Exploiting Morphological Regularities in Distributional Word Representations

Arihant Gupta **Syed Sarfaraz Akhtar** **Avijit Vajpayee**
Arjit Srivastava **Madan Gopal Jhanwar** **Manish Shrivastava**
{arihant.gupta, syed.akhtar, arjit.srivastava, madangopal.jhanwar}@research.iiit.ac.in,
avijit.vajpayee@students.iiit.ac.in,
manish.shrivastava@iiit.ac.in
Language Technologies Research Center(LTRC)
Kohli Center On Intelligent Systems (KCIS)
International Institute of Information Technology Hyderabad

Abstract

We present an unsupervised, language agnostic approach for exploiting morphological regularities present in high dimensional vector spaces. We propose a novel method for generating embeddings of words from their morphological variants using morphological transformation operators. We evaluate this approach on MSR word analogy test set (Mikolov et al., 2013d) with an accuracy of 85% which is 12% higher than the previous best known system.

1 Introduction

Vector representation of words are presently being used to solve a variety of problems like document classification (Sebastiani, 2002), question answering (Tellex et al., 2003) and chunking (Turian et al., 2010).

Word representations capture both syntactic and semantic properties (Mikolov et al., 2013d) of natural language. Soricut and Och (2015) exploited these regularities to generate prefix/suffix based morphological transformation rules in an unsupervised manner. These morphological transformations were represented as vectors in the same embedding space as the vocabulary.

Using Soricut's transformation rules, the major problem is identifying correct rule to apply to a word, i.e. if we have to generate an embedding for "runs", which rule to apply on "run". Experimental results showed that "walk - walks" gives better results than rules like "invent - invents" or "object - objects" in generating word embedding for "runs". In this paper, we try to explore if we can harness this morphological regularity in a much better way, than applying a single rule using vector arithmetic.

Hence, we tried to come up with a global transformation operator, which aligns itself with the source word, to give best possible word embedding for target word. We will have a single transformation operator for each rule, irrespective of the form of root word (like verb or a noun). Our transformation operator is in the form of a matrix, which when applied on a word embedding (cross product of vector representation of word with transformation matrix) gives us a word embedding for target word.

The intuition is not to solve for "invent is to invents as run is to ?" or "object is to objects as run is to ?", but instead we are solving for "walk is to walks, object is to objects, invent is to invents, ... as run is to ?". A transformation operator aims to be a unified transition function for different forms of the same transition.

The idea of projection learning has been applied to a multitude of tasks such as in the learning of cross lingual mappings for translation of English to Spanish (Mikolov et al., 2013b). Our approach has its basis on the same lines but with a different formulation and end goal to learn morphological rules rather than semantic associations and translational constraints (as done by Mikolov).

In summary, the main contributions of this paper is a new method to harness morphological regularities present in high dimensional word embeddings. Using this method, we present state of the art results on MSR word analogy dataset.

This paper is structured as follows. We first discuss the corpus used for training the transformation operators in section 2. In section 3, we discuss how these transformation operators are trained. Later in sections 4, we analyze and discuss the results of our experiments. We finish this paper with future scope of our work in section 5.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 292–297
Copenhagen, Denmark, September 7–11, 2017.

2 Datasets

We are using word embeddings trained on Google News corpus (Mikolov et al., 2013c) for our experiments. For the model trained in this paper, we have used the Skip-gram (Mikolov et al., 2013a) algorithm. The dimensionality has been fixed at 300 with a minimum count of 5 along with negative sampling. As training set and for estimating the frequencies of words, we use the Wikipedia data (Shaoul, 2010). The corpus contains about 1 billion tokens.

The MSR dataset (Mikolov et al., 2013d) contains 8000 analogy questions. This data set has been used by us for testing our model. The relations portrayed by these questions are morpho-syntactic, and can be categorized according to parts of speech - adjectives, nouns and verbs. Adjective relations include comparative and superlative (good is to best as smart is to smartest). Noun relations include singular and plural, possessive and non-possessive (dog is to dog's as cat is to cat's). Verb relations are tense modifications (work is to worked as accept is to accepted).

For all the experiments, we have calculated the fraction of answers correctly answered by the system on MSR word analogy dataset.

3 Transformation Matrix

The thresholds mentioned in this section have been determined after empirical fine tuning.

To compute the transformation matrix of a rule, we first extract in an unsupervised way all the word pairs following that transition rule. For example, in case of the rule <null,s>, we find word pairs such as <boy,boys>, <object,objects> and <invent,invents>. In this paper, we used the data structure TRIE for computational optimization. However, we don't want cases which do not follow the general regularity of a rule. One such case may be <hat,hated> which does not follow the general trend of the transformation <null,ed>. For eliminating these cases, we set a threshold of cosine similarity of the word vectors of the two words of the pair at 0.2. Also, the frequency of both these words should be greater than 1000 (so that they are well trained). Since our transformation matrix is derived from all the word pairs following a particular transition rule, we carefully use only those word pairs which are of high frequency. We do so because highly frequent words have better trained word embeddings.

Suppose we get "N" highly frequent word pairs following the same regularity(transition rule). For our experiments, the lower threshold of "N" is set at 50. Dimensions of word embedding of a word in our model is "D". Using first word of our "N" chosen word pairs, we create a matrix "A" of dimensions N*D, where each row is vector representation of a word. Similarly, we create another matrix B, of similar dimensions as A, using second word of our chosen word pairs.

We now propose that a matrix "X" (our transformation matrix) exists such that,

$$\begin{aligned} A * X &= B \\ or, X &= A^{-1} * B \end{aligned} \tag{1}$$

(all instances of A that we encountered were non-singular). Our matrix "X" will be of dimensions "D*D" and when applied to a word embedding (matrix of dimensions 1*D, it gives a matrix of dimensions 1*D as output), it results in the word embedding of the transformed form of the word.

Due to inverse property of a matrix, it accurately remembers the word pairs used for computing. The matrix also appears to align itself with the word embedding of other words (not used for its training) to transform them according to the rule that the matrix follows. Some interesting results are shown in table 1.

Word1	Word2	Word3	Operator	Word4	Cosine
decides	decided	studies	<s , d>	studied	0.89
reach	reaches	go	<null , es>	goes	1.0
ask	asks	reduce	<null , s>	reduces	0.91

Table 1: Some example results of transformation operators.

While testing, we extract the syntactic transition using the first two words of the analogy question. For example, for pairs like <reach, reached>, <walk, walked>, we are able to extract that they follow <null, ed> rule syntactically. But, for <go, went>, we are not able to find any transformation operator after syntactic analysis, and for such cases, we fall back on CosSum/CosMul (Levy et al., 2014) approaches as our backup. Mikolov et al. showed that relations between words are reflected to a large extent in the offsets between their vector embeddings (queen - king = woman - man), and thus the vector of the hidden word b^* will be similar to the vector $b - a + a^*$, suggesting that the

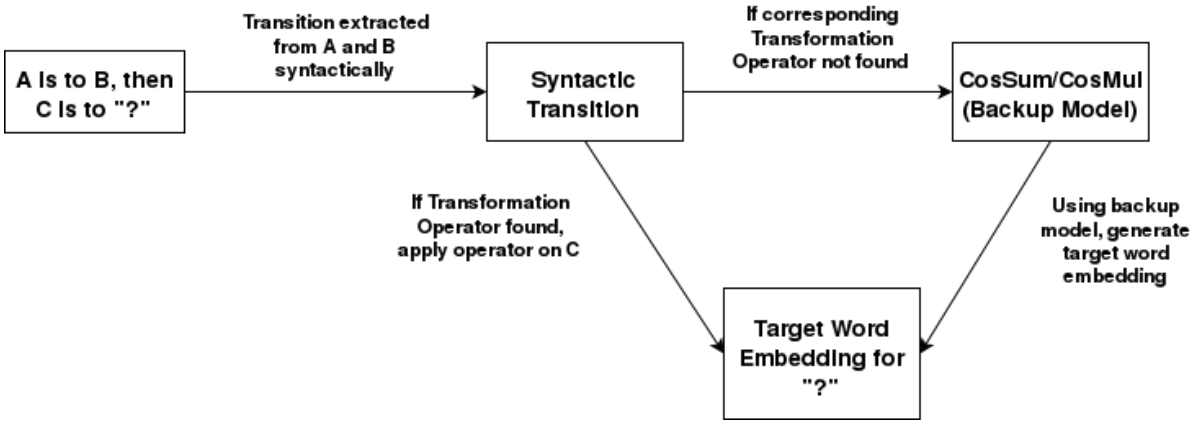

Figure 1: System Workflow

analogy question can be solved by optimizing:

$$arg \max_{b^* \in V} (sim(b^*, b - a + a^*)) \quad (2)$$

where V is the vocabulary and sim is a similarity measure. Specifically, they used the cosine similarity measure, defined as:

$$cos(u, v) = \frac{u \,.\, v}{||u|| \,.\, ||v||} \quad (3)$$

resulting in:

$$arg \max_{b^* \in V} (cos(b^*, b - a + a^*)) \quad (4)$$

Equation 4 has been referred to as CosAdd model.

While experimenting, Omer Levy (Levy et al., 2014) found that for an analogy question "London is to England as Baghdad is to - ?", using CosAdd model, they got *Mosul* - a large Iraqi city, instead of *Iraq* which is a country, as an answer. They were seeking for Iraq because of its similarity to England (both are countries), similarity to Baghdad (similar geography/culture) and dissimilarity to London (different geography/culture). While Iraq was much more similar to England than Mosul was (because both Iraq and England are countries), the sums were dominated by the geographic and cultural aspect of the analogy.

Hence to achieve better balancing among different aspects of similarity, they proposed a new model, where they moved from additive to multiplicative approach:

$$arg \max_{b^* \in V} \frac{cos(b^*, b) \,.\, cos(b^*, a^*)}{cos(b^*, a) + \epsilon} \quad (5)$$

$$(\epsilon = 0.001 \; to \; prevent \; division \; by \; zero)$$

This was equivalent to taking the logarithm of each term before summation, thus amplifying the differences between small quantities and reducing the differences between larger ones. This model has been referred to as CosMul model.

Even though our transformation operator can handle any sort of transformation, but if we are not able to detect the rule syntactically, we are not able to determine which transformation operator to use, and hence, we fall back on CosSum/CosMul. Like for the above mentioned examples, we will use transformation operator (if existing) for transformations like <reach, reached>, since we can find the rule syntactically, but for <go, went>, we can not, since we can not extract the corresponding rule itself - even if the matrix can handle such transitions.

If a transformation matrix exists for a transition rule, we apply the corresponding transformation matrix on the word embedding of the third word and search the whole vocabulary for the word with an embedding most similar to the transformed embedding (ignoring the third word itself). If the similarity of the resultant word's embedding with our transformed embedding is less than 0.68 (determined empirically) or the transformation matrix itself does not exist, we fall back on the CosSum/CosMul techniques.

Levy et. al. (2015) proposed the systems CosSum and CosMul in which they showed that tuning the hyperparameters has a significant impact on the performance. Hyperparameters are all the modifications and system design choices which are a part of the final algorithm.

Figure 1 gives an overview of how target word

embedding is generated using transformation operators and our backup models.

4 Result and Analysis

Model	CosSum	CosSum w/ M	CosMul	CosMul w/ M
SGNS-L	0.69	-	**0.729**	-
Glove-L	0.628	-	0.685	-
SG	0.269	0.554	0.282	0.566
GN	0.646	0.718	0.67	**0.733**
GN-SG Hybrid	0.674	0.835	0.698	**0.85**

Table 2: Scores on MSR word analogy test set.

Word1	Word2	Word3	Operator	Word4	Cosine
decides	decided	studies	<s , d>	studied	0.89
reach	reaches	go	<null , es>	goes	1.0
member	members	school	<null , s>	schools	0.88
ask	asks	reduce	<null , s>	reduces	0.91
resident	residents	rate	<null , s>	rates	0.86
get	gets	show	<null , s>	shows	0.83
higher	highest	stricter	<r , st>	strictest	1.0
wild	wilder	harsh	<null , er>	harsher	0.91

Table 3: Example results of transformation operators for regular transformations.

Word1	Word2	Word3	Operator	Word4	Cosine
joined	joins	became	<ed , s>	becomes	0.68
turned	turns	said	<ed , s>	says	0.74
learn	learned	build	<null , ed>	built	0.80
support	supported	see	<null , ed>	saw	0.72

Table 4: Example results of transformation operators for irregular transformations.

In table 2, GN denotes the scores of Google-News word embeddings on the test set. SGNS-L and Glove-L (Levy et al., 2015) denote the results of Skip-gram with negative sampling and Glove word embeddings respectively, both trained on large datasets. SG denotes the scores of our word2vec trained model (on 1B tokens). "w/ M" implies that we have used matrix arithmetic (along with CosSum/CosMul as backup) for word analogy answering questions. Our model uses "CosSum" and "CosMul" as backup transformation method in case a transformation operator (matrix) does not exist. We see that the results of GN+Matrix are better than the previously used models.

However, one thing we noticed was that the model trained on Google-News did not contain words with apostrophe sign(s) and 1000 out of 8000 words in MSR word analogy test set contained apostrophe sign(s). Also, we noticed that in

Word1	Word2	Word3	Operator	Word4	Cosine
reach	reached	go	<null , ed>	went	0.80
recognize	recognizes	be	<null , s>	is	0.70

Table 5: Example results of transformation operators for complete change of word form.

SG, the matrix approach was able to answer word analogy queries where words contained apostrophe sign(s), with an accuracy of 93.7% since it is a very common transformation - which resulted in well trained transformation matrix. So, we used SG as a backup for words which were not found in GN. The results of this hybrid model are denoted by GN-SG Hybrid. We see that this model performs considerably better than the existing state of the art system.

As we can see in table 3, our approach works really well for analogy questions where target word experiences regular transformation, i.e. the transformation type is simple addition/subtraction of suffix/prefix.

In table 4 and table 5 we observe that transformations are irregular transformations i.e there is slight change in word form while addition/subtraction of suffix/prefix or there is complete change in word form in the target word of our analogy question. This is an interesting observation, because even though our rule extraction (as explained above) is syntactic in nature, our method still learns and can apply transformation rules on words which undergo such irregular/complete transformations.

In operator "<null,s>", we see that our transformation matrix works pretty well irrespective of the form the word. For example, it works for "school-schools" and "reduce-reduces" which are noun and verb word pairs respectively. Our approach works by statistically creating global transformation operators and is agnostic in applying them (i.e. applied on a verb or a noun). Our transformation rules learn from both noun transitions and verb transitions and hence, even though we agree that linguistically there is a difference between noun and verb transitions, our approach performed better than previously existing systems

We also observed that in some cases, cosine similarity score is 1. This is mostly because "stricter-strictest" was used for training transformation matrix of "<r,st>" operator.

Although our cosine scores for irregular/complete transformations are not that high

with respect to scores for regular transformations, our system still performs at par or better than previous known systems. It is still able to predict words with high accuracy using its limited training corpora.

These observations can also help us analyze how certain complex transformations (irregular/complete) still behave similar to their regular counterpart computationally, as is apparent from our transformation matrix - which has learnt itself from rules that were extracted via all possible prefix and suffix substitutions from w1 to w2, and thus irregular/complete transformations would not be present in training our transformation matrix (where w1 and w2 belong to our vocabulary V - the size of our corpus).

We conclude that our matrix is able to harness morphological regularities present in word pairs used for training.

5 Future Work

The main application of this approach lies in its ability to generate representations for unseen/unreliable words on the go. If we encounter a word such as "preparedness" for which we do not have a representation or our representation is not reliable, we can identify any reliable form of the word, say "prepared" and apply <null,ness> operator on it, resulting in a representation for "preparedness". In a similar case, we can generate embeddings for words such as "unpreparedness" from "prepared" by sequentially applying <null,ness> and a prefix operator trained in a similar manner - <null,un>. Overall, this results in a much larger vocabulary than of the model initially being used.

We observe that for a transformation matrix to exist, we need enough word pairs (frequent enough to be included) to train a transformation matrix, and to cover majority of the transformation rules. Since many languages face problem of data scarcity, we face problem of "missing" transformation matrix for a transformation rule.

We will also explore if we can generate better word embeddings for words that are not highly frequent (hence less reliable), by applying transformation operators on their morphological variants, which are highly frequent and hence more reliable. For example, if for word pair "<walked,walking>", "walked" was not highly frequent in our corpora, we would not include it in training our transformation matrix for operator "<ed,ing>" because "walked" didn't have a well trained word embedding (being less frequent). But, if we have walk as highly frequent, and a transformation matrix for rule "<null,ed>", we can generate a better word embedding for "walked", and in turn use "<walked,walking>" for training our transformation matrix for rule "<ed,ing>".

In our current approach, for problem "If A is to B, then C is to ?", we do syntactic analysis on "A" and "B" to find out the transformation rule (and hence the transformation matrix) to be applied on "C" to find our "?". Rather than doing syntactic analysis, we will focus on finding out the correct transformation matrix by applying all transformation matrices on "A" and then analyzing output of each matrix. The one which will give closest result to "B" can be safely assumed to be the correct rule (transformation matrix), and hence will be applied on "C" to find "?". We will also analyze this approach's impact in terms of space and time complexities with respect to our current system. This will also enable us to find transformation matrix for word pairs which do not follow a syntactic transformation.

References

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225.

Omer Levy, Yoav Goldberg, and Israel Ramat-Gan. 2014. Linguistic regularities in sparse and explicit word representations. In *CoNLL*, pages 171–180.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. 2013b. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*.

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

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013d. Linguistic regularities in continuous space word representations. In *hlt-Naacl*, volume 13, pages 746–751.

Fabrizio Sebastiani. 2002. Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, 34(1):1–47.

Cyrus Shaoul. 2010. The westbury lab wikipedia corpus. *Edmonton, AB: University of Alberta*.

Radu Soricut and Franz Josef Och. 2015. Unsupervised morphology induction using word embeddings. In *HLT-NAACL*, pages 1627–1637.

Stefanie Tellex, Boris Katz, Jimmy Lin, Aaron Fernandes, and Gregory Marton. 2003. Quantitative evaluation of passage retrieval algorithms for question answering. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 41–47. ACM.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394. Association for Computational Linguistics.