Word Vector Embeddings and Domain Specific Semantic based Semi-Supervised Ontology Instance Population

Vindula Jayawardana\textsuperscript{31}, Dimuthu Lakmal\textsuperscript{32}, Nisansa de Silva\textsuperscript{33}, Amal Shehan Perera\textsuperscript{34}, Keet Sugathadasa\textsuperscript{35}, Buddhi Ayesha\textsuperscript{36}, Madhavi Perera\textsuperscript{37}

Abstract—An ontology defines a set of representational primitives which model a domain of knowledge or discourse. With the arising fields such as information extraction and knowledge management, the role of ontology has become a driving factor of many modern day systems. Ontology population, on the other hand, is an inherently problematic process, as it needs manual intervention to prevent the conceptual drift. The semantic sensitive word embedding has become a popular topic in natural language processing with its capability to cope with semantic challenges. Incorporating domain specific semantic similarity with the word embeddings could potentially improve the performance in terms of semantic similarity in specific domains. Thus, in this study, we propose a novel way of semi-supervised ontology population through word embeddings and domain specific semantic similarity as the basis. We built several models including traditional benchmark models and new types of models which are based on word embeddings. Finally, we ensemble them together to come up with a synergistic model which outperformed the candidate models by 33\% in comparison to the best performed candidate model.

Keywords—Ontology, Ontology Population, Word Embeddings, word2vec, Semantic Similarity.

I. INTRODUCTION

In various computational tasks in many different fields, the use of ontologies is becoming increasingly involved. Many of the research areas such as knowledge engineering and representation, information retrieval and extraction, and knowledge management and agent systems \cite{1} have been incorporated with the use of ontologies to a greater extent. As defined by Thomas R. Gruber \cite{2}, an ontology is a “formal and explicit specification of a shared conceptualization”. Due to the evolving ability of ontologies to overcome limitations in traditional natural language processing methods, the popularity of using ontologies in modern computation tasks are getting increased day by day. For an example, text classification \cite{3}, \cite{4}, word set expansions \cite{5}, linguistic information management \cite{13}, \cite{14}, and information extraction \cite{15}, \cite{16} emphasize the growing popularity of the ontology based computations and processing.

According to Carla Faria et al. \cite{7}, ontology population looks for instantiating the constituent elements of an ontology like properties and non-taxonomic relationships. However, most of the time, ontology populations are done by domain experts and knowledge engineers as a manual process, which is both time consuming and expensive. As the majority of the world’s knowledge is encoded in natural language text, automating the population of these ontologies using results obtained from Natural Language Processing (NLP) based analysis of documents, has recently become a major challenge for NLP applications \cite{8}.

In this study, we propose a novel way for semi-supervised instance population of an ontology using word vector embeddings. Word Embeddings could be identified as a collective name for a set of language modelling and feature learning techniques in natural language processing. The basic idea behind word embedding is based on the concept where words or phrases from the vocabulary are mapped out into vectors of real numbers. We use these vectors as a method of arriving at instance population in an ontology. For this purpose, we built an iterative model based on the class representative vector for ontology classes \cite{16}. In our implementation, we built multiple models based on different methodologies. In one model, we assigned membership to natural language tokens by distance to the representative vectors. In another, we used dissimilar exclusion method to identify the membership. Set expansion as described by \cite{5}, was used in another model for the purpose of ontology population. Finally, we used two semi-supervised models based on k-means clustering and hierarchical clustering. As each model outputs a set of candidate words for a given class, we then collaborate with domain experts and knowledge engineers to identify the performance of each model and to build an ensemble model as the final resultant model. We intend to demonstrate the use of domain specific semantic similarity in defining the similarities between instances and classes. As allowed by the nature of the defined models, we use the domain specific semantic similarity measure \cite{20} as the distance measure.

Semi-supervised learning falls between unsupervised learning (without any labelled training data) and supervised learning (with completely labelled training data). It has been observed that many machine leaning approaches elucidate a considerable improvement in learning accuracy, when unlabelled data is used in conjunction with a small amount of labelled data.
The legal context contains jargon which is complex and most of the time, it is impossible to store them in the mind; whether it be an average person or a paralegal, given that it consists of a terminology derived from ancient Latin terms, as well as various distinctive terminology depending on the category of laws and the geographical settings of practice. Therefore, knowing them manually is rather an impossible task which drives us to select the legal domain for this study of semi-supervised ontology population.

The rest of this paper is organized as follows: In Section II, we review previous studies related to this work. The details of our methodology for semi-supervised instance population of an ontology using word vector embeddings is introduced in Section III. In Section IV, we demonstrate that our proposed methodology produces superior results outperforming traditional approaches. Finally, we conclude and discuss some future works in Section V.

II. BACKGROUND AND RELATED WORK

The following sections depict the background of this study and other related studies.

A. Ontologies

Ontologies are mainly used to organize information as a form of knowledge representation in many areas. As defined by Thomas R. Gruber [2], ‘ontologies are an explicit and formal specifications of the terms in the domain and the relations among them’. Ontologies have been expanding out from the realm of Artificial-Intelligence to domain specific tasks such as: Linguistics [4, 5, 9, 21, 22], Law [19], Medicine [10, 11, 13, 14, 16]. Ontologies have become common on the semantic iteration of the World-Wide Web [12]. An ontology may model either the world or a part of it as seen by the said area’s viewpoint [5].

The basic ground units of an ontology are the Individuals (instances). By grouping these Individuals which can either be concrete objects or abstract objects, the structures called classes are built. A class in an ontology is a representation of a concept, type, category, or a kind. However, these definitions may be altered depending on the domain of the ontology. Often these classes form taxonomic hierarchies among themselves by subsuming, or being subsumed by, another class.

B. Word Vector Embeddings

As first proposed by Tomas Mikolov et al. [22] word embedding systems, are a set of natural language modelling and feature learning techniques, where words from a domain are mapped to vectors to create a model that has a distributed representation of words. Word2vec[1] [23], GloVe [24], and Latent Dirichlet Allocation (LDA) [25] are the leading Vector Embedding systems. However, due to the flexibility and ease of customization, we picked word2vec as the word embedding method for this study.

Word2vec is a neural network with two layers, which uses a large corpus of text as an input and outputs a vector space, typically of several hundred dimensions for the given corpus of text. Word2vec trains neural network to reconstruct the linguistic contexts of words utilizing either of two methods: continuous bag-of-word (CBOW) or continuous skip-gram. In continuous bag-of-words method, the model predicts the current word from a windows of surrounding context words. In the continuous skip-gram method, the model uses the current word to predict the surrounding window of the context words. Word2vec can be adapted to provide similar terms for an input term and facilitate vector operations with a high degree of accuracy.

Word2vec has been used in many areas due to its capability in coping up with the challenge of preserving the semantic sensitivity of a given context. It has been used in sentiment analysis [27–30] and text classification [31]. Gerhard Wohlgemann et al. [32]’s approach to emulate a simple ontology using word2vec and Harmen Prins [33]’s usage of word2vec extension: node2vec [34], to overcome the problems in vectorization of an ontology, are two major works that have been carried out in relation to ontologies with the use of word2vec. More recently, there have been successful studies on using word2vec on the legal domain [19, 20].

C. Word Set Expansion

Word lists that contain closely related sets of words is a critical requirement in machine understanding and processing of natural languages. Creating and maintaining such closely related word lists is a complex process that requires human input and is carried out manually in the
The said word-lists usually contain words that are deemed to be homogeneous in the level of abstraction involved in the application. Thus, two words W1 and W2 might belong to a single word-list in one application, but belong to different word-lists in another application. This fuzzy definition and usage is what makes creation and maintenance of these word-lists a complex task.

De Silva et al. [5] describe a supervised learning mechanism which employs a word ontology to expand word lists containing closely related sets of words. This study has been an extension of their previous work [21], which was done to enhance the refactoring process of the RelEx2Frame component of OpenCog AGI Framework, by expanding concept variables used in RelEx. The expected outcome of the project has a significant effectiveness upon applications which fit into real life IT solutions which are related to the natural language domain. Mainly, AI related applications which require English language processing would benefit from the project. Moreover, the output of the project can be utilized in a vast area of applications related to English language such as chat applications, text critiquing, information retrieval from the web, question answering, summarization and translations, rather than focusing on a specific area of applications of English Language.

D. Ontology Population

Being a knowledge acquisition task, ontology population is inherently a complex activity. Ontology population has been approached by using techniques such as rule based and machine learning. SPRAT [35] combines aspects from traditional named entity recognition, ontology-based information extraction, and relation extraction, in order to identify patterns for the extraction of a variety of entity types and relations between them, and to re-engineer them into concepts and instances in an ontology.

Since the majority of world’s knowledge concentrated on natural language text, it is vital to take the knowledge extracted from natural language analysis, into account when populating an ontology in any given domain. Natural language analysis frameworks such as GATE have been introduced with the aim of facilitating NLP application development. In GATE, natural language processing tasks such as tokenization, POS tagging, or chunking are supported by integrating existing components into complex application pipelines. Nevertheless, exporting results of GATE natural language analysis into ontology still requires high degree of human intervention. Rene Witte et al. [18] have implemented a GATE processing resource namely OWLExporter that empowers automation of ontology population from text for an existing application pipeline. It yields a number of novel features such as exporting sentences, noun and verb phrase chunks, and integrating reasoning support for conference chains, to overcome said issues with ontology population using GATE. Moreover, it allows language engineers to create ontology population systems without requiring extensive knowledge of ontology APIs.

However, modern day researches are more focused on semi supervised ontology population due to the nature of less manual intervention.

E. Domain Specific Semantic Similarity

In almost all Natural Language Processing (NLP) tasks such as Information Retrieval, Information Extraction, and Natural Language Understanding [15], Semantic Similarity measurements based on linguistic features are a fundamental component. Methods that treat words as independent atomic units are not sufficient to capture the expressiveness of language [24]. A solution to this is word context learning methods [21, 36]. Another solution is lexical semantic similarity based methods [37]. Both of these approaches try to capture semantic and syntactic information of a word.

Lexical Semantic similarity of two entities is a measure of the likeness of the semantic content of those entities. This likeness of the semantic content of the entities are most commonly calculated with the help of topological similarity existing within an ontology such as WordNet [6]. Wu and Palmer proposed a method to give the similarity between two words in the 0 to 1 range [38]. In comparison, Jiang and Conrath proposed a method to measure the lexical semantic similarity between word pairs using corpus statistics and lexical taxonomy [39]. Hirst & St-Onge’s system [40] quantifies the amount that the relevant synsets are connected by a path that is not too long and that does not change direction often. In [4], the strengths of each of these algorithms were evaluated by means of the tool WS4J3.

However, the semantic similarity measures built for general use do not perform well within specific domains. Law and Medical [41] fields are the fields which suffer from this issue drastically. Therefore, Sugathadasa et al. [20] have introduced a domain specific semantic similarity measure that has been created by the synergistic union of word2vec, a word embedding method that is used for semantic similarity calculation and lexicon based (lexical) semantic similarity methods. According to Sugathadasa et al., while for word context learning, Word Embedding method, word2vec [19, 24], has been used, number of lexical semantic similarity measures [38–40] have been used to augment and improve the results.

F. Semi Supervised Ontology Population

Although supervised machine learning methodologies have shown promising results when it comes to information extraction, they accumulate more cost for training since they require a vast number of labelled training data. As a solution, semi-supervised machine learning methodologies have been introduced, requiring considerably less amount of labelled training data.

Carlson [42] proposed a semi-supervised learning model to populate instances of a set of target categories and relations of an ontology by providing seed labelled data and a set of constraints which couples classes and relationships of an ontology. Semi-supervised algorithms tend to show unacceptable results due to ‘semantic drift’ and constraints have been introduced to overcome the issue. Carlson has used ‘Bootstrapping’ method for semi-supervised learning which starts with a small number of labelled data and grows labelled data iteratively, which are chosen from a set of candidates, which is classified using the current semi-supervised model. Three types of constraints have been introduced by Carlson to conform mutual exclusion, type checking, and text features.

Carlson [43] has expanded coupled semi-supervised learning [42] to never-ending language learning (NELL); an agent that runs forever to extract information from the web.
and populate them continuously into a knowledge base. A prototype of the system that they have implemented is able to extract noun phrases related to various semantic categories, and semantic relations between categories. Its information extracting ability increases day by day which is evidenced by the ability to extract more information from previous day's text sources more accurately. The input ontology in the system was included with seed instances for each ontology class and then sub systems which consist of previously described coupled semi supervised methodologies extract candidate instances and relationships from the text corpus. The Knowledge Integrator of the system choose strongly supported sets of instances and relations from the candidate set, as new beliefs of the system.

Zhilin Yang [4] has presented a semi supervised learning methodology based on graph embeddings. The system was developed based on Findlaw [46] as the reference. The ontology creating process was an iterative process where, upon adding parts of legal domain knowledge to the ontology, a validation phase is run by the domain experts. However, to improve the clarity of this paper, we extract a sub-ontology from it and use it to explain the methodology to make the process simple and intuitive to understand. In selecting a part of the ontology, we mainly focused on more sophisticated relationships and taxonomic presences. An overview of the selected part of ontology is illustrated in Fig. 2.

After the creation of sub-ontology, we manually populated the ontology with seed instances for each ontology class. For this phase as well, we incorporated the domain experts’ knowledge and the collaboration of knowledge engineers.

B. Training word Embeddings

The word embeddings method used in this study was built using a word2vec model. We obtained a large legal text corpus from Findlaw [46] and built a word2vec model using the corpus. The reason for selecting word2vec word embedding for this study is the success demonstrated by other studies such as [19] and [20] in the legal domain that uses word2vec as the word embedding method. The text corpus consisted of legal cases under 78 law categories. In creating the legal text corpus, we used the Stanford CoreNLP for pre-processing the text with tokenizing and sentence splitting. Fig. 3(a) illustrates the Natural Language Processing pipeline that we used in pre-processing the text corpus.

Following are the important parameters that we specified in training the model.
- size (dimensionality): 200
- context window size: 10
- learning model: CBOW
- min-count: 5
• training algorithm: hierarchical softmax

C. Deriving Representative Class Vectors

Ontology classes are sets of homogeneous instance objects that can be converted into a vector space by word vector embeddings. A methodology to derive a representative vector for ontology classes, whose instances were mapped to a vector space is presented in [19]. We followed the same approach and started by deriving five candidate vectors which are then used to train a machine learning model that would calculate a representative vector for each of the classes in the selected sub-ontology shown in Fig. 2. In the following sections, we describe in-depth, the manner in which we used this devised class vectors in our proposed methodology.

D. Instances Corpus for Ontology Population

In order to perform semi-supervised ontology population, we used legal cases from Findlaw [46] to create an instances corpus. We performed Stanford CoreNLP based pre-processing on the raw text with tokenizing and sentence splitting to generate the instance corpus. This legal corpus was used in the subsequent models for the purpose of ontology population.

E. Domain Specific Semantic Similarity Measure

In order to measure the domain specific semantic similarities, we used the methodology proposed by Sugathadasa et al. [20]. Fig. 3(b) indicated the high-level overview of building the domain specific semantic similarity model as per [20]. Depending upon the nature of the models that we train, we intend to use this trained model in subsequent actions.

F. Candidate Model Building

Based on the aforementioned components, we built five candidate models for semi-supervised population of the ontology. The five models are illustrated below:

• Membership by distance model (M1)
• Membership by dissimilar exclusion model (M2)
• Set expansion based model (M3)
• Semi-supervised K-Means clustering based model (M4)
• Semi-supervised hierarchical clustering based model (M5)

1) Membership by Distance Model (M1): In this model, the candidate vectors for the ontology are generated from the instance corpus based on the minimum distance to the representative class vector derived in Section III-C. By taking the vector similarity, we use cosine similarity. Given an instance \( i \) which has the vector embedding \( X_i \), Equation 6 describes which class the particular instance belongs to:

\[
C_{Mi} = \left\{ j \mid \argmax_{c_j \in C} \text{distance}(X_i, c_j) \right\}
\] (1)

Here, the set \( C \) denotes the set of representative class vectors. \( C_{Mi} \) is the selected class index of the instance \( i \) out of class set \( C \). distance \((X_i, c_j)\) is a function provides domain specific semantic similarity between the given instances. In measuring the semantic similarity between the given instance and derived ontology class vector, we could encounter a situation where the derived ontology class vector may not be in the vector space model. In such a situation, a semantic similarity was taken by identifying the closest vector available in the vector space to the derived ontology class vector and then taking domain specific semantic similarity between the identified vector and the instance vector. Here, the closest vector to the given ontology class vector was found based on the cousin similarity.

2) Membership by dissimilar exclusion model (M2): In this model, we use word2vec based dissimilar exclusion method in identifying the membership of a particular instance to a given class. This is a utilization of an internal method of word2vec where given a set of members, it would return the member that should be removed from the set in order to increase the set cohesion. For example, given the set of instances: breakfast, cereal, dinner and lunch, the word2vec dissimilar exclusion method would identify the instance cereal as the item that should be removed from the set to increase the set cohesion. We define this method as shown in Equation 2 where \( S \) is the set provided and \( e \) is the member selected to be excluded.

\[
e = \text{Exclusion}(S)
\] (2)

Here, the Exclusion(S) is defined as below. For a given \( n \) number of words, we obtain word embeddings of them using word2vec. Let \( W_v \) denotes the word vectors of \( n \) words.

\[
W_v = [v_1, v_2, v_3, ..., v_p]
\] (3)

Now, we take for each word vector and the average distance from the rest of the other word vectors as per the Equation 4.

\[
d_i = \frac{\sum_{j=1}^{p} \text{Distance}(v_i, v_j)}{p-1}
\] (4)

Here, \( D_i \) denotes the average distance from the rest of the other word vectors for the word vector \( i \). Distance \((v_i, v_j)\) function performs the distance calculation based on domain specific semantic similarity measure as per [20].

Upon defining the \( d_i \) as per Equation 4, we then define \( D \) as per Equation 5.

\[
D = [d_1, d_2, d_3, ..., d_p]
\] (5)

Finally, we identify \( e \), the member selected to be excluded as Equation 6

\[
E_{i,j} = \left\{ e \mid e \in S_j \mid \argmin_{d_i \in D} |d_i| \right\}
\] (6)

Here, \( w_j \) is the corresponding instance for the \( j \)-th index. We used the Equation 3 to decide whether the instance \( i \) should belong to class \( j \). Here, \( S_j \) is the seed set of class \( j \) and \( X_i \) is the vector representation of the instance \( i \). If the value \( E_{i,j} \) gets evaluated to TRUE we declare that instance \( i \) should belong to class \( j \) under model \( M_2 \).

\[
E_{i,j} = \left\{ e \mid e \in S_j \mid e = \text{Exclusion}(S_j \cap X_i) \right\}
\] (7)

When using the aforementioned method in identifying the membership of an instance, there is a possibility of getting
more than one class for a given instance as a possible parent class.

\[ C_{M2} = \{ C_k | 0 < k \leq N \} \] (8)

Here in Equation 4, \( C_{M2} \) is the set of classes for a given instance \( i \). \( C_k \) denote the common representation of those set of classes and \( N \) denotes the total number of classes we have in the ontology.

3) Set Expansion Based Model (\( M_3 \)): For the purpose of set expansion based model, we selected the algorithm presented in [5] which was built on the earlier algorithm described in [21]. The rationale behind this selection is the fact that as per [5], WordNet [6] based linguistic processes are reliable due to the fact that the WordNet lexicon was built on the knowledge of expert linguists.

In this model, the idea is to increase the ontology class instances based on a WordNet hierarchy based expansion. Simply put, it discovers the WordNet \textit{synsets} pertaining to the seed words and proceeds up the hierarchy to find the minimum common ancestors for each of the senses of the words. The next most common word sense is selected by the majority. The relevant rooted tree is extracted and the gazetteer list of that rooted \textit{synset} tree is created. The gazetteer list is subjected to set subtraction of the original seed set. The set intersection of the remaining set with the candidate word set is declared to be the word set assigned to the given class. However, it should be noted that as we showed in model \( M_2 \), after running the set expansion algorithm, one candidate instance may be tentatively assigned to more than one class. Fig. 4 illustrates the flow for
the simplified architecture of the concept expanding using WordNet as per the algorithm that we used [5].

4) Semi-Supervised K-Means Clustering Based Model (M₄): Out of the models proposed in this study so far, this model is the first semi-supervised model. First, the seed instances are put together with the unlabelled data from instance corpus. Let Nₜₐₗₜₜₑₙₑ be the number of labelled (seed) instances and Nₜₐₜₜₑₜₑₙₑ be the total number of unlabelled instances. Thus, by mixing up the labelled and unlabelled data, we get a total of Nₜₐₜₜₑₜₑₙₑ + Nₜₐₜₜₑₚₜₑₚₑ Number of instances. Next, all the instances are subjected to the k-means algorithm iteration where there are no new assignments. Finally, all the remaining unassigned classes are put in ct and Equation 11 is executed repetitively with tie breaking, done with precedence until all the clusters are uniquely assigned to some class.

5) Semi-Supervised Hierarchical Clustering Based Model (M₅): The next model that we used is a semi-supervised method based on hierarchical clustering. Hierarchical clustering is a method of cluster analysis which seeks to build a hierarchy of clusters. We built a model which creates such hierarchy of clusters using the word embeddings taken from the word2vec model, of the entire corpus similar

\[
C_i = \left\{ j \mid \arg\max_{c \in C} \left( \sum_{y \epsilon L} | d_i \right) \right\}
\]

\[
d_i = \begin{cases} 
1 & \text{if } y_i \epsilon c_j \\
0 & \text{otherwise}
\end{cases}
\]

At this point, it should be noted that there can be three situations where it is possible not to get a C_j value assigned to some class L by Equation 5 without ambiguity: (1) L not having any seed instances to vote. (2) L has multiple seed instances but the majority voting ending in a tie. (3) Two (or more) clusters, claim the same class. To solve these problems, we defined Equation 7, which selects the unassigned class that is closest to an unassigned cluster. Here, an unassigned cluster L' is considered. C' is the set of representative class vectors of unassigned classes. C_i is the selected class index of the cluster L'.

\[
C'_i = \left\{ j \mid \arg\max_{c \in C'} \left( \sum_{x \epsilon L'} | \frac{X_i, c_j}{|X_i||L'|} \right) \right\}
\]

The first problem to be solved is the problem of L having multiple seed instances but the majority voting ending in a tie. In this case, the Ct of Equation 11 is limited to the set intersection of tied classes and unassigned classes. Next, the problem of Two (or more) clusters, claiming the same class is solved. In this case Ct of Equation 11 is limited to the contested class. These steps are repeated until there is an
Recall\(_{M_{ij}}\) = \(\frac{W_{M_{ij}} \cap W_{M_{ij}}}{W_{M_{ij}}}\)  

Here, \(W_M\) denotes words by the model \(M\), and \(W_{ij}\) denotes the set of the words proposed by domain experts to be the golden standard for class \(j\). The model precision and recall of \(M\) were calculated by averaging the class values for precision and recall for those models.

\[
F1_{M_i} = 2 \cdot \frac{\text{Precision}_{M_i} \cdot \text{Recall}_{M_i}}{\text{Precision}_{M_i} + \text{Recall}_{M_i}}
\]

(H. Ensemble Model)

Next, we came up with an ensemble model based on the models identified earlier. In the task of creating the ensemble model, we allocated a candidate weight for each model based on \(F1\) measure of each model as calculated in the previous step.

Let \(M_i\) be a model out of the models and let \(F1_M\) be the \(F1\) measure of model \(M_i\). Hence, with the models in consideration, weight of the model \(W_i\) is calculated as shown in Equation 11, where \(p\) is the total number of models.

\[
V_e = \sum_{i=1}^{p} V_i W_i
\]

As identified above, upon calculating the weight of each model, we created the ensemble model as shown in Equation 12. Given an unlabelled instance \(Y\), let \(M_{\text{ensemble}}\) be a \(p \times n\) matrix where \(n\) denotes the number of classes in the ontology and \(p\) denotes the number of basic models. Each column of the matrix corresponds to a class in the ontology and each row corresponds to a model while each \(m_{ij}\) is derived from Equation 13.

\[
M_{\text{ensemble}} = \begin{bmatrix}
m_{1,1} & \cdots & m_{1,n} \\
\vdots & \ddots & \vdots \\
m_{p,1} & \cdots & m_{p,n}
\end{bmatrix}
\]

\[
m_{ij} = \begin{cases} 
1 & \text{if } Y \in M_i \\
0 & \text{otherwise}
\end{cases}
\]

Let \(M_{\text{weights}}\) be the \(p\) length vector which defines the weights of each model calculated by Equation 11.

\[
M_{\text{weights}} = [w_1 \ w_2 \ w_3 \ \ldots \ w_p]
\]

Then we calculate the total score vector for the instance \(Y\) by,

\[
S = M_{\text{weights}} \ast M_{\text{ensemble}}
\]

Here, \(S\) is the score vector of size \(n\) where element \(i\) in the vector denotes the total score for instance \(Y\) for the membership in Class \(C_i\). Next, we selected the class with the highest membership score as the parent class of instance \(Y\). It is illustrated in Equation 16.

\[
C_{M_{\text{ensemble}}} = \arg \max_{C_i \in C} (S_{C_i})
\]

With that, we get the final class of the instance \(Y\). Hence, we populate that selected class with the instance \(Y\).

IV. Results

In this section, we illustrate the results that we obtained through our proposed methodology for semi supervised instance population of an ontology using word vector embeddings as the basis. We intend to illustrate and compare the results that we obtained with domain specific semantic similarity incorporated and without incorporating it. It should be noted that the domain specific semantic similarity was incorporated only in the models, membership by distance model (M1) and membership by dissimilar exclusion model (M2).

In testing our ensemble model, we used another instance corpus. In this corpus, we sub divided in the order of 70%, 20%, and 10% as the training set, validation set, and test set respectively. The training set was used in training the models individually. The validation set was used to fine tune the models. Finally, the testing test was used in verifying the accuracy of the models. We have reported our findings below in the Table 1, where we compare the individual models: membership by distance model (M1), membership by dissimilar exclusion model (M2), set expansion based model (M7), k-means clustering based model (M8), hierarchical clustering based model (M9) and the ensemble model as a whole. In Fig. 7, we compare the precision, recall and \(F1\) of each of the candidate models along with the ensemble model with the domain specific semantic similarity. In Fig. 8, we compare the models’ performance, membership by distance model(M1) and membership by dissimilar exclusion model(M2) with and without domain specific semantic similarity incorporated.

<table>
<thead>
<tr>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>M1</td>
<td>0.08</td>
<td>0.22</td>
</tr>
<tr>
<td>M2</td>
<td>0.15</td>
<td>0.36</td>
</tr>
<tr>
<td>M3</td>
<td>0.24</td>
<td>0.30</td>
</tr>
<tr>
<td>M4</td>
<td>0.07</td>
<td>0.20</td>
</tr>
<tr>
<td>M5</td>
<td>0.06</td>
<td>0.23</td>
</tr>
<tr>
<td>Mensemble</td>
<td>0.51</td>
<td>0.63</td>
</tr>
</tbody>
</table>

Fig. 8. Comparison of precision, recall and \(F1\) of the models M1 and M2 with and without domain specific semantic similarity
In defining the ensemble model, Equation 21 defines the calculated weights of each model in the order of models $M1$ to $M5$ without domain specific semantic similarity incorporated. Equation 22 defines the calculated weights of each model in the order of models $M1$ to $M5$ with domain specific semantic similarity incorporated. These calculations are based on the above-mentioned training set.

$$M_{\text{weights}} = [0.15 \ 0.27 \ 0.33 \ 0.13 \ 0.12] \quad (21)$$

**TABLE II**

**COMPARISON OF PERFORMANCE OF MODELS WITH DOMAIN SPECIFIC SEMANTIC SIMILARITY**

<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M1$</td>
<td>0.17</td>
<td>0.33</td>
<td>0.22</td>
</tr>
<tr>
<td>$M2$</td>
<td>0.27</td>
<td>0.45</td>
<td>0.34</td>
</tr>
<tr>
<td>$M3$</td>
<td>0.24</td>
<td>0.30</td>
<td>0.26</td>
</tr>
<tr>
<td>$M4$</td>
<td>0.07</td>
<td>0.20</td>
<td>0.10</td>
</tr>
<tr>
<td>$M5$</td>
<td>0.06</td>
<td>0.23</td>
<td>0.10</td>
</tr>
<tr>
<td>$M_\text{ensemble}$</td>
<td>0.65</td>
<td>0.70</td>
<td>0.67</td>
</tr>
</tbody>
</table>

$$M_{\text{weights}} = [0.22 \ 0.33 \ 0.25 \ 0.10 \ 0.10] \quad (22)$$

It can be seen that; domain specific semantic similarity measures have improved the performance of membership by distance model ($M1$) and membership by dissimilar exclusion model ($M2$) by 10% and 13% respectively. Also with that performance change, the ensemble model has an improvement of 20% compared to the ensemble model we obtained without domain specific semantic similarity measures incorporated.

As can be seen in Table II, our ensemble model’s F1 has been improved by 33%, compared to the best of the candidate models with semantic similarity measures being used. Hence, from the results obtained, as a proof of concept, we can demonstrate that word embeddings can be used effectively in semi-supervised ontology population.

**V. CONCLUSION AND FUTURE WORKS**

Through this work we demonstrated the use of word embeddings on semi-supervised ontology population. We mainly focused on semi-supervised population which basically falls between the supervised population and unsupervised population. The main motive behind making the process semi-supervised is to reduce the level of manual interventions in ontology populations while maintaining a considerable amount of accuracy. As shown in the results, our ensemble model outperforms the five individual models in populating the selected legal ontology. The findings in this study is mainly important in two ways as mentioned below.

Firstly, an important part of the ontology engineering cycle is the ability to keep a handcrafted ontology up to date. Through the semi-supervised ontology population, we can reduce the hassle involved in manual intervention to keep the ontology updated.

Secondly, there is novelty in the methodology proposed in our study. We proved that, since word embeddings map words or phrases from the vocabulary to vectors of real numbers based on the semantic context, a methodology based upon it can yield more sophisticated results when it comes to context sensitive tasks like ontology population. This indeed is a step up from the traditional information extraction based ontology population and maintenance processes, towards new horizons.

We can improve the methodology proposed, to yield better accuracy performances. For an example, we only considered the single word instances in populating the ontology using the defined models. However, in some of the scenarios, phrases also could be instances of ontology classes. Hence, it is important to convert phases into vectors and use them in the methodology as well. Also, as illustrated with models $M4$ and $M5$, we can perform more sophisticated semi-supervised ontology populations based on the concept of this study with more improvements. We keep them to be the future works of this study.

**VI. ACKNOWLEDGEMENT**

The authors would also like to thank Menuka Warushavithana, Thejan Rupasinghe, Rathnayaka and Viraj Salaka from the Department of Computer Science and Engineering of the University of Moratuwa, Sri Lanka, for the immense assistance that they provided in preparing this work.

**REFERENCES**

Word Vector Embeddings and Domain Specific Semantic based Semi-Supervised Ontology Instance Population

10


[33] H. Prins, “Matching ontologies with distributed word embeddings.”


