# Leveraging Ontological Knowledge for Neural Language Models

Ameet Deshpande[*]
Indian Institute of Technology Madras
Chennai, India
ameetsd97@gmail.com

Monisha Jegadeesan
Indian Institute of Technology Madras
Chennai, India
monishaj@cse.iitm.ac.in

## ABSTRACT

Neural Language Models such as Word2Vec and GloVe have been shown to encode semantic relatedness between words. Improvements in unearthing these embeddings can ameliorate performance in numerous downstream applications such as sentiment analysis, question answering, and dialogue generation. Lexical ontologies such as WordNet are known to supply information about semantic similarity rather than relatedness. Further, extracting word embeddings from small corpora is daunting for data-hungry neural networks. This work shows how methods that conflate Word2Vec and Ontologies can achieve better performance, reduce training time and help adapt to domains with a minimum amount of data.

## KEYWORDS

Word Vectors, Domain-transfer, Ontology, Hierarchy

**ACM Reference Format:**
Ameet Deshpande and Monisha Jegadeesan. 2019. Leveraging Ontological Knowledge for Neural Language Models. In *6th ACM IKDD CoDS and 24th COMAD (CoDS-COMAD '19), January 3–5, 2019, Kolkata, India.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3297001.3297059

## 1 INTRODUCTION

The Word2Vec model [10] has been used for many problems and possesses interesting semantic and arithmetic properties. However, it encodes semantic relatedness rather than semantic similarity, has no notion of hierarchies, and is data-hungry (the best performing vectors were trained on 100 billion words). Ontologies like WordNet [11] are more specific in the information they possess, which is manifested in the form of its structure and informative glosses. Thus, it makes sense to combine these two methods to derive the best of both the worlds. The rest of the paper is structured as follows. Section 2 summarizes previous relevant work, sections 3 and 4 show the advantages of weight initialization by achieving better performance, section 5 introduces a relation constrained model adapted for domains, section 6 proposes a method to incorporate hierarchies in the word vectors, and section 7 lists our directions for future work.

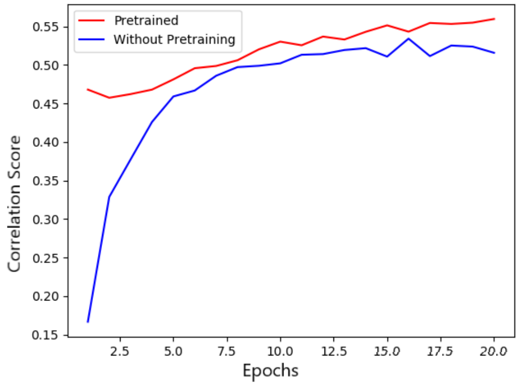

**Figure 1: British National Corpus**

## 2 RELATED WORK

There have been a few efforts which have tried to address the Data-Knowledge Tradeoff. The Joint-RCM model proposed in [17] attempts to add constraints in the form of pairs $(w_1, w_2)$ such that the respective vectors have higher similarity, and outperforms vanilla Word2Vec when Mean Reciprocal Rank is used as an evaluation measure. Dict2Vec [13] defines these constraints based on natural language dictionaries, motivated by the fact that they capture contextual co-occurrences more concisely. There has been work [2] on the extension of such constraints to entity relationships as well. Ordinal Knowledge Constraints proposed in [7] allow weaker constraints like $sim(w_1, w_2) > sim(w_1, w_3)$ to drive learning. Thus, instead of having a large number of word pairs which are synonyms, ordinal knowledge constraints can be extracted more easily and can represent multiple types of relationships. RC-NET [16] jointly learns word embeddings using the skip-gram objective combined with a knowledge base consisting of relations between words or full-fledged word clusters, making it even easier to automate the process of curating constraints because of the prevalence of clustering techniques and entity-relationship extractors.

## 3 WEIGHT INITIALIZATION

Weight Initialization has been shown [5] to influence the minima that the neural network reaches. Further, weight initialization gives a warm-start for training. We use the WordNet glosses for learning embeddings and use that as a pre-trained model. Subsets of British National Corpus (BNC) [4] and Wikipedia are used as training data, and correlation score on WordSim353 [1] is used as the evaluation metric. Results are shown in Figures 1 and 2.

Training on BNC shows the advantage that warm-start gives, and that pre-training allows the model to achieve a higher score faster. Training curves on Wikipedia Corpus corroborate that Weight

[*] Fourth Year Undergraduate Student

*CoDS-COMAD '19, January 3–5, 2019, Kolkata, India*

ACM ISBN 978-1-4503-6207-8/19/01…$15.00
https://doi.org/10.1145/3297001.3297059

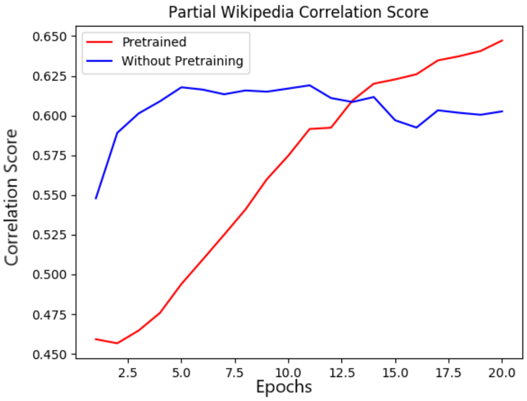

**Figure 2: Wikipedia Corpus**

**Table 1: Effect of varying corpus size on performance**

| Corpus | Not Pretrained | Pretrained |
|---|---|---|
| $\frac{enwiki}{4}$ | 0.6113 | 0.6479 |
| $\frac{2*enwiki}{4}$ | 0.6215 | 0.6565 |
| $\frac{3*enwiki}{4}$ | 0.6186 | 0.6479 |
| $\frac{4*enwiki}{4}$ | 0.6026 | 0.6472 |

**Table 2: Convergence Time on BNC**

| Rel. Size | Score for vectors without pretraining | # epochs to reach score for pre-trained |
|---|---|---|
| 1 | 0.5159 | 9 |
| 3 | 0.5435 | 13 |
| 6 | 0.5636 | 15 |

Initialization will enable the model to explore parts of the space which would have otherwise been inaccessible.

Further, it was interesting to note that the pre-trained model achieves better performance with lesser data. The Wikipedia Corpus used was divided into 4 parts, and the pre-trained model outperformed all four of the non-pre-trained models (Table 1).

In many cases, the task at hand only requires the model to achieve a threshold score. In such cases, the training or convergence time is a better evaluation metric than performance. Different subsets of BNC corpus were used for training a model from scratch for 20 epochs, and the number of epochs for the pre-trained model to achieve that score was recorded. As shown in table 2, the latter consistently outperforms the former, thus giving better embeddings in significantly lesser number of epochs. Rel. Size is the relative size of the corpus and the second column represents the score after training a model from scratch.

Even a method as naive as weight initialization can be powerful when WordNet, which encodes structure and semantic similarity, is used. The vectors are initialized to crude values which represent relationships and fine-tuning them gives an unprecedented performance. This motivated us to explore other ways to leverage ontologies.

**Table 3: Construction Domain Task**

| Training Data | Num epochs | Accuracy (%) |
|---|---|---|
| Construction Corpus | 2 | 73.2 |
| Construction Corpus | 4 | 74.8 |
| Construction Corpus with WordNet Weight Initialization | 2 | 74.4 |

## 4 DOMAIN TRANSFER USING WEIGHT INITIALIZATION

Domain Transfer involves using word-embeddings learned on a different domain or dataset and adapting it to the one at hand. In some cases, it might not suffice to learn embeddings just on a corpus related to the target domain. The following are a few reasons.

- Some domains do not have large datasets, especially niche ones, and this exacerbates the of performance data-hungry Neural Networks
- The target domain's corpus may not represent relationships with more general words and may thus lack "common sense" knowledge

A fairly straight-forward way to address this is to use a pre-trained model and then fine tune it on the target domain corpus. The pre-trained model is expected to possess "common sense" knowledge and possibly encode some crude relationships between words in the target domain, and WordNet seems like a good choice because of the same. Similar to the previous section, we use the glosses in WordNet to initialize the model and adapt it to the *construction* domain.

The skip-gram model was trained on the WordNet glosses for 25 epochs, followed by the construction corpus [14] for 2 epochs. These embeddings were used to evaluate the performance on a classification task in the construction domain. The injury report dataset [14] is used to predict the class of injury (fatality, non-hospitalized and hospitalized) from the injury report and the concatenated word embeddings of keywords are used as features. A single layer Neural Network is used as the classifier.

As can be seen in Table 3, higher accuracy is reached in a lesser number of epochs (2) when weight initialization is used. Moreover, WordNet glosses constitute a relatively small corpus, and the training required for weight initialization is a one-time process and can be used for multiple tasks, thus reducing the amortized cost. However, training on the domain corpus is expensive.

## 5 USING RELATION CONSTRAINED MODEL

The joint model proposed in [17] uses both a corpus to train, and knowledge in the form of constraints $(w_1, w_2)$, whose aim is to bring the embeddings of these two words closer. The Continuous Bag-of-Words (CBOW) version of Word2Vec is used. Note that $c$ represents the window of the context.

$$\frac{1}{T}\sum_{t=1}^{T}\log p(w_t|w_{t-c}^{t+c}) + \frac{C}{N}\sum_{i=1}^{N}\sum_{w\in R_{w_i}}\log p(w|w_i) \qquad (1)$$

**Table 4: Similarity on Construction Domain**

| Word 1 | Word 2 | Without RCM | With RCM |
|---|---|---|---|
| *Synonymous Words* | | | |
| baseboard | mopboard | 0.06 | 0.35 |
| perspective | view | 0.52 | 0.52 |
| trump | trumpet | 0.21 | 0.25 |
| horizon | skyline | 0.22 | 0.28 |
| *Unrelated Words* | | | |
| alcove | capital | 0.04 | 0.02 |
| flat | service | 0.11 | 0.10 |

The first term is CBOW model's loss function, and the second term is from the Relation Constrained Model (RCM). The RCM objective expects the model to predict $w_2$ given $w_1$, thus bringing their vector representations closer. Though the relations extracted in [17] are mainly synonyms, the above formulation is general and can be used to capture entity relationships, structural relationships and even common misspellings. We extend the above formulation to include relation-specific weights, which allows us to emphasize on different relations with different degrees.

$$\frac{1}{T}\sum_{t=1}^{T}\log p(w_t|w_{t-c}^{t+c}) + \frac{C}{N}\sum_{i=1}^{N}\sum_{w\in R_{w_i}}\log p(w|w_i)\times\lambda(w,w_i) \quad (2)$$

One crucial condition (not mentioned in [17]) for the constraints to give better performance is to ensure that there are multiple occurrences of the words $w_1$ and $w_2$ with other words. CBOW has two matrices, the *Word Matrix* and the *Context Matrix*. Training on pairs like $(w_1, w_2)$ brings the vector representing $w_1$ in the *word matrix* and the vector representing $w_2$ in the *context matrix* closer, but what we actually want is to bring both their representations in the word matrix closer. This is ensured in [17] by using a very large number of relations ($\approx$ 6 lakh), but we use a much smaller set of 1000 constraints obtained from WordNet Domain [8], with $C = \frac{1}{12}$. The constraints are obtained by an automated process which considers synonyms of domain-specific words.

We train on the *construction* domain with corpus from [14]. We evaluate the similarity of words from the domain in both cases and observe that the RCM model improves the similarity scores and brings synonymous words closer to each other. Due to space constraints, we report only a few word pairs in Table 4, but the trend is the same across a large number of other word pairs.

## 6 INDUCING HIERARCHIES

Hierarchies are not only useful for computational advantages such as organization and faster retrieval, but their structure also represents a similarity between concepts and words. Word2Vec learns "flat" embeddings. However, there may be many advantages of using WordNet's hierarchy to train vectors. Consider a hierarchy as in Figure 3. The word embeddings would be expected to increase the similarity between the words "surgeon" and "doctor", and ensure that it is more than the similarity between "surgeon" and "nurse". Such useful information can be excavated without the need of any data, and to this end, we propose three models.

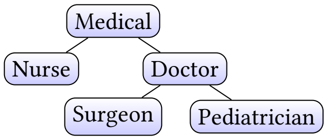

**Figure 3: Example Hierarchy**

### 6.1 Hierarchical RCM (HRCM)

While equation 2 is powerful because of relation-dependent weights, it is hard to tune them in practice. Instead, we could decide them based on the level in the hierarchy. This is motivated by WordNet similarity scores like LCH [6] and WUP [15]. Let's say $lvl(w)$ is a function that denotes what level in the hierarchy the word is present in and $lcs$ returns the Lowest Common Subsumer. The constraint which is equivalent to the aforementioned description is

$$\sum_{i=1}^{N}\sum_{w\in R_{w_i}}\log p(w|w_i)\times\lambda(lvl(w), lvl(w_i), lvl(lcs(w, w_i))) \quad (3)$$

The function $\lambda$ represents the following properties.

- Words on the same branch and deeper in the hierarchy should be more similar than shallow pairs
- Words on the same level but different branches should be less similar when compared to their LCS

These properties are easy to express as a function, and the hope is that HRCM results in word vectors which respect the hierarchy. Further, the number of constraints can be reduced by realizing that three constraints (or generally, $\binom{n}{2}$) of the form $(w_1, w_2), (w_2, w_3)$, $(w_3, w_1)$ can be reduced to 2 constraints (or generally, $n - 1$) of the form $(w_1, w_2), (w_2, w_3)$ because of transitivity. For a node with an out-degree of $m$ in a tree, there will be $m$ constraints, and thus the number of constraints will be $O(|E|) \approx O(|V|)$, where $|E|$ is the number edges and $|V|$ is the number of vertices.

### 6.2 Hierarchical Ordinal Knowledge Constraints

Work proposed in [7] allows an easy extension to the case at hand. Since we want the similarity scores to respect certain inequalities, as mentioned at the beginning of the section, we can perform a Depth-First Search on the tree and assign higher similarity scores to parent-children pairs which are deeper down in the hierarchy. In the case of Figure 3, the list of constraints for the node *doctor* will be *sim*(medical, doctor) < *sim*(doctor, surgeon), and *sim*(medical, doctor) < *sim*(doctor, pediatrician).

### 6.3 Using keywords as pivots

Often, there are specific words in domains which are infrequent in corpora which do not target that domain. We would still want to have two-level hierarchies, where the first level indicates the domain of the word and the second one indicates the word itself. Figure 4 illustrates how the words in leaf nodes are uncommon and might end up being misrepresented because of a general corpus.

The upshot of word embeddings representing semantic similarity is that words from domains tend to form close clusters. This is manifested in works like [3] which use those clusters for downstream

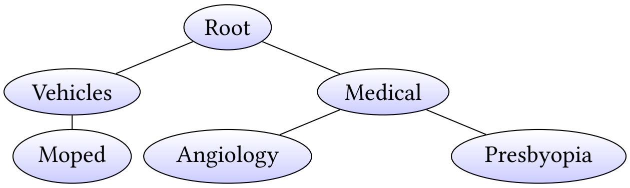

**Figure 4: Two-level Hierarchy**

applications. These clusters allow us to define the word vectors of infrequent words either in terms of keywords from that domain, or as an average of words in that domain, and then fine tune it on domain-specific datasets. We choose the former for illustration in Algorithm 1. Note that it is easy to extract keywords from text using algorithms like RAKE [12] or TextRank [9].

**Algorithm 1:** Pseudo-code for Hierarchies Using Pivots

```
word_vectors ← train(general_corpus);
domain_keywords ← RAKE(domain_corpus);
domain_representative ←
  Mean(word_vectors(domain_keywords));
Initialize infrequent_domain_words with
  domain_representative;
Fine Tune on domain_corpus by fixing other word_vectors;
```

In the last step of the algorithm, vectors from other domains are not changed, and only the ones within the domains are changed. This is not a severe restriction because these words can still move closer to words outside of the domain, though the other way round is not allowed.

## 7 CONCLUSION AND FUTURE WORK

Several papers reviewed in this work show that Ontological information can aid in training word vectors by either improving performance, reducing the amount of data needed, or improving training efficiency. These problems can be direr than they seem at first sight because many domains do not even have enough data for training.

The experiments we performed using Weight Initialization showed that informative glosses and structure inherent in WordNet could not only accelerate learning but also improve performance, a phenomenon commonly observed in unsupervised pre-training [5]. We further showed that it is fairly straightforward to extend it to domain transfer. It is essential to use a pre-trained model which already encodes relationships between general words, and ontologies like WordNet can realize such models.

From the experiments involving the extension of the RCM model for domain adaptation, we observed that ontologies like WordNet Domain could provide a boost in performance by explicitly enforcing similarity between synonyms.

While most of the methods described so far capitalize on the structure that ontologies provide, none of them use the tree or DAG structure explicitly. We proposed three models, the HRCM, Hierarchical Ordinal Knowledge Constraints, and Keywords as Pivots, which incorporate constraints like relative ordering between similarity and explicit coalescing of words which are synonyms. We believe that hierarchies are important if we want the inferred word vectors to emulate how human beings represent words.

The optimistic goal of this work is to try and quantify the Data-Knowledge trade-off. After having sufficiently motivated that knowledge bases like ontologies can help reduce the requirement of data, it is important to quantify the reduction. We hope we will be able to compare the quality and quantity of corpus with the number of ontological relations. An informative example would be to say that 1 million words more in the corpus is equivalent to using an ontology instead. This will answer important questions regarding the amount of data that needs to be collected, which in turn finds the sweet spot for the trade-off. After all, data acquisition is the most expensive part of the pipeline in many Machine Learning problems.

## REFERENCES

[1] Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Paşca, and Aitor Soroa. 2009. A study on similarity and relatedness using distributional and wordnet-based approaches. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, 19–27.

[2] Asli Celikyilmaz, Dilek Hakkani-Tur, Panupong Pasupat, and Ruhi Sarikaya. 2010. Enriching word embeddings using knowledge graph for semantic tagging in conversational dialog systems. *genre*.

[3] Miriam Cha, Youngjune Gwon, and HT Kung. 2017. Language modeling by clustering with word embeddings for text readability assessment. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. ACM, 2003–2006.

[4] The British National Corpus. 2007. *version 3 (BNC XML Edition)*. Distributed by Bodleian Libraries, University of Oxford, on behalf of the BNC Consortium. http://www.natcorp.ox.ac.uk/.

[5] Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. 2010. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11, Feb, 625–660.

[6] Claudia Leacock and Martin Chodorow. 1998. Combining local context and wordnet similarity for word sense identification. *WordNet: An electronic lexical database*, 49, 2, 265–283.

[7] Quan Liu, Hui Jiang, Si Wei, Zhen-Hua Ling, and Yu Hu. 2015. Learning semantic word embeddings based on ordinal knowledge constraints. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Vol. 1, 1501–1511.

[8] Bernardo Magnini and Gabriela Cavaglia. 2000. Integrating subject field codes into wordnet. In *LREC*, 1413–1418.

[9] Rada Mihalcea and Paul Tarau. 2004. Textrank: bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*.

[10] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, 3111–3119.

[11] George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38, 11, 39–41.

[12] Stuart Rose, Dave Engel, Nick Cramer, and Wendy Cowley. 2010. Automatic keyword extraction from individual documents. *Text Mining: Applications and Theory*, 1–20.

[13] Julien Tissier, Christophe Gravier, and Amaury Habrard. 2017. Dict2vec: learning word embeddings using lexical dictionaries. In *Conference on Empirical Methods in Natural Language Processing (EMNLP 2017)*, 254–263.

[14] Antoine J-P Tixier, Michalis Vazirgiannis, and Matthew R Hallowell. 2016. Word embeddings for the construction domain. *arXiv preprint arXiv:1610.09333*.

[15] Zhibiao Wu and Martha Palmer. 1994. Verbs semantics and lexical selection. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 133–138.

[16] Chang Xu, Yalong Bai, Jiang Bian, Bin Gao, Gang Wang, Xiaoguang Liu, and Tie-Yan Liu. 2014. Rc-net: a general framework for incorporating knowledge into word representations. In *Proceedings of the 23rd ACM international conference on conference on information and knowledge management*. ACM, 1219–1228.

[17] Mo Yu and Mark Dredze. 2014. Improving lexical embeddings with semantic knowledge. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Vol. 2, 545–550.