Harnessing Morphological Regularities for Representation Learning for Low Resourced Languages

Thesis submitted in partial fulfillment
of the requirements for the degree of

MS by Research
in
Computer Science and Engineering

by

ARIHANT GUPTA
201202003
arihant.gupta@research.iiit.ac.in

International Institute of Information Technology
Hyderabad - 500 032, INDIA
January 2018

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled **"Harnessing Morphological Regularities for Representation Learning for Low Resourced Languages"** by Arihant Gupta, has been carried out under my supervision and is not submitted elsewhere for a degree.

6/1/18

Date

Adviser: Dr. Manish Shrivastava

To All and To None

Acknowledgments

Thank you, Dr Manish Shrivastava for being my advisor, and always reminding me that I can indeed complete my research. If it wasn't for you, I would still be stuck figuring out an escape plan, instead of working towards my masters. Your suggestions have shaped my thoughts, and given me direction when I needed it the most. Your lectures introduced me to NLP, and applications of it. I truly appreciate the freedom that you gave me as a student, allowing me to work on my own time. You let me explore all different options, and helped me settle down on one that worked the best for me. I am grateful to you for being extremely patient with me, and entertaining my pointless queries at times. I sincerely respect you for always having faith in me, even though I was irresponsible at times, and being so forgivable about it. Finally, thank you for being a friend, with whom I could discuss from complex linguistic tasks to stories about army and which battalion is the most fearsome. Having you as my advisor was an unforgettable experience which I will always remember.

I thank Prof. Dipti Misra Sharma, for introducing me to linguistics. You always encouraged me to go from the basics, and were a constant support, academically and non-academically. Thank you for helping me get a better insight into the problems I was working with by questioning me at each step.

Thank you, Sarfaraz, for being my partner in crime - even though you were mostly the one who led the team. If not for you, I do not think I would be writing this acknowledgment. You always had unique ideas, and you were a constant motivation for doing the research that I have done till now. It would not be wrong to say that you were my mentor in certain sense, and you taught me quite a lot. I will always be grateful for being there for me at each step.

Thank you, Arjit and Avijit, for being a constant support. Avijit, you have always guided me whenever I was stuck with a problem(linguistically), even though you had your corporate work to take care of. I can never forget your contribution to all the work I have done till now. Your dedication towards helping me out even after graduating, puts me in awe of you. Arjit, you have always been practical about things, and have always showed me that it can be done. Thank you for being present whenever I needed you to help me in understanding and figuring out scope of a problem. Also, thank you for keeping me sane throughout this journey.

Special thanks to Madan and Nazrul, without whom I wouldn't be completing my research in time. You guys helped me get out of my lazy attitude and get done with my work. Thanks to all my wingmates, who survived through my idiocy, and always supported and guided me. Thanks to all the members of

Language Technologies and Research Center, who always encouraged me to pursue my ideas and for being there to help me out.

Thank you IIIT-H, for giving me this wonderful experience.

Thank you Micky(brother), for always being there. Your random phone calls, and 3AM encouraging pep talks is something that made me believe, that maybe I can do this. You often gave me insights and discussed at length about what I was doing, which often gave me new ideas to work on, or on how to improve my work. Thank you for being there, always.

Finally, Mom and Dad, thank you for always believing in me. If it wasn't for your constant encouragement, I would still be struggling. You gave me all the love and joy, which kept me going, even when times got tough. You always supported and guided me, no matter how irrational or irritating I got. Thank you for making me who I am today. And mom, finally you can tell people that your son is not 12th pass but a graduate(:P).

Abstract

Morphology is the branch of linguistics that deals with words, their internal structure, how they are formed, and their relationship to other words in the same language. It involves analyzing the structure of words and parts of words, such as stems, root words, prefixes, and suffixes. It also looks at parts of speech, intonation and stress, and the ways context can change a word's pronunciation and meaning.

In most languages, if not all, many words can be related to other words by rules that collectively describe the grammar for that language. For example, English speakers recognize that the words dog and dogs are closely related, differentiated only by the plurality morpheme "-s", only found bound to nouns.

With recent advancements in computational linguistics, we can now learn one-hot vector representation for each word, also called word representations, from monolingual corpus of a language (training corpus). Word representations have been shown to contain syntactic as well as semantic (morphological) regularities. These word representations are being widely used to solve problems of various areas of natural language processing. These include but are not limited to dependency parsing, named entity recognition and parsing.

One major requirement for learning good word representations (word embeddings) is large enough corpus to train. Size of training corpus directly affects the corresponding quality of word representations our model learns. Many languages, even though widely spoken, suffer from being computationally resource poor, which results in relatively poorer trained word embeddings. On top of it, morphologically rich languages suffer from morphologically induced data sparsity, since there are cases, where one morphological form of a word is common but other is rare in the same training corpus.

Hence to learn better word representations for low resourced languages and to better exploit morphological regularities present in distributional word representations, we present a language independent, unsupervised method for building word embeddings using morphological expansion of text by exploiting morphological regularities present in distributed word representations. Our model handles the problem of data sparsity and yields improved word embeddings by relying on training word embeddings on artificially generated sentences. We evaluate our method using small sized training sets on eleven test sets for the word similarity task across seven languages. Further, for English, we evaluated the impacts of our approach using a large training set on three standard test sets. Our method improved results across all languages.

We also 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 with an accuracy of 85% which is 12% higher than the previous best known system.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

Most tasks in natural language processing involve looking at words, and could benefit from word representations that do not treat individual words as unique symbols, but instead reflect similarities and dissimilarities between them.

The distributional hypothesis of Harris [41] states that words in similar contexts have similar meanings. This hypothesis is the primary motivation for deriving word representations. Initial distributional semantic models also known as "word space" or "distributional similarity", dynamically built semantic representations in the form of high-dimensional vector spaces through a statistical analysis of the contexts in which words occur.

LSA is one of the classic methods for generating dense vectors. It analyzes relationships between a set of documents and the terms they contain by producing a set of concepts related to the documents and terms. LSA assumes that words that are close in meaning will occur in similar pieces of text (the distributional hypothesis). A matrix containing word counts per paragraph (rows represent unique words and columns represent each paragraph) is constructed from a large piece of text and SVD is used to reduce the number of rows while preserving the similarity structure among columns. Words are then compared by taking the dot product between the normalizations of the two vectors formed by any two rows. Values close to 1 represented very similar words while values close to 0 represented very dissimilar words.

Singular Value Decomposition (SVD) is a method for finding the most important dimensions of a data set, those dimensions along which the data varies the most. It can be applied to any rectangular matrix. SVD is part of a family of methods that can approximate an N-dimensional dataset using fewer dimensions, including Principle Components Analysis (PCA), Factor Analysis, and so on.

Many works also proposed representing words as dense vectors derived using various training methods inspired from neural-network language modeling. These representations, referred to as "neural embeddings" or "word embeddings", have been shown to perform well in a variety of NLP tasks [15].

Since, a word is known by the company it keeps i.e words in similar context have similar meanings (Harris [41]), we can also have one-hot vector representation for each word. We would expect our word representation to follow certain semantic rules, few of which could be:

- Words with similar meaning should be closer to each other.
- Direction from "walk" to "walking" should be similar to "run" to "running" since they follow the same transformation rule.
- It should follow vector arithmetic, i.e "King" - "Man" + "Woman" = "Queen".

Finally, Tomas Mikolov devised the training method which is both efficient to train and provides state-of-the-art results on various linguistic tasks [36], [37], [38]. The training method (as implemented in the word2vec software package) is a highly popular method for training word embeddings.

Word2vec can utilize either of two model architectures to produce a distributed representation of words: continuous bag-of-words (CBOW) or continuous skip-gram. In the continuous bag-of-words architecture, the model predicts the current word from a window of surrounding context words. The order of context words does not influence prediction (bag-of-words assumption). In the continuous skip-gram architecture, the model uses the current word to predict the surrounding window of context words. The skip-gram architecture weighs nearby context words more heavily than more distant context words. According to Mikolov, CBOW is faster while skip-gram is slower but does a better job. Hence, for all our experiments, we use skip-gram model.

1.2 Motivation and Contribution

Word representations learnt using either of the two model architectures - CBOW and Skipgram, have shown syntacto-semantic and morphological regularities. For eg: "boy" and "man" are more similar than "boy" and "animal". Similarly, direction from "go" to "going" is similar to "talk" to "talking" as they follow the same morphological rule. These representations are not different from LSA (with SVD) as has been proved by Levy and Goldberg [26] where they showed that SVD can achieve solutions that are at least as good as SGNS's (Skip-gram with Negative Sampling) solutions when dense low-dimensional vectors are preferred. In models using large corpora and a high number of dimensions, the skip-gram model yields the highest overall accuracy, and consistently produces the highest accuracy on semantic relationships, as well as yielding the highest syntactic accuracy in most cases.

The use of different model parameters and different corpus sizes can greatly affect the quality of a word2vec model. Accuracy can be improved in a number of ways, including the choice of model architecture (CBOW or Skip-Gram), increasing the training data set, increasing the number of vector dimensions, and increasing the window size of words considered by the algorithm. Each of these improvements comes with the cost of increased computational complexity and therefore increased model

generation time. But primary contributor to good representation learning is the size of the training corpus. These dense vector representations are very dependent on the size of training corpus since they are context dependant. Small training corpus means lack of context, which in turn means weaker learnt word representations.

Even if we have large training corpora, there are words which do not occur as frequently as others. In turn there are some words, which can be considered rare in the corpus because their frequency in the entire corpus is really small (generally < 100). Word representations of such words are not reliable because of their low frequency. There are also cases, where one morphological form of a word is common but other is rare in the same training corpus which leads to morphologically induced data sparsity for morphologically rich languages. One example could be that "walk" and "walking" are common in the training corpus, but "walked" is not. Even though "walk", "walking" and "walked" have similar context, we would still have an unreliable word representation learnt for "walked" (because of its low frequency).

With the advent of word representations, word similarity and word analogy tasks are becoming increasingly popular as an evaluation metric for the quality of the representations. They have acted as benchmarks for evaluating various word representation learning models. English rare-word word similarity dataset (En-RW) and MSR word analogy dataset are one of the most popular datasets for evaluating word representations. A word similarity dataset contains a pair of words, and each pair has a human given score regarding how similar those two words are. A word analogy dataset has a series of questions where each question is of form,"if A is to B, then C is to ?". Both these types of datasets help in evaluating syntacto-semantic and morphological regularities found in word representations.

Learning good word representations for Indian languages becomes all the more harder since they suffer from both morphological richness and lack of training data (and hence fall into category of low resourced languages). Even if there are word representation models trained for Indian languages, we do not have the resources to evaluate the reliability of the learnt word representations. There is a very evident lack of word similarity or word analogy datasets for representation learning for Indian languages.

Hence, primary motivation behind this thesis is to learn better word representations for low resourced languages and to better exploit morphological regularities present in distributional word representations given that all other parameters - vector dimension, window size etc are constant and well within our computational complexity space. In this thesis, we show how we developed a new architecture where we harness morphological regularities and synthesize artificial sentences, which act as training corpus for our word representation learning models. Our model showed significant improvement in quality of learnt word representations with respect to the ones that were learnt from initial training corpus, using the same word representation learning model. We go on to show how morphological richness of a language can be harnessed to boost the quality of word representations learnt. We also achieved results comparable to models trained using previous best known systems which were trained on much larger data sets. We evaluated our model across seven different languages with varied training corpus sizes via various word similarity datasets for these languages, hence showing that our architecture indeed improves on previous

best known systems and tackles the problem of data sparsity (small training corpus) with a significant improvement in quality of learnt word representations.

We also created a word similarity dataset for Hindi (which is fourth most spoken language in the world) - Hin-WS235 to facilitate evaluation of word representations learnt for Hindi. This will help in furthering research in various areas of NLP tasks for Hindi language. We also give detailed explanation of how this dataset was created, to aid development of similar datasets for different languages in future.

We also developed a new architecture for exploiting morphological regularities present in distributional word representations in form of global transformation matrix operators. We show that these operators performs significantly better standard vector arithmetic in understanding various transformation rules and their corresponding transformations. For eg, "walking" is present continuous transformation of "walk". We also show that our model works better than previous best known systems by evaluating it on MSR word analogy dataset.

We then present our contributions in the development of an architecture where we show that models from vector space of a language can be mapped to vector space of another language. For eg, we show how a model trained on Hindi can be mapped to work for Urdu (which has relatively smaller training corpus with respect to Hindi) via our vector mapping architecture. We go on to show how this architecture significantly boosts the quality of word representations, and that this boost is more significant for similar language pairs as compared to other language pairs. We also show, how this approach can be used to better representation learning for a language, by combining multiple models (learnt on different training corpuses) of the same language into one. We also realize that Indian languages severely lack in evaluation metrics available for all the representation learning models that have been built. And hence, we conclude this thesis by showing our contribution in development of word similarity datasets for six Indian languages, and we also provide baseline scores using current state of the art word representation learning models.

1.3 Thesis Structure

This thesis is structured as follows:

- In chapter 2, we give an introduction to the existing models that have been used along with introduction to related work that has been done in this area.

- In chapter 3, we present our approach where we do unsupervised artificial text generation to improve word embeddings. We go on to describe our approach, later run our model on 11 different word similarity datasets. We also compare our results with previous state of the art models, and standard skip-gram word2vec model.

- In chapter 4, we present an approach for exploiting morphological regularities using global transformation operators. We evaluate our approach using MSR word similarity dataset.

- In chapter 5, we present our contribution in developing a technique for porting vector spaces from one language space to another, and developing six another word similarity datasets along with baseline systems for six Indian languages. We also describe how we created our Hindi word similarity dataset - Hin-WS235, which contains 235 word similarity word pairs. We present guidelines and steps followed while creating this dataset. This dataset was used for evaluating our approach on Hindi language in chapter 3.
- We conclude our work in chapter 6 along with possbile future work.

Chapter 2

Related Work

Mikolov et. al. [36] introduced the Skip-gram (SG) and the continuous bag-of-words (CBOW) models trained on untagged text. Both of them followed a single layered feedforward architecture. CBOW's training algorithm relied on predicting the current word based on context and SG tried to predict the context using the current word. Mikolov et. al. [37] also showed the semantic and syntactic regularities found in these embeddings.

Pennington et. al. [13] introduced the GloVe (Global Vectors for Word Representations) model in which they analyzed the properties responsible for the morphological regularities in the earlier models. They combined the effects of local context window and global matrix factorization methods resulting in a global log-bilinear regression model. The proposed model out-performed SG and CBOW on standard test sets.

Word Representations, as learned by the above methods, deal with word as the basic unit and do not exploit the morphological relations present between words. However, these relations are present in the embeddings as regularities in the vector space [37]. These are specially useful for words which are unseen or have a low frequency in the training set and are not trained well. Luong et. al. [21] and Botha and Blunsom [12] used Morfessor [20] for word segmentation and used a combination of morpheme and word level models. Both of these approaches handled rare and unseen words using their basic morpheme units. Luong et. al. [21] used morphological recursive neural networks (RNNs) for constructing representation for a word using its morphemes. Botha and Blunsom [12] used log-bilinear models for building and combining representations of morphemes for constructing representations of rare or unseen words. Luong et. al. [21] also introduced the Stanford Rare-Word Dataset which contains a large number of rare and morphologically complex words.

In contrast to Luong et. al. [21] and Botha and Blunsom [12] where an external morphological analyzer was used, Soricut and Och [29] induced morphological transformations in an unsupervised manner using SG (Skip-gram) word embeddings. These morphological transformations were represented as word pairs in the same embeddings space and they used simple vector arithmetic to calculate vectors for rare and unseen words. For example, for building the embedding for "nationalism", they evaluated the expression "functionalism - function + nation".

2.1 Previous Models

Many current NLP systems and techniques treat words as atomic units - there is no notion of similarity between words, as these are represented as indices in a vocabulary. This choice has several good reasons - simplicity, robustness and the observation that simple models trained on huge amounts of data outperform complex systems trained on fewer data.

With the progress of machine learning techniques in recent years, it has become possible to train more complex models on a much larger data set, and they typically outperform the simple models. The most successful concept is to use distributed representations of words. For example, neural network based language models significantly outperform N-gram models.

Neural networks are typically organized in layers. Layers are made up of a number of interconnected 'nodes' which contain an 'activation function'. Patterns are presented to the network via the 'input layer', which communicates to one or more 'hidden layers' where the actual processing is done via a system of weighted 'connections'. Most NNs contain some form of 'learning rule' which modifies the weights of the connections according to the input patterns that it is presented with.

Typically, a neural network is initially trained or fed large amounts of data. Training consists of providing input and telling the network what the output should be. For example, to build a network to identify the faces of actors, initial training might be a series of pictures of actors, non-actors, masks, statuary, animal faces and so on. Each input is accompanied by the matching identification, such as actors' names, "not actor" or "not human" information. Providing the answers allows the model to adjust its internal weightings to learn how to do its job better.

In defining the rules and making determinations - that is, each node decides what to send on to the next tier based on its own inputs from the previous tier - neural networks use several principles. These include gradient-based training, fuzzy logic, genetic algorithms and Bayesian methods. They may be given some basic rules about object relationships in the space being modeled. We can give our Neural Network some rules to start with, which might be specific to the domain of the problem. Preloading rules can make training faster and make the model more powerful sooner. But it also builds in assumptions about the nature of the problem space, which may prove to be either irrelevant and unhelpful or incorrect and counterproductive, making the decision about what, if any, rules to build in very important.

2.1.1 NNLM

NNLM is a probabilistic feedforward neural network language model [3]. It consists of input, projection, hidden and output layers. At the input layer, N previous words are encoded using 1-of-V coding, where V is the size of the vocabulary. The input layer is then projected to a projection layer P that has dimensionality $N * D$, using a shared projection matrix. As only N inputs are active at any given time, the composition of the projection layer is a relatively cheap operation. The NNLM architecture becomes complex for computation between the projection and the hidden layer, as values in the projection layer are dense. For a common choice of N = 10, the size of the projection layer (P) might be 500 to

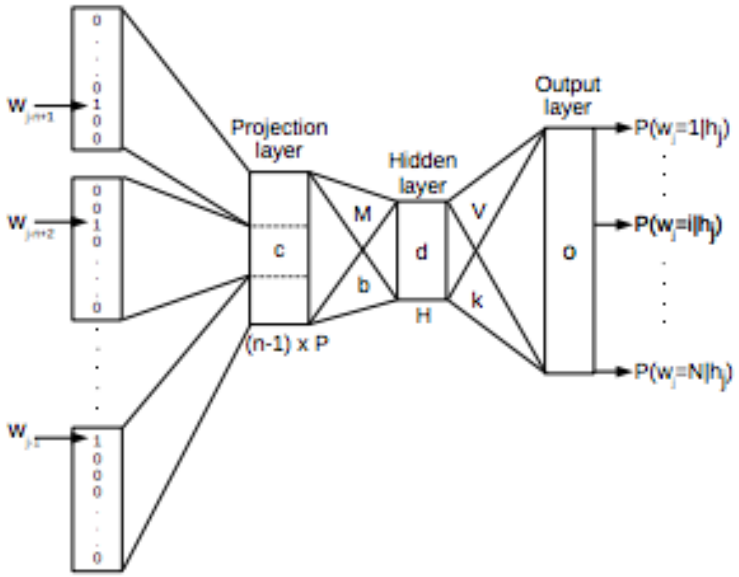

Figure 2.1 NNLM Architecture

2000, while the hidden layer size H is typically 500 to 1000 units. Moreover, the hidden layer is used to compute probability distribution over all the words in the vocabulary, resulting in an output layer with dimensionality V. Thus, the computational complexity per each training example is

$$Q = N * D + N * D * H + H * V, \tag{2.1}$$

where the dominating term is $H*V$. However, several practical solutions were proposed for avoiding it one of them being using hierarchical versions of the softmax. With binary tree representations of the vocabulary, the number of output units that need to be evaluated can go down to around $\log_2(V)$. Thus, most of the complexity is caused by the term $N * D * H$.

While this is not crucial speedup for neural network LMs as the computational bottleneck is in the $N * D * H$ term, Mikolov proposed architectures that do not have hidden layers and thus depend heavily on the efficiency of the softmax normalization.

2.1.2 Recurrent Neural Net Language Model (RNNLM)

Recurrent neural network based language model has been proposed to overcome certain limitations of the feedforward NNLM, such as the need to specify the context length (the order of the model N), and because theoretically RNNs can efficiently represent more complex patterns than the shallow neural networks. The RNN model does not have a projection layer; only input, hidden and output layer. What is special for this type of model is the recurrent matrix that connects hidden layer to itself, using time-delayed connections. This allows the recurrent model to form some kind of short term memory, as

information from the past can be represented by the hidden layer state that gets updated based on the current input and the state of the hidden layer in the previous time step.

The complexity per training example of the RNN model is

$$Q = H * H + H * V, \tag{2.2}$$

where the word representations D have the same dimensionality as the hidden layer H. Again, the term $H * V$ can be efficiently reduced to $H * \log_2(V)$ by using hierarchical softmax. Most of the complexity then comes from $H * H$.

2.2 New Log Linear Models

The main observation from the previous section is that most of the complexity is caused by the non-linear hidden layer in the model. While this is what makes neural networks so attractive, Tomas Mikolov decided to explore simpler models that might not be able to represent the data as precisely as neural networks, but can possibly be trained on much more data efficiently.

Finally, Mikolov devised the training method which is both efficient to train and provides state-of-the-art results on various linguistic tasks [36], [37], [38]. The training method (as implemented in the word2vec software package) is a highly popular method for training word embeddings.

Word2vec can utilize either of two model architectures to produce a distributed representation of words: continuous bag-of-words (CBOW) or continuous skip-gram (SG).

2.2.1 Continuous Bag-of-Words Model

This architecture is similar to the feedforward NNLM, where the non-linear hidden layer is removed and the projection layer is shared for all words (not just the projection matrix); thus, all words get projected into the same position (their vectors are averaged). This architecture is called a bag-of-words model as the order of words in the history does not influence the projection. Furthermore, Mikolov et. al. also used words from the future; they obtained the best performance by building a log-linear classifier with four future and four history words at the input, where the training criterion is to correctly classify the current (middle) word.

Training complexity is then

$$Q = N * D + D * \log_2(V) \tag{2.3}$$

Unlike standard bag-of-words model, it uses continuous distributed representation of the context. Note that the weight matrix between the input and the projection layer is shared for all word positions in the same way as in the NNLM.

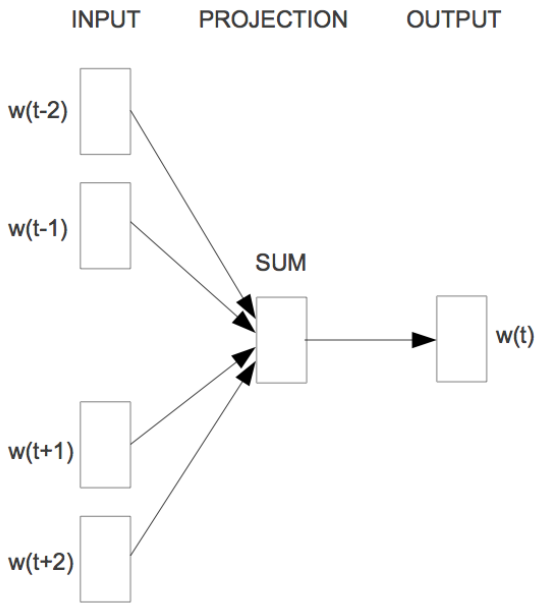

Figure 2.2 CBOW Architecture

2.2.2 Continuous Skip-gram Model

This architecture is similar to CBOW, but instead of predicting the current word based on the context, it tries to maximize classification of a word based on another word in the same sentence. More precisely, each current word is used as an input to a log-linear classifier with continuous projection layer, and predict words within a certain range before and after the current word. Mikolov found that increasing the range improves the quality of the resulting word vectors, but it also increases the computational complexity. Since the more distant words are usually less related to the current word than those close to it, he gave less weight to the distant words by sampling less from those words in his training examples.

The training complexity of this architecture is proportional to

$$Q = C * (D + D * \log_2(V)) \tag{2.4}$$

where C is the maximum distance of the words. For example, for C = 5, for each training word, the model will select randomly a number R in the range $< 1; C >$, and then use R words from history and R words from the future of the current word as correct labels. This will require the model to do $R * 2$ word classifications, with the current word as input, and each of the $R + R$ words as output.

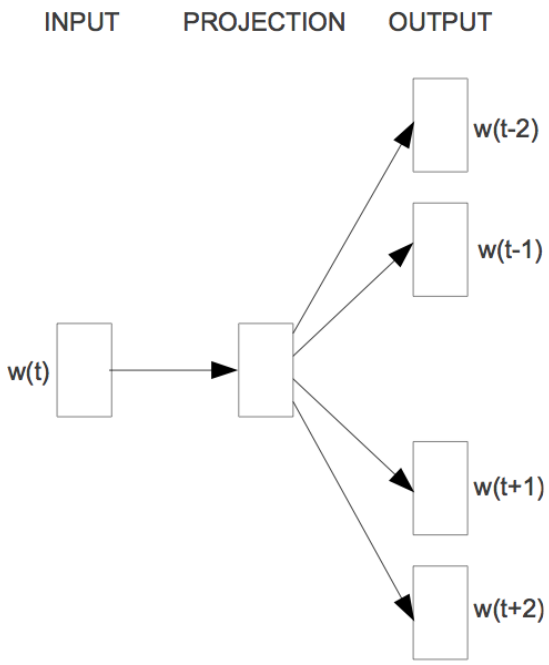

Figure 2.3 Skip-gram Architecture

Chapter 3

Unsupervised Morphological Expansion of Small Datasets for Improving Word Embeddings

3.1 Introduction

Word representations have been shown to contain syntactic as well as semantic regularities [37]. Such regularities also extend to morphological relations (vector of "ran" is close to the vector resulting from the expression "walked - walk + run").

The basis of our approach lies in morphological treatment of text before training word embeddings. In this chapter, we present a method for learning word representations which, for training, take morphological regularities into consideration by generating artificial sentences. These sentences contain automatically learned morphological variants of words in the corpus. This harnesses morphology to reduce data sparsity by improving the training of low frequency words with the help of their more common morphological variants (as word embeddings train better on high frequency words).

We show that our method performs well on small datasets of seven languages with significant increase across all languages except Arabic (discussed in section 3.7). We further analyze the impact of our approach on a large training set of English. Our evaluations show that when being applied on large quantities of training data, this method is comparable to models trained on even much larger datasets.

3.2 Datasets

For all the models trained in this chapter, we have used the Skip-gram [36] algorithm. The dimensionality has been fixed at 500 with a minimum count of 5 along with negative sampling. These parameters are identical to the ones used by Soricut and Och [29] for the purpose of extracting morphological transformations (Though, they used their own implementation of Word2Vec). Data is pre-processed to replace digits and special characters to avoid sparsity.

As training set for English, we use the Wikipedia data [6]. Soricut and Och [29] and Luong et. al. [21] had used the same training corpus for their models. The cleaned corpus contains about 1

billion tokens. For German and French, we use News Crawl (Articles from 2010) released as a part of ACL 2014 Ninth Workshop on Statistical Machine Translation. For Arabic, Persian and Spanish, we used Wikipedia Monolingual Corpora(2014) which is licensed under "Creative Commons Attribution-ShareAlike 4.0 International Public License". For Hindi, we used the Hindi corpus created by Bojar, Ondrej, et al. [1].

We use standard word-similarity datasets for testing. For English, we use Stanford English Rare-Word (RW) dataset [21], the WS353 [17] and the RG65 dataset [11]. The Stanford Rare-Word dataset contains comparatively more rare words and morphological complexity than other datasets and is central to our experiments. For German, we use the Gur350 and ZG222 datasets [39] and the German RG65 dataset. For French we use the French RG65 [5] dataset. For Spanish and Persian, we use Spanish-RG65 and Persian-RG65 test data-sets [14].

For Hindi (having fourth highest number of native speakers in the world) we are releasing a word similarity (Hin-WS235) dataset containing 235 word pairs. This data-set was created by manually translating and re-annotating the English WS353 [17] dataset. This dataset is crucial since its used for direct evaluation of word embeddings. This dataset will be helpful for future work on Hindi.

For rest of the chapter, we have calculated the Spearman ρ (multiplied by 100) between human assigned similarity and cosine similarity of our word embeddings for the word-pairs.

3.3 Morphological Expansion

Since we are generating new sentences using morphology, morphological sets of words that should be replaced have to be constructed. A morphological set of a word contains its "first cousins". First cousins are pairs of words which are morphological forms of each other, are semantically similar and also one can be reached from the other by a transformation involving addition/removal/replacement either suffix or prefix. For example, "nation" and "national" are first cousins (addition), "nationalism" and "national" are first cousins (removal), "nations" and "national" are first cousins (replacement), but "nation" and "nationalism" are second cousins as they involve two transformation operations. We go about doing this by extracting morphological transformations in a manner similar to that of Soricut and Och [29] using Skip-Gram word embeddings trained on the text.

Soricut and Och considered two main transformation types, namely prefix and suffix substitutions. The following steps were applied to monolingual training data over a finite vocabulary V:

- Extract candidate prefix/suffix rules from V.
- Train embedding space $E^n \subset C^n$ for all words in V.
- Evaluate quality of candidate rules in E^n.
- Generate lexicalized morphological transformations.

Regularity	Set of Stems	Regularity Type
ed	reduc, unannounc, walk, ...	Suffix
ly	on, unstab, respectab, ...	Suffix
un	clear, dress, derstand, ...	Prefix
dis	assemble, close, connect, ...	Prefix

Table 3.1 Examples of Regularities Sets

For extracting rules, starting from $(w1, w2) \in V^2$, the algorithm extracts all possible prefix and suffix substitutions from w1 to w2. At this stage, the candidate rules set contains both rules that reflect true morphology phenomena but also rules that simply reflect surface-level coincidences. For eg, 'stops' to 'stop', 'moves' to 'move' is a valid transition rule where suffix 's' is removed, but 'scope' to 'cope' is not - i.e removal of prefix 's'.

For training, using a large monolingual corpus, they trained a word-embedding space E^n of dimensionality n for all words in V using the SkipGram model (explained in previous sections).

For evaluation, each proposed rule is applied on word w1, where w1 belongs to our vocabulary V, to get target word w2, and applied an evaluation function on w2 to remove rules that are non-meaning-preserving or invalid.

We have followed the same approach with minor changes to reduce the overall complexity of the approach while still maintaining its authenticity. For example, we use TRIE based approach to extract morphological rules, which is much faster complexity-wise with respect to Soricut's approach, which takes $O(V^2)$ time. Our approach has been explained in detail in later sections.

All the thresholds mentioned in this chapter have been decided after empirical fine tuning. Even though our experiments were computationally optimized, time and space complexities also played a part in deciding our thresholds.

In order to learn initial representations of the words, we train word embeddings (word2vec) using the parameters described above on the training set. This model is referred to as SG (Skip-gram).

3.3.1 Morphological Transformations

Morphological transformations are word pairs representing morphological regularities present at large in the corpus. Following Soricut and Och's [29] methodology, we extract representative candidate word pairs which exhibit same morphological behavior as at least ten other word pairs in the corpus. These candidates allow us to learn rules of morphological transformations which may not be linguistically perfect but capture orthographic regularities. We refer to the final morphological transformations extracted as "transformation rules".

For extracting these transformation rules, we follow these steps:

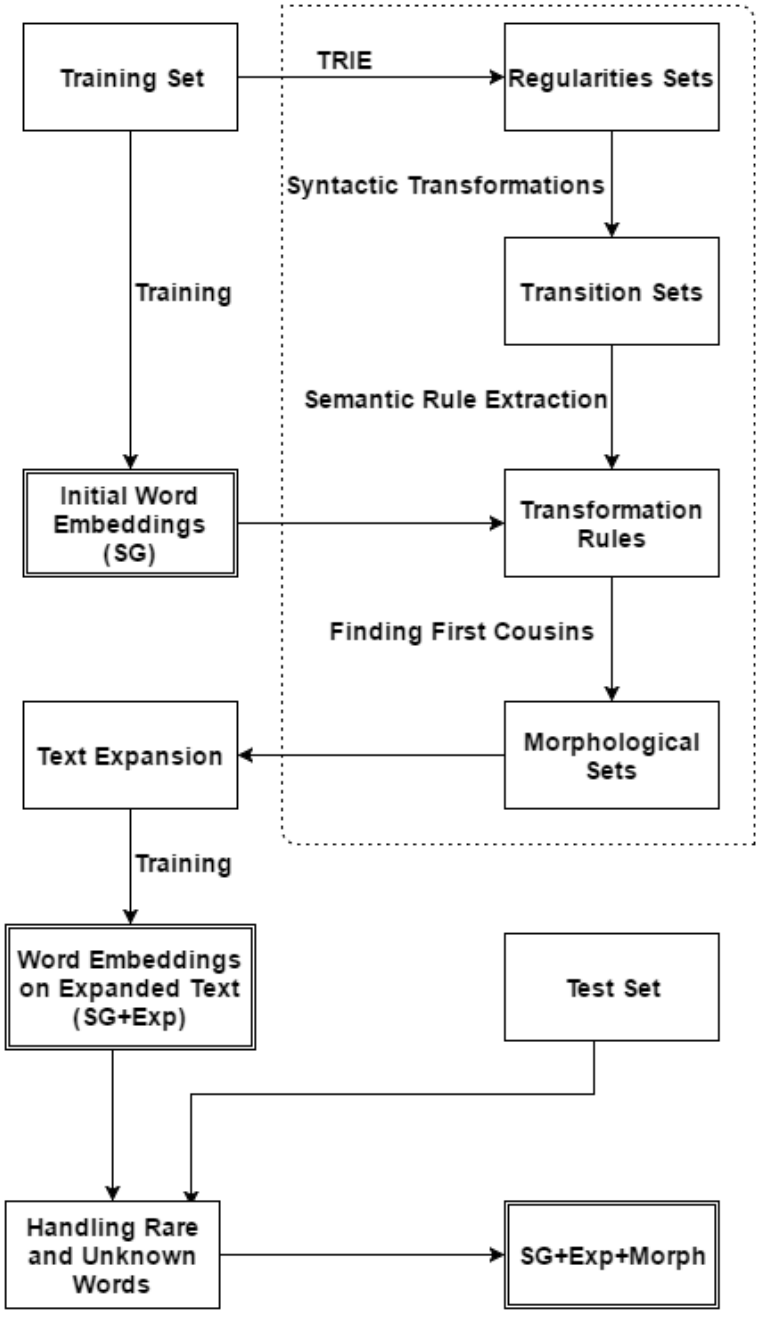

Figure 3.1 System Workflow

Transition	Set of Words	Regularity
<null, ed>	succeed, seem,..	Suffix
<null, ing>	read, poison,..	Suffix
<ed, ing>	viewed, documented,..	Suffix
<able,null>	sustainable, reasonable,..	Suffix
<null, un>	acceptably, accounted,..	Prefix

Table 3.2 Examples of Transition Sets

- Extract **Regularities** - Regularities sets (see Table 3.1) are the set of stems associated with each candidate prefix/suffix.

  Regularities Sets: We use a TRIE for the extraction of candidate suffixes/prefixes. We insert all the words of the training set into the TRIE and then extract candidate prefixes. A candidate prefix is one which has more than 10 children in the TRIE. For candidate suffixes, we follow the same procedure with the only change being that the TRIE is constructed with all the words reversed. The output of this step is a candidate prefix/suffix and a set of all its stems. We call this a regularities set. As can be surmised, a number of these sets may not be linguistically correct. Also, some of these may not be true regularities and might just be data artifacts.

- Construct **Transition Sets** using these morpheme sets - Transitions sets (see Table 3.2) are the set of word pairs which follow the same syntactic transformation.

  Transition Sets: There are two types of transition sets. One of them involving a prefix/suffix going to null and the other is a transition in which we have to both add and delete characters to get one word from the other. We call them null transitions and cross transitions, respectively. For evaluating null transitions, we find the intersection of each regularities set with the vocabulary of the training set. For extracting cross transitions, we find the set intersection between regularities sets. Since this leads to a large number of combinations, we evaluate only those cross sets in which both the regularities sets are large (a frequency greater than or equal to 500 for small training sets and 30000 for large training sets were used for our experiments). For both the null and cross transitions, we evaluate only those sets whose sizes are greater than 10. Also, we down-sample the transition sets thus generated to 1000 for time optimization.

- Extract **Transformation Rules** from these transition sets - Transformation rules (see Table 3.3) capture word pairs which are morphologically similar (belong to same transition sets) apart from being syntactically similar according to initial word embeddings learned in SG.

  Transformation Rules: For a word pair to be considered as a part of the transformation rules, both of its words should be frequent; a frequency threshold of 500 for small training sets and 1000 for large training sets. This ensures that both the words of the pair are trained well.

Word Pair	Regularity Type
<side, beside>	Prefix
<eighty, eight>	Suffix
<transmitted, transmit>	Suffix
<cinematic, cinema>	Suffix
<reminder, remind>	Suffix
<after, afterward>	Suffix

Table 3.3 Examples of Transformation Rules

For every transition set, we find out which pair represents the maximum number of word pairs (the cosine similarity of their vector differences is above a threshold: 0.15 for prefix rules and 0.25 for suffix rules). If the count is greater than 10, we make it a transformation rule and remove it along with the other word pairs it represents from the transition set (because we want a limited number of morphological transformations representing similar morphological regularities). We then recursively follow the said technique for the transition set until we stop getting transformation rules from it. This approach results in multiple transformation rules from the same transition set. We see that it is needed because different forms of same transition exist (walk-walks, invention-inventions, object-objects - both verbs and nouns in this case).

We observed that there were more impurities in words of smaller length. They were not removed at the time of extraction of transformation rules because these words are large in number. In this step, those rules in which either or both of the words have a length of less than or equal to three are eliminated. We also know that words which are morphological forms of each other have similar word embeddings and our aim is to create new sentences similar to the original sentence, so, we used a cosine similarity threshold of 0.1 for filtering the transformation rules.

3.3.2 Morphological Sets

A morphological set (see Table 5.1) of a word is a collection of all morphological forms we detect. For words with frequency more than a threshold (100 for small and 1000 for large training sets were used in our experiments) and length greater than three, we attempt to construct morphological sets. We apply all the transformation rules on the word. For every resultant word with frequency more than or equal to 5 (hence has a trained word embedding in SG), we check if it is similar to the original word (cosine similarity greater than 0.15) and add it to the set. Now, we proceed to evaluating the morphological set. There may be many issues with the morphological set, homo-morphs being one. A homo-morph is a word which has more than one meaning. An example of a homo-morph is the word “state” with its morph set {stately, state-hood, stated, re-stated}. We observe that using cosine similarity, we are

Word	Morphological Set
comically	comical, comic
hanging	hang, overhanging, rehanging, hangings
woody	non-woody
localized	unlocalized, localize, non-localized, local
cityhood	city
trawling	trawl

Table 3.4 Examples of Morphological sets of some words

able to eliminate such sets as they can be broken up into more than one subsets ({stately, state-hood}, {stated,re-stated}) whose words are not similar to each other. We select another random word from the set and assure that its cosine similarity with every word in the morph set is more than 0.15. This technique also removes morphologically unrelated words sharing a common prefix/suffix that may have crept in. For example, the word "define" has {definition, defined, fine} as its morph set. We found that large morph sets usually contained more impurities. As we had only applied a single level of transformation for extracting these sets, we discard those sets whose size is greater than 6.

3.3.3 Text Expansion

Now that we have an approximation of word similarities based on morphology, we would like to use this information to learn better word embeddings. We know that rare words' embeddings are not learned well due to lack of context. Based on words' morphological similarities we would want to overcome this shortfall by training embeddings on corpus augmented with artificially generated sentences. We want to generate new sentences using the morphological sets that we have generated. However, a brute force approach results in combinatorial explosion with a single sentence expanding to thousands of sentences. Therefore, we use a random function to choose which sentences to generate to prevent bias towards any specific morphological form.

In this step, for each sentence in the training set, we attempt to create a maximum of one new sentence using a random function and morphological sets. For each word in the sentence, if that word has a morphological set, we choose with 50 percent probability to replace that word with a random word from its morph set. If the sentence thus generated is different from the original sentence, it becomes a part of our new training set upon which we train word embeddings in the next step.

In model SG + 2-Exp, we try to generate a maximum of two sentences from each sentence and a maximum of three new sentences in 3-Exp. Shown below is a sample sentence of the original text.

"*Maintenance is necessary for a software product to be successful* "

Now in expanded text (Exp).

"*Maintenance is necessary for a software product to be successful*
High-Maintenance is necessary for a software production to be successful"

In expanded text (2-Exp).

"*Maintenance is necessary for a software product to be successful*
Maintenance is unnecessary for a software product to be successful
Maintenance is necessary for a software production to be unsuccessful"

In expanded text (3-Exp).

"*Maintenance is necessary for a software product to be successful*
Self-Maintenance is unnecessary for a software product to be successful
Maintenance is unnecessary for a software production to be unsuccessful
High-Maintenance is necessary for a software production to be successful"

We observe "necessary" and "successful" were replaced by their negation forms. This happened because their word vectors are similar which is due to the fact that they are often used interchangeably in natural language.

As you can also see from the above example, a lot of sentences generated were incorrect. But one thing to note is that even though these sentences violate global real world knowledge, and are counter intuitive, semantic context still exists. We still keep these sentences and use them as our training samples because our initial word embeddings were poorly trained due to lack of data, and these grammatically incorrect sentences do help in improving corresponding word embeddings since their main purpose is to provide more training samples, or artificial data for word2vec to train on.

3.3.4 Word Embeddings on Expanded Text (SG+Exp)

We train word embeddings on the expanded text to train a new model. We will call this model SG + Exp (Skip-gram + Expansion). For exploring the effects of this method, we also train models SG + 2-Exp and SG + 3-Exp on the large dataset of English. For all the models tested, we have initialized vectors of unseen words with zero vectors.

In table 3.5, we observe that for languages like Arabic, which show inherent complex nature, our method was unable to perform as it did with respect to other languages. This can be accounted to the method being unable to reconcile infixes as opposed to suffixes and prefixes resulting in incorrect morphological sets. Hence future work can also be directed towards generating transformations which involve changes inside the word (apart from just changes to starting and endings) for improving representation learning for languages like Arabic.

On the other hand, other languages showed significant improvement which goes on to show the morphological induced data sparsity present in many languages. Our approach is able to reduce this data sparsity, at least for languages which have suffix/prefix based morphology.

Language	Size	SG	OOV	SG+EXP	OOV
Hindi	34M	50.3	0	56.8	0
Arabic	34M	46.17	9	46.12	6
Persian	43M	20.5	15	22.7	15
Spanish	44M	61	4	73.1	0
DE Gur	44M	38.8	54	47.5	32
DE RG	44M	14.7	1	28.2	0
DE ZG	44M	22	74	27.8	55
French	39M	48.2	3	61.4	2
EN WS	53M	71.6	0	74.1	0
EN RG	53M	64.5	0	71.3	0
EN RW	53M	18.7	719	22	255

Table 3.5 Small training samples (Spearman ρ). OOV represents Out of Vocabulary words encountered while testing.

Language	Size	SG	OOV	SO [29]	Size (SO)
Hindi	0.75b	61	0	-	-
Arabic	63M	48.3	4	43.1	0.45b
Persian	59M	22.1	15	-	-
Spanish	0.45b	82.5	0	47.3	0.56b
DE Gur	0.5b	57.1	23	64.1	1.2b
DE RG	0.5b	67.1	1	-	-
DE ZG	0.5b	28.1	38	21.5	1.2b
French	0.2b	64.6	1	67.3	1.5b
EN WS	1b	74.4	0	71.2	1.1b
EN RG	1b	77.9	0	75.1	1.1b
EN RW	1b	42.1	66	41.8	1.1b

Table 3.6 Large training samples (Spearman ρ). SO[29] and Size (SO) represents scores of [29] and size of datasets used respectively. These scores act as reference for our scores on small training samples.

System	RW	WS	RG
SG	42.1	74.4	77.9
SG + Exp	45.6	73.8	80.4
SG + 2-Exp	45.9	72.9	80.3
SG + 3-Exp	45.3	71.3	78.3

Table 3.7 Trained on 1B tokens(English) - Comparison between different degrees of expansion (Spearman ρ)

Model	Sim	Closest to "censorships"
SG	0.43	POVs, normies
SG + Exp	0.87	censorship, self-censorship

Table 3.8 Comparison between SG and SG + Exp regarding word pair <censorship,censorships>

Table 3.7 interestingly shows how effective our approach is when evaluated against RW, WS and RG datasets. As more artificial sentences are introduced into our training corpora, quality of word representations start getting deteriorated which is apparent as we can see that SG+3-Exp performs worst out of all expansion models. We can see that SG + 2-Exp performs better than all others for RW since RW dataset contains rare words - words whose frequency is on the lower spectrum with respect to other words. In general SG + Exp performs the best out of all, since amount of artificial sentences introduced into the training corpora do not overpower our original training corpus.

Table 3.8 gives an insight into the effectiveness of the proposed method. We see that the words "censorship" and its rarer morphological variant "censorships" have very low similarity score and the similar words for "censorships" are also not very informative. By training new embeddings on the expanded corpus we find that not only does the similarity score increase but the other morphological variants also come closer to the word.

3.4 Handling Rare and Unseen Words

Since we assume rare words – frequency lower than a threshold, have unreliable word embeddings, in this section, we explain how we build word embeddings of rare and unseen words at the time of evaluation using the transformation rules. Our approach relies on the regularities illustrated by Mikolov et. al. [37] which were used successfully by Soricut and Och [29].

System	RW	WS	RG
SG + Morph	45.1	74.4	77.9
SG + Exp + Morph	47.9	73.8	80.4
SG + 2-Exp + Morph	47.6	72.9	80.3
SG + 3-Exp + Morph	46.8	71.3	78.3

Table 3.9 Tranined on 1B tokens(English) - Comparison between different systems after Morph step (Spearman ρ). Note that the scores on WS and RG are unchanged because the frequencies of all the words are above the threshold

In our experiments, we classify a word as rare if its frequency is less than 20. For building word embeddings for such words, we first find a reliable base word using Algorithm 3.4, from which rare word can be generated after successive application of transformation rule vectors.

Algorithm 1 Find Reliable Base Word

```
1:  procedure EXPLORE LEVEL(curWords)
2:      Words ← Initialize with empty list
3:      for each word w ∈ curWords do
4:          for each rule r ∈ Rules do
5:              word ← apply rule r on w
6:              Words ← append word
7:      return Words

8:  procedure FIND RELIABLE BASE(rareWord)
9:      word ← rareWord
10:     for level l ∈ [1, 3] do
11:         Reliable ← l*50
12:         words ← ExploreLevel(words)
13:         FreqWord ← MostFrequent(words)
14:         if Freq[FreqWord] > Reliable then
15:             return FreqWord
```

Our algorithm uses level order search and returns us the most reliable base word (if any) along with the transformation rules required to generate our rare word from the base word.

We increase our reliability threshold (Reliable) as the search level increases. This is done to because application of multiple transformation rules generally results in more errors. If even after searching for three levels we do not get our embedding, we use the embedding of the word itself if its present in the model (even though its probably poorly trained). In case the word is not present in the model, we try the above mentioned technique of level order search after capitalisation, followed by the lower case form of the word.

If we still do not have any embedding of the word, we recursively remove characters from the beginning and end of the word and generate all words with a frequency greater than 50 and a length greater than 3 (as the probability of error increases with small lengths) . We use the embedding of the word with the maximum length and use frequency in case of a tie. The same technique is applied on after capitalisation and then the lowercase version of the word in case we do not get any embedding.

If we fail to get any embedding, we assign it a zero vector. This step is referred to as "Morph". The system SG + Morph indicates that we have handled rare and unknown words using transformation rules when evaluating the model SG.

System	**Size**	**RW**	**WS**	**RG**
LSM13 [21]	1B	34.4	64.6	65.5
Glove [13]	6B	38.1	65.8	77.8
Glove [13]	42B	47.8	**75.9**	**82.9**
SO15 [29] w/o M	1.1B	35.8	71.2	75.1
SO15 [29] w/ M	1.1B	41.8	71.2	75.1
LTM16 [18]	0.3B	47.1	-	-
SG	1B	42.1	74.4	77.9
SG + Morph	1B	45.1	74.4	77.9
SG + Exp	1B	45.6	73.8	80.4
SG + Exp + Morph	1B	**47.9**	73.8	80.4

Table 3.10 Comparison with previously proposed methods (Spearman ρ)

3.5 Result and Analysis

The method seems to perform well on small datasets with an increase in accuracy on all datasets across all languages - English, German, French, Spanish, Persian and Hindi. Morphologically richer languages show more increase with respect to others. (see Table 3.5).

We see that the embeddings trained on a large dataset also tackles the data sparsity present in them (in the form of low frequency words). This is seen by the increase in the accuracies of RW and RG datasets which contain more low frequency words.

After analyzing these three datasets, we compute their rarity which is the average of the frequencies of the rarer word of the pair. The order of rarity of the datasets is RW(2253) $<$ RG(6505) $<$ WS(34152). The order helps in explaining the observation in terms of better word embeddings for low frequency words. For RW dataset, accuracy increases on expansion and the highest accuracy is found on SG + 2-Exp; for RG dataset, accuracy increases on expansion and the highest is found on SG + Exp. While for the WS dataset, accuracy decreases with any expansion.

We observe that the accuracy of SG + Exp + Morph $>$ SG + 2-Exp + Morph on RW dataset even though the accuracy of SG + 2-Exp $>$ SG + Exp (see Table 3.9). This happens because of the decrease in analogical regularities as more artificial sentences are introduced.

3.6 Discussion

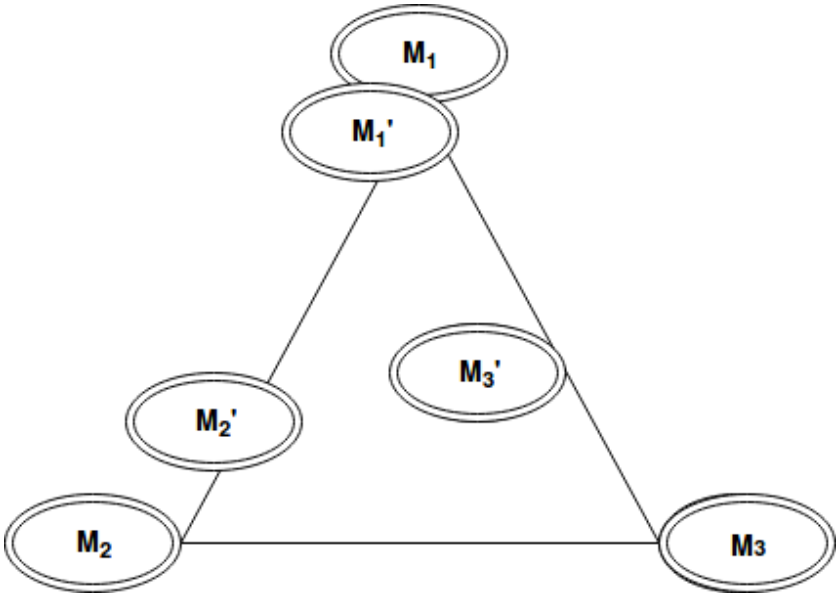

Figure 3.2 2D representation of shift in the word embeddings. M_1 is initial and M_1' is the word embedding after expansion step(s). Similarly for M_2 and M_3.

The major achievement of our approach is improving the word embeddings of most of the words in the dataset. Let us consider three words – M_1, M_2 and M_3, which are semantically similar morphological forms (also first cousins) of each other. For example, M_1, M_2 and M_3 may be "walk", "walker" and "walking". Their frequencies are in the order $M_1 > M_2 > M_3$. Because of its high frequency, it is reasonable to assume that M1 has better trained word embedding than M_2 or M_3.

While generating artificial sentences, M_1 and M_2 are replaced with a certain probability by their other two morph forms. However, because of its low frequency, M_3 will have a null morph set and will not be replaced by M_1 or M_2. As shown in Fig. 3.2, a higher frequency word shows lesser shift as compared to its initial trained word embedding. Also, a word will show higher shift towards its more frequent semantically similar morphological forms.

Hence our method helps in greatly improving word embeddings of lower frequency words by shifting their embedding towards its higher frequency semantically similar first cousins. As frequency of words increases, similar phenomena can be observed involving smaller shifts.

We also observe that there is no significant fall in the number of OOV (out of vocabulary) words, but there is a decrease in some small datasets due to text expansion. This happens because low frequency

words which were not being trained previously (their frequency being lower than the threshold), are now frequent enough to be trained, as they occur in artificially generated sentences. Hence, when encountered in test data, we have some representation for these OOV words in our model unlike being assigned zero vector initially.

3.7 Conclusion

As expected, the results for Arabic show the inherent complex nature of the language and can be accounted to the method being unable to reconcile infixes as opposed to suffixes and prefixes resulting in incorrect morphological sets.

Our method successfully exploits morphological regularities to produce high quality of word embeddings. Evaluations show that the method may be used to deal with the problem of data sparsity across different languages, the effect is particularly noticeable for morphologically rich languages. Our results are comparable to the ones presented earlier [13] which was trained on a much larger dataset (42 billion tokens) compared to a much smaller training data (1 billion tokens) used in this method.

Chapter 4

Exploiting Morphological Regularities in Distributional Word Representations

Word representations capture both syntactic and semantic properties [37] of natural language. Soricut and Och [29] 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 chapter, 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 - for each of the transformation rules, which aligns itself with the source word, to give 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) i.e, irrespective of the fact that root word (source word) is a 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.

This chapter is structured as follows. We first discuss the corpus used for training the transformation operators in section 4.1. In section 4.2, we discuss how these transformation operators are trained. Later in section 4.3, we analyze and discuss the results of our experiments.

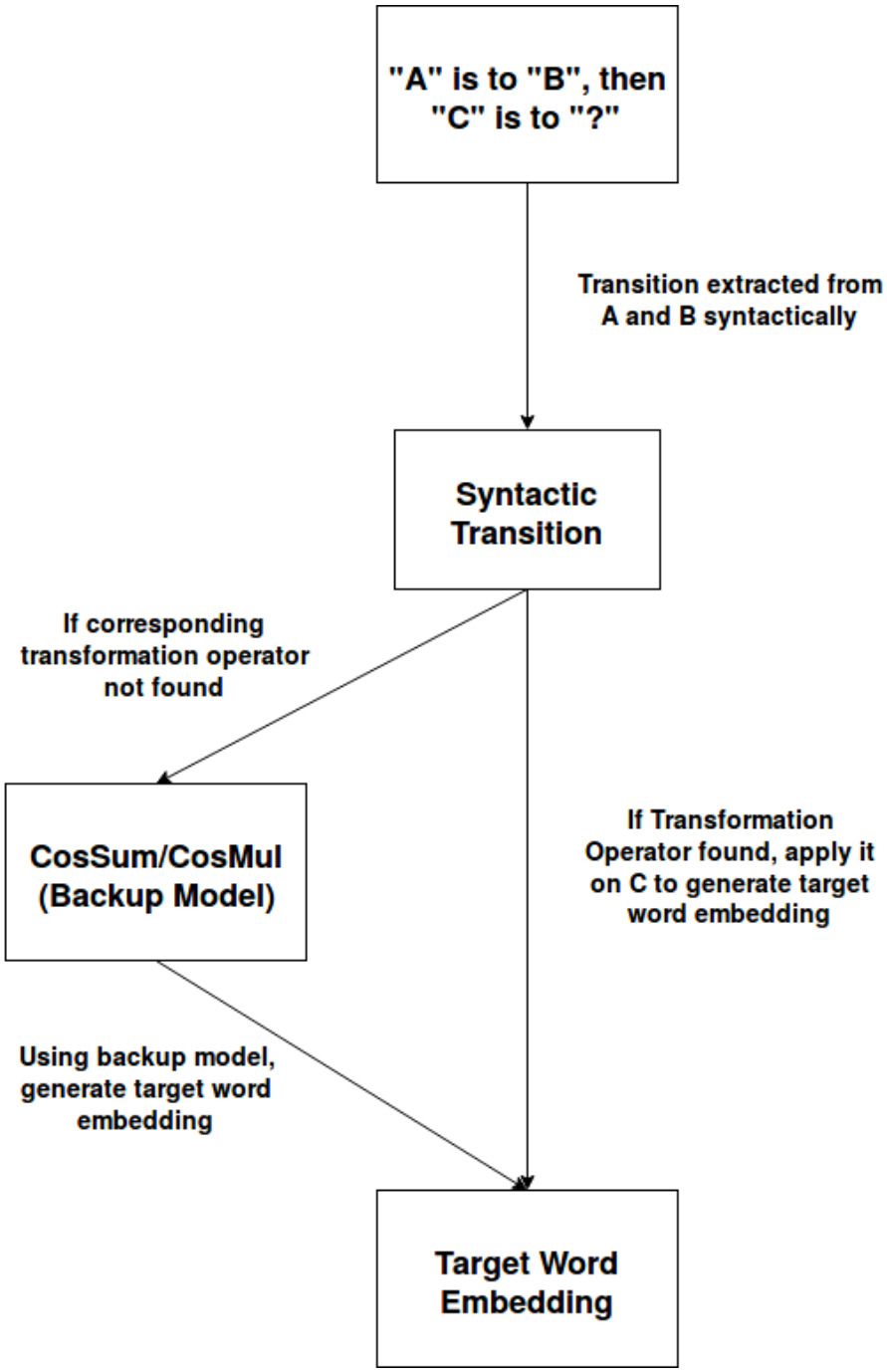

Figure 4.1 System Workflow

4.1 Datasets

We are using word embeddings trained on Google News corpus [38] for our experiments. For the model trained in this chapter, we have used the Skip-gram [36] 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 [6]. The corpus contains about 1 billion tokens.

The MSR dataset [37] 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.

4.2 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 chapter, 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 $A * X = B$, i.e. $X = A^{-1} * B$ (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 4.1.

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 4.1 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 [25][24] 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 analogy question can be solved by optimizing:

$$arg \max_{b^* \in V} (sim(b^*, b - a + a^*)) \tag{4.1}$$

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

$$cos(u, v) = \frac{u \,.\, v}{||u|| \,.\, ||v||} \tag{4.2}$$

resulting in:

$$arg \max_{b^* \in V} (cos(b^*, b - a + a^*)) \tag{4.3}$$

Equation 4.3 has been referred to as CosAdd model.

While experimenting, Omer Levy [24] 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:

$$arg\ \max_{b^* \in V} \frac{cos(b^*, b)\ .\ cos(b^*, a^*)}{cos(b^*, a) + \epsilon} \tag{4.4}$$
$$(\epsilon = 0.001\ to\ prevent\ division\ by\ zero)$$

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. [25] 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 4.1 gives an overview of how target word embedding is generated using transformation operators and our backup models.

4.3 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 4.2 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 4.3 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 4.4 Example results of transformation operators for irregular transformations.

In table 5.4, GN denotes the scores of Google-News word embeddings on the test set. SGNS-L and Glove-L [25] 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 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.

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

Table 4.5 Example results of transformation operators for complete change of word form.

As we can see in table 4.3, 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 4.4 and table 4.5 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.

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).

4.4 Discussion

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.

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 transformation matrix also works for more complex word transformations like “<ed,s>” where “said” successfully transforms to “says”.

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.

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

Chapter 5

Mapping between Vector Spaces & Word Similarity Datasets

While we were working on exploiting morphological regularities in distributional word representations, we discovered that we can extend our approach to develop cross lingual vector mapping too. In this chapter, we will give you an overview of how this approach works, datasets used, our supporting experimental results, and our contribution in development of the approach.

5.1 Mapping between Vector Spaces

We try to exploit similarities in linguistically similar languages by transforming word embeddings of source language - which is highly resource rich and hence well trained, to a corresponding model of target language - which is relatively resource deficient. This technique is similar to that used by Mikolov et. al. [38] for machine translation. We further extend our approach by applying it to different models trained for one particular language. This enables us to incorporate the best of various models.

Given a source of structured connections between words, Faruqui et.al [19] proposed a technique to combine embeddings learned from distributional semantics of unstructured text, known as "retrofitting". Speer and Chin [33] extended this technique to produce state of the art word embeddings for English. The method proposed by them resulted in a 16% increase on the Stanford English Rare-Word (RW) dataset [21].

We propose a method to transform words from one vector space to another. We use this technique to transform word embeddings between languages and also within the same language. Some languages are richer in resources than others. For example, Hindi is richer in resources than Urdu. We also evaluate our approach on dissimilar language pairs like English - French and English - German. Our aim is to use resource rich techniques like ConceptNet Ensemble [33] for languages poor in resources, for which we need to learn a mapping between their vector spaces.

Using this technique, we are also able to get embeddings of words which are unknown for one embedding space by importing the transformed embedding of the same word from another model of the same language. This method proved to be faster than training a combined embedding space from scratch, while giving high quality word embeddings.

The basis of our approach lies in having a sufficient number of frequent word pairs in both source and target languages to successfully train our transformation matrix. Each word pair is of form <word from source language, translated word of target language>. In this paper, we present a method for transforming word representations which, for training, take word representations of these word pairs to create a single transformation matrix, which when applied on any word representation of source word, will give us word representation of corresponding word in target language. We emphasize on highly frequent words, because we believe that highly frequent words have better trained word embeddings and thus result in better results for our approach.

We rely on a bi-lingual dictionary of the language pair for training and evaluating our transformation matrix. For training our matrix, we generate a bi-lingual dictionary from parallel corpus of source and target language in an unsupervised way. But for evaluation, the bi-lingual dictionary was missing most of the word pairs present in our test dataset because the parallel corpus for all the language pairs that we worked on in this paper were not large enough. Hence, only for evaluating our transformation matrix, we manually created a bilingual dictionary from the test sets - which was not used for training our transformation matrix because transformation matrix tends to completely remember the word representations it was generated from.

We tried our approach on two different types of language pairs - diverse and similar. For similar language pairs, we chose Hindi and Urdu and for diverse, we tested our approach between English - French and English - German.

5.1.1 Datasets

For some experiments, we are using word embeddings trained on Google News corpus [38]. For all the models trained, we have used the Skip-gram [36] algorithm. The dimensionality has been fixed at 300 with a minimum count of 5 along with negative sampling.

As training set for English, we use the Wikipedia data [6] (SG/en-SG). Soricut and Och [29] and Luong et. al. [21] had used the same training corpus for their models. The corpus contains about 1 billion tokens. For German and French, we use News Crawl (Articles from 2010) released as a part of ACL 2014 Ninth Workshop on Statistical Machine Translation (de-SG and fr-SG respectively). For Urdu, we use the untagged corpus released by Jawaid et. al. [2]. For Hindi, we use a monolingual corpus containing 31 million tokens, for training.

As a parallel corpus for Urdu, we use the Hindi-Urdu parallel corpus released by Durrani et. al. [23]. We have used the Europarl parallel corpus version 7 [27] for parallel sentences of English-French and English-German.

We use standard word-similarity datasets for testing. For English, we use Stanford English Rare-Word (RW) dataset [21] and the RG65 dataset [11]. The Stanford Rare-Word dataset contains comparatively more rare words and morphological complexity than other datasets and is central to our experiments. For German, we use the German RG65 [39] dataset. For French we use the French RG65 [5] dataset. In case of Urdu, we are using the word similarity dataset WS-UR-100 [35].

For testing analogical regularities, we have used the MSR word analogy dataset [37]. It contains 8000 analogy question. This dataset 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 rest of the paper, we have calculated the Spearman ρ (multiplied by 100) between human assigned similarity and cosine similarity of our word embeddings for the word-pairs.

All the thresholds mentioned have been decided after empirical fine tuning. Even though our experiments were computationally optimized, time and space complexities also played a part in deciding our thresholds.

In order to learn initial representations of the words, we train word embeddings (word2vec) using the parameters described above on the training set. This model is referred to as SG (Skip-gram).

5.1.2 Results

System	**WS-UR-100**
ur-SG	34.50
hi-SG-T	**50.08**

Table 5.1 WS-UR-100 is the **Urdu** version of en-RG-65 and en-WS353 test sets. Note that hi-SG-T denotes the scores of transformed word embeddings of hi-SG.

System	**de-RG-65**
SO [29]	62.4
SO w/ Morph [29]	64.1
de-SG	64.96
en-SG-T	**67.3**
ConceptNet-T	**83.17**

Table 5.2 de-RG-65 is the **German** version of en-RG-65 test set. Note that en-SG-T and ConceptNet-T denote the scores of transformed word embeddings of en-SG and ConceptNet.

We can see the increase in accuracy for Hindi-Urdu language pair in Table 5.1. In table 5.3 we can see that en-SG gave a score of 68.62 on fr-RG-65 after transformation, whereas for ConceptNet, the score is 80.13. In Table 5.2, we see that there is a proportional increase in the scores of the transformed embeddings of en-SG-T and ConceptNet-T (as we saw in case of French).

System	fr-RG-65
SO [29]	63.6
SO w/ Morph [29]	67.3
fr-SG	62.48
en-SG-T	**68.62**
ConceptNet-T	**80.13**

Table 5.3 fr-RG-65 is the **French** version of en-RG-65 test set. Note that en-SG-T and ConceptNet-T denote the scores of transformed word embeddings of en-SG and ConceptNet.

We see that there is a large increase in the scores of the transformed embeddings of hi-SG-T, considerably greater than en-SG-T or de-SG-T over en-SG owing to similarity between the two languages. Please note that the comparison is not made between ConceptNet-T and hi-SG-T but between en-SG-T (of both French and German) and hi-SG-T because anything of similar nature to ConceptNet does not exist for Hindi.

5.1.3 Transformation between Different Models of Same Language

While evaluating and testing our approach, we realized that it might be possible to port and use multiple models as and when needed, for a single language. We often encounter models trained for different types of data, for different purposes, but for the same language. Our approach enables us to generate word embeddings for words that are missing in one model but are present in another model. This enables us to reduce number of unseen words encountered, and after careful evaluation, we found that this approach indeed helps us. We tested this approach using word similarity and word analogy tasks and it showed significant improvement in results.

For creating transformation matrix for two different models of same language, we initially require two different models trained on two different datasets - so that there are certain set of words which are not present in both the models. Then we take a small corpus, for determining the frequent words of the language (above a frequency threshold of 500 and also present in both the models). Using these frequent words, we create our transformation matrix.

We now propose that a matrix "X" (our transformation matrix) will exist such that $A * X = B$, i.e. $X = A * B$. 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 it. 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 patterns that the matrix follows.

After creating this matrix, now whenever we encounter a word which is not present in our first model, we look for the word in our second model, and if found, we apply our transformation matrix to its embedding in our second model. This results in a representation of the word, which proved to be good enough, when we ran word similarity and word analogy tasks on it. Figure 3.1 gives a high level overview of how we try to incorporate different models of the same language via transformation matrix.

System	RW	Unseen Words
GN	45.27	173
SG	40.08	88
GN+SG	48.56	58

Table 5.4 Scores on Stanford Rare word test set.

In Table 5.4, GN denotes the scores of embeddings trained in Google-News word embeddings. SG denotes the scores of the embeddings trained by us. GN+SG denotes the system in which we import the embeddings of any word missing in GN from SG.

5.2 Word Similarity Datasets

Word Similarity task is a computationally efficient method to evaluate the quality of word vectors. It relies on finding correlation between human assigned semantic similarity (between words) and corresponding word vectors. We have used Spearman's Rho for calculating correlation.

Unfortunately, most of the word similarity tasks have been majorly limited to English language because of availability of well annotated different word similarity test datasets and large corpora for learning good word representations, where as for Indian languages like Marathi, Punjabi, Telugu etc - which even though are widely spoken by significant number of people, are still computationally resource poor languages. Even if there are models trained for these languages, word similarity datasets to test reliability of corresponding learned word representations do not exist.

5.2.1 Construction of Hindi Word Similarity Dataset: Hin-WS235

English RG-65 and WordSim-353 were used as base for creating our word similarity dataset. Translation of English data set to target language - Hindi, was manually done by a set of three annotators who are native speakers of the target language and are fluent in English. Initially, translations are provided by two of them, and in case of disparity, third annotator was used as a tie breaker.

Finally, all three annotators reached a final set of translated word pairs in target language, ensuring that there were no repeated word pairs. This approach was followed by Camacho-Callados et al. [4] where they created word similarity datasets for Spanish and Farsi in a similar manner.

5.2.1.1 Scoring

Eight native speakers were asked to manually evaluate word similarity data set individually. They were instructed to indicate, for each pair, their opinion of how similar in meaning the two words are on a scale of 0-10, with 10 for words that mean the same thing, and 0 for words that mean completely different things. The guidelines provided to the annotators were based on the SemEval task on Cross-Level Semantic Similarity [16], which provides clear indications in order to distinguish similarity and relatedness.

The results were averaged over the 8 responses for the word similarity data set, and dataset saw good agreement amongst the evaluators.

5.2.1.2 Evaluation

Inter Annotator Agreement (IAA) The meaning of a sentence and its words can be interpreted in different ways by different readers. This subjectivity can also reflect in annotation of sentences of a language despite the annotation guidelines being well defined. Therefore, inter-annotator agreement is calculated to give a measure of how well the annotators can make the same annotation decision for a certain category.

Fleiss' Kappa Fleiss' kappa is a statistical measure for assessing the reliability of agreement between a fixed number of raters when assigning categorical ratings to a number of items or classifying items. This contrasts with other kappas such as Cohen's kappa, which only work when assessing the agreement between not more than two raters or the interrater reliability for one appraiser versus himself. The measure calculates the degree of agreement in classification over that which would be expected by chance [40].

We have calculated Fleiss' Kappa for our word similarity dataset to be **0.773589**.

5.2.2 Creation of Word Similarity Datasets for 6 Indian Languages

Primary motivation for creation of these six word similarity datasets has been to provide necessary evaluation resources for all the current and future work in field of word representations on these six Indian languages - all ranked in top 25 most spoken languages in the world, since no prior word similarity datasets have been publicly made available.

Creation of these newly created word similarity datasets would allow for fast and efficient comparison between word embeddings trained on different models. Word similarity is one of the most important evaluation metric for word representations and hence as an evaluation metric, these datasets would promote development of better techniques that employ word representations for these languages. We also present baseline scores using state-of-the-art techniques which were evaluated using these datasets.

Multitude of word similarity datasets have been created for English, like WordSim-353 [17], MC-30 [10], Simlex-999 [9], RG-65 [11] etc. RG-65 is one of the oldest and most popular datasets, being used as a standard benchmark for measuring reliability of word representations.

Language	Inter Annotator Agreement
Urdu	0.887
Punjabi	0.821
Marathi	0.808
Tamil	0.756
Telugu	0.866
Gujarati	0.867

Table 5.5 Inter Annotator Agreement (Fleiss Kappa) scores for word similarity datasets created for six languages.

System	Score	OOV	Vocab
CBOW	28.30	19	130K
SG	34.40	19	130K
FastText	34.61	19	130K
FastText w/ OOV	45.47	14	-

Table 5.6 Baseline results for **Urdu**

RG-65 has also acted as base for various other word similarity datasets created in different languages : French [5], German [39],Portuguese [30], Spanish and Farsi [4]. While German and Portuguese reported IAA (Inter Annotator Agreement) of 0.81 and 0.71 respectively, no IAA was calculated for French. For Spanish and Farsi, inter annotator agreement of 0.83 and 0.88 respectively was reported. Our datasets were created following guidelines mentioned in previous subsection, using RG-65 and WordSim-353 as base.

System	Score	OOV	Vocab
CBOW	36.16	3	194K
SG	41.22	3	194K
FastText	33.68	3	194K
FastText w/ OOV	38.66	0	-

Table 5.7 Baseline results for **Marathi**

System	Score	OOV	Vocab
CBOW	26.01	14	174K
SG	27.04	14	174K
FastText	34.29	14	174K
FastText w/ OOV	46.02	0	-

Table 5.8 Baseline results for **Telugu**

5.2.3 IAA Scores and Baseline Scores

We present IAA (Inter-Annotator Agreement) scores and baseline scores using state of the art techniques - CBOW and Skipgram [36] and FastText-SG [28], evaluated using our word similarity datasets in tables 5.6, 5.7 and 5.8. As we can see the models trained encountered unseen word pairs when evaluated on their corresponding word similarity datasets. This goes on to show that all word pairs in our word similarity sets are not too common, and contain word pairs with some rarity.

Chapter 6

Conclusions

Even though our expansion method successfully exploits morphological regularities to produce high quality of word embeddings, while synthesizing artificial sentences, a lot of sentences generated were grammatically incorrect. But since our initial word embeddings were poorly trained due to lack of data, these grammatically incorrect sentences do help in improving corresponding word embeddings since their main purpose is to provide more training samples, or artificial data for word2vec to train on.

Although these sentences might be undesirable, we can further explore how this artificially generated data might help us in other linguistic tasks, but for languages that are computationally poor, it helped in improving their word embeddings. We also evaluated our approach on languages that are resource rich, and as expected, word embeddings were affected negatively by our artificial sentences because initial word embeddings were already well trained. For languages that are resource rich, we can apply techniques like HMM or doc2vec on synthesized sentences to keep only those which are correct to a certain degree and study their impact on the word embeddings.

We also observed that for languages like Arabic, which show inherent complex nature, our method was unable to perform as it did with respect to other languages. This can be accounted to the method being unable to reconcile infixes as opposed to suffixes and prefixes resulting in incorrect morphological sets. Hence future work can also be directed towards generating transformations which involve changes inside the word (apart from just changes to starting and endings) for improving representation learning for languages like Arabic.

Using operator matrices, we successfully exploited morphological regularities to generate global transformation operators, but we observed 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. This often results in our backup model being used quite often.

But by using our approach, we can 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 approaches, for a problem "If A is to B, then C is to ?", we have done 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 can 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.

By extending our transformation matrices to cross lingual vector transformation operators, we were able to significantly improve on baseline scores of Urdu and beat previous state of the art systems for German and French by generating cross lingual vector mapping between different vector spaces. This approach was extended and was able to improve scores by porting two models to generate word embeddings for missing words. We can further extend this approach by creating transformation matrix for not only two, but for all possible combinations of models available. This way, when ever a particular model is being used for a particular task, we can look for a missing word in other models and use its transformed representation accordingly. Significant improvement in training word embeddings by mapping from a source language to a linguistically similar target language, gives us the hypothesis that similar improvement can be achieved for training between different dialects of the same language and we would like to test the approach further on such pairs.

What we could also do is choosing best available word representation of a word in two or more models. For example, even though we have a word present in our model, but its representation is not reliable because of its low frequency in the corpus it was trained on or we are able to detect somehow that it is not well trained. We could then use a more reliable word embedding from other models, transform it for our model, and then use it. We would still need to run various evaluation tasks on this approach to see its impact and usage in future. We will also try to see if it can be modified to retain its character in one model, and import characteristics from its other word representations in other models, which are relatively more reliable. We could define a heuristic for the same, and evaluate on various different values of parameters in the heuristics.

Even though most of our models are unsupervised and language agnostic, it does not change the fact that there are a lot of Indian languages that are still computationally resource poor even though they are widely spoken by significant number of people. Creating word similarity datasets is a small step towards generating resources to further the research involving word representations on Indian languages.

To further extend our work, we can create rare-word word similarity datasets for six languages we worked on in this paper, and creating word similarity datasets for other major Indian languages as well.

We also realize that even though creating word similarity datasets provides us with necessary evaluation metrics for evaluating word embeddings, we still need to build training corpus for languages that do not have any for training models to start with. Hence, this will require us to build new corpus to train our models for three languages that we couldn't provide baseline scores for - Punjabi, Tamil and Gujarati and build more corpus for Urdu, Telugu and Marathi to train better word embeddings. We can also work on improving word representations for the languages we worked on, hence improve the baseline scores that we presented here.

Related Publications

1. Arihant Gupta, Syed Sarfaraz Akhtar, Avijit Vajpayee, Arijt Srivastava and Manish Shrivastava. **Unsupervised Morphological Expansion of Small Datasets for Improving Word Embeddings.** International Conference on Computational Linguistics and Intelligent Text Processing, CICLing, 2017

2. Arihant Gupta, Syed Sarfaraz Akhtar, Avijit Vajpayee, Arijt Srivastava and Manish Shrivastava. **Exploiting Morphological Regularities in Distributional Word Representations.** Conference on Empirical Methods in Natural Language Processing, EMNLP, 2017

3. Syed Sarfaraz Akhtar, Arihant Gupta, Avijit Vajpayee, Arjit Srivastava and Manish Shrivastava. **An Unsupervised Approach for Mapping between Vector Spaces.** International Conference on Computational Linguistics and Intelligent Text Processing, CICLing, 2017

4. Syed Sarfaraz Akhtar, Arihant Gupta, Avijit Vajpayee, Arjit Srivastava and Manish Shrivastava. **Word Similarity Datasets for Indian Languages: Annotation and Baseline Systems.** European Chapter of the Association for Computational Linguistics - Language Annotation Workshop, EACL-LAW, 2017

Bibliography

[1] Bojar, Ondrej and Diatka, Vojtech and Rychlỳ, Pavel and Stranák, Pavel and Suchomel, Vít and Tamchyna, Ales and Zeman, Daniel 2014. *HindEnCorp-Hindi-English and Hindi-only Corpus for Machine Translation*. LREC 2014.

[2] Bushra Jawaid, Amir Kamran, and Ondrej Bojar. 2014 *A Tagged Corpus and a Tagger for Urdu*. LREC 2014.

[3] Bengio Y, Ducharme R, Vincent P, Jauvin C 2003 *A neural probabilistic language model*. Journal of machine learning research. 2003;3(Feb):1137-55.

[4] Camacho-Collados, Jos, Mohammad Taher Pilehvar, and Roberto Navigli. 2015. *A Framework for the Construction of Monolingual and Cross-lingual Word Similarity Datasets*. In ACL (2) (pp. 1-7).

[5] Colette Joubarne and Diana Inkpen. 2011. *Comparison of semantic similarity for different languages using the Google n-gram corpus and second-order co-occurrence measures*. In Advances in Artificial Intelligence - 24th Canadian Conference on Artificial Intelligence, pages 216-221.

[6] Cyrus Shaoul and Chris Westbury. 2010. *The Westbury lab Wikipedia corpus*. In Edmonton, AB: University of Alberta.

[7] Fabrizio Sebastiani. 2002. *Machine learning in automated text categorization*. ACM Computing Surveys (CSUR), volume 34, pages 1-47.

[8] Fei Huang, Arun Ahuja, Doug Downey, Yi Yang, Yuhong Guo, and Alexander Yates. 2014. *Learning representations for weakly supervised natural language processing tasks*. Computational Linguistics, volume 40, number 1, pages 85-120.

[9] Felix Hill, Roi Reichart, and Anna Korhonen. 2016. *Simlex-999: Evaluating semantic models with (genuine) similarity estimation*. Computational Linguistics.

[10] George A. Miller, and Walter G. Charles. 1991. *Contextual correlates of semantic similarity*. Language and cognitive processes 6.1 (1991): 1-28.

[11] Herbert Rubenstein and John B. Goodenough. 2006. *Contextual correlates of synonym*. Communications of the ACM, volume 8, number 10, pages 627-633.

[12] Jan A. Botha and Phil Blunsom. 2014. *Compositional Morphology for Word Representations and Language Modelling*. In ICML, pages 1899-1907

[13] Jeffrey Pennington, Richard Socher and Christopher D. Manning. 2014. *GloVe: Global Vectors for Word Representation*. In EMNLP, volume 14, pages 1532-43.

[14] Jos Camacho-Collados, Mohammad Taher Pilehvar and Roberto Navigli 2015. *A Framework for the Construction of Monolingual and Cross-lingual Word Similarity Datasets*. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL 2015), Beijing, China, July 27-29, 2015.

[15] Joseph Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. *Word representations: A simple and general method for semi-supervised learning*. In Proceedings of ACL.

[16] David Jurgens, Mohammad Taher Pilehvar, and Roberto Navigli. 2014. *Semeval-2014 task 3: Cross-level semantic similarity*. Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014), in conjunction with COLING. 2014.

[17] Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman and Eytan Ruppin. 2002. *Placing search in context: The concept revisited*. Proceedings of the 10th international conference on World Wide Web, pages 406-414.

[18] Luong, Minh-Thang, and Christopher D. Manning. 2016. *Achieving open vocabulary neural machine translation with hybrid word-character models*. arXiv preprint arXiv:1604.00788.

[19] Manaal Faruqui, Jesse Dodge, Sujay K. Jauhar, Chris Dyer, Eduard Hovy. 2015. *Retrofitting word vectors to semantic lexicons*. In proceedings of NAACL.

[20] Mathias Creutz and Krista Lagu. 2007. *Unsupervised models for morpheme segmentation and morphology learning*. In ACM Transactions on Speech and Language Processing (TSLP), version 4, number 1, page 3

[21] Minh-Thang Luong, Richard Socher and Christopher D. Manning. 2013. *Better Word Representations with Recursive Neural Networks for Morphology*. In CoNLL. Pages 104-113.

[22] Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2014. *Tailoring continuous word representations for dependency parsing*. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, Baltimore, MD, USA, Volume 2: Short Papers, pages 809815.

[23] Nadir Durrani, Hassan Sajjad, Alexander Fraser, and Helmut Schmid. 2010. *Hindi-to-Urdu machine translation through transliteration*. in Proceedings of the 48th Annual meeting of the Association for Computational Linguistics, pp. 465-474. Association for Computational Linguistics, 2010.

[24] Omer Levy, Yoav Goldberg, and Ramat-Gan, Israel 2014 *Linguistic Regularities in Sparse and Explicit Word Representations* . In CoNLL, pp. 171-180. 2014.

[25] 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 (2015): 211-225

[26] Omer Levy, Yoav Goldberg 2014. *Neural word embedding as implicit matrix factorization*. Advances in neural information processing systems (pp. 2177-2185).

[27] Philipp Koehn. 2002. *Europarl: A multilingual corpus for evaluation of machine translation*. Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics. Association for Computational Linguistics, 2005.

[28] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov 2016. *Enriching word vectors with subword information*. arXiv preprint arXiv:1607.04606 (2016).

[29] Radu Soricut and Franz Och. 2015. *Unsupervised Morphology Induction using Word Embeddings*. In Proceedings of NAACL.

[30] Roger Granada, Cassia Trojahn, and Renata Vieira. 2014. *Comparing semantic relatedness between word pairs in Portuguese using Wikipedia*. International Conference on Computational Processing of the Portuguese Language. Springer International Publishing, 2014.

[31] Richard Socher, John Bauer, Christopher D. Manning, and Andrew Y. Ng. 2013. *Parsing With Compositional Vector Grammars*. In ACL, pages 455-465.

[32] Scott Miller, Jethran Guinness, and Alex Zamanian. 2004. *Name tagging with word clusters and discriminative training*. In Proceedings of HLT-NAACL, volume 4, pages 337-342.

[33] Robert Speer 2016. *Ensemble Method to Produce High-Quality Word Embeddings*. arXiv preprint arXiv:1604.01692 (2016).

[34] 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 information retrieval, pages 41-47.

[35] Syed Sarfaraz Akhtar, Arihant Gupta, Avijit Vajpayee, Arjit Srivastava, Manish Shrivastava 2017. *Word Similarity Datasets for Indian Languages: Annotation and Baseline Systems*. Proceedings of the 11th Linguistic Annotation Workshop, pages 91-94.

[36] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeff Dean. 2006. *Efficient estimation of word representations in vector space*. In In arXiv preprint arXiv:1301.3781

[37] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013. *Linguistic regularities in continuous space word representations*. In Proceedings of HLT-NAACL, volume 13, pages 746-751.

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

[39] Torsten Zesch and Iryna Gurevyc. 2006. *Automatically creating datasets for measures of semantic relatedness*. In Proceedings of the Workshop on Linguistic Distances, pages 16-24.

[40] Wikipedia contributors. 2017. *"Fleiss' kappa." Wikipedia, The Free Encyclopedia.*. Wikipedia, The Free Encyclopedia, 6 Feb. 2017. Web. 16 Feb. 2017.

[41] Zellig S. Harris 1954 *Distributional structure*. Word, 10(23):146162, 1954.