have developed various approaches in which the similarity and relatedness between concepts are modeled in the form of graph-like structures such as WordNet [4], BabelNet [5], MeSH [6] and word embeddings such as word2vec [7]. Modern deep learning models like Elmo [8] and BERT [9] are trained to classify text based on word embeddings, practical in text summarization and resolving word-sense disambiguation. These state-of-the-art techniques require a lot of data and hours of training over expensive resources to produce reasonable results.

The emergence of big data in the NLP domain requires a scalable approach for quick estimations and analysis. MinHashing [10] is a highly scalable technique that quickly estimates the similarity (Jaccard) between two sets. This approach has been used in Locality Sensitive Hashing for Cross-Lingual Similarity [11], Graph Summarization [12], and Estimating Web Document Similarity [13]. To obtain decent results, we aim to determine the semantic similarity between documents without training on gigabytes of data. Textual similarity, a.k.a. lexical similarity, is efficiently computed through K-shingle tokens and MinHash algorithm [13]. However, this technique fails to consider semantics as illustrated in Fig. 1. In D1 and D2, different or synonymous words tell the same information that a person is good. If we measure the similarity score by MinHashing with stop words, it approximates 60%; while removing the stop words, the score drops to 0 when these words are 100% similar semantically. Integrating knowledge into the word vectors in the MinHashing approach is expected to produce reasonable results efficiently.

In this paper, we propose an efficient technique inspired by MinHashing to compute the semantic similarity of documents. We model the data so that it preserves the semantic information of a document by increasing the syntactical similarity of the documents achieved by embedding related concepts in the extending vector representation of the document. We transform the documents into tokens to form a vector of words, where we obtain the associated words for each token from the knowledge base like WordNet. The idea is to add knowledge to the word vectors using a well-structured knowledge base. These vectors are then encoded to pass them on to the MinHash algorithm. The MinHashing technique produces signatures against those vectors as codes such that similar words get similar MinHash values. These extended encoded vectors are then passed through a set of hash functions to obtain their short signatures. These signatures are meant to preserve the similarity of the documents, which is approximately equal to the Jaccard similarity. Our approach not only measures the

This research is funded by the Deanship of Scientific Research of the Islamic University of Madinah, KSA, under the research groups (first) project no. 956.

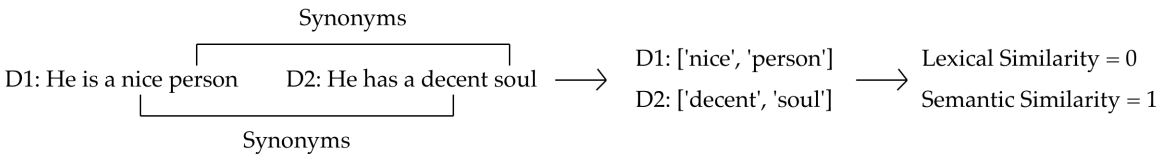

Fig. 1. Problem Illustration

structural similarity of the documents but semantic similarity as well in O(mnk $+m^2k$) run time complexity, which is the complexity of the MinHash algorithm. This method can be helpful in computational linguistics to find semantically similar items, such as tweets, research articles, and clustering semantically similar documents on a large scale.

II. Literature Review

This section briefly discusses closely related existing studies in the context of semantic similarity computations using knowledge bases, MinHash algorithms, and applications.

A. MinHashing Techniques

MinHashing is frequently used to find similar items in extensive data sets to estimate the Jaccard similarity between two sets. It represents large sets in small signatures while preserving the similarity among the entities. The probability of two sets having the same MinHash value is approximately equal to the sets' Jaccard Similarity (Intersection over Union) [10]. Since it is costly to generate random permutations over a large set explicitly, universal hashing is used on indexes to produce permutations. Many variations have been introduced in MinHashing to improve its efficiency and effectiveness. The standard MinHash considers all elements equally, while the weighted MinHash associates a probability with each component according to its importance [14]. The added weights increase the bias of the items in the sets, thus improving the score's accuracy. The B-bit MinHash approach [15] limits the fingerprint size to reduce space complexity, where B is an arbitrary number and the fingerprint represents an item in the set, i.e., hash value. However, it may miss some elements in the sets if their hash value does not occur in b-bits, or it may override an existing element. On the other hand, the one-permutation MinHash technique [16] limits the number of permutations to 1, assuming that the hash value will not collide, which is very unlikely and only depends on the type of data being used. None of these variants considers the semantics while estimating the similarity between items efficiently.

Finding similar documents is one of the problems solved by MinHashing-based approaches [10]. The documents are represented in the form of sets. Since MinHash approximates the Jaccard similarity of the sets, we can compute the lexical similarity of the documents. MinHash has also been used to measure cross-lingual similarity [11]. Wikipedia Corpus is available in multiple languages. The association of multilingual documents has been exploited to obtain similar cross-lingual documents. In the graph summarization, [12], a bias-free MinHash clusters similar nodes together. The similarity of the web documents has been computed using K-shingle tokens with MinHashing [13]. This approach tokenizes the documents into K-shingles so that structurally (non-semantically) similar documents can be grouped.

B. Knowledge-base Approaches

There are several studies on the estimation of the semantic similarity score that depends on the knowledge bases and ontologies like WordNet[1] [4], MeSH[2] [6], BabelNet[3] [5], SENSUS[4] [17], Wikipedia [5]. These knowledge bases are structured hierarchically while preserving the concepts and words' relatedness so that similar words are grouped. Longitude, depth, and density are the main factors that can be translated into semantics for similar terms. Based on factors, semantic measures can be categorized as; Structure-based measures [18], [19], Information content measures [20], [21], Feature-based measures [22], and hybrid measures [23], [24]. These measures give semantic scores between two concepts or words. Structure-based measures [18], [19], relying on the *is-a* relationship in a single ontology, have limitations, as the distance between similar words can vary in the network. Information content methods [20], [21] are computationally expensive, considering both ontology and corpus for the computation of probability-based similarity. Feature-based methods [22] assess similarity by comparing properties or relationships of terms from knowledge bases. Hybrid approaches combine these techniques, but feature-based methods are preferable for cross-ontology similarity. Despite their utility, these methods are computationally expensive to measure semantic similarity. The authors in [23] proposed a graph-based approach, extracting relations from a semantic network for each keyword to compute semantic similarity. However, this method is not exhaustive or scalable. Another study [24] focuses on semantically similar scientific articles, calculating similarity based on synonyms, "is-a" and "part of" relations in titles, abstracts, and keywords. This approach, while limiting the use of information, is also computationally expensive.

[1]https://wordnet.princeton.edu/
[2]https://www.ncbi.nlm.nih.gov/mesh/
[3]https://babelnet.org/
[4]https://www.isi.edu/natural-language/resources/sensus.html
[5]https://dumps.wikimedia.org/

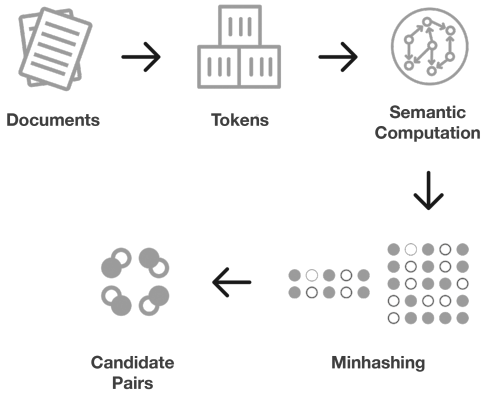

Fig. 2. High-level Overview of the Proposed Solution

C. Corpus-based Approaches

Recently, word embeddings and autoencoders have been in the limelight. It represents a document in the form of vectors, and there are several techniques, some of which are Word2vec [25], Glove [26], and fastText [27]. Word2Vec's [25] embeddings help improve most NLP tasks, but subword information is not captured. FastText [27] uses the word2vec model and improves its efficiency and performance, helping to capture the context of subwords. However, Glove [26] aims to bring word prediction algorithms and word statistics together in the corpus. It considers the corpus's co-occurrence statistics and the effectiveness of prediction-based approaches. These techniques are not contextual, meaning they do not believe the context in which the word is used, hence losing the semantics. ELMo [8] introduced contextualized embedding that captures semantics with context. BERT [28] uses auto-encoders for text representation in the form of vectors. KnowBert [29] is a generic and efficient model for incorporating prior knowledge into a deep neural network. Train entity linkers by self-supervision on unlabeled data, resulting in general-purpose knowledge-enhanced representations that may be used for various downstream applications.

D. Summary

The survey on document clustering based on semantic similarity reveals that most methods employ the term-frequency inverse document-frequency or WordNet as a knowledge base to derive semantic information, utilizing K-means clustering and its variants for document clustering. In particular, none of the techniques explored using the MinHashing technique to compute semantic similarity. MinHashing, typically employed in clustering, can effectively estimate the Jaccard similarity, providing a scalable approach amid rapidly growing data. While machine learning and deep learning models are applied in word-sense disambiguation, they demand extensive pre-training on large corpora. In contrast, Jaccard similarity and MinHash present efficient and scalable alternatives to approximating similarity while preserving semantic information, especially when working with word embeddings represented as vectors in sentence pairs.

III. Methodology

In this section, we discuss the proposed solution as briefly illustrated in Fig. 2. We aim to capture the semantically similar candidate pairs through MinHashing. The knowledge bases like WordNet provide semantic information in synonyms, antonyms, meronyms, etc. These relationships between words can be modeled in such a way that it can increase the bias, which in turn captures the semantics of the documents.

The objective is to provide a scalable solution for finding semantically similar documents by exploiting the relationships of words in the taxonomy. The associations between words such as "is-a", "part-of", and "has-a" provide a basis for exploring the semantics. MinHash provides the advantage of scalability, as each instance is reduced to a fixed-size signature. The following subsections discuss the proposed approach step by step. Fig. 3 shows the flow chart of the process, which is discussed later.

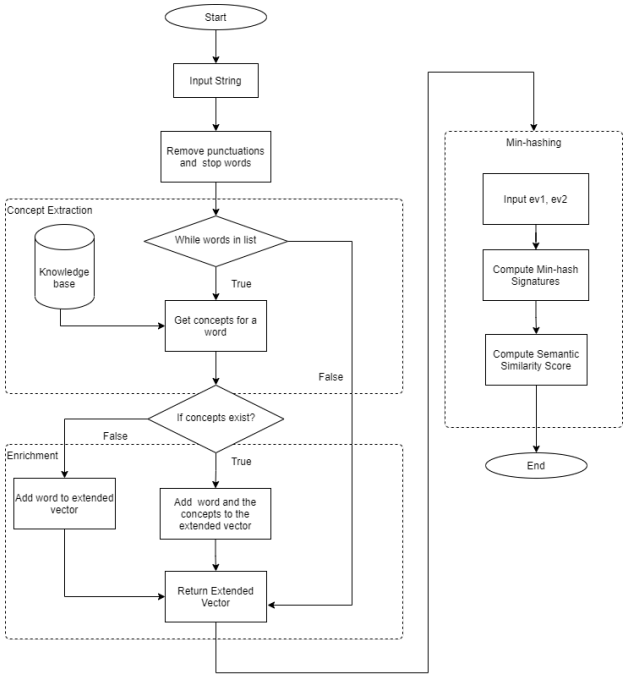

Fig. 3. Flow of the semantic score computation

MinHashing is widely used to calculate the syntactical similarity of documents in information management systems [22], to search encrypted data [30], and to summarize graphs [12]. We can embed semantic information in the original vector using knowledge bases such as WordNet to get the semantic similarity score. MinHash takes data in the form of sets to perform the computation. This can be achieved by transforming documents into words. Assume that we have two documents, i.e., d1 and d2.

d1: ['he', 'is', 'a', 'nice', 'person']
d2: ['he', 'has','a', 'decent', 'soul']

After removing the stop words, the vectors will have words that add meaning to the sentence.

d1: ['nice','person']
d2: ['decent','soul']

We can see that the documents are similar to each other syntactically, approximately 50%. Still, they are 100% similar semantically, meaning that both sentences have the same meaning, but their similarity score is meager. For considering the semantics, we propose to obtain closely related words from the semantic network, as shown in Fig. 4, to be used in the enrichment phase.

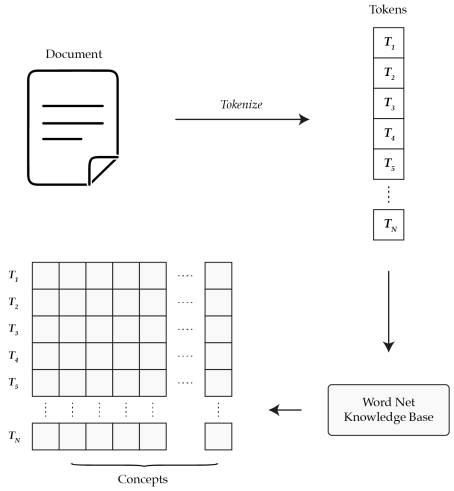

Fig. 4. Concept Extraction Mechanism

This vector is then encoded as a sequence passed to the MinHash algorithm to calculate the scores. For example, the word 'nice' has a synonym 'decent', and the word 'decent' has 'nice' as its synonym; we can see that both words occur in synsets of each other. Similarly, words like 'person' and 'soul' are synonyms in WordNet. Adding these words to the word vector will increase the bias of the semantic terms, which add meaning to a phrase or sentence, causing the collisions in the MinHash and resulting in the increased score in terms of semantics. The semantic similarity of the above sentences can be estimated in three steps. Each of the above steps is discussed below in detail.

A. Step 1: Concept extraction

This phase extracts concepts from a knowledge base to enrich feature vectors. Knowledge bases preserve the semantics between words in the form of different relationships such as synonyms, antonyms, hyponyms, hypernyms, and meronyms, and we call these relationships *concepts*. The feature vector after the pre-processing contains only the meaningful terms, hence removing the bias added due to stop words. The concepts are obtained against each word in the original sentence vector. The number of concepts obtained against each word may vary (it can be zero). Fig. 4 shows the concept extraction phase where related concepts are obtained against each vector. We emphasize the following relationships.

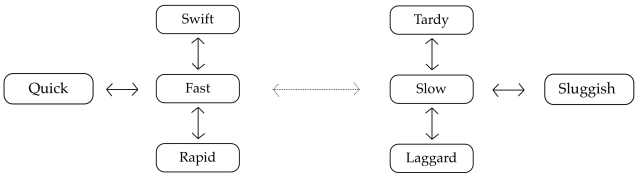

Fig. 5. The illustration of Synonym-Antonym relationship, where solid-line represents synonyms-relation and dotted-line shows the antonyms.

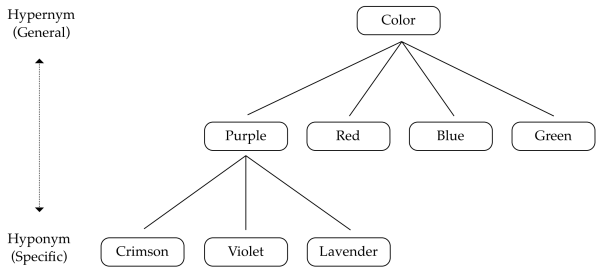

Fig. 6. Hyponym-Hypernym Relationship

- **Synonyms:** Two words having the same meaning are called synonyms of each other, like *big* is the synonym of *large*. WordNet contains synonyms in the form of synsets, which are lists of grouped words with similar meanings. The context changes as we move away from a word by replacing it with its synonyms.
- **Antonyms:** Two words having opposite meanings are called antonyms, such as big is an antonym of small. Some words may or may not have opposite words. Multiple words have the same opposite meaning. The synonym-anonym relationship can be seen in Fig. 5
- **Hyponyms:** The words having a specific example of a general word are hyponyms of each other; for example, color is a general word, and red is a specific color, so red is the hyponym of color. If we consider WordNet as a tree, the leaves of each node are the specific terms of its general form at the node. For this research, we have considered one-hop away hyponyms, as there is much less chance of concepts colliding if we go deeper into specific words, which can be seen through Fig. 6.
- **Hypernyms:** In contrast to hyponyms, hypernyms are the general form of a specific word. For example, color is the hypernym of Red, as it represents a broader meaning, as shown in Fig. 6. Therefore, if different colors are used in a sentence, we can add color for each word to increase semantic similarity. For this research, we have considered concepts up to three hops away from hypernyms, as there is a chance of getting common words for two closely related words.

WordNet is rich in synonyms, as shown in Fig. 7. The number of concepts obtained for each relation against each

token was counted to analyze the resourcefulness of WordNet. It gives a significantly high average of 115 for synonyms, while antonyms are very few. In addition to antonyms, the number of hyponyms and hypernyms is insignificant compared to synonyms.

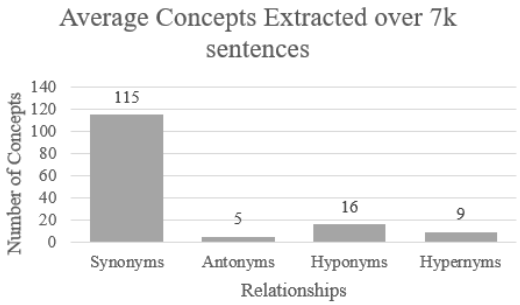

Fig. 7. Average number of concepts obtained for each relation

Each of the above relations is independent of the other; therefore, only one relationship is considered to make a meaningful enriched word vector. In the example below, we obtained synonyms from the extraction process of the words 'decent', 'person', 'nice', and 'soul'. The extracted concepts for *person* are ['individual', 'somebody', 'someone', 'mortal', 'person', 'soul'] and for *soul* are ['individual','somebody','someone', 'psyche','mortal', 'person','soulfulness','soul']. We can see that few synonyms are common for both person and soul.

B. Step 2: Vector enrichment

This phase is responsible for the formation of extended vectors. These vectors are obtained by adding extracted terms to the original vector so that the representation of each token increases by its related terms, as given in the following Eq. 1.

$$\sum_{i=1}^{N} T_i + C_t \tag{1}$$

where C is the concept for Tokens T from 1 to N (N is the number of tokens per document, and t is the size of the concept vector). To remain in the context, a threshold value of t=3 is determined for enrichment, which means that we will use the first three values of the extracted concepts for each word, which results in enriched vectors called *extended vectors*. If two words are related, the extracted word set must have at least one similar word, causing the collision. The process is demonstrated in Fig. 8. The output of this phase is an encoded vector, which is encoded using a straightforward approach, i.e., each word is given a number in a sequence starting from 1, where similar words get the same identification number.

The formation of extended vectors depends on one relationship at a time. Using the synonyms obtained from extraction phase, the resulting extended vectors are: ev1 = ['nice', 'good', 'decent', 'gracious', 'person', 'individual', 'somebody', 'someone'] and ev2 = ['decent', 'good', 'nice', 'dainty, 'soul', 'individual', 'somebody', 'someone']. The extended vectors

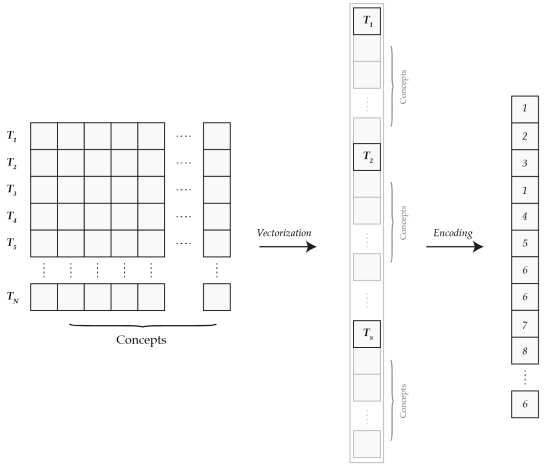

Fig. 8. Enrichment Phase

are now highly similar as each word is present in both vectors, increasing the representation of the similar words. The size of the extended vectors adds to the complexity of the space by a factor of *t*. The size of the extended vector is given by $size(ev) = n + t * n$ where n is the size of the original vector, and t is the threshold that determines the number of concepts to be added to the extended vector. Fig 9 shows the average increase in the extended vector for the relationships used in this research.

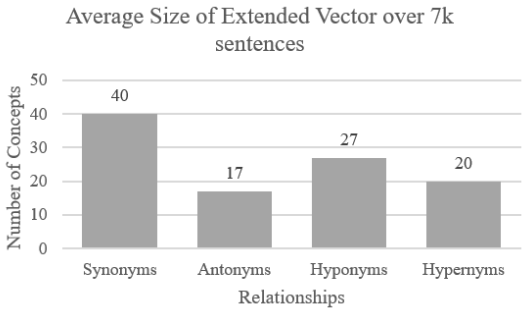

Fig. 9. Average size of extended vectors after vectorization

The size of the extended vector is directly proportional to the threshold, that is, $size(ev) \propto t$ where ev is the extended vector, and t is the threshold that determines the number of concepts used for each token. The higher the threshold, the larger the size of the extended vector, and vice versa. In Fig 9, synonyms give the maximum size of the extended vectors when the threshold is 3, where hypernyms are half the size of the synonyms, and antonyms result in the smallest vector size.

C. Step 3: MinHashing

MinHash algorithm represents large sets into more miniature representations by hashing, therefore significantly reducing the size of the sets to be compared quickly while preserving the similarity of the sets. Leveraging this property of MinHash to reduce the size of the extended vectors and obtain equal-sized

encoded vectors where the size of the vectors depends on the number of hash functions used.

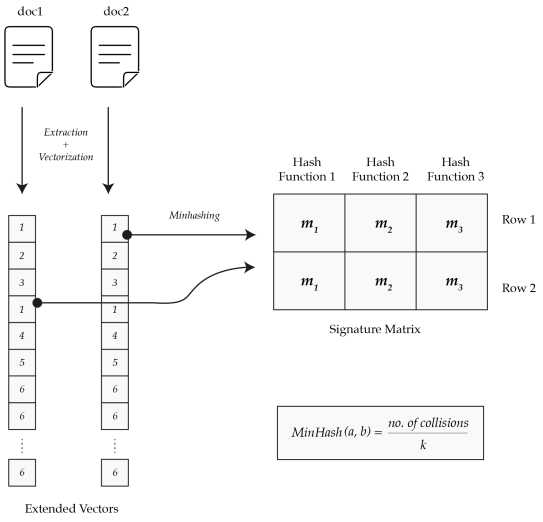

Fig. 10. An overview of MinHashing strategy

The k-length vectors obtained from the MinHash algorithm are then compared to calculate the MinHash semantic similarity score (ms3), which is given by the formula $ms3(sig1, sig2) = \frac{number\ of\ collisions}{k}$ where sig1 and sig2 are a set of integers that preserve the semantics and similarity of the two documents, obtained from MinHash, and k is an arbitrary number of hash functions used to calculate signatures. This score is approximately equal to the Jaccard similarity of the semantic set, ranging between 0 and 1. The extended vectors ev1 and ev2 obtained in the previous section give a significant similarity score of 0.8, which shows that the two sentences are highly similar despite the structure of the sentences, making them different from each other.

We omit the detailed discussion about the proposed algorithms due to space limitations.

IV. Empirical Analysis

A. Datasets

The proposed approach is tested on two viral datasets; one is the Microsoft Research Paraphrase Corpus (MRPC) [31], which consists of 5,800 pairs of sentences in the English language, annotated by humans. The sentences are rephrased, meaning the pair have the same meaning, but their syntactical structure is different. The pairs are annotated in terms of quality. A score of 1 is given to the pair with good paraphrasing, while 0 is assigned to low-quality pairs. Sentence-involving compositional knowledge (SICK) dataset [32] consists of 10,000 pairs of sentences in English. Each pair of sentences is annotated for relatedness in meaning, giving a relatedness score that ranges between 1-5. This score has been normalized using a threshold of 2.5. Values greater than this threshold are classified as 1 and 0 otherwise. It also contains the annotations of entailment that are out of the scope of this report. We used the relatedness score to obtain the performance statistics for our approach.

B. Computing environment

The hardware used is a standard PC with Intel(R) Core(TM) i5-10210U CPU @ 1.60GHz 2.11GHz processor and 16GB RAM running 64-bit Windows 10 for the development and testing of the proposed approach. However, this implementation could be efficiently run on a low-resource machine to compare two documents.

The proposed method is implemented using the Natural Language Processing Toolkit (NLTK)[6] in Python 3[7]. We have used ntlk corpus for WordNet for the concept extraction phase. Other than this regex tokenizer, stop words and n-grams have been used from NLTK. Pandas package is used to read and process data.

Evaluations have been performed using the sklearn metrics library[8]. Evaluation metrics include the accuracy score and the entire classification report that consists of precision, recall, and the F1 score, as well as sensitivity and specificity. Scores ranging from 0-1 are classified as 0 and 1 using a threshold of 0.5.

C. Results and discussion

The results are obtained at different levels, as different relations are exploited to obtain a semantic score. As mentioned in the methodology section, we focus on four relations: synonyms, antonyms, hyponyms, and hypernyms. For each relation, we have obtained the precision, recall, and F1 score along with specificity and sensitivity. These metrics show the performance of our approach on the aforementioned datasets.

TABLE I
Evaluation on Synonyms

Datasets	Measure	Metrics		
		Precision	Recall	F1 Score
MRPC	Sensitivity	0.79	0.61	0.69
	Specificity	0.45	0.66	0.54
SICK	Sensitivity	0.99	0.50	0.66
	Specificity	0.24	0.96	0.38

TABLE II
Evaluation on Antonyms

Datasets	Measure	Metrics		
		Precision	Recall	F1 Score
MRPC	Sensitivity	0.78	0.62	0.69
	Specificity	0.45	0.65	0.53
SICK	Sensitivity	0.99	0.50	0.66
	Specificity	0.24	0.96	0.38

- **Synonyms** Synonymous words have similar meanings. WordNet is rich in synonyms, as shown in Fig. 7. For simplicity, we used $\leq$ three concepts for each word in the original sentence in the enrichment phase. Using this

[6]https://www.nltk.org/
[7]https://www.python.org/downloads/
[8]https://scikit-learn.org/stable/

TABLE III
EVALUATION ON HYPONYMS

Datasets	Measure	Metrics		
		Precision	Recall	F1 Score
MRPC	Sensitivity	0.76	0.68	0.72
	Specificity	0.46	0.56	0.50
SICK	Sensitivity	0.97	0.56	0.71
	Specificity	0.25	0.91	0.39

TABLE IV
EVALUATION ON HYPERNYMS

Datasets	Measure	Metrics		
		Precision	Recall	F1 Score
MRPC	Sensitivity	0.76	0.69	0.72
	Specificity	0.46	0.55	0.50
SICK	Sensitivity	0.98	0.56	0.71
	Specificity	0.25	0.92	0.39

approach, we obtained an accuracy 63% in the MRPC and 56% in the SICK data set. Table I shows the detailed classification report. It shows an F1 score of 0.69 on MRPC and 0.66 on SICK for positive class. F1 score shows that we have relatively low false positives and low false negatives, which means that our approach accurately classifies true positives and true negatives with little error. The precision goes very high to 0.99 on the SICK dataset for synonyms.

- **Antonyms** Antonyms are words that have opposite meanings. WordNet is not very rich in antonyms, as shown in Fig. 7; on average, only five antonyms are obtained for more than 7k sentences, a significantly small number compared to synonyms. The notion behind using the antonyms is that the two words may have the exact opposite meaning word, thus reducing the size of the extended vector. Since the antonyms occurrences are very low, we cannot rely on antonyms for enrichment. However, the antonyms give an accuracy score of 55% and 63% on the SICK and MRPC datasets, respectively. Table II shows the statistics obtained after enrichment with antonyms containing precision, recall, and F1 score. The scores show that there is not much difference between the metrics score of synonyms and antonyms, even when very few are obtained on average for a sentence.
- **Hyponyms** Hyponymy is a relation between words from general to specific. If we consider WordNet as a graph, then the hyponyms are the leaflets of a node. As we move down the graph, the leaves contain more specific terms; for example, red is the hyponym of the word color. So, the notion behind using hyponyms is to use particular terms that are one hop away from the node word. Hypnonyms are pretty reasonable in numbers compared to antonyms but still significantly less than synonyms. We obtained 61% accuracy on SICK and 64% on MRPC. The F1 score has increased to 0.72 and 0.71 for the MRPC and SICK data sets, respectively, as shown in Table III. The improved metrics show that hyponymy is a more meaningful relationship in terms of capturing semantics from WordNet compared to synonyms and antonyms.
- **Hypernyms** Hypernymy is the relationship between words from specific to general. It is the opposite of hyponymy, just as synonyms and antonyms. For example, color is the hypernom of the word Red. This relationship is attractive; two or more leaves may end up in the same word, hence better chances of collision. We can better generalize the terms used in a document using this relationship. However, the accuracy obtained is 61% on SICK and 64% on MRPC, which is similar to the accuracy of the hyponyms. Table IV shows the scores of the metrics, which are also similar to those of hyponyms.

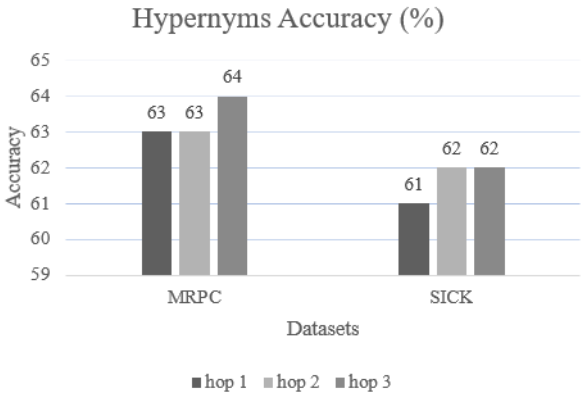

Fig. 11. Accuracy Scores for Hypernyms up to 3 hops away

Since hypernyms can generalize, we have tested our approach on three-hops away hypernyms, which means that three-hops-away words are more general terms. Fig 11 shows the accuracy scores for hypernyms up to 3 hops away. It can be seen that there is no significant change in accuracy. So, we can conclude that the words three hops away from the given word do not significantly affect the semantic similarity obtained from our proposed method.

The comparison of the accuracy scores of all relationships used in this article is shown in Fig. 12. It can be seen that *Hypernyms* give the most consistent result, since they have the same accuracy score of 64% on both datasets, which is also the maximum.

V. CONCLUSION AND FUTURE WORK

Addressing semantic similarity remains challenging due to the traditional trade-off between accuracy and computational complexity. Past knowledge-based approaches lacked a holistic view of document semantics, while corpus-based methods required extensive training on massive datasets. To overcome this, we leveraged MinHash techniques for efficient lexical similarity estimation. Our proposed method enhances document vectors by incorporating knowledge-based concepts, resulting in extended vectors emphasizing similar words. Experimentally, hypernyms demonstrated the most promising result, achieving 64% accuracy on MRPC and SICK data sets with minimal pre-processing.

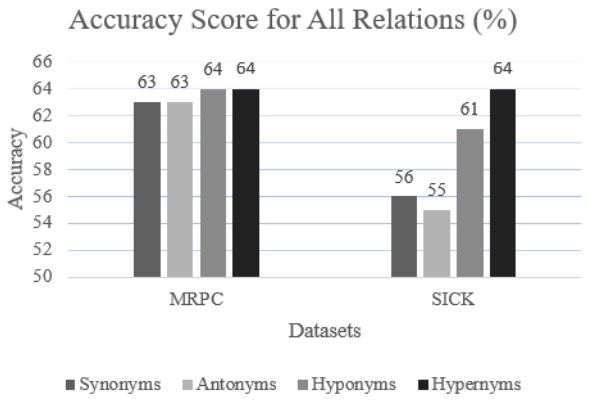

Fig. 12. Accuracy Scores for All Relationships used for Enrichment

Our approach has significant potential for data mining, facilitating semantic analysis, and clustering semantically similar documents to achieve meaningful results. Future directions include exploring additional word relationships, leveraging part-of-speech tags for deeper semantic understanding, and combining multiple relations in a distinct vector space for comprehensive embeddings.

REFERENCES

[1] S. J. Athenikos and H. Han, "Biomedical question answering: A survey," *Computer methods and programs in biomedicine*, vol. 99, no. 1, pp. 1–24, 2010.

[2] M. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," *CoRR*, vol. abs/1508.04025, 2015. [Online]. Available: http://arxiv.org/abs/1508.04025

[3] Z. Zhang, Y. Wu, H. Zhao, Z. Li, S. Zhang, X. Zhou, and X. Zhou, "Semantics-aware bert for language understanding," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 05, 2020, pp. 9628–9635.

[4] G. A. Miller, "Wordnet: a lexical database for english," *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995.

[5] R. Navigli and S. P. Ponzetto, "BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network," *Artificial Intelligence*, vol. 193, pp. 217–250, 2012.

[6] S. J. Nelson, D. Johnston, and B. L. Humphreys, "Relationships in medical subject headings, relationships in the organization of knowledge," *Bean and Green, eds. Kluwer Academic Publishers*, pp. 171–84, 2001.

[7] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[8] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," *arXiv preprint arXiv:1802.05365*, 2018.

[9] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," *CoRR*, vol. abs/1810.04805, 2018. [Online]. Available: http://arxiv.org/abs/1810.04805

[10] A. Z. Broder, "On the resemblance and containment of documents," in *Proceedings. Compression and Complexity of SEQUENCES 1997 (Cat. No.97TB100171)*, 1997, pp. 21–29.

[11] F. Ture, T. Elsayed, and J. Lin, "No free lunch: Brute force vs. locality-sensitive hashing for cross-lingual pairwise similarity," in *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 01 2011, pp. 943–952.

[12] K. U. Khan, W. Nawaz, and Y.-K. Lee, "Set-based unified approach for summarization of a multi-attributed graph," *World Wide Web*, vol. 20, no. 3, pp. 543–570, 2017.

[13] M. Manna and G. Abdulameer, "Web documents similarity using k-shingle tokens and minhash technique," *Journal of Engineering and Applied Sciences*, vol. 13, pp. 1499–1505, 05 2018.

[14] W. Wu, B. Li, L. Chen, J. Gao, and C. Zhang, "A review for weighted minhash algorithms," *IEEE Transactions on Knowledge and Data Engineering*, vol. PP, pp. 1–1, 09 2020.

[15] P. Li and C. König, "B-bit minwise hashing," in *Proceedings of the 19th International Conference on World Wide Web*, ser. WWW '10. New York, NY, USA: Association for Computing Machinery, 2010, p. 671–680. [Online]. Available: https://doi.org/10.1145/1772690.1772759

[16] A. Shrivastava and P. Li, "Densifying one permutation hashing via rotation for fast near neighbor search," in *Proceedings of the 31st International Conference on Machine Learning*, ser. ICML'14. JMLR.org, 2014.

[17] K. Knight and S. K. Luk, "Building a large-scale knowledge base for machine translation," in *AAAI*, vol. 94, 1994, pp. 773–778.

[18] C. Leacock and M. Chodorow, "Combining local context and wordnet similarity for word sense identification," *WordNet: An electronic lexical database*, vol. 49, no. 2, pp. 265–283, 1998.

[19] R. Rada, H. Mili, E. Bicknell, and M. Blettner, "Development and application of a metric on semantic nets," *IEEE transactions on systems, man, and cybernetics*, vol. 19, no. 1, pp. 17–30, 1989.

[20] P. Resnik, "Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language," *Journal of artificial intelligence research*, vol. 11, pp. 95–130, 1999.

[21] J. J. Jiang and D. W. Conrath, "Semantic similarity based on corpus statistics and lexical taxonomy," *arXiv preprint cmp-lg/9709008*, 1997.

[22] E. Petrakis, G. Varelas, A. Chliaoutakis, and P. Raftopoulou, "X-similarity: Computing semantic similarity between concepts from different ontologies." *JDIM*, vol. 4, pp. 233–237, 12 2006.

[23] P. Chahal, M. Singh, and S. Kumar, "An ontology based approach for finding semantic similarity between web documents," *International Journal of Current Engineering and Technology*, vol. 3, no. 5, pp. 1925–1931, 2013.

[24] M. Nasab and R. Javidan, "A new approach for finding semantic similar scientific articles," *Journal of Advanced Computer Science & Technology*, vol. 4, 12 2015.

[25] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *arXiv preprint arXiv:1310.4546*, 2013.

[26] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[27] A. Joulin, E. Grave, P. Bojanowski, M. Douze, H. Jégou, and T. Mikolov, "Fasttext. zip: Compressing text classification models," *arXiv preprint arXiv:1612.03651*, 2016.

[28] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[29] M. E. Peters, M. Neumann, R. L. Logan IV, R. Schwartz, V. Joshi, S. Singh, and N. A. Smith, "Knowledge enhanced contextual word representations," *arXiv preprint arXiv:1909.04164*, 2019.

[30] M. Kuzu, M. S. Islam, and M. Kantarcioglu, "Efficient similarity search over encrypted data," in *2012 IEEE 28th International Conference on Data Engineering*. IEEE, 2012, pp. 1156–1167.

[31] W. B. Dolan and C. Brockett, "Automatically constructing a corpus of sentential paraphrases," in *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*, 2005. [Online]. Available: https://aclanthology.org/I05-5002

[32] M. Marelli, S. Menini, M. Baroni, L. Bentivogli, R. Bernardi, R. Zamparelli *et al.*, "A sick cure for the evaluation of compositional distributional semantic models." in *Lrec*. Reykjavik, 2014, pp. 216–223.