Synergistic Union of Word2Vec and Lexicon for Domain Specific Semantic Similarity

Keet Sugathadasa, Buddhi Ayesha, Nisansa de Silva, Shehan Perera, Vindula Jayawardana, Dimuthu Lakmal
University of Moratuwa
Sri Lanka
keetmalin.13@cse.mrt.ac.lk
buddhiayesha.13@cse.mrt.ac.lk
nisansadds@cse.mrt.ac.lk
shehan@cse.mrt.ac.lk
vindula.13@cse.mrt.ac.lk
kjtdimuthu.13@cse.mrt.ac.lk

Madhavi Perera
University of London
Sri Lanka
madhaviperera58@gmail.com

ABSTRACT: *Semantic similarity measures are an important part in Natural Language Processing tasks. However Semantic similarity measures built for general use do not perform well within specific domains. Therefore in this study we introduce a domain specific semantic similarity measure that was 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. We prove that this proposed methodology out performs word embedding methods trained on generic corpus and methods trained on domain specific corpus but do not use lexical semantic similarity methods to augment the results. Further, we prove that text lemmatization can improve the performance of word embedding methods.*

Keywords: Word Embedding, Semantic Similarity, Neural Networks, Lexicon, word2vec

Received: 1 May 2017, Revised 30 May 2017, Accepted 6 June 2017

1. Introduction

Semantic Similarity measurements based on linguistic features are a fundamental component of almost all Natural Language Processing (NLP) tasks: Information Retrieval, Information Extraction, and Natural Language Understanding [1]. In the case of NLP based Information Retrieval (IR), it plays into the task of obtaining the items that are most relevant to the query whereas

Information Extraction (IE), plays into the task of correctly recognizing the linguistic elements to be extracted be it the Part of Speech (PoS) tags or Named Entities in Named Entity Recognition (NER). In the case of Text Understanding which is also known as Natural Language Understanding (NLU), it helps in identifying semantic connections between elements on the document that is being analyzed.

Law and order could be rather regarded as the cloak of invisibility that operates and controls the human behavior to its possible extents in the name of justice. Thus in terms of maintaining social order, quiddity of law within the society is mandatory. John Stuart Mill articulated a principle in On Liberty, where he stated that *The only purpose for which power can be rightfully exercised over any member of a civilized community, against his will, is to prevent harm to others* [2]. Being such a vital field, acquisition of laws and legal documents through technological means is on the list of necessities on the rise. Also, ample justification for the need of semantic disambiguation in the legal domain can be found in seminal cases such as: *Fagan v MPC and R v Woollin*. Therefore we selected the law as the domain for this study.

The legal domain contains a considerable amount of domain specific jargon where the etymology lies with mainly Latin and English. To complicate this fact, in certain cases, the meaning of the words and context differs by the legal officers interpretations.

Another field which suffers similarly from this issue is the medical industry [3]. Non-systematic organization and complexity of the medical documents result in medical domain falling victim to loss of having proper representations for its terminology. PubMed [4] attempts to remedy this. Therefore, it is possible to claim that the problem addressed in this study is one that is not just limited to the legal domain but is one that transcends into a numerous other domains as well.

Methods that treat words as independent atomic units is not sufficient to capture the expressiveness of language [5]. A solution to this is word context learning methods [6]– [8]. Another solution is lexical semantic similarity based methods [9]. Both of these approaches try to capture semantic and syntactic information of a word. In this study we proposea methodology to have a synergistic union of both of these methods. First for word context learning, we used a Word Embedding [6] method, word2vec [5]. Then we used a number of lexical semantic similarity measures [10]–[12] to augment and improve the result.

The hypothesis of this study has three main claims: (1) A word embedding model trained on a small domain specific corpus can outperform a word embedding model trained on a large but generic corpus, (2) Word lemmatization, which removes inflected forms of words, would improve the performance of a word embedding model, (3) Usage of lexical semantic similarity measures trained over a machine learningsystem can improve the overall system performance. Our results sufficiently prove all of these claims to be true.

The structure of the paper is as follows. Section 2 gives a brief overview on the current tools being used and domains that have tackled this problem of word representations. Section 3 gives a description on the methodology being used in this research in order to obtain the results and conclusions as necessary. That is followed by Section 4 that presents and analyses results. The paper ends with Section 5 which gives the conclusion and discusses future work.

2. Background And Related Work

This section illustrates the background of the techniques used in this study and work carried out by others in various areas relevant to this research. The subsections given below are the important key areas in this study.

2.1 Lexical Semantic Similarity Measures

Lexical Semantic similarity of two entities is a measure of the likeness of the semantic content of those entities, most commonly calculated with the help of topological similarity existing within an ontology such as WordNet [13]. Wu and Palmer proposed a method to give the similarity between two words in the 0 to 1 range [10]. In comparison, Jiang and Conrath proposed a method to measure the lexical semantic similarity between word pairs using corpus statistics and lexical taxonomy [11]. Hirst & St-Onge's system [12] quantifies the amount that the relevant synsets are connected by a path that is not too long and that does not change direction often. The strengths of each of these algorithms were evaluated in [9] by means of [14].

2.2 Word Vector Embedding

Traditionally, in Natural Language Processing systems, words are treated as atomic units ignoring the correlation between the words as they are represented just by indices in a vocabulary [5]. To solve the inadequacies of that approach, distributed

representation of words and phrases through word embeddings was proposed [15]. The idea is to create vector representations for each of the words in a text document along with word meanings and relationships between the words all mapped to a common vector space.

A number of Word Vector Embedding systems have been proposed such as: GloVe [16], Latent Dirichlet Allocation (LDA) [17], and word2vec1 [18]. GloVe uses a *word to neighboring word* mapping when learning dense embeddings, which uses a matrix factorization mechanism. LDA uses a similar approach via matrices, but the concept is based on mapping words with relevant sets of documents. word2vec uses a neural network based approach that uses *word to neighboring word* mapping. Due to the flexibility and features it provided in terms of parameter passing when training the model using a text corpus, we use word2vec in this study. word2vec supports two main training models: Skip-gram [19] and Continuous Bag Of Words (CBOW) [18].

2.3 Legal Information Systems

Schweighofer [20], claims that there is a huge vacuum that should be addressed in eradicating the information crisis that the applications in the field of law suffer from. This vacuum is evident by the fact that, despite being important, there is a scarcity of legal information systems. Even though the two main commercial systems; WestLaw[2] and LexisNexis[3] are widely used, they only provide query based searching, where legal officers need to remember keywords which are predefined when querying for relevant legal information, there is still a hassle in accessing this information.

One of the most popular legal information retrieval systems is KONTERM [20], which was developed to represent document structures and contents. However, it too suffered from scalability issues. The currently existing implementation that is closest to our proposed model is Gov2Vec [21], which is a system that creates vector representations of words in the legal domain, by creating a vocabulary from across all corpora on, supreme court opinions, presidential actions and official summaries of congressional bills. It uses a neural network [6] to predict the target word with the mean of its context words' vectors. However, the text copora used here, itself was not sufficient enough to represent the entire legal domain. In addition to that, the Gov2Vec trained model is not available to use used by legal professionals or to be tested against.

3. Methodology

This section describes the research that was carried out. Each section below, addresses a component in the overall methodology. An overview of the methodology we propose is illustrated in Fig. 1.

The first phase of this study was to gather the necessary legal cases from on-line repositories. We obtained over 35000 legal case documents, pertaining to various areas of practices in law, from Findlaw [22] by web crawling. Due to this reason, the system tends to be generalized well over many aspects of law.

3.1 Text Lemmatization

The linguistic process of mapping inflected forms of a word to the word's core lemma is called lemmatization [23]. The crawled natural language text would contain words of all inflected forms. However, the default word2vec model does not run a lemmatizer before the word vector embeddings are calculated; which results in each of the inflected forms of a single word ending up with a separate embedding vector. This in turn leads to many drawbacks and inefficiencies. Maintaining a separate vector for each inflected form of each word makes the model bloat up an consume memory unnecessarily. This especially leads to problems in building the model. Further, having separate vector for each inflected form weakens the model because the values for words originating from the same lemma will be distributed over those multiple vectors. For example, when we search for similar words to the noun input "judge", we get the following words as similar words: *Judge, judges, Judges*. Similarly, for verb input "train", the model would return the following set of words: *training, trains, trained*.

As shown in Fig. 1, we use the raw legal text to train the $word2vec_{LR}$ model. But for both $word2vec_{LL}$ model and $word2vec_{LLS}$ model, we first lemmatize the legal document corpus to map all inflected forms of the words to their respective lemmas. For this task we use the Stanford CoreNLP library [24].

[1] https://code.google.com/p/word2vec/
[2] https://www.westlaw.com/
[3] https://www.lexisnexis.com

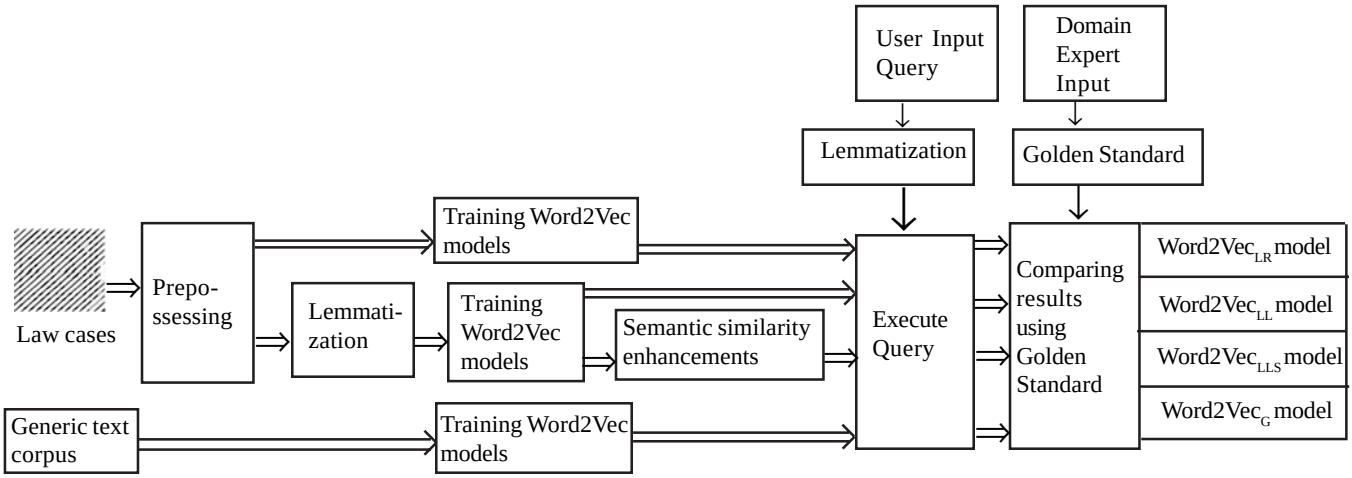

Figure 1. Flow diagram for the Overall Methodology

3.2 Training word2vec models

In this stage we trained word2vec models. One was on the raw legal text corpus and the other was on the lemmatized law text corpus. Each input legal text corpus included text from over 35000 legal case documents amounting to 20 billion words in all. The training took over 2 days. As mentioned in the Section 3.1, the model trained by the raw legal text Fig. 1. Flow diagram for the Overall Methodology corpus is the *word2vec*$_{LR}$ model shown in Fig. 1. The model trained on lemmatized law text corpus is *word2vec*$_{LL}$. Further, a clone of the trained *word2vec*$_{LL}$ is passed forward to 3.3 in order to build the *word2vec*$_{LLS}$ model. Following are the important parameters we specified in training these models.

- size (dimensionality): 200
- context window size: 10
- learning model: CBOW
- min-count: 5
- training algorithm: hierarchical softmax

For this phase, we used a neural network with a single hidden layer as depicted in figure 2. As mentioned above, to train the system to output a user-specified dimensional vector space, we picked the Continuous Bag Of Words approach for learning weights. The rationale as to why CBOW learning model was picked over skip-gram is that, Skip-gram is said to be accurate for infrequent words whereas CBOW is faster by a factor of window size which is also more appropriate for larger text corpora.

In CBOW, the current word or the target word is being predicted based on the context words, within the specified context window size.

In addition to the above trained models, we obtained the word2vec$_{G}$ model which was trained by Google on the Google News dataset. As shown in Fig. 1, this is a generic text corpus that contain data pertaining to a large number of topics. It is not fine tuned to the legal domain as the three models (*word2vec*$_{LR}$, *word2vec*$_{LL}$, *and word2vec*$_{LLS}$) that we train. However, Google's model was trained over on around 100 billion words that add up to 3,000,000 unique phrases. This model was built with layer size of 300. Due to the general nature and the massive case of *word2vec*$_{G}$ model, it is possible to use the comparison of results obtained by our models against the results obtained by this model to showcase the effectiveness of a model trained using a specific domain in the applications of that domain over a model trained using a generic domain in application on the same specific domain.

3.3 Lexical Semantic Similarity Enhancements

At this step we used established lexical semantic similarity measures to enhance the output. As mentioned in Section 3.2, we get a clone of the trained *word2vec*$_{LL}$ model to use in this process. Unlike in the cases of the previous models where the training system is internal to the word2vec model, here it was important that the model is trained explicitly using n-fold cross validation.

As such, a training dataset where each entry is a key value pair is obtained. The key *k* is a lemma of a word in the legal domain and the value is an array of lemmas of words *G* that are most relevant in the legal domain to the word lemma used as the key. The

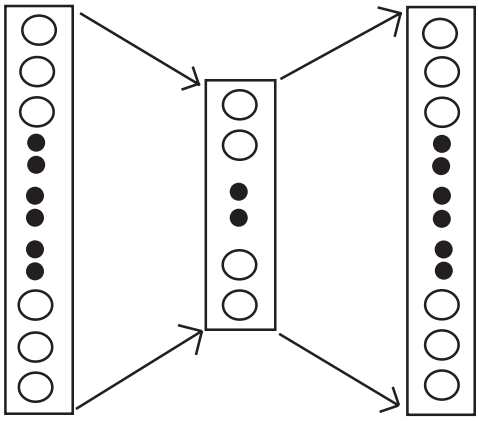

Figure 2. word2vec neural network model: This is a neural network model with a single hidden layer, which is used in the word2vec training process. Input layer and output layer both consist of v neurons where v is the number of words in the vocabulary of the given raw text corpus. The hidden input layer consists of n neurons where n is the intended dimensionality of the vector representation of each word

said array is of length *l*. Following paragraphs explain how the training of the model and the testing happen for the first fold in n-fold cross validation. It is imperative to understand that the same process will be done for each of the folds subsequently.

1) Mathematical model: The first step was to obtain an entry from the training set and to query the $word2vec_{LL}$ model with the key lemma. Let us define an integer *n* such that $n = Cl$ where $C > 1$. When executing the query, $word2vec_{LL}$ model was instructed to return the first *n* elements that matches the query best along with the word2vec similarity values. Let us call this resultant Matrix *R*. *R* has *n* columns where w_i is the word similar to *k* and d_i is the word2vec similarity between *k* and w_i. Equation 1 shows *R*.

$$R = \begin{bmatrix} w_1 & w_2 & w_3 & ... & w_n \\ d_1 & d_2 & d_3 & ... & d_n \end{bmatrix} \quad (1)$$

From *R*, we created the vectors *W* and *D* as shown in Equation 2 and Equation 3 respectively. Each w_i and d_i has the same value as they had in *R*.

$$W = \{w_1, w_2, w_3, ..., w_n\} \quad (2)$$

$$D = \{d_1, d_2, d_3, ..., d_n\} \quad (3)$$

We defined the following functions for words w_i and w_j. The lexical semantic similarity calculated between w_i and w_j using the Wu & Palmer's model [10] was given by $wup(w_i, w_j)$. The lexical semantic similarity calculated by Jiang & Conrath's model [11] was given by $jcn(w_i, w_j)$. $hso(w_i, w_j)$ was used to indicate the lexical semantic similarity calculated using the Hirst & St-Onge's system [12].

Next we created the lexical semantic similarity matrix *M*. The matrix *M* is a $4 \times N$ matrix where *N* has the same value as in Equation 1. The first row element $m_{1,i}$ of *M* was calculated by taking the Wu & Palmer similarity between *k* and w_i. Here w_i element at index *i* of *W*. Thus, $m_{1,i}$ is equal to $wup(k, wi)$. Similarly, the second row element $m_{2,i}$ of *M* was calculated by taking the Jiang & Conrath similarity between *k* and w_i. The third row consists of Hirst & St-Onge similarities while the forth row contains a series of 1s for the sake of the bias. The matrix *M* is shown in Equation 4.

$$M = \begin{bmatrix} wup\ (k,w_1) & wup\ (k,w_2) & .... & wup\ (k,w_n) \\ jcn\ (k,w_1) & jcn\ (k,w_2) & .... & jcn\ (k,w_n) \\ hso\ (k,w_1) & hso\ (k,w_2) & .... & hso\ (k,w_n) \\ 1 & 1 & .... & 1 \end{bmatrix} \quad (4)$$

We defined the normalizing function given in Equation 5 to return a value x_{norm} when given a value x_{raw} that exists between the maximum value v_{max} and the minimum value v_{min}. The returned x_{norm} is a double value that has the range [0; 1].

$$x_{norm} = normalize\ (x_{raw}, v_{min}, v_{max}) \quad (5)$$

We created the matrix *SM* from the matrix *M* by calculating each $sm_{i,j}$ using the Equation 5 on each $m_{i,j}$. For this we used the *min* and *max* values shown in table 1.

Model	**Min**	**Max**
Wu & Palmer	0.0	1.0
Jiang & Conrath	0.0	∝
Hirst & St-Onge	0.0	16.0

Table 1. Lexical Semantic Similarity Statistics

We defined the value matrix *V* using the vector *D* and the Matrix *SM* as shown by Equation 6.

$$V = \begin{bmatrix} D \\ SM \end{bmatrix}^T \quad (6)$$

We defined the vector *E* where each element e_i is given by activating a neural network by values v_i, where v_i is the *i*th row of *V*. The training of the aforementioned neural network is explained in Section 3.3.2.

2) Machine Learning for weight calculation: The motive behind using machine learning is to train the system to take into account how each similarity measure value can be used to derive a new, compound, and better representative value for similarity. As mentioned in Section 3.3.1, a neural network was chosen as the machine learning method. Initially the weight values were initialized to random values and *E* was calculated. Next the matrix *MI* was defined as shown in Equation 7.

$$MI = \begin{bmatrix} W \\ E \end{bmatrix} \quad (7)$$

The matrix *Y* was obtained by sorting the columns of matrix *MI* in the descending order of elements in the second row. We defined a seeking function given in Equation 8 to return the index of word w in the first row of matrix *P*. If the word *w* does not exist in the first row of matrix *P*, it returns the column count of the matrix *P*. Also, observe that the new matrix *Y* has the same form as the initial matrix *R* shown in Equation 1. This symmetrical representation is important because it gives us the opportunity to use the same accuracy measures on all the word2vec models in Section 4 to achieve a fair comparison.

$$s = seek\ (w, P) \quad (8)$$

Next we defined the error *err* according to Equation 10 where the value of x_i was derived from Equation 9 and $\in$ is a small constant.

$$x_i = \begin{cases} 1, & \text{if } seek\ (g_i, Y) < l \\ \dfrac{n\text{-}seek\ (g_i, Y)}{n\text{-}l}, & \text{otherwise} \end{cases} \quad (9)$$

$$err = 1 - \frac{\epsilon + \sum_{i=1}^{l} x_i}{|G \cap W| + \epsilon} \quad (10)$$

This error *err* is used to adjust the neural network. This training cycle is continued until convergence. The completed model that is trained this way is named $word2vec_{LLS}$ model.

3.4 Query processing

In order to use and test our models, we built a query processing system. A user can enter a query in the legal domain using the provided interface. The system then takes the query and applies natural language processing techniques such as *PoS* tagging until the query is through the *NLP* pipeline to be lemmatized. We used the same Stanford Core *NLP* pipeline that we used in Section 3.1 for this task. The reason for this is to bring all the models to the same level to be compared equally. We have shown this step in Fig. 1.

3.5 Experiments

We got experts in the legal field to create a golden standard to test our models. The golden standard includes 100 concepts with each containing 5 words that are most related to the given concept in the legal domain picked from a pool of over 1500 words by the legal experts.

The accuracy levels of these experiments are measured in terms of precision and recall [1], which are common place measures in information retrieval. The logical functionality of these two are based on the comparison of an expected result and the effective result of the evaluated system.

If the Golden Standard Word Vector is *G* and the word vector returned by the model is *W* (Same naming conventions as Section 3.3.1), the recall in this study is calculated with equation 11. This measures the completeness of the model. Our recall calculation used the same function suggested in [1].

$$recall = \frac{|G \cap W|}{|G|} \quad (11)$$

The precision calculation in this study is not as clear cut as it is described in [1]. This is because in those systems the precision is only a matter of set membership and thus would simply be ratio between the correctly found similar words over the total number of returned words. However, in the case of word2vec models, it is the user who input the number of matching words to retrieve. Thus, it is wrong to use the total number of returned words to calculate the precision in cases where there is prefect recall. In the cases of imperfect recall, the classical precision is adequate.

While word2vec make precision calculation difficult as shown above, it also has a quality that makes finding the solution to that problem somewhat easy. That is the fact that the returning word list of similar words is sorted in the descending order. This is the same property that we used in the above Section 3.3.2 to calculate the error. Thus it is logical to derive that the precision is given by equation 12 where *err* is the error calculated by equation 10.

$$precision = 1 - err \quad (12)$$

4. Results

This section includes results obtained from the four different models ($word2vec_G$, $word2vec_{LR}$, $word2vec_{LL}$, and $word2vec_{LLS}$) as introduced in Section 3.2. Results shown in table 2 were obtained for different *k* values, where *k* is the number of words requested from each of the models. As expected, the *F1* of each model increases with *k*, where the possibility of finding the correct similar words against the golden standard increases. Given that the task here is to return the expected set of words, the recall is more important than precision (i.e: False-Negatives are more harmful than False-Positives). In that light, it is obvious that the $word2vec_{LLS}$ performs better than all other models because it consistently has the highest recall for all values of *k*. In addition to that, the $word2vec_{LLS}$ model also has the highest *F1* for all values of *k*, which is sufficient proof that the small loss in precision does not adversely affect the overall result.

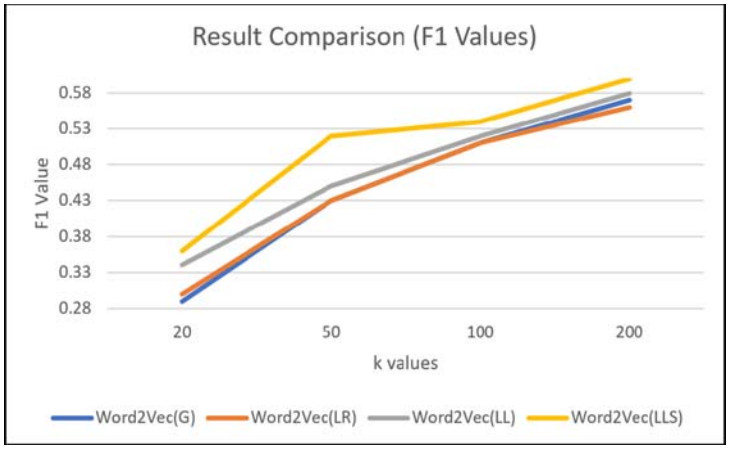

Figure 3. F1 value comparison

A graphical comparison of the changes in the *F1* measure is shown in Fig. 3. As shown, the domain specific models such as $word2vec_{LR}$, $word2vec_{LL}$, and $word2vec_{LLS}$, show better results than the general $word2vec_{G}$ model. It should be noted that this performance enhancement has happened despite the fact that $word2vec_{G}$ was trained on a text corpus 3 times bigger than the text corpora we have used in this study for the models $word2vec_{LR}$, $word2vec_{LL}$, and $word2vec_{LLS}$.

In the comparison of domain specific models, we can see a clear distinction between the $word2vec_{LR}$ and $word2vec_{LL}$ models, where the $word2vec_{LL}$ generally performs better. Further, it can be observed that the $word2vec_{LLS}$ model outperforms both $word2vec_{LR}$ and $word2vec_{LL}$ models.

5. Conclusion And Future Works

The hypothesis of this study had three main claims and each of these claims were justified by the results presented in Section 4. The first claim of the hypothesis is that a word embedding model trained on a small domain specific corpus can out perform a word embedding model trained on a large but generic corpus. The success of $word2vec_{LR}$ model over the $word2vec_{G}$ model justifies this claim. In Section 3.1 we proposed the second claim: word lemmatization, which removes inflected forms of words and improve the performance of a word embedding model. $word2vec_{LL}$ model obtaining better results than $word2vec_{LR}$ model proved this claim to be true. The third claim was made in Section 3.3. There, we proposed that usage of lexical semantic similarity measures trained over a machine learning system can improve the overall system performance. The significant improvement that we show for the $word2vec_{LLS}$ model over the $word2vec_{LL}$ model verified this claim also to be accurate. Therefore we can conclude that the proposed methodology of word vector embedding augmented by lexical semantic similarity measures, gives a more accurate evaluation of the extent of which a given pair of words is semantically similar in the considered domain.

Model	k=20			k=50			k=100			k=200		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
$word2vec_{G}$	0.57	0.19	0.29	0.62	0.33	0.43	0.67	0.41	0.51	0.74	0.46	0.57
$word2vec_{LR}$	0.75	0.19	0.30	0.71	0.31	0.43	0.74	0.38	0.51	0.77	0.44	0.56
$word2vec_{LL}$	0.73	0.22	0.34	0.72	0.32	0.45	0.75	0.40	0.52	0.76	0.47	0.58
$word2vec_{LLS}$	0.66	0.24	0.36	0.73	0.33	0.52	0.72	0.43	0.54	0.74	0.50	0.60

Table 2. Results Comparison (P =Precision, R = Recall)

Semantic similarity measures are important in many areas of applications. Out of those, for future work, we expect to extend the findings of this study to the document level. Word based semantic similarity is the building block for sentence similarity

measures, which in turn aggregates to build document similarity measures. This is the direction in which we intend to move. We will be using this word semantic similarity measure to build up to a document similarity measure which can be used for more efficient domain based document retrieval systems.

References

[1] Wimalasuriya, D. C., Dou, D. (2010). Ontology-based information extraction: An introduction and a survey of current approaches. *Journal of Information Science*.

[2] Mill, J. S., On liberty. (1999). Broadview Press.

[3] Oliver, D. E., Shahar, Y., Shortliffe, E. H., Musen. (1999). Representation of change in controlled medical terminologies. *Artificial Intelligence in Medicine*, 15 (1), p. 53–76.

[4] N. C. for Biotechnology Information. (2017) Pubmed help. [Online]. Available: https://www.ncbi.nlm.nih.gov/books/NBK3827/

[5] Mikolov, T., Chen, K., Corrado, G., Dean, J. (2013). Efficient estimation of word representations in vector space. arXiv preprint arXiv: 1301- 3781.

[6] Bengio, Y., Ducharme, R., Vincent, P., Jauvin, C. (2003). A neural probabilistic language model. *Journal of Machine Learning Research*, 3. p. 1137–1155.

[7] De Silva, N., Fernando, C., Maldeniya, M., Wijeratne, D., Perera, A., Goertzel, B. (2011). Semap-mapping dependency relationships into semantic frame relationships. *In:* 17th ERU Research Symposium, vol. 17. Faculty of Engineering, University of Moratuwa, Sri Lanka.

[8] Ng, H. T., Lee, H. B. (1996). Integrating multiple knowledge sources to disambiguate word sense: An exemplar-based approach. *In:* Proceedings of the 34th annual meeting on Association for Computational Linguistics. Association for Computational Linguistics, p. 40–47.

[9] De Silva, N. (2015). Safs3 algorithm: Frequency statistic and semantic similarity based semantic classification use case. In Advances in ICT for Emerging Regions (ICTer), *2015* Fifteenth International Conference on. IEEE, p. 77–83.

[10] Wu, Z., Palmer, M. (1994). Verbs semantics and lexical selection. *In*: Proceedings of the 32 nd Annual Meeting on Association for Computational Linguistics, ser. ACL '94. Stroudsburg, PA, USA: Association for Computational Linguistics, p. 133–138. [Online]. Available: http://dx.doi.org/10.3115/981732.981751

[11] Jiang, J.J., Conrath, D.W. (1997). Semantic similarity based on corpus statistics and lexical taxonomy. *In:* Proceedings of the10th International Conference on Research in Computational Linguistics, ROCLING97.

[12] Hirst, G., St-Onge, D. (1998). Lexical chains as representations of context for the detection and correction of malapropisms. WordNet: An electronic lexical database, 305 p. 305–332.

[13] Miller, G. A., Beckwith, R., Fellbaum, C., Gross, D., Miller, K. J. (1990). Introduction to wordnet: An on-line lexical database. *International Journal of Lexicography*, 3 (4) 235–244.

[14] Shima. H. (2016). Wordnet similarity for java (ws4j). [Online]. Available: https://code.google.com/p/ws4j/

[15] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *In:* Advances in Neural Information Processing Systems, p. 3111–3119.

[16] Pennington, J., Socher, R., Manning, C. D. (2015). Glove: Global vectors for word representation. In EMNLP, 14, 2014, p. 1532–1543.

[17] Das, R., Zaheer, M., Dyer, C. *Gaussian lda for topic models with word embeddings*. *In*: ACL (1) p. 795–804.

[18] Goldberg, Y., Levy, O. (2014). word2vec explained: Deriving mikolov et al.'s negative-sampling word-embedding method. arXiv preprint arXiv: 1402.3722.

[19] Melamud, O., Levy, O., Dagan, I., Ramat-Gan, I. (2015). A simple word embedding model for lexical substitution. *In:* Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing. p. 1–7.

[20] Schweighofer, E., Winiwarter, W. (1993). Legal expert system kontermautomatic representation of document structure and contents. *In*: International Conference on Database and Expert Systems Applications. Springer, p. 486–497.

[21] Nay, J.J. (2016). Gov2vec: Learning distributed representations of institutions and their legal text. arXiv preprint arXiv:1609.06616.

[22] Hughes, J. (1999). Rules for mediation in findlaw for legal professionals.

[23] Korenius, T., Laurikkala, J., arvelin, K. J., Juhola, M. (2004). Stemming and lemmatization in the clustering of finnish text documents. *In:* Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management. ACM, p. 625–633.

[24] Manning, C.D., Surdeanu, M., Bauer, J., Finkel, J.R., Bethard, S., McClosky, D. (2014). The stanford corenlp natural language processing toolkit. *In:* ACL (System Demonstrations), p. 55–60.