Multi-sense Embeddings Using Synonym Sets and Hypernym Information from Wordnet

Krishna Siva Prasad Mudigonda*, Poonam Sharma

Department of Computer Science and Engineering, Visvesvaraya National Institute of Technology, Nagpur (India)

Received 17 November 2019 | Accepted 8 May 2020 | Published 2 July 2020

Abstract

Word embedding approaches increased the efficiency of natural language processing (NLP) tasks. Traditional word embeddings though robust for many NLP activities, do not handle polysemy of words. The tasks of semantic similarity between concepts need to understand relations like hypernymy and synonym sets to produce efficient word embeddings. The outcomes of any expert system are affected by the text representation. Systems that understand senses, context, and definitions of concepts while deriving vector representations handle the drawbacks of single vector representations. This paper presents a novel idea for handling polysemy by generating Multi-Sense Embeddings using synonym sets and hypernyms information of words. This paper derives embeddings of a word by understanding the information of a word at different levels, starting from sense to context and definitions. Proposed sense embeddings of words obtained prominent results when tested on word similarity tasks. The proposed approach is tested on nine benchmark datasets, which outperformed several state-of-the-art systems.

Keywords

Hypernym Path, Multi-sense Embeddings, Synonym Sets, Word Embeddings, Word Similarity.

DOI: 10.9781/ijimai.2020.07.001

I. Introduction

Classical problem like semantic analysis continues to grab the attention of researchers since it is important to different fields of study. Semantic similarity has gained importance in fields like information retrieval [1], bio-medical domains [2], creating knowledge graphs [3], sentence clustering [4], cross-lingual text similarity [5]. Primitive semantic similarity measures are based on the distance between the concepts. Measures like Jaccard distance [6], Euclidean distance [7], and dice coefficient [8] are examples of it. These measures are not suitable for natural language processing activities as there is no prior knowledge taken into account for assessing the similarity. Researchers use the measures which consider background knowledge and the relation between the concepts to assess semantic similarity.

Knowledge-based measures, which are one of the kinds of semantic similarity measures use relations between the concepts. State-of-the-art knowledge-based measures use a lexical database like Wordnet [9] to measure semantic similarity. Wordnet is a structured organization of terms, referred to as synonym sets or concepts. Knowledge-based measures are formulated based on distance, depth or information content of concepts. Measures proposed by Rada [14], Wu and Palmer [17], Leacock and Chodorow [11], Li [12] specify different ideas of using the distance between the concepts to calculate the semantic similarity. The measure proposed by Li [12] uses distance and depth between the concepts as metrics to evaluate the similarity between the concepts. Resnik [15] calculates similarity by using the information content of the lowest common subsumer between the concepts. Lin [13], Jiang and Conrath [10] also use the information content between the concepts. Zhu [16] proposed a hybrid measure by combining the Resnik measure [15] and path measure [14]. Zhu [16] extracted semantic similarity by using knowledge-based measures on knowledge graphs like YAGO [38]. These measures do not discuss vector representations. These measures are not substantially strong to perform present-day NLP activities. Hence, the measures which address the distributional semantics of words are required in the current era.

In recent years, distributional semantic representations of words gained popular attention. Distributional representations of words well known as word embeddings set the state-of-art systems for core natural language processing activities (NLP). Word embeddings [18], [19], represent words or phrases with vectors to extract relationships between them. Word embedding techniques improved the efficiency of various NLP tasks, like word similarity [20], sentiment analysis [21], [22], text classification [23], question answering tasks [24].

Even though word embedding techniques improved the accuracy of core NLP tasks, these approaches fail to address the polysemy problems [25]. Research related to polysemy, dealing context, and senses of words is needed in the present scenario. Based on the word's sense and context, recent research is trying to improve the semantic representations. Word embedding techniques generally neglect to explore lexical structures with valuable semantic relations from lexical databases like Wordnet [9]. To improve the efficiency of word embeddings, word vectors should consider different contexts and multiple senses of words. Multi-sense embeddings require a well-formed semantic network to extract multiple senses of the word, alongside word vector representations.

Semantic similarity requires the knowledge of concepts that are obtained from background sources. Well defined structures like Wordnet are limited, yet this knowledge cannot be ignored.

* Corresponding author.
E-mail address: krishnasivaprasad536@gmail.com

Please cite this article in press as:
K. S. Prasad Mudigonda, P. Sharma. Multi-sense Embeddings Using Synonym Sets and Hypernym Information from Wordnet, International Journal of Interactive Multimedia and Artificial Intelligence, (2020), http://dx.doi.org/10.9781/ijimai.2020.07.001

Hence, sources like Wikipedia and Wordnet can be combined to obtain the context. Wordnet establishes the relationship between the concepts using hierarchical relations. Each concept in Wordnet has synonym sets, which help to predict the multiple senses of words. Hence, this paper develops semantic vector representations of words considering their context, sense, definitions, and hypernym path using the lexical structure of Wordnet and word embedding techniques like Word2vec models.

The following are the contributions of this paper:

- This paper uses the rich semantic structure of Wordnet to generate the senses and definitions of each word.
- Regarding each word, suitable contexts are generated from a large source of Wikipedia text.
- Multiple sense based word embeddings are generated.
- This paper aims at developing parts-of-speech related synonym set embeddings to carry out word similarity tasks.
- Detailed comparison and analysis of state-of-art techniques are mentioned.
- Nine benchmark datasets are compared to portray the significance of the proposed approach.

The paper is organized as follows. Section II gives the related work, which covers the existing word embedding techniques. Section III gives the proposed approach and covers the senses extracted from Wordnet, context from Wikipedia and word vector representations. In section IV, experimentation details of existing measures and the proposed measure are mentioned, followed by discussion in section V. Section VI gives the conclusion of the work done and recommends some suggestions for future work.

II. Related Work

This section discusses the existing distributed representations of words. Distributed representations of words proposed by Mikolov et al, [18], [19] become popular among NLP related tasks. Working with contexts has its roots with the distributional hypothesis proposed by Harris [26]. After that, the bag-of-words [27] approach also discussed the distributed representations of words, but these approaches suffer from drawbacks like data sparsity, not maintaining word order, and dimensionality related issues. Approaches based on language prediction also exist in the literature of the NLP [28]. The language models are transformed with continuous bag-of-words and skip-gram models.

Mikolov et al, [18], [19] proposed Word2vec with continuous skip-gram and continuous bag-of-words (CBOW) which portrayed the importance of word embeddings. This approach gained importance due to its efficient log-linear neural network language model, low-dimensionality vector representations. Skip-gram and CBOW models produce vector representations of words. Word embedding representations like SENNA [29], GloVe [30], and fastText [31] exist in the literature. All these are single vector representations. These models fail to address the polysemy of the words. Word vector representations can be enhanced by combining multiple senses of words. Examining the sense, context, and definitions of words while deriving word embeddings improves the efficiency of the NLP tasks. The proposed approach is developed using the Word2vec model to train the corpus and later the vectors are used based on context, sense, and definitions of each word.

Researchers after the invention of distributional semantic representations of words developed sense embeddings [25], [32], [33], [34], [35] to perform NLP tasks like word similarity. Interestingly, Li [25] presented the idea of developing separate vectors for each sense. The approach mentioned by Iacobacci et al, [32] obtains sense embeddings using lexical resource BabelNet [52] for measuring semantic similarity. This approach is an effective measure for word similarity tasks, yet this can be enhanced by making the model understand the context of words. Chen et al. [33] presented a model for word sense representation. This model considers a single representation per single sense. This model is considered as the basis for the innovation of presenting the word sense disambiguation with embedding techniques for similarity tasks. This model does not learn the relationships like hypernyms and hyponyms between the concepts from Wordnet. The model presented by Oele et al. [34], combines word-sense, context, and word-definition to develop embeddings. This approach [34] developed lexeme embeddings for senses using the Lesk algorithm [36] and an AutoExtend [37] training procedure. The approach mentioned by Ruas et al, [35] disambiguates the text using a context window. The authors [35] explain the limitations of single vector representations and the advantages of multi-sense embeddings. This model [35] derived synonym set based embeddings by integrating Wordnet synonym sets and Word2vec model [18] model.

Word embeddings mechanisms with single vector representations present all the senses in a single vector. There is limited work done in this area. Hence, this paper focuses mainly on generating vector representations for each sense and the context of words is selected from top-n context-rich sentences from Wikipedia. Word embedding models capture the taxonomic information of words but fail in capturing hyponymy and entailment relations. To understand the relation between two words, the word`s synonyms and hyponyms play a crucial role. The above mentioned works in the literature [32-35] developed multi-sense embeddings but with little attention paid on synonym and hypernym relations. This paper projects the importance of the Wordnet hierarchy for understanding synonym sets, hypernyms, lowest common subsumer of concepts to generate multi-sense embeddings for word similarity tasks.

The next section portrays the proposed approach for generating multi-sense embeddings based on the word's synonyms, parts-of-speech, hypernyms. In the next section, a detailed exploration of Wordnet is presented at the start, followed by the explanation of working with multi-sense embeddings.

III. Proposed Methodology

In this section, the main idea of multi-sense embeddings using Wikipedia data and lexical database Wordnet is discussed. This section discusses developing Wikipedia corpus at the start, which is used by the proposed algorithm. The methodology presented in this section has two main tasks: (i) exploring Wordnet lexical structure (ii) using knowledge of synonym sets, hypernyms to produce multi-sense embeddings. The last subsection applies the multi-sense embeddings developed to perform the word similarity tasks.

A. Pre-Processing of Wikipedia Articles

The initial process is to transform the Wikipedia articles into a corpus and then map the words with synonym sets of Wordnet. Initially, the Wikipedia dump is preprocessed to form the corpus. The latest Wikipedia dump is around 15.7 GB of size and has 4,677,566 documents with more than 19,508,987 articles. The articles in the dump are in the XML format, which needs to be converted to text format. This research converts the Wikipedia dump into text format. Articles shorter than 50 words, article name starting with numbers, articles which are not standard words are pruned. From the text, punctuation and other unknown symbols are removed. The tokens which occurred more than ten times in the articles are preserved, and rest is removed since it is challenging to obtain their context. After cleaning the entire text, more than 2 million unique tokens are formed.

Later, these tokens are trained using the Word2vec model. To train the model, this paper uses the Word2vec with default hyperparameters. CBOW model of Word2vec [18] is the training algorithm, with a window size of 15 and vectors of size 300 dimensions (300 d) are the parameter specifications to generate embeddings. After this, all the possible senses obtained from Wordnet also are assigned vectors from the trained Word2Vec model. For each sense of the word, 300 dimensional vectors are generated. This paper proposes an efficient and simple training phase with minimal hyperparameters to understand the sense embeddings. Once this training phase is completed, Wordnet is explored to understand to generate all possible senses of a given word.

B. Exploring Wordnet Hierarchy

One of the main relationships that can exist among the concepts in Wordnet is synonymy. Wordnet has 117,000 synonym sets (synsets), and each of these is linked to one another using conceptual relations. A word may have many forms, every form has a distinct synset and these are uniquely represented in Wordnet. The relation between the synsets is an ancestor-child relation, this kind of relationship is referred to as hyponymy or IS-A relation.

Polysemous words can be used in many senses; the synonym set of a concept refers to a particular sense. If two words are similar, then they share a common synset in the network. Traditional knowledge-based measures [11]-[16] handle the polysemy and synonymy of the words using the taxonomic structure of the semantic network. These measures derive the synonym sets of the words using the hierarchical semantic relationships. Any model developed using ontologies for deriving semantic similarity should be capable of exploring different kinds of relations like hierarchical (IS-A), has-a, hypernyms, etc. The hierarchy between the concepts is a significant relationship as it maps the category of objects into a taxonomy.

The hierarchical structure of the concepts is provided by taxonomy. Each taxonomy concept is represented as a node, and the nodes are connected based on hierarchical relations in the network. The edges of the concepts represent the semantic relationship. WordNet [12] is a semantic network of English words. In this network, each word is represented as a synset. The noun network in WordNet was developed very richly using hyponymy/hypernymy hierarchy.

The maximum depth of the noun hierarchy in this network is 16 nodes. The noun network includes nine types of relations; synonymy relation, hyponymy (IS-A) relation, and its inverse hypernymy, and six meronymic (PART-OF) relations. Synonymy relations account for 80 percent of the relations. Hyponymy relation between the words articulates IS-A relationship between two words and the inverse of this relation is hypernymy. COMPONENT-OF, MEMBER-OF, SUBSTANCE-OF and their inverses are meronymic relations.

Fig. 1 shows an example of Wordnet's "is-a" relation structure. The concepts in Wordnet are arranged in the hierarchical structure. From Fig. 1, it can be noticed that the root of all concepts is 'Entity.' The figure shows only a fragment of Wordnet structure. The leaves 'mammal,' 'tree,' of the tree is under 'organism.' The leaf node 'mammal' is-a 'vertebrate' and 'vertebrate,' is-a 'chordate.' The concept 'chordate,' is-a 'animal.' Similarly, 'carrot' is-a 'root,' and 'root' is-a 'plant-organ,' which is-a 'plant-part.' This organized representation in Wordnet helps in obtaining various senses of concepts. Fig. 1 represents the basic hierarchical cover of three concepts 'tree,' 'mammal,' and 'carrot.' This figure does not cover the various classifications under the concepts like 'organism' or under 'plant.' Once again, the concept like 'mammal' has various subconcepts like cat, dog etc. All these are well organized in the Wordnet.

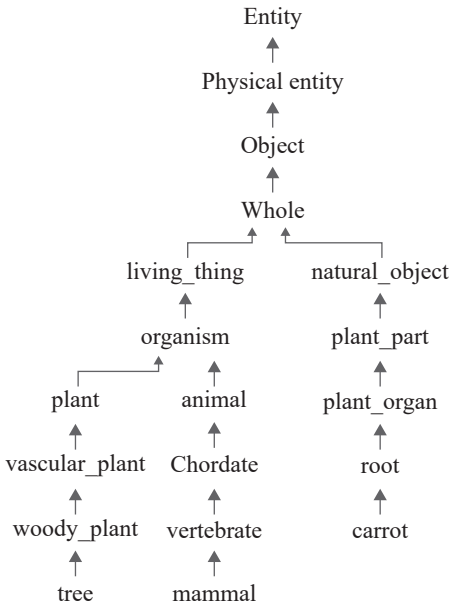

Fig. 1. Wordnet is-a relation example.

A concept is a synset of Wordnet that exists in the form as 'concept.pos.number.' The first part in the synset represents the concept, pos represents the possible parts-of-speech (noun, verb, adjective and adverb), and the number indicates the number of applicable parts-of-speech that exist for the concept. Consider the concept 'advance,' the possible synsets for the concept is shown in Fig. 2. From Fig. 2, it can be noted that the concept 'advance' is having a relationship with six noun synsets, twelve verb synsets, and two adjective synsets. Each synonym set in Fig. 2 is a different sense of the word 'advance.'

Hypernyms in Wordnet specify the hierarchical structure of a concept and from the hypernyms of the two concepts, common ancestors can be checked. Checking the path of the hypernyms of two concepts helps in recognizing additional senses of word pairs. Hypernyms of two concepts 'accentuate' and 'highlight' are shown in Fig. 3. From the figure, it can be noticed that the concepts have more common ancestors. The length of the hypernym path for the concept 'accentuate' is 9, and that

```
Synset('progress.n.03'), Synset('improvement.n.01'),  Synset('overture.n.03'),
Synset('progress.n.02'), Synset('advance.n.05'),  Synset('advance.n.06'),

Synset('advance.v.01'), Synset('advance.v.02'),  Synset('boost.v.04'),
Synset('promote.v.02'), Synset('advance.v.05'),  Synset(gain.v.05'),
Synset('progress.v.01'), Synset('advance.v.08'),  Synset('promote.v.02'),
Synset('advance.v.10'), Synset('advance.v.11'),  Synset('advance.v.12'),

Synset('advance.s.01'), Synset('advance.s.02')
```

Fig. 2. Possible synonym sets for the concept 'advance' obtained from Wordnet.

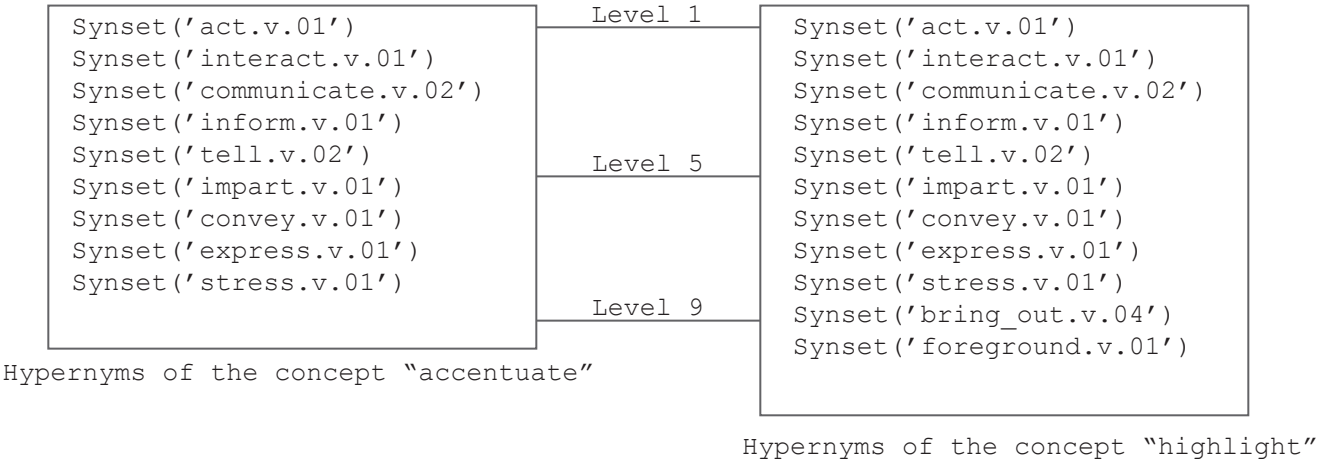

Fig. 3. Hypernym paths for concepts 'accentuate' and 'highlight'.

of 'highlight' is 11. Ancestors of concept 'accentuate' are up to level 9, the depth of the concept from the root is 9, and that of 'highlight' is 11 from the root. Both concepts have the same root up to level 9. Hence, these concepts are more similar. Wordnet provides different modules to obtain synonym sets, hypernyms, and other relations between the concepts. The proposed approach uses the information of synonym sets of a word, hypernym path of the words to generate the appropriate synset, hypernym path based sense embeddings for evaluating word similarity. Another important information this paper considers to derive multi-sense embeddings while performing the word similarity task is least common subsumer (LCS). Least common subsumer (LCS) is the least possible ancestor of the two concepts in the taxonomy. With the concepts represented in Fig. 1, it is not possible to understand entirely about LCS. There are many underlying concepts under a category, hence a more complex tree structure is shown in Fig. 4, to understand the LCS of the concepts.

Fig. 4 is an instance of Wordnet taxonomy. In Fig. 4, A is the root node and every node is a concept (represented as the synonym set) of Wordnet. Distance between two nodes is the shortest path possible from one node to another. Consider nodes H and L, the length between H and L is 6, least common subsumer of H, L is A. Similarly, LCS(F, L) is C and LCS(E, I) is B.

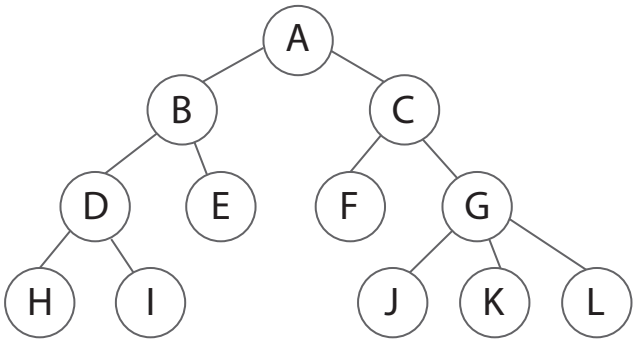

Fig. 4. A sample Tree in WordNet.

C. Methodology

In this subsection, the process of generating the multi-sense embeddings using synonym sets and hypernyms is presented. Fig. 5 shows the block diagram of the proposed approach. Multi sense embeddings are generated using the Wordnet and Wikipedia text. Extracting multiple senses of a word is the crucial step in the entire process. In order to extract the senses, synonym sets and hypernym path of the word from Wordnet is used.

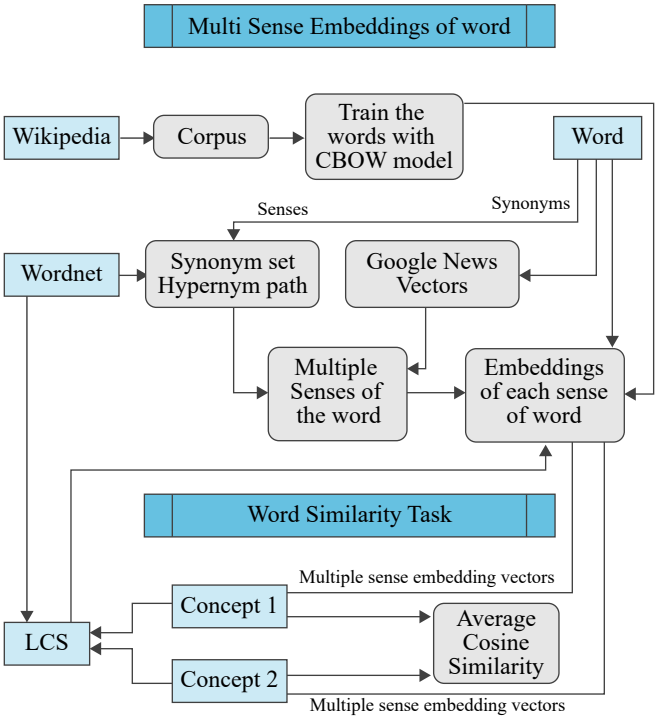

Fig. 5. Block Diagram of generating Multi-sense Embeddings and performing Word similarity task.

There are some concepts with fewer or zero synsets in Wordnet. This paper handles the words with fewer synsets using Google News vectors[1] to derive synonyms.

The similarity of the synonyms with the respective word is calculated. Synonyms which are in certain threshold are considered as possible senses of the respective word. Once the senses are obtained, each sense is converted into a 300-dimensional vector from the pre-trained Wikipedia CBOW model. The words in the corpus are trained on Wikipedia text using the CBOW model as it is an efficient model and deals with the context of words in more massive datasets quite well. The block diagram also shows how to perform the word similarity task. The similarity of the two words is obtained using their multi-sense embedding vectors. Since there are multiple sense vectors for each word, the average cosine similarity between them is calculated to get the final similarity. This paper also inspects another similarity

1 https://code.google.com/archive/p/word2vec/

metric called Tanimoto similarity [42] along with cosine similarity. The discussion about these metrics is mentioned in the next subsection. The block diagram shows how to obtain the multi-sense embeddings of a single word and similarity between two concepts. For the entire corpus with more than 2 million tokens, the same procedure is applied.

Once the information of synonym sets and hypernyms path are extracted from Wordnet, the next step is to generate the sense embeddings. The approach mentioned in this section is not explored in any of the previous works, the results obtained with this approach highlights the efficiency of this work. Algorithm I, Sense Embedding using Wordnet Synonym sets, gives the idea of generating sense embedding of a word C_i.

Algorithm I: Sense Embedding using Wordnet Synonym sets: Procedure SenseEmbed(C_i)

Input: Word (C_i), preprocessed Wikipedia Corpus, Wordnet, context of the word.

Output: Multi-Sense embeddings of the word

Step 1: Generate the synonym sets for concept C_i,

$SS(C_i) = [s_1, s_2, s_3, ...s_n]$, n synonym sets for C_i

$SS\_Preserve(C_i) = \{\}$, empty set

Step 2: Group synonym sets based on parts-of-speech.

Step 3: Preserve the most suitable sense by checking the definitions of each synonym set in Wordnet

```
pos ← POS(word (C_i)) from context
SS_POS ← pos(SS(C_i))
for j in 1 to len(SS_POS):
    if def(SS_POS[j]) matches context:
        SS_Preserve(C_i) ← SS_POS[j]
        SS_Preserve(C_i) ← lemmas(SS_POS[j])
    end if
end for
```

Step 4: Generate the hypernym path of each synonym set to add appropriate senses.

```
for j in 1 to n:
    Hypernyms = synsets(hypernym_path(s_j))
    if hypernym_path(s_j) exists:
        for q in 1 to len(Hypernyms):
            if Similar( Hypernyms[q], word(C_i)) >= δ
                Add Hyper to SS_Preserve(C_i)
            end if
        end for
    end if
end for
```

Step 5: Extract top-m related synonyms of C_i

```
top_m(Syn)<-Google News vectors
```

Step 6: Update $SS(C_i)$ of C_i

```
for k in 1 to m:
    if top_m(Syn) not in SS(C_i)[k]
        Add item to the candidates of SS_Preserve(C_i)
    end if
end for
```

Step 7: Generate word vector representations for each sense in $SS\_Preserve(C_i)$ using trained Word2vec representations.

```
for k in 1 to len(SS_Preserve(C_i)):
    SEV(C_i) ← Word2Vec(SS_Preserve(C_i)[k]
end for
```

Algorithm I, SenseEmbed(C_i), takes a word (C_i), preprocessed Wikipedia corpus, Wordnet as input. The output of the algorithm is multi-sense embeddings $SEV(C_i)$ of the word C_i. The articles in Wikipedia are preprocessed by removing punctuations, HTML tags, followed by removing stop words from the document. The words are then lemmatized to their root form. A corpus of Wikipedia words is formed after this. Wikipedia corpus is an input for the algorithm to check the word for which multi-sense embeddings need to be generated is present or not. Lexical database Wordnet and the context of the word are other important inputs to the algorithm. Context usually is a sentence or a sequence of words in which the required word (C_i) is present. Based on context, different related synonyms of the sense are extracted. Algorithm I generates multi-sense embedding vectors as output for the given input word (C_i) based on seven steps.

The first part of the algorithm from steps 1 to 4 majorly relies on Wordnet to extract multiple senses of the word. After step 1 in the algorithm, the possible synsets of the word are in $SS(C_i)$. Appropriate synsets of the words are stored in $SS\_Preserve(C_i)$, which is empty at the start and is updated in steps 3, 4 and 6. In step2, the synonym sets are grouped based on their parts-of-speech and the tagging of synonym sets is removed. Grouping synonym sets based on their parts-of-speech helps to obtain synsets required for the context.

In step 3, POS(word (C_i)) is an internal procedure that returns the parts-of-speech of the input word. Based on this, the synsets of the word are selected and stored in *SS_POS*. The synsets definitions are matched with the context of the word to update $SS\_Preserve(C_i)$. To explain in further consider the words shown as examples in Table I. Table I shows the information of two words, 'tiger' and 'sofa.' For the word 'tiger,' a context, synsets and definitions of synsets are mentioned in the table. Similarly, for the word 'sofa,' synsets and lemmas are mentioned in Table I. The context of 'tiger,' matches the definition of 'tiger.n.02', and the lemmas of this sense are added to $SS\_Preserve(C_i)$. The lemmas of a synset are the synonyms of the word in a particular sense. There are certain cases where the algorithm has to handle the words with no context. For these words, the algorithm considers all the synsets and their lemmas. The second word 'sofa' in the table has only one synset and the context of this word is not given. The lemmas of the words are added to $SS\_Preserve(C_i)$.

TABLE I. Synsets, Definitions and Lemmas for Example Words 'Tiger' and 'Sofa' from Wordnet

Word ------- 'tiger'	
Context	'tiger of snows'
Synsets	Synset('tiger.n.01'), Synset('tiger.n.02')
Definition of 'tiger.n.01' Definition of 'tiger.n.02'	'a fierce or audacious person' 'large feline of forests in most of Asia having a tawny coat with black stripes; endangered'
Lemmas of 'tiger.n.02'	'tiger', 'Panthera_tigris'
Hypernyms of 'tiger.n.02'	Synset('big_cat.n.01')
Hypernym path of 'tiger.n.02'	'entity.n.01'→'physical_entity.n.01'→ 'object.n.01' → 'whole.n.02' → 'living_thing.n.01' → 'organism.n.01' → 'animal.n.01' → chordate.n.01' → 'vertebrate.n.01'→ 'mammal.n.01' → 'placental.n.01' → 'carnivore.n.01' → 'feline.n.01'→ 'big_cat.n.01'→'tiger.n.02'
Word ------ 'sofa'	
Context	----
Synsets	Synset('sofa.n.01')
Definition of 'sofa.n.01'	'an upholstered seat for more than one person'
Lemmas of 'sofa.n.01'	'sofa', 'couch', 'lounge'

There are certain words with fewer or no synsets in Wordnet, for these words hypernyms and hypernym paths are considered to extract possible senses. Step 4 discusses these details. The word 'tiger' in Table I, has only two senses with only one sense is matching the context. At the same time, the lemmas (synonyms) of the word are also two. Hence, navigating through the hypernym path of such words, helps in increasing the multiple senses. The hypernym path of synset 'tiger.n.02' is shown in Table I, from the path it is observed that the head portion of the path contains more generalized synsets. But the synsets which are two or three levels up the synset 'tiger.n.02' are more specific to the synset. Hence, these senses are also examined to update them in *SS_Preserve*(C_i). To determine, whether these can be added or not, cosine similarity between the synset in the hypernym path and word are determined. If the similarity is greater than δ, the synset is added to *SS_Preserve*(C_i). The value this paper considers for the δ is 0.5.

Steps 5 and 6 in Algorithm I handle the words with no synsets and hypernym paths in Wordnet by extracting synonyms of the word using pre-trained Google News vectors. From step 5, top-m related synonyms of Ci are extracted. The value of m chosen is 10. In step 6, these are added to *SS_Preserve*(C_i).

After step 6, *SS_Preserve*(C_i) contains synsets, lemmas (synonyms) and certain synsets from hypernym path or synonyms from Google News vectors. As the synonym sets are in the form 'concept.pos. number,' it is not possible to generate the embeddings for words with these representations hence tagging is removed. For each sense present in *SS_Preserve*(C_i), a vector of 300 dimensions is obtained from the trained model discussed in subsection A. The context sliding window in this approach is similar to the one used in CBOW [18]. The word vector representations for each sense of *SS_Preserve*(C_i) are stored in *SEV*(C_i).

The vector representations stored in *SEV*(C_i) are based on the synsets, lemmas (synonyms), hypernym path information from Wordnet, and synonyms extracted from Google News vectors. These vectors can be used in word similarity, analogy, entailment tasks. In the next subsection, the multi-sense embeddings are used to carry out word similarity tasks.

D. Word Similarity Task using Multi-Sense Embeddings

In the previous subsection, the procedure to obtain multi-sense embedding representations of a word is presented. This subsection uses the methodology of multi-sense embeddings to carry out word similarity tasks. For two concepts or words, C_i, C_j, Algorithm I generates the multi-sense vector representations as,

$$SEV(C_i) = [s_{11}, s_{12}, \ldots . s_{1n}]$$
$$SEV(C_j) = [s_{21}, s_{22}, \ldots . s_{2m}]$$

Where C_i, C_j represents words, s_{1i} represents sense embedding vector 'i' of word C_i and s_{2j} represents the sense embedding vector 'j' of C_j. In the above representations, the number of senses for concepts C_i and C_j are n and m, and both greater than one.

Algorithm II obtains the similarity of two words C_i, C_j, using *SEV(C)*, *SEV*(C_i) vectors. The initial step in the algorithm is to generate Multi sense embedding vectors of the concepts using Algorithm I. The next step is to update *SEV*(C_i), *SEV*(C_j). The idea here is to utilize all the related senses of the words to derive similarity. Wordnet is searched for the least common subsumer (LCS) of words C_i, C_j. LCS is the shared ancestor of two concepts.

Using LCS information of C_i and C_j, *SEV*(C_i), *SEV*(C_j) are updated. Step 2 in Algorithm II updates the sense vectors by checking the path in between LCS to C_i. The list *Synsets*(C_i, *LCS*) in the algorithm holds all the synsets in the path from C_i to LCS. The synsets which are

Algorithm II : Word Similarity using Sense Embed vectors and Wordnet information:: WordSimilarity(C_i,C_j)

Input: Words C_i, C_j,

Output: Similarity of C_i, C_j

Step 1: Multi sense vectors for C_i, C_j

SEV(C_i) ← SenseEmbed(C_i)

SEV(Cj) ← SenseEmbed(C_j)

Step 2: Update *SEV*(C_i), *SEV*(C_j) *using LCS information from Wordnet, hypernym path of both concepts*

```
if LCS(C_i, C_j) is TRUE:
    Synsets(C_i, LCS) ← Path(C_i,LCS)
    for j in 1 to len(Synsets(C_i, LCS)):
        if Similar(Synsets(C_i, LCS) [j], C_i) >= δ
            Add Vec(Synsets(C_i, LCS) [j]) to SEV(C_i)
        end if
    end for
end if
// Repeat above for SEV(C_j), C_j
    else
    path1← hypernym_path(C_i)
    path2 ← hypernym_path(C_j)
    for k in len(path1):
        for l in len(path2):
            if path1[k]== path2[l]
                if path1[k] not in SEV(C_i)
                    update SEV(C_i)
                end if
                if path1[k] not in SEV(C_j)
                    update SEV(C_j)
                end if
            end if
        end for
    end for
```

Step 3: Similarity of the concepts C_i, C_j

```
for p in 1 to len(SEV(C_i)):
    for q in 1 to len(SEV(C_j)):
        sim[p][q] = M(SEV(C_j)[p], SEV(C_j)[q])
        //Where M is either Cosine or Tanimoto similarity
    end for
end for
//average of similarity
```

similar to the word C_i are added to *SEV*(C_i), by calculating the *Similar*(*Synsets*(C_i, *LCS*) [*j*], C_i). If this value is greater than δ, then the synset is converted into a vector of 300 dimensions from the pretrained CBOW model on Wikipedia text. The same procedure is repeated to update *SEV*(C_j). If LCS of concepts does not exist, the hypernym path of each concept is examined. It is observed from the experimentations that this step hardly updates the sense vectors, since individual hypernym paths are checked in Algorithm I. But this is done not to miss out on any possible senses. Algorithms I and II check all the possible senses, which increases the overall efficiency of the proposed approach.

Step 3 in Algorithm II calculates the average similarity of the two multi vectors, *SEV*(C_i), *SEV*(C_j). MSSA approach [35] gives two metrics for evaluating the approaches, namely average similarity and maximum similarity. If the maximum similarity is considered, similarity obtained will be of only one sense in each word. Hence, this

paper uses average similarity of all the senses. Given $SEV(C_i)$, $SEV(C_j)$, the similarity is calculated using the following equation,

$$Similarity(C_n, C_m) = avg(M(s_{1i}, s_{1j})), \quad i \in (1,n), j \in (1,m) \tag{1}$$

Where $M(s_{1i}, s_{1j})$ gives the similarity of sense embedding vectors s_{1i}, s_{1j}, M is either Cosine or Tanimoto similarity metric for assessing the vectors. If the words are represented as vectors, the task of finding the similarity between them is simpler. There are many measures that calculate the similarity between the vectors, like Cosine, Tanimoto [42], etc. The most used metric for Word2vec models is cosine similarity. The vectors with the same orientation are more similar and tend to have a value of 1 with cosine similarity. The results of state-of-art approaches are based on the cosine similarity between the vectors. To estimate the strength of the proposed approach with state-of-art techniques cosine similarity is considered. Tanimoto metric [42] is mainly used to calculate the similarity of vectors with binary data. But this metric can be easily enhanced to apply on vectors with continuous data. Tanimoto [42] metric for continuous data is used in this paper. Moreover, this metric is analogous to cosine similarity. Hence, this metric is also considered for measuring the similarity.

$$Cosine(V_1, V_2) = \frac{V_1 * V_2}{|V_1||V_2|} \tag{2}$$

$$Tanimoto(V_1, V_2) = \frac{V_1 * V_2}{|V_1^2 + V_2^2 - V_1 * V_2|} \tag{3}$$

This section proposes multi-sense embeddings using Wikipedia data and lexical database Wordnet. Multiple senses of words are extracted from the synsets, definitions, hypernym path and Google News vectors. These vectors are enhanced using LCS information of concepts to perform word similarity tasks. The next section discusses the experimentation details, datasets and models compared with the proposed approach.

IV. Results

In this section, experiments related to word similarity task are presented. The proposed algorithm and its comparison against other approaches are shown in detail.

A. Experimentation Setup

The proposed methodology in this paper is implemented using Python 3.6 with NLTK 3.4[2] and gensim 3.4[3] libraries. NLTK is used to get the information of Wordnet's synsets, definitions, lemmas, LCS of concepts and hypernym path. This paper builds a corpus from Wikipedia dump[4] which is open-source. CBOW Word2vec [18] is used to train the corpus of words. The hyperparameters of the model are as follows: CBOW model for training, a window size of 15, vector size of 300 dimensions. Google News Vectors[5] used in Algorithm I is a pre-trained model with tokens and respective word vectors.

B. Datasets

The benchmark datasets used in this paper are described as follows:

1. MC dataset [43]: MC dataset is a subset of the RG dataset [44], this dataset has 30 word pairs. Two word pairs are not present in Wordnet and hence the researchers work on 28 word pairs. The rating for each word pair in the dataset is in the range from 0 to 4, higher rating indicates word pairs are more similar.
2. RG dataset [44]: This dataset contains 65 word pairs and rating is in the range 0 to 4.
3. WS-353 dataset [45]: This dataset has 353 noun pairs, the rating of word pairs are in a range from 0 to 10. The dataset is divided into two sets. The first set is with 153 word pairs and the later with 200 word pairs. The complete version is considered in experimentation.
4. MTurk771 dataset: Halawi et al, [46] proposed this dataset with 771 word pairs and each word pair is given a rating in the range 0 to 5.
5. MTurk287 dataset [47]: This dataset has 287 word pairs, the rating of each word is in the range 0 to 5. All the word pairs are noun pairs.
6. SCWS (Stanford Context Word Similarity) dataset [48]: This dataset has 1997 word pairs, for each word in the word pair context is mentioned in the dataset. Word pairs are given a rating in range from 0 to 10.
7. Rare words dataset [49]: This dataset has 2034 word pairs and each word pair is given a rating in the range 0 to 4.
8. Men dataset [50]: This dataset has 3000 word pairs. This dataset has word pairs of noun, verb, and adjectives. The rating of each word ranges from 0 to 1.
9. Simlex999 dataset [51]: This dataset has 999 word pairs, out of these 999 word pairs, 666 word pairs are nouns, 222 are verb pairs and 111 adjective pairs. For experimentation purposes, 666 noun pairs (Simlex666) are considered.

C. Models Compared

The following models are compared with the proposed approach on benchmark datasets,

1. CBOW: Mikolov et al. [18] proposed Word2vec with two word embedding models, namely Skip-gram and CBOW. Skip-gram aims to predict a context word in a window around a target word. CBOW predicts the current target word using the representations for its context words. It is mentioned in several works that CBOW is more efficient, hence this approach is selected as one of the state-of-art methods for comparison.
2. GloVe: Pennington et al. [30] proposed a log-bilinear model called GloVe for learning word vectors. This is based on global matrix factorization and local context window methods. Local window methods like Word2vec[18], fail to capture global statistical information in the corpus, GloVe overcomes this drawback.
3. fastText [31]: Word2vec [18] and GloVe [30] ignore morphology of words. Hence, these approaches produce more out of vocabulary words. The fastText approach overcomes this problem by representing each word as a bag of character n-grams.
4. SenseEmbed [32]: SensEmbed uses BabelNet [52] for word sense disambiguation. This model trains a word2vec model (400 dimensions). This approach integrates structural knowledge from BabelNet and the distributional sense representations.
5. Chen et al. [33]: This approach performs disambiguation of words to learn word sense representations. It has two sub-tasks, namely L2R (left to right) and S2C (simple to complex). L2R disambiguates words from left to right and S2C selects senses of the word. S2C selects only those senses with a certain threshold.
6. MSSA: MSSA approach [35], mentions that Chen et al. [33] considers only specific senses leading to poor representations. Hence, MSSA explores all senses available for a word. MSSA-T and MSSA-D approaches mentioned in this work are taken for comparison.

2 https://www.nltk.org/

3 https://radimrehurek.com/gensim/models/word2vec.html

4 https://dumps.wikimedia.org/enwiki/latest/

5 https://code.google.com/archive/p/word2vec/

7. Paragram : Wieting et al. [42] proposed a word embedding model based on paraphrase phrases in PPDB database [53].
8. Counter-fitting [41]: This approach considers antonymy and synonymy constraints from the PPDB database and Wordnet.
9. Attract-Repel: Mrksic et al. [39] proposed Attract–Repel approach with synonymy and antonymy constraints to develop embeddings. In this approach, semantic relations are taken from BabelNet [52].

D. Metrics

This paper uses two evaluation metrics, the Spearman rank correlation, the Pearson correlation factor. Pearson correlation was first given in the studies of synonymy between words by Rubenstein and Goodenough [43]. A majority of the researchers in their works evaluated their approaches with Pearson correlation. Hence, this paper uses this evaluation metric to compare the results with state-of-the-art approaches. Another metric that is used by all the researchers for word similarity tasks is Spearman correlation. Hence, this metric is also used for comparing the results. Pearson correlation coefficient measures the strength and direction of the linear relationship between the variables. Whereas, the Spearman coefficient measures the correlation between the ranks of two variables. There are fewer publications that reported the results of both Spearman and Pearson correlation coefficient values. The unavailable data are not mentioned in the results.

The experimentations of the approaches Word2vec [18], GloVe [30], fastText [31] are implemented by us. The results of the approaches [39], [41], [42] are obtained from the HESML library[6]. The results of the approaches [32] and [33] for some datasets are reported in [35]. The MSSA [35] approach reported results for only Spearman correlation. The results reported in the tables for [32], [33] are taken from MSSA [35], as it became difficult to set up the same environment and to produce results.

Table II presents the results of the proposed approach against several state-of-the-art distributional approaches on the MC dataset [43]. On this dataset, Attract-Repel, SenseEmbed, and proposed approaches have obtained the highest Spearman correlation results. The proposed method obtained the highest results for the Pearson correlation, which is 0.851.

TABLE II. Spearman and Pearson Correlations of Proposed and Existing Word Embedding Approaches on MC-28 Dataset, Highest Results Highlighted in Boldface

Method	Spearman correlation	Pearson correlation
SenseEmbed [32]	0.880	--
MSSA-T [35]	0.796	--
MSSA-D [35]	0.835	--
Attract-Repel [39]	**0.884**	0.837
Counter-fitting [41]	0.857	0.806
Paragram [42]	0.824	0.796
CBOW [18]	0.781	0.796
GloVe [30]	0.862	0.845
FastText [31]	0.845	0.842
Proposed approach (cosine)	0.865	**0.851**
Proposed approach (Tanimoto)	0.860	0.845

Table III shows the results of the proposed approach against several models for the RG [44] benchmark dataset. In this experiment, SensEmbed and proposed approach obtained the highest results for the Spearman correlation. The approach SenseEmbed [32] develop multi-sense vectors using BabelNet [51]. BabelNet is a lexical database composed of different resources. This approach trains the Word2vec model with 400 dimensions. This approach obtained better results on smaller datasets when the size of the dataset increases, the results of SenseEmbed approach are not promising as they are for MC and RG datasets.

MC [43] benchmark is a subset of the RG [44] benchmark dataset. The words in the MC benchmark and RG are nouns. Algorithm I to generate the multi-sense embeddings of these datasets uses noun synsets from Wordnet. Some of the word pairs in these datasets are having very few noun synsets. For the words with one or two synsets, lemmas (synonyms) are generated. For words with fewer synonyms tracing the hypernym path gives multiple senses. There is minimal variation in the Spearman and Pearson correlation for cosine and Tanimoto metrics because both are analogous. The human ratings of half of the word pairs in MC benchmark are meager, suggesting that these word pairs are less similar. Out of 28 word pairs in the dataset, the human ratings of 10 word pairs are below 1.0, the scale of the ratings is in the range [0,4]. Hence, the approaches like Attract-Repel [39], Counter-fitting [41] handle the antonym relations work well on this dataset. The multi-sense embeddings approaches also worked well on both MC and RG benchmarks since all the senses are covered. The SenseEmbed [32] approach uses BabelNet [52], which has a wide coverage of synsets. Hence it is able to achieve better results on these two benchmarks and also on other datasets.

TABLE III. Spearman and Pearson Correlations of Proposed and Existing Word Embedding Approaches on RG-65 Dataset, Highest Results Highlighted in Boldface

Method	Spearman correlation	Pearson correlation
SenseEmbed [32]	**0.871**	--
MSSA-T [35]	0.776	--
MSSA-D [35]	0.801	--
Attract-Repel [39]	0.825	0.840
Counter-fitting [41]	0.808	0.806
Paragram [42]	0.813	0.810
CBOW [18]	0.760	0.772
GloVe [30]	0.755	0.770
FastText [31]	0.801	0.793
Proposed approach (cosine)	0.863	**0.851**
Proposed approach (Tanimoto)	0.865	0.847

Table IV presents the Spearman and Pearson correlation values of the approaches on the WS-353 dataset. From the results, it can be observed that the proposed approach outperformed all the other state-of-art approaches. WS-353 dataset is an interesting one because the dataset is used for carrying out semantic relatedness and semantic similarity tasks. The dataset has a collection of nouns, verbs, and adjectives. Algorithm I groups the synsets based on the parts-of-speech while generating the synsets, and hence, the proposed approach is able to achieve better results for both cosine and Tanimoto similarities.

Tables V and VI represent the results on Turk771 and Turk287 datasets. Semantic relatedness tasks use these two datasets. The difference between semantic similarity datasets and relatedness datasets is in the former, the concepts are more specific, and in the later, the concepts are more general. The proposed approach handles both the types because it observes both specific (using synonyms) terms, and general (using hypernyms) terms efficiently. The proposed approach obtained the highest results for Spearman correlation on both benchmarks. The single vector representation model GloVe achieved the highest results for the Pearson correlation.

6 https://github.com/jjlastra/HESML

TABLE IV. Spearman and Pearson Correlations of Proposed and Existing Word Embedding Approaches on WS-353 Dataset, Highest Results Highlighted in Boldface

Method	Spearman correlation	Pearson correlation
SenseEmbed [32]	0.779	--
MSSA-T [35]	0.694	--
MSSA-D [35]	0.708	--
Attract-Repel [39]	0.666	0.608
Counter-fitting [41]	0.680	0.615
Paragram [42]	0.764	0.679
CBOW [18]	0.603	0.642
GloVe [30]	0.716	0.713
FastText [31]	0.738	0.698
Proposed approach (cosine)	**0.817**	**0.836**
Proposed approach (Tanimoto)	0.814	0.832

TABLE V. Spearman and Pearson Correlations of Proposed and Existing Word Embedding Approaches on Turk771 Dataset, Highest Results Highlighted in Boldface

Method	Spearman correlation	Pearson correlation
Attract-Repel [39]	0.599	0.590
Counter-fitting [41]	0.701	0.666
Paragram [42]	0.745	0.704
CBOW [18]	0.672	0.698
GloVe [30]	0.715	**0.749**
FastText [31]	0.661	0.728
Proposed approach (cosine)	**0.761**	0.733
Proposed approach (Tanimoto)	0.757	0.727

TABLE VI. Spearman and Pearson Correlations of Proposed and Existing Word Embedding Approaches on Turk287 Dataset, Highest Results Highlighted in Boldface

Method	Spearman correlation	Pearson correlation
Attract-Repel [39]	0.606	0.618
Counter-fitting [41]	0.639	0.630
Paragram [42]	0.699	0.701
CBOW [18]	0.674	0.650
GloVe [30]	0.724	**0.749**
FastText [31]	0.709	0.728
Proposed approach (cosine)	**0.726**	0.736
Proposed approach (Tanimoto)	0.724	0.732

SCWS dataset has 1997 word pairs. Each word in the word pair has a context sentence, parts-of-speech of the word is mentioned for each word. SCWS dataset has four different sets of word pairs. The majority of the word pairs in the dataset have the same parts-of-speech, and some word pairs are with different parts-of-speech.

- The dataset has 1323 pairs, which are nouns.
- The dataset has 399 verb pairs.
- The dataset has 97 adjective pairs.
- Remaining 178 word pairs are with different parts-of-speech like noun-verb, verb-noun, verb-adjective, etc.

The results of the SCWS dataset are given in Table VII. The state-of-art methods produced competitive results with Spearman correlation but failed to produce the same with Pearson correlation. But the proposed approach is consistent in obtaining the highest results on this dataset also. Algorithm I handles words with different parts-of-speech, that enabled the proposed approach to achieve better results on this dataset in comparison with state-of-art methods.

TABLE VII. Spearman and Pearson Correlations of Proposed and Existing Word Embedding Approaches on SCWS Dataset, Highest Results Highlighted in Boldface

Method	Spearman correlation	Pearson correlation
SenseEmbed [32]	0.624	--
MSSA-T [35]	0.649	--
MSSA-D [35]	0.640	--
Chen et al. [33]	0.662	--
Attract-Repel [39]	0.587	0.113
Counter-fitting [41]	0.611	0.114
Paragram [42]	0.691	0.115
CBOW [18]	0.643	0.105
GloVe [30]	0.624	0.106
FastText [31]	0.652	0.106
Proposed approach (cosine)	**0.693**	**0.658**
Proposed approach (Tanimoto)	0.690	**0.658**

Table VIII shows the results of the approaches to the Rare words dataset. This dataset has nouns, verbs and adjective word pairs. The proposed approach has obtained the highest results of Spearman and Pearson correlation on this dataset also because Algorithm I handles different parts-of-speech efficiently.

TABLE VIII. Spearman and Pearson Correlations of Proposed and Existing Word Embedding Approaches on Rare words Dataset, Highest Results Highlighted in Boldface

Method	Spearman correlation	Pearson correlation
Attract-Repel [39]	0.273	0.319
Counter-fitting [41]	0.207	0.288
Paragram [42]	0.536	0.505
CBOW [18]	0.456	0.438
GloVe [30]	0.451	0.440
FastText [31]	0.464	0.432
Proposed approach (cosine)	**0.543**	**0.544**
Proposed approach (Tanimoto)	0.535	0.535

Table IX shows the Spearman and Pearson correlation values on Men dataset. This dataset is a collection of 3000 word pairs, with all the parts-of-speech included in it. The proposed approach continues to outperform all the state-of-art approaches on this dataset also. The proposed approach obtained the highest values of *0.836, 0.815* for Spearman and Pearson correlation, respectively. From the results obtained on all the datasets, it is observed that the proposed approach performed well on different benchmarks. It is because of the procedure followed to generate the multi-sense embeddings.

Table X shows the analysis on Simlex666 dataset, the results on this dataset for the proposed approach and other multi-sense approaches are marginally less when compared with Attract-Repel [39], counter-fitting [41], paragram [42]. These three approaches Attract-Repel, counter-fitting, paragram outperformed other single vector and multi-vector representations on the Simlex666 dataset. The reasons are mentioned in the discussion section.

TABLE IX. Spearman and Pearson Correlations of Proposed and Existing Word Embedding Approaches on Men Dataset, Highest Results Highlighted In Boldface

Method	Spearman correlation	Pearson correlation
SenseEmbed [32]	0.805	--
MSSA-T [35]	0.769	--
MSSA-D [35]	0.768	--
Chen et al. [33]	0.620	--
Attract-Repel [39]	0.709	0.655
Counter-fitting [41]	0.741	0.680
Paragram [42]	0.799	0.754
CBOW [18]	0.732	0.723
GloVe [30]	0.801	0.800
FastText [31]	0.762	0.755
Proposed approach (cosine)	**0.836**	**0.815**
Proposed approach (Tanimoto)	0.831	0.810

TABLE X. Spearman and Pearson Correlations of Proposed and Existing Word Embedding Approaches on Simlex666 Dataset, Highest Results Highlighted in Boldface

Method	Spearman correlation	Pearson correlation
MSSA-T [35]	0.460	--
MSSA-D [35]	0.425	--
Chen et al. [33]	0.430	--
Attract-Repel [39]	0.690	0.691
Counter-fitting [41]	**0.698**	**0.697**
Paragram [42]	0.645	0.662
CBOW [18]	0.454	0.461
GloVe [30]	0.429	0.467
FastText [31]	0.410	0.411
Proposed approach (cosine)	0.522	0.531
Proposed approach (Tanimoto)	0.521	0.527

V. Discussion

In the results section, nine benchmark datasets are analyzed with the state-of-art systems and the proposed approach. The state-of-the-art systems compared in this paper are categorized as follows:

1. Symmetric pattern vector representations: Approaches like CBOW of Word2vec [18], GloVe [30], fastText [31] fall in this category. These approaches use shallow linguistic information like word and context co-occurrences within a single window.
2. Inject approaches : The approaches such as Attract-Repel [39], counter-fitting [41], Paragram [40] fall in this category. In these approaches, word embeddings are enriched with synonymy and antonymy constraints and paraphrase relations. These three inject approaches use synonymy and antonymy relations of the words.
3. Multi-sense representations: Approaches like SenseEmbed [32], Chen et al. [33], MSSA [35], fall in this category. These approaches use multi-sense vector representations to carry out various NLP tasks.

On MC and RG datasets, there is a little variation in the results of all the approaches. This is because of the less number of word pairs in the datasets. Most of the researchers use these two benchmarks as these are more popular datasets in word similarity tasks. From, the results it is observed that the proposed approach performed consistently on all the datasets except Simlex666 dataset. The proposed approach obtained the first or second highest results of Spearman and Pearson correlations. The single vector representations CBOW [18], GloVe [30], fastText [31] when compared with multi-sense representations obtained lesser results. The CBOW [18], GloVe [30], fastText [31] approaches laid the basis of modern research for carrying out multiple NLP tasks. As the proposed approach generates all the multiple senses based on context, definitions, hypernym path, and LCS, the results are consistent on all the datsets.

Inject approaches like Attract-Repel [39], counter-fitting [41], paragram [40] performed well on WS-353 and Simlex666 datasets. The reason why these methods obtained better results on these datasets is because of using synonymy and antonymy relations. The multi-vector representations performed well on the WS-353 dataset but failed to perform well on the Simlex666 dataset. Compared to the WS-353 dataset, the Simlex666 dataset has more antonym word pairs, the inter-annotator agreement is also less in this dataset. As the multi-vector representations do not consider dealing with antonym relations, injection approaches produced better results on these datasets compared to multi-sense representations. Turk771 and Turk287 datasets have only noun pairs and the proposed approach obtained better results on these two datasets. These datasets are used mostly in semantic relatedness purposes. The proposed approach obtained the highest results on the WS-353 dataset because of handling synsets of different parts-of-speech.

On larger datasets like SCWS, Men and Rare words, the performance of the multi-vector representations and the proposed approach are consistent. These three datasets have different parts-of-speech word pairs. Algorithm I generates synsets with respect to their parts-of-speech to produce suitable multi-sense vector representations of the word. On larger datasets, the proposed multi-vector representation model outperformed the other state-of-art systems. The overall performance of the proposed approach is stable on all the datasets.

VI. Conclusion

This paper discusses the importance of multi-sense embedding for carrying out word similarity tasks. The proposed approach overcomes the limitations of single vector representations. This paper understands the importance of synonym sets, hypernyms, LCS, and definitions of words to generate multi-sense embeddings. This inspection of Wordnet for various parameters is a novel idea, which is not explored by the NLP community. With improved results of word similarity, this paper shows the significance of multi-sense representations.

The proposed approach performs disambiguation of words using Wordnet. A library of words from Wikipedia is created, and for each word, multi-sense embeddings are generated. Sense embedding vectors are enriched by using the LCS, hypernym information between the words. A comparison with recent state-of-art methods confirmed the efficiency of the proposed approach.

The proposed multi-sense representation produced state-of-art results on nine benchmark datasets. With these considerations, the next challenge is to see how multi-sense embeddings help to improve the efficiency of other NLP tasks like text summarization and document classification.

Acknowledgment

The authors thank the Ministry of Electronics and Information Technology for extending their support to carry out research (Grant No: MLA/MUM/Ga/10(37)B).

References

[1] X. Ji, A. Ritter, P. Y. Yen, "Using ontology-based semantic similarity to facilitate the article screening process for systematic reviews," *Journal of biomedical informatics,* 2017, vol. 69, pp. 33-42.

[2] M. B. Aouicha, and M. A. H. Taieb, "Computing semantic similarity between biomedical concepts using new information content approach," Journal of biomedical informatics, 2016, vol. 59, pp. 258-275.

[3] G. Zhu, and C. A. Iglesias, "Computing semantic similarity of concepts in knowledge graphs," *IEEE Trans. Know. Data Engg.,* 2016, vol. 29(1), pp. 72-85.

[4] A. Skabar, and K. Abdalgader, "Clustering Sentence-Level Text Using a Novel Fuzzy Relational Clustering Algorithm," *IEEE Trans. Know. Data Engg*., 2013, vol. 25(1), pp. 62-75.

[5] G. Glavas, M. Franco-Salvador, S. P. Ponzetto, and P. A. Rosso, "A resource-light method for cross-lingual semantic textual similarit," *Knowl.-Based Syst*., 2018, vol. 143, pp. 1–9.

[6] Jaccard, "Etude comparative de la distribution florale dans une portion des Alpes et des Jura," *Bulletin de la Société Vaudoise des Sciences Naturelles,* 1945, vol. 37, pp. 547-579.

[7] V. Levenshtein, "Binary Codes capable of correcting deletions, insertions, and reversals," *Cyber. and Control Theory,* 1966, vol.10, pp. 707–710.

[8] Dice, "Measures of the amount of ecologic association between species," *Ecology*, 1945.

[9] C. Fellbaum, "WordNet: An Electronic Lexical Database," *Cambridge, MA: MIT Press,* 1998.

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

[11] C. Leacock, M. Chodorow, "Combining local context and WordNet similarity for word sense identification," *WordNet: An electronic lexical database,* 1998, vol. 49, pp. 265-283.

[12] Li, McLean, "An approach for measuring semantic similarity between words using multiple information sources," *IEEE Trans. on know. and data eng.,* 2003, vol. 15, pp. 871-882.

[13] Lin, Dekang, "An information-theoretic definition of similarity," *Proc. Of Icml,* 1998.

[14] R. Rada, H. Mili, E. Bicknell, and M. Blettner, "Development and application of a metric on semantic nets," *IEEE Trans. Syst. Man. Cyber.,* 1989, vol. 19(1), pp. 17-30.

[15] Resnik, "Using information content to evaluate semantic similarity in a taxonomy," *arXiv preprint cmp-lg/9511007*, 1995.

[16] G. Zhu, C. A. Iglesias, "Computing semantic similarity of concepts in knowledge graphs," *IEEE Trans. Knowl. and Data Eng.*, 2017, vol. 29, pp. 72-85.

[17] Z. Wu, M. Palmer, "Verbs semantics and lexical selection," *In Proceedings of the 32nd annual meeting on ACM,* 1994, pp. 133-138.

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

[19] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and Dean, "Distributed representations of words and phrases and their compositionality," *in NIPS'13*, 2013, pp. 3111–3119.

[20] T. Chen, R. Xu, and Y. He, and X. Wang, "Improving distributed representation of word sense via wordnet gloss composition and context clustering," *International Joint Conference on Natural Language Processing*, 2015, pp. 15-20.

[21] S. M. Rezaeinia, R. Rahmani, A. Ghodsi, and H. Veisi, "Sentiment analysis based on improved pre-trained word embeddings," *Expert Systems with Applications,* 2019, vol. 117, pp. 139-147.

[22] T. Khai Tran, and T. Thi Phan, "Deep Learning Application to Ensemble Learning—The Simple, but Effective, Approach to Sentiment Classifying," *Applied Sciences*, 2019, vol. 9(13).

[23] F. Li, Y. Yin, J. Shi, X. Mao, and R. Shi, "Method of feature reduction in short text classification based on feature clustering," *Applied Sciences,* 2019, vol. 9(8).

[24] D. Dimitriadis, and G. Tsoumakas, "Word embeddings and external resources for answer processing in biomedical factoid question answering," *Journal of biomedical informatics*, 2019, vol. 92, pp. 103-118.

[25] J. Li, D. Jurafsky, "Do multi-sense embeddings improve natural language understanding?" *Proceedings of Empirical methods in natural language processing,* 2015, pp.1722–1732.

[26] Z. Harris, "Distributional structure," *Word*, 1954, vol. 10(23), pp. 146–162.

[27] G. Salton, A. Wong, C. S. A. Yang, "A vector space model for automatic indexing," *Communications of the ACM*, 1975, vol. 18(11), pp. 613–620.

[28] P. D. Turney, and P. Pantel, "From frequency to meaning: Vector space models of semantics," *CoRR . ArXiv: 1003.1141,* 2010.

[29] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. P. Kuksa, "Natural language processing (almost) from scratch," *Journal of Machine Learning Research*, 2011, vol. 12, pp. 2493–2537.

[30] J. Pennington, and R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," *Proceedings of Empirical Methods in Natural Language Processing*, 2014, pp. 1532–1543.

[31] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the Association for Computational Linguistics*, 2017, vol. 5, pp. 135–146.

[32] I. Iacobacci, M. T. Pilehvar, and R. Navigli, "Sensembed: Learning sense embeddings for word and relational similarity," *International Joint Conference on Natural Language Processing*, 2015, pp. 95-105.

[33] X. Chen, Z. Liu, M. A. Sun, "A unified model for word sense representation and disambiguation," *Proceedings of Empirical Methods in Natural Language Processing*, 2014, pp.1025-1030.

[34] D. Oele, G. van Noord, "Simple Embedding-Based Word Sense Disambiguation," *Proceedings of the 9th Global WordNet Conference*, 2018.

[35] T. Ruas, W. Grosky, and A. Aizawa, "Multi-Sense embeddings through a word sense disambiguation process," *Expert Systems with Applications*, 2019 (In press).

[36] M. Lesk, "Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone," *Proceedings of the 5th annual international conference on systems documentation SIGDOC'86*, 1986, pp. 24–26.

[37] S. Rothe, and H. Schutze, "Autoextend: Extending word embeddings to embeddings for synsets and lexemes," *arXiv preprint arXiv:1507.01127,* 2015.

[38] M. Fabian, M. F. Suchanek, G. Kasneci, G. Weikum, "Yago: A Core of Semantic Knowledge," *16th International Conference on the World Wide Web,* 2007, pp. 697–706.

[39] N. Mrksic, I. Vulic, D. O. Seaghdha, I. Leviant, R. Reichart, M. Gašić, A. Korhonen, and S.Young, "Semantic specialisation of distributional word vector spaces using monolingual and cross-lingual constraints," *Trans. ACL*., 2017, vol. 5, pp. 309–324.

[40] J. Wieting, M. Bansal, K. Gimpel, K. Livescu, and D. Roth, "From paraphrase database to compositional paraphrase model and back," *Trans. ACL.,* 2015, vol. 3, pp. 345–358.

[41] N. Mrksic, D. O Seaghdha, et al., "Counter-fitting word vectors to linguistic constraints," *In Proceedings of NAACL-HLT,* 2016, pp.142–148.

[42] T. Tanimoto, "An Elementary Mathematical theory of Classification and Pre-diction," *IBM Internal Report 17th IBM,* 1957.

[43] G. A. Miller, W. G. Charles, "Contextual correlates of semantic similarity," *Lang. Cogn. Process.,* 1991, vol. 6 (1), pp. 1–28.

[44] H. Rubenstein, J. B. Goodenough, "Contextual correlates of synonymy," *Commun. ACM,* 1965, vol. 8(10), pp. 627–633.

[45] L. Finkelstein, et al., "Placing search in context: The concept revisited," *ACM Trans. Inf. Syst*., 2002, vol. 20(1), pp. 116–131.

[46] G. Halawi, G. Dror, E. Gabrilovich, and Y. Koren, "Large-scale learning of word relatedness with constraints," *In Proc. of ACM SIGKDD*. 2012, pp. 1406–1414.

[47] K. Radinsky, et al., "A word at a time: computing word relatedness using temporal semantic analysis," *In Proc. of the Intl. Conf. on WWW. ACM,* 2011, pp.337–346.

[48] E. H. Huang, R. Socher, C. D. Manning, and A. Y. Ng, "Improving word representations via global context and multiple word prototypes," *In Proc. of the Annual Meeting of the ACL,* 2012, pp. 873–882.

[49] T. Luong, R. Socher, C. D. Manning, "Better word representations with recursive neural networks for morphology," *In: Proc. of CoNLL.,* 2013, pp. 104–113.

[50] E. Bruni, N. K. Tran, and M. Baroni, "Multimodal distributional semantics," *Journal of Artificial Intelligence Research,* 2014, vol. 49(1), pp. 1–47.

[51] F. Hill, R. Reichart, A. Korhonen, "SimLex-999: Evaluating semantic models with (genuine) similarity estimation," *Comput. Linguist.,* 2015,

vol. 41 (4), pp. 665–695.

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

[53] J. Ganitkevitch, B. Van Durme, C. Callison-Burch, "PPDB: The paraphrase database," *In: Proc. of HLT-NAACL,* 2013, pp. 758–764.

Krishna Siva Prasad Mudigonda

Krishna Siva Prasad Mudigonda is presently a Research Scholar in Computer Science and Engineering department at Visvesvaraya National Institute Technology, Nagpur, India. He received his M.Tech and B.Tech degrees from Jawaharlal Nehru Technological University, Kakinada, Andhra Pradesh, India. His research area includes natural language processing, deep learning and text processing.

Poonam Sharma

Poonam Sharma is presently Assistant Professor in Computer Science and Engineering department at Visvesvaraya National Institute Technology, Nagpur, India. She completed her PhD degree from Maulana Azad National Institute of Technology, Bhopal, India. Her research area includes Biometrics, Image processing, soft computing and pattern recognition. She is a recipient of Visvesvaraya young faculty fellowship from the government of India. She has completed a project sponsored by Department of Science and Technology, New Delhi.