Article

Generation of Cross-Lingual Word Vectors for Low-Resourced Languages Using Deep Learning and Topological Metrics in a Data-Efficient Way

Sanjanasri JP ^{1,*}, Vijay Krishna Menon ² , Soman KP ¹, Rajendran S ¹ and Agnieszka Wolk ^{3,4}

¹ Center for Computational Engineering and Networking (CEN), Amrita School of Engineering, Amrita Vishwa Vidyapeetham, Coimbatore 641112, India; kp_soman@amrita.edu (S.K.); s_rajendran@cb.amrita.edu (R.S.)

² Gadgeon Systems Privated Limited, Kochi 682021, India; vijay.km@gadgeon.com

³ Multimedia Department, Polish-Japanese Academy of Information Technology, Koszykowa 86, 02-008 Warsaw, Poland; awolk@pja.edu.pl

⁴ The Institute of Literary Research of the Polish Academy of Sciences, Nowy Świat 72, 00-330 Warsaw, Poland

* Correspondence: jp_sanjanasri@cb.amrita.edu

Abstract: Linguists have been focused on a qualitative comparison of the semantics from different languages. Evaluation of the semantic interpretation among disparate language pairs like English and Tamil is an even more formidable task than for Slavic languages. The concept of word embedding in Natural Language Processing (NLP) has enabled a felicitous opportunity to quantify linguistic semantics. Multi-lingual tasks can be performed by projecting the word embeddings of one language onto the semantic space of the other. This research presents a suite of data-efficient deep learning approaches to deduce the transfer function from the embedding space of English to that of Tamil, deploying three popular embedding algorithms: Word2Vec, GloVe and FastText. A novel evaluation paradigm was devised for the generation of embeddings to assess their effectiveness, using the original embeddings as ground truths. Transferability across other target languages of the proposed model was assessed via pre-trained Word2Vec embeddings from Hindi and Chinese languages. We empirically prove that with a bilingual dictionary of a thousand words and a corresponding small monolingual target (Tamil) corpus, useful embeddings can be generated by transfer learning from a well-trained source (English) embedding. Furthermore, we demonstrate the usability of generated target embeddings in a few NLP use-case tasks, such as text summarization, part-of-speech (POS) tagging, and bilingual dictionary induction (BDI), bearing in mind that those are not the only possible applications.

Keywords: semantic interpretation; transfer learning; bilingual embedding; cross-lingual embedding; English–Tamil; low resourced languages; topological measures; ontology engineering

Citation: JP, S.; Menon, V.K.; KP, S.; S, R.; Wolk, A. Generation of Cross-Lingual Word Vectors for Low-Resourced Languages Using Deep Learning and Topological Metrics in a Data-Efficient Way. *Electronics* **2021**, *10*, 1372. <https://doi.org/10.3390/electronics10121372>

Academic Editor: Rui Pedro Lopes

Received: 10 May 2021

Accepted: 26 May 2021

Published: 8 June 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

The mapping of a word to a representation of its meaning is termed semantic representation. Such abstract representations can be comprehended by human cognition, but their reproductions have so far been mainly qualitative. The definition of semantics is largely based on linguistic theory, which encompasses lexical, syntactic, morphological, and other complex phenomena, such as ambiguity, negation, inferences, and lemmas [1,2].

Prior work apropos analysis and quantification of semantics has been exclusively approached either by statistical techniques such as Latent Semantic Analysis (LSA) [3] or by leaning on lexicographic knowledge bases such as WordNet and Thesauri. LSA applies Singular Value Decomposition (SVD) to the term-document matrix, in order to learn word representations. Lexicographic databases [4] encode certain relations among words—synonymy, hypernymy, and meronymy. In general, these resources are a reproduction of human interpretation, which is qualitative and example-based, representing only a

fragment of semantic information. The creation of lexical resources requires linguistic human expertise, a time- and labor-intensive manual process. Unfortunately, this process is not amenable to automation.

Quantitative representation of semantics is made possible with the evolution of word embeddings; these are densely distributed vector representations of words. These vectors translate relative semantics between words to spatial positions. The first-ever word embedding model learned the word vector representation by predicting the next word in a sequence within a local context window, using a neural network model [5]. Subsequently, all NLP tasks were effectively rebooted using deep neural networks [6]. Their overwhelming success in terms of state-of-the-art accuracy and excellent benchmark results, compared to extant statistical and other traditional methods, were supervised by the features of the deep learning network, automatically extracted from the corpus. In addition, the process is completely unsupervised. These neural features are now what we call *word vectors* or *word embeddings*. The ability of word embeddings to represent semantic relations between words as spatial distances is called the *semantic property* of the whole embedding set of vectors.

GloVe [7] and Word2Vec [8] and other contemporary vector training algorithms [9] are more accurate in capturing word-to-word semantics (relative semantics) than conventional vector space models such as LSA; the former performs better in almost all downstream tasks in NLP [10–12]. Vectors, pre-trained on a very large text $[(1-10) \times 10^6]$ corpus (one to ten million words), are readily available for almost all European languages and Asian languages such as Arabic, Chinese, Hindi, and Korean [13–15]. Morphologically rich languages like Slavic can also have equally effective vector representations [16–20].

1.1. Overview

In this paper, cross-lingual embedding is accomplished by mapping the vectors from one language's embedding space into that of the other language through a transfer function. Multiple experiments with various methodologies are carried out to obtain target word vectors for English–Tamil language pairs. Recently developed contextual embeddings methods such as Contextual word Vectors (CoVe), Embeddings from Language Models (ELMo), and Bidirectional Encoder Representations from Transformers (BERT) do not support the transfer of knowledge across languages, mainly from resource-rich to resource-poor languages as they use robust baseline architectures specific to each task. Hence, experiments are carried with the three most popular embedding algorithms, Word2Vec, Glove, and FastText. The trained cross-lingual model, **Transfer Function-based Generated Embedding (TFGE)** synthesizes new vectors for unknown words by transfer learning with a minimal seed dictionary (five thousand words) from a resource-rich source language (English) to a resource-poor target language (Tamil).

The topology-based comparative assessment (neighborhood analysis) was considered to assess the quality of the generated embedding, as word embedding has no ground truth data available [21]. The trained cross-lingual model is language-independent. The built-in model can be shared by the languages that share a lot of common syntax and vocabulary with the target language. Hence pre-trained Hindi and Chinese embeddings (Word2Vec) were piped through the cross-lingual model on the target side to show the sharing property (transferability). The generated embeddings were further validated with real NLP tasks such as Text Summarisation, a multi-class model of the Part-Of-Speech Tagging and Bilingual Dictionary Induction (BDI) for low-resource languages featuring Tamil.

1.2. Motivation

The primary downside of vector representations is that they quantify only relative semantics between words. It is evident that for the same corpus, vectors will manifest disparately each time a different vector training algorithm is used. *These vectors have no absolute position*. In linear algebraic terms, the vector space spanned by each embedding model is different [22]. The embeddings offer the best opportunity to address NLP use cases such as machine learning (ML) tasks, which call for comparison of vectors (projections

and linear combinations) in multilingual scenarios. Evidently, this creates a problem, as embeddings are not in the same vector space; they are trained from multiple disparate corpora. *The vectors cannot be compared.* To attain comparability, the generated vectors need to be mapped independently to a single vector space or have a transfer function to project them to other vector spaces. This was the main motivation for the exploration of bilingual and cross-lingual models. In the subsequent sections, transfer learning monolingual embeddings were devised for the generation of cross-lingual target vectors, using an ML model and a bilingual dictionary (English–Tamil language pair). The results were validated by verification of the semantic and topological properties of the generated vectors.

2. Bilingual Embeddings and TFGE

Several approaches have been adopted to bring about the common semantic representation (preserving common vector space) of words across languages. Consider the word “boy” in English. Suppose we built a bilingual representation between English and Esperanto; then, the Esperanto lexicon “*knabo*” should mimic the same semantic behavior as “boy”. The goal of any bilingual model is to capture the common semantics between languages. Another example is the word “run” in English, which has the equivalent Tamil word “*Oodu*”. However, “run” and “*Oodu*” have other incompatible synonyms, such as in an English sentence, “A small river runs into the sea”. Bilingual embeddings typically capture the common semantics to the exclusion of alternative interpretations; an obligatory trade-off in the interest of a common vector space.

Mapping of bilingual embeddings can be effectuated in two broad approaches. One method purely leverages bilingual training with sentence/document-aligned bilingual corpora [23–27]. The other method is to learn transfer functions, noted before, that will project the vectors from the embedding space of one language to that of the other. We call this method TFGE. Bilingual embeddings call for heavy resources in the form of parallel, comparable corpora with word-aligned and sentence-aligned subsets. Bilingual embeddings are a trade-off between the actual semantics of both languages. These embeddings may not be the felicitous choice for monolingual tasks, punctuated by the compromise over language-specific semantics, in the interest of common vector space. This is yet another justification to consider the cross-lingual transfer function model where the embedding spaces are monolingually trained.

2.1. Case of a Low-Resource Target Language

When a language pair is such that one language is corpus-rich with resources and the other is low-resourced, the cross-lingual embedding model can be used to augment the embedding space of the low-resourced target language. The augmentation is a multi-purpose process:

- Vectors for the unknown words of the target language can be generated using a bilingual dictionary
- The effectiveness of the target language embeddings can be improved

The suitability of “embeddings” can be evaluated, which simply connote better representation of relative semantics between the translational equivalence (bilingual) words of languages. Bilingual embeddings require an aligned corpus from which bilingual information is derived. A transfer learned cross-lingual model derives bilingual information merely from a bilingual dictionary. The creation of a bilingual dictionary is of much less effort compared to the creation and alignment of a parallel corpus. This paper explores the following applications:

- Transfer learning functions that generate cross-lingual embeddings for the target language;
  - Transfer functions defined on three word embeddings, Word2vec, GloVe, and Fast-Text;

- Three techniques are used, linear mapping as mentioned in [8] and two deep learning networks, One Dimensional Convolutional Neural Network (1D-CNN) and Multi-Layer Perceptron (MLP);
- A standard bilingual embedding algorithm, Bilingual Bag-Of-Words without Alignments (BilBOWA) is also considered for relative comparison;
- Achieve the above objective in a data-efficient way;
  - Transfer functions over various embedding is learned using different dictionary size as low as 1000 English–Tamil word pair;
  - Parallel or comparable corpora are not used to generate embedding, only the learned transfer function;
- Evaluation of the generated embeddings for their efficacy;
  - Embeddings are evaluated quantitatively using topological measures using Pair-wise Accuracy and Neighborhood Accuracy;
  - Visually verified using t-SNE plots for each of the TFGes categories;
  - Usability is tested for real NLP use-cases POS tagging, Extractive summarization, and BDI.

The three most popular contemporary vector training algorithms, namely Word2Vec, GloVe and FastText [7,28,29], are used to generate cross-lingual embedding. Bilingually trained BilBOWA [26] embeddings were used as a baseline reference to compare our TFGes. The primary dataset used is a cEnTam English–Tamil corpus [30], which has monolingual English, Tamil, and sentence-aligned English–Tamil data.

2.2. Premise

A transfer function (F) enables mapping from a pre-trained source embedding (X), to the target embedding (Y). The target embedding is derived by the use of a contemporary vector training algorithm on a limited target corpus. Source embeddings are pre-trained on a billion-word text corpus, readily retrievable from various online sources. ML model (M) is used to learn the mapping from X to Y . However, this is possible only if the bilingual information of X matches with that of Y . The bilingual information is obtained from the dictionary (D), which has to be created. This method obviates the use of any aligned bilingual corpus.

Once the model M is trained, it provides the transfer function F , which is used to generate vectors for unknown target words by providing a vector of the known source word. It is imperative to note that the target is the low-resource language. Figure 1 schematically explains the flow of the working premise. The creation of a bilingual dictionary is easier and is a well-defined process in preference to the creation of an aligned corpus.

Transfer learning is achieved in two ways. The embedding set (not all words) for a language is not a feature set by itself, but a fairly good language model. One way is to use pre-trained source embedding to generate vectors of the unknown target words by mapping the vector spaces, i.e., by transferring the information in the source embedding to improve the target embedding. *It is of vital significance that the target embedding is obtained with the same vector training algorithm that was employed for pre-trained source embedding.* The generated vectors exist in the target embedding space.

Similar languages may roughly align their embedding spaces. This will enable re-use of the transfer function trained on a target language, directly with another language, which shares common semantic properties. Empirical proof of this hypothesis is presented in future sections. The second transfer learning method fits parlance with ML, where a model trained with one pair serves another without augmentation/retraining. In summary, instead of generating embedding from a large monolingual corpus, the pre-trained embedding of one language is used to generate a bulk of the unknown vectors, using a machine-learned transfer function. The generated cross-lingual embedding exists in the same space of monolingual embedding of the target language. Doing so averts the need to compromise on language specific semantics, unlike bilingual models.

Figure 1. Flow chart of the proposed transfer learning model.

In order to test the effectiveness and usability of the generated embedding, two evaluation metrics are proposed:

- Pairwise cosine accuracy—a measure of semantics between similar words
- Global cosine neighbourhood, which measures how words are separate from each other.

These relative measures compare generated embeddings with the original target embeddings.

3. State-of-the-Art Transfer Learning Techniques in NLP

In a typical deep learning algorithm, a model is trained to learn patterns from training data to efficiently classify and predict unseen data [31,32]. With transfer learning, a model is generalized by reuse of knowledge learned from one task to an entirely different task. With NLP, transfer learning is anticipated to be a useful option in the development of efficient models, given the noisy, diversity, and unstructured characteristics of text data. The principal challenge in the application of transfer learning across languages in NLP is the language dependency of numerous tasks, which are inexpedient for processing by ML models generally.

Optimal leveraging of existing datasets is crucial as creation of parallel corpora for low-resourced language is expensive. With the proliferation of word embedding methods like Word2Vec, GloVe, and FastText, pre-trained (generic) embeddings are exploited in a wide variety of tasks, even if there is a lack of adequate data. Leveraging prior knowledge from pre-trained embedding to solve a completely different task is a perfect example of transfer learning.

The following research papers highlight some current developments in transfer learning techniques in NLP. Doc2Vec [33] leverages the information in pre-trained word embeddings (Word2Vec) for the generation of the embedding for larger chunks of text, like sentences and documents. CoVe is a type of word embedding learned by an encoder in an attentional seq-to-seq machine translation model [34]. CoVe are learned on top of the original word vectors, Word2Vec, GloVe, or FastText vectors to generate slightly different embeddings for each word based on its context. The authors of CoVe devised the Machine Translation–Long Short-Term Memory (MT-LSTM) system that encodes words in

context and decodes them into another language. This model uses a two-layer bidirectional LSTM-based encoder initialized with GloVe and a two-layer unidirectional LSTM with an attention mechanism as the decoder. The pre-trained encoder of MT-LSTM, CoVe, is applied across various downstream NLP tasks such as sentiment analysis and question classifier based on the transfer learning idea. CoVe and GloVe embedding's unified approach is said to be more reliable than the application of GloVe alone.

Differently from CoVe, the recently developed contextualized word embedding algorithms such as Embeddings from Language Models (ELMo) [35] and Bidirectional Encoder Representations from Transformers (BERT) [36] learn contextual word representation by pre-training a language model in an unsupervised way. The vector generated is a weighted combination of all layers in the network. ELMo uses a combination of independently trained bidirectional LSTMs. BERT uses the Transformer, a neural network architecture based on a self-attention mechanism. The Transformer has demonstrated superior performance in modelling long-term dependencies in the text, compared to the RNN architecture [37]. Thus, the ELMo and BERT capture the syntax, semantics, and polysemy of a word using the deep embeddings from the language model and can be used for a multitude of NLP activities. The integration of the contextual word embeddings into neural architectures has led to consistent improvements in important NLP tasks such as sentiment analysis, question answering, reading comprehension, textual entailment, semantic role labelling, co-reference resolution, and dependency parsing.

Although traditional word embedding algorithms, including Word2Vec, GloVe, Fast-Text and the contextualized embedding, transfer knowledge from a general-purpose source task to a more specialized target task, there are some significant differences. The features of attention models such as ELMo and BERT are more specific since they are context-dependent and cannot be generalized to a new task (transfer learned across languages) because specific features are less useful for transfer learning. The standard word embedding algorithms are not context-dependent, and the various semantics concerning the word are mixed, hence a generic representation. The contextualized embedding computes vectors dynamically as a sentence or a sequence of words is being processed, so it is necessary to provide a model for the downstream tasks. The standard word embedding algorithms generate a matrix of word vectors that can be plugged into the neural network model to perform a lookup operation by mapping a word to a vector.

A few propitious examples of transfer learning in NLP based on topological properties are noted next. The authors of [38] present a comprehensive study of the machine-translation-based cross-lingual approach of sentiment analysis in Bengali. The paper compares and provides a detailed analysis regarding the performance of ML classifiers in the Bengali and machine-translated datasets (English). The performance of simple transfer learning that utilizes the cross-domain data is presented. The authors use multiple cross-domain datasets from the English language, IMDB, TripAdvisor, etc. to train the Logistic Regression (LR) classifier. The trained model predicts the semantic orientations of reviews from the machine-translated (Bengali–English) corpus. The authors of [39] use monolingual resources and unsupervised techniques to induce cross-lingual task-specific word embeddings for the tasks of emoji prediction and sentiment classification of micro-blog posts from Twitter and Sina Weibo. Enormous Mandarin Chinese language datasets were utilized to train a monolingual model for emoji prediction, and the trained embedding layer was adapted to support the English language. The cross-lingual English models achieved 11.8% accuracy at emoji prediction (out of 64 emojis) and 73.2% at binary sentiment classification. Lastra-Díaz et al. [40] developed a software Half-Edge Semantic Measures Library (HESML) to implement various ontology-based semantic similarity measures proposed to evaluate word embedding models and have shown an increase in performance time and scalability of the models. Classifying Interactively with Multilingual Embeddings (CLIME) efficiently specializes cross-lingual embedding using the annotated task-specific key-words by bilingual speakers [41].

Therefore, the proposed methodology uses standard word embedding algorithms such as Word2Vec, GloVe, and FastText to build a learning conversion (cross-lingual) model trained with limited resources from English to Tamil. Furthermore, the model can be reused for languages with comparable semantics with Tamil as the features learned are generic.

4. Dataset Description

We used cEnTam, an English–Tamil bilingual dataset [30]. The dataset consists of a sentence-aligned English–Tamil corpus, a good chunk of crawled monolingual corpus of Tamil, and a stand-alone comparable corpus of English. We used this dataset for the generation of Tamil embeddings for all our experiments, and the bilingual embeddings were generated using BilBOWA algorithms. Details of the dataset used for training cross-lingual embedding are shown in Table 1. The data are organized as four instances.

Table 1. Dataset descriptions and dictionary sizes.

Source Data	Target Data	Algorithm	#. of Dic. Words	Attribute
Wikipedia pre-trained (en)	Wikipedia pre-trained (ta)	FastText	10,786	Comparable
GloVe 840b pre-trained (en)	cEnTam-Monolingual Tamil (ta)	GloVe	10,861	Monolingual
Google new pretrained (en)	cEnTam-Monolingual Tamil (ta)	Word2Vec	10,723	Monolingual
cEnTam (en)	cEnTam (ta)	BilBOWA	6088	Parallel/Comparable

The first instance was trained using the Word2Vec algorithm. The English source is constituted of pre-trained Google news embeddings trained on 100 billion words. Correspondingly, cEnTam is trained on Word2Vec for Tamil.

The second instance had to be trained with the GloVe algorithm. Here, the source was a set of pre-trained English embeddings, which use a common crawled corpus of 840 billion words. The Tamil GloVe embeddings were obtained from the cEnTam corpus. With FastText models, pretrained Wikipedia corpus embeddings and a cEnTam corpus for Tamil embedding were used.

The cEnTam corpus was used for training bilingual embedding using BilBOWA. All four dictionaries were carefully hand-crafted while maintaining good dispersion of all categories of words over the Tamil vocabulary.

Pre-trained Hindi and Chinese vectors were sourced online [42], along with the respective online dictionaries [43,44]. Hindi and Chinese dictionary sizes were capped at 1000 words in order to induce a resource constraint. The purpose of these embeddings is to demonstrate transferability.

5. Learning Transfer Functions

In Section 2.2, we introduced the premise of a transfer function that maps word embeddings from one language to another. We employed ML and Deep Learning (DL) techniques to learn these mappings. No effort was spent in tuning the hyper-parameters of these models in order to improve their model accuracy. Deep Neural networks are used as a transformative model to generate the transfer function that will map a known English Vector to an unknown Tamil vector. As the physicality of the numbers is not known, the only way to map vectors sensibly is to use deep learning architectures, which are data-driven and hence self-guided. Multi-Layer Perceptron (MLP) and One Dimensional Convolutional Neural Network (1D-CNN) were specifically chosen for this experiment as MLP is a fully-connected network and the latter is a non-fully connected network.

5.1. Linear Mapping

An elementary way of mapping to and from and between vector spaces is by projection. A vector in English embedding space is projected to the Tamil embedding space. If x represents a vector for an English word and y represents a vector for a Tamil word, then we can compute a matrix T . Here, x is 300×1 , T is 300×300 , and y is 1×300 .

$$Tx = y \quad (1)$$

Now consider matrix X spanning the embedding space of English and matrix Y spanning the embedding space of Tamil. Then T can be computed as shown in Equation (2), where X^+ is Moore–Penrose pseudo inverse and $X^+ = (X^T X)^{-1} X^T$. Equation (2) presents the transfer function as a matrix operator, T [45].

$$\begin{aligned} TX &= Y \\ T &= YX^+ \end{aligned} \quad (2)$$

T will be more accurate if X and Y are synthesized appropriately such that their columns are populated by the most diverse (semantically unrelated) word, from the corresponding embedding spaces. Here, we apply the concept of semantics between words as a linear combination of semantics of the other words. The projection TX is the target embedding space. The linear operator is easy to compute as it does not have any iterative training. T is an $m \times m$ square matrix, where m is the embedding dimension and X is $m \times n$, where m is the word embedding dimension and n is the bilingual vocabulary / dictionary size.

5.2. Multi-Layer Perceptron

Although linear mapping seems amicable, it may be inadequate for distinct language pairs, like English–Tamil. Attainment of better accuracy with disparate language pairs calls for non-linear projections. In addition, as the vocabulary size increases, it exacerbates the computation complexity of the transformation matrix. Linear mapping is not a scalable approach, although [28] presented it using Stochastic Gradient Descent. This stimulates the search for more apt ML approaches besides linear transformation. Initially, we considered Multi-Layer Perceptron (MLP), which is a basic deep-learning neural network (DNN) with fully connected layers. MLP will attempt to learn non-linear vector valued function f such that $f(x) = y$, where x is the English vector and y is the Tamil vector. The loss function has *cosine proximity* between the predicted ($\hat{y}$) and actual monolingual vectors (y) from the Tamil embedding space. The training phase typically implies minimization of a loss function over target vectors. Cosine proximity loss is usually negative (making proximity as high as possible by minimizing a negative scalar). Equation (3) presents the inverse cosine proximity, where the cosine proximity (K) between predicted ($\hat{y}$) and actual monolingual vectors (y) is maximized.

$$\begin{aligned} \cos\_prox_{inv} &= 1 - K(\hat{y}, y) \\ K(\hat{y}, y) &= \frac{\hat{y}^T \times y}{\|\hat{y}\| \times \|y\|} \end{aligned} \quad (3)$$

This was implemented in the Keras library in Python. Figure 2 shows the basic architectural pipeline for MLP, enabled by usage of certain hyper-parameter values. The MLP architecture is constituted of three dense layers: Rectified Linear Unit (ReLU) as its activation layer, and the dense layer followed by the Dropout Layer, to avert over-fitting in training. The cosine proximity is used as the loss function and the RMSprop as the optimizer.

Figure 2. Architecture of MLP.

5.3. One Dimensional—Convolutional Neural Network

Groundbreaking results for computer vision with the advent of DNNs, like AlexNet and ResNet, accentuated Convolutional Neural Networks (CNN) [46,47]. In NLP, instead of convolving over pixels, convolutional filters were applied and pooled sequentially, over individual or groups of word vectors [48]. MLPs work well by capturing the non-linearities of the transfer function. MLPs, being fully connected (dense network), are unable to ignore noisy aspects of the data, whereas CNN is ideally suited for disregarding noise and filtering in the aspects that are most prominent in the data. Transfer function learned by CNN can be more focused without loss of generality. Figure 3 explains the architectural pipeline of the CNN network employed in the study reported in this paper.

Figure 3. Architecture of CNN

The network has three layers, a CNN layer followed by a Max Pooling layer and a Dense layer; each layer uses ReLU as an activation function. The CNN filter defines the number of features to be learned; this investigation used twenty-two filters of kernel size seven. Cosine proximity and RMSprop were used as the loss function and optimizer, respectively.

6. Comparison of Various Monolingual Word Embedding Models

The three most popular generic embedding algorithms in NLP are Word2Vec, GloVe, and FastText. Word2Vec [28] remembers the forward–backward context of a word. It is designed as a pair of feed-forward neural networks—a Continuous Bag Of Words (CBOW) model and the skip-gram model. GloVe [7] is another popular count-based model, trained on counts of global co-occurrent words and minimization of least-square error to produce a word vector representation.

FastText [29] is an open-source library, designed by the Facebook research team for learning efficient word representations and classification of text/documents. As FastText treats each word as character n-grams, the word embedding generated for rare words can come in handy, as character n-grams are shared with other words. In Word2Vec and GloVe, a rare word that occurs less than 10 times and has fewer neighbors has poor embedding quality compared to the vector of a word that occurs more than 100 times. Both of the algorithms fail to provide good vector representations for OOV and compound words. For instance, if a compound word “earache” is not in vocabulary, Word2Vec and GloVe may return either a zero vector or a random numbered vector with low magnitude. However, FastText can produce a vector whose magnitude is closer either to the vector ‘ear’ or ‘ache’ by breaking the word ‘earache’ into chunks.

FastText training is computationally heavy compared to Word2Vec and GloVe. Since the training is at character n-gram level, it takes longer to generate FastText embedding. As the corpus size grows, the memory requirement grows too; the number of n-grams that are hashed into the same n-gram bucket would grow.

7. Evaluation Tasks

Word embeddings have no measurable ground truths to verify their semantic properties [49]. Evaluation of these vector representations requires conception of precise linguistic use cases. Word vectors translate semantic relationships to spatial distances. When two words are semantically related, their respective embeddings are expected to have high similarity measures. Assessment of word embeddings, using a crowd-sourced scoring scheme, is detailed in [50].

Here, the original monolingual embedding is treated as ground truth for the evaluation of TFGE. Embeddings can be evaluated quantitatively with respect to original ones and qualitatively by visualization, by plotting them on two-dimensional graphs that show the position of words in relation to other words, followed by visualization using the t-distributed Stochastic Neighbor Embedding (t-SNE) method [51]. Visual verification of the quality of the embeddings facilitates precise estimation of their usability. Nevertheless, one has to have a basic idea of the semantic relation of a target language space prior to visual inspection. In this case, this gap shall be filled by comprehensive explanation of the t-SNE plots.

7.1. Quantitative Evaluation

Instead of evaluating vectors in an absolute sense, the requirement is to compare the original target vectors to the generated target vectors. Semantic relationships between words translate to neighborhood distances between word vectors. The entire semantics is captured in the topology of position vectors in word embeddings. Neighborhood analysis is a direct measure of the information captured by the generated vectors [40].

Two complementary measures were used: one that deals with ontologically related words such as synonyms and antonyms [40], and the global neighborhood, which is the bearing of the current word in relation to the rest of the language representation (a set of prime words from the corpus). Both the approaches are topological measures; one is specific and the other is general. Both measures have almost an equal amount of semantic information for a word.

7.1.1. Pairwise Accuracy of Similar Words

This evaluation model assesses the efficacy of the obtained embedding in reciprocating the semantic relatedness among the word pairs with respect to the original embedding. For assessment, semantically related word pairs are collected based on known linguistic relations (synonyms, antonyms, meronyms, and any kind of etymological relationship). A target language version of word pairs similar to the Simlex999 [52] English word pair dataset was developed in a prior study. A sample of word pairs used for computing pairwise cosine accuracy of three languages is listed in Table 2. In Section 7.1, we put to use Hindi and Chinese word pairs as shown in Table 2 to evaluate the transferability of trained DNNs and linear mapping. Figure 4 explains the process flow pipeline for computing pairwise cosine accuracy (Paccuracy). The cosine of word pairs in Figure 4 is calculated as given in Equation (3), where K is the cosine distance between any two vectors.

Table 2. A sample word pairs list for computing pairwise cosine accuracy for all three evaluating languages.

Tamil	Hindi	Chinese
mother–motherhood	fruit–Chinese	English–Chinese
attracted–attractive	grand–majestic	representative–administrative
attractive–ugly	charming–wonderful	enter–entry
liver–cell	liver–cell	political–democracy,
critised–uninomous	cultivation–paddy	time–period

Figure 4. Pipeline for computing pairwise accuracy for similar words.

7.1.2. Neighborhood Accuracy

While pairwise cosine accuracy measures the retention of known linguistic relations as it was in the original embedding, global neighborhood accuracy (N.accuracy) measures the retention of the overall topology with respect to the original embedding. Clarifying further, the neighborhood measures the distance between a word and every other word in the vocabulary for the generated embedding and compares it with the same measure in the original embedding. Figure 5 shows the pipeline of computing the neighborhood accuracy. Empirical observations show that the neighborhood follows at least some linear relationship between the word pairs. This is discussed further in Section 8. The similarity metric between two words is cosine distance, K , as given in Equation (3).

Figure 5. Pipeline for computing cosine neighborhood accuracy.

7.2. Qualitative Evaluation

Word embedding maps semantic relations between words as spatial distances in the vector space. Since word vectors are of very high dimensionality (generally 300 dimensional vectors), ideally, it is an insurmountable challenge for most humans to visualize the vectors or their relations. This calls for employment of a dimensionality reduction algorithm that will help enervate them to visualizable dimensions (two- or three-dimensional). Application of a normal Principle Component Analysis (PCA) for such drastic reductions in

dimensionality is ineffective. An alternative to this is to resort to t-SNE [51], which affords minimal loss transformation. t-SNE is an ideal tool for visualization of word vectors and generation of word clouds. In this context, t-SNE is used to visualize the semantic relations between word pairs in the evaluation dataset. Figure 6 shows the t-SNE visualization of vectors over word pairs, selected for assessment of Tamil embedding from Table 2. There are three t-SNE plots

- Embedding generated by original embedding algorithm
- Embedding generated by trained MLP and
- Embedding generated by trained CNN networks

Figure 6b,c displays plots of the generated embeddings; the distance between word pairs either have changed or have been retained from the original embedding. This is a direct qualitative measure of neighborhood accuracy. A fair evaluation of such accuracy calls for clear perception of the semantic relations of word pairs. For example, the relation between the word pairs “attracted”–“attractive” and “mother”–“motherhood” are maintained in both MLP and CNN. The distance between the pairs may have changed, but the nature of relationships between similar word pairs is preserved.

Figure 6. t-SNE visualization of transfer-learned Tamil vectors (MLP& CNN) and original vectors generated by Word2Vec. (a) Original Embeddings; (b) Embeddings generated by MLP; (c) Embeddings generated by CNN.

7.3. Evaluation Based on Usability Tests

Standard NLP tasks were employed that utilize the embeddings to generate results. The accuracies of these results can be quantitatively assessed as they are mainly ML, data-driven tasks. However, a successful task implies that a set of good-quality vectors was used versus unsuccessful tasks, which point to poor quality. Hence, the assessment of the vectors is qualitative. To gauge TFGE vectors, the following three NLP use cases were employed: Text Summarization, Part-of-Speech (POS) Tagging, and Bilingual Dictionary Induction (BDI).

7.3.1. Text Summarization

For the text summarization task, an extractive SVD-based summarization algorithm was used [53]. Every sentence in the text had a pairwise score with every other sentence. Word embeddings were used to perform sentence-level alignments and derive sentence-to-sentence pairwise scores. The summary from texts of varying sizes with the original embeddings, and TFGE are observed separately.

The generated summary with original embeddings is taken as a benchmark summary, and that generated by TFGE is considered a referring summary. For example, if the original summary consists of sentence indexes 1, 8, 10, and 18 and the referring summary has 1, 7, 10, and 18, then the count of differing sentences is 1 (sentence # 7 is extracted in the referee summary instead of sentence # 8 in the benchmark summary). Differing sentences in the referring summary are counted and used as an entropy measure—the higher the entropy, the poorer the quality.

7.3.2. Part-of-Speech (POS) Tagging

Part-of-Speech Tagging forms the part of all downstream tasks like Name Entity Recognition (NER), Semantic Role Labelling (SRL), Word Sense Disambiguation (WSD), Chunking, Machine Translation (MT), and Parsing (syntax analysis). POS tags include nouns, verbs, adjectives, adverbs, and their subcategories. POS tagging is a string-labeling exercise where the sentence is fed in as an input, and each word in the sentence is labeled with the name of a tag indicating the POS category. The standard Pen Tagset was employed for tagging POSs in Tamil, which is the target embedding space. Another CNN was trained using an annotated Tamil corpus provided with the cEnTam dataset [30].

Figure 7 shows the pipeline architecture of a POS Tagger using a CNN network. The class-wise testing accuracy of each of the categories and the average accuracy of prediction over all POS classes are measured in the prediction of the right POS tag for each word. This accuracy is used as a measure of success. Typically, similar to the text summarization tasks, the CNN trained on the original Tamil embedding is deemed as a benchmark, and the one trained on TFGE is used as the referee.

Figure 7. Pipeline of POS Tagger trained using CNN.

7.3.3. Bilingual Dictionary Induction

BDI entails the task of guessing the source word for a given foreign word. Traditionally, there are various methods to induct dictionaries, especially the bilingual embedding method [25]. Recently, [54] used topological metrics to induct BDI, but the authors only used linear mapping and global neighborhood measures (Iterative Mapping). We have, however, observed that BDI can be achieved using TFGEs [55]. This paper uses cross-lingual embeddings, aka TFGEs, to study an ML model and reverse lookup of the source embedding. This method achieves high accuracy.

8. Results and Discussion

This section presents and discusses all the results of evaluation carried out on TFGEs apropos original embeddings. DNNs were trained using cosine proximity as the loss function. With minimal data (dictionary size of 10,000+ words noted in Table 1) and diverse vector spaces, it would be impetuous to expect the networks to yield high testing/training (model) accuracies. This can be improved if we have more data to tune the hyper-parameter of the network. Resource-constrained target language impeded trainability of the proffered model and supervised by the unavailability of a large dataset. Even in the best-case

scenario, the highest TFGE model's testing accuracy achieved was limited to 62% with any of the transfer learning models. In spite of moderately high training error (MSE as high as (30–40)%), the model still yielded fairly good embedding. Figure 8a,b shows the 1D-CNN TFGE model's training accuracy/loss vs. epoch. In addition, the TFGE model afforded all the desirable semantic properties that were verified and evaluated, using the quantitative (topological) and qualitative metrics, mentioned in Section 7.

Figure 8. Graphical representation of Training Accuracy/Loss vs. Epochs for 1D-CNN model of TFGE. (a) Training Accuracy vs. Epochs; (b) Training Loss vs. Epochs.

8.1. Quantitative Evaluation Results

We used three transfer function models, Linear Mapping (LM), MLP, and CNN and a bilingual model, BilBOWA. Each model was trained on the three respective word embedding algorithms: Word2Vec (W), GloVe (G), and FastText (F). As a composite, twelve experiments were performed—combining four models across three different word embeddings ($4 \times 3 = 12$). The experiments were designated using the model name subscripting the embedding algorithm name. For example, LM_W refers to the Linear Mapping (LM) model trained on Word2Vec (W) algorithm. Table 3, enlists pairwise cosine accuracy (P.accuracy) and neighborhood accuracy (N.accuracy) for all of the models trained across various algorithms.

In Table 3, global neighborhood accuracy (N.accuracy) was cross-validated on random validation datasets of 300 words over 50 epochs, whereas pair-wise accuracy, P.accuracy, was computed from the pairwise data retrieved from [50]. It is a set of 146 word pairs, sampled and translated from SimLex-999 [52]. Among the topological accuracies, P.accuracy is the most difficult to achieve compared to N.accuracy, as the former are linguistically verifiable and not linear, like the latter. This explains why linear mapping was able to achieve better scores in N.accuracy, rather than P.accuracy.

In order to measure the transferability, another experiment was undertaken that used Hindi and Chinese as target languages. Hindi, semantically distant from Tamil, is another Indian language that has readily available pretrained vectors. Chinese, quite distinct from Tamil, also has readily available rich resources. We hypothesize that if we are able to find Hindi and Chinese monolingual vectors trained with the same Word2Vec algorithm that was used to train the Tamil embeddings, they may reveal semantic alignment. As the CNN model gained better accuracy than MLP, CNN was used to validate the property of transferability.

Initially, Hindi and Chinese vectors were used only to evaluate the CNN_W —generated vectors for Tamil. Surprisingly, the model gave a good pairwise accuracy of 76.95% for Hindi and 70.52% for Chinese. In addition, retraining the models for Hindi and Chinese embeddings yielded considerably augmented accuracies, with Hindi reaching 83.02% and Chinese reaching 87.59%. Table 4 depicts the accuracy measurements for Hindi and Chinese, with and without retraining, on linear mapping and CNN models.

Table 3. Neighborhood and Pairwise Accuracy (N.accuracy and P.accuracy) of English–Tamil TFGE trained on various models. LM, MLP, and CNN refers to the transfer function learning models, Linear Mapping, Multi Layer Perceptron, and Convolutional Neural Network, respectively. The subscripts W, G and F refer to the vector training algorithms, Word2Vec, GloVe, and FastText, respectively.

Transfer Functions	N.accuracy	P.accuracy
LM_W	81.70	12.13
LM_G	76.40	20.64
LM_F	77.28	11.96
MLP_W	80.56	82.68
MLP_G	87.36	89.70
MLP_F	89.72	90.67
CNN_W	85.53	86.34
CNN_G	89.95	91.38
CNN_F	80.33	92.68
$BilBOWA_W$	78.36	73.12
$BilBOWA_G$	68.97	71.38
$BilBOWA_F$	77.28	73.61

Table 4. Neighbourhood and Pairwise Accuracy (N.accuracy and P.accuracy) of Asian TFGE Embeddings trained by various models.

Language	Transfer Function	Model Type	P.accuracy	N.accuracy
Hindi	Linear Mapping	Eng-Tam Matrix	12.91	77.16
		Re-computed Matrix	17.34	88.47
	CNN	Eng-Tam Network	76.95	72.15
		Re-trained Network	83.02	80.40
Chinese	Linear Mapping	Eng-Tam Matrix	12.08	75.13
		Re-computed Matrix	17.28	90.23
	CNN	Eng-Tam Network	70.52	71.32
		Re-trained Network	87.59	84.21

One of the priorities of transfer learning embedding is data efficiency. This implies that the transfer function can be trained to obtain TFGEs with dictionaries that contain no more than 1000 words. All the TFGE models were trained with this premise in mind. The resource constraint was simulated by reduction of the size of the dictionary and training the model on varying data instances. As the linear mapping model achieved good neighborhood accuracy but failed miserably on pairwise accuracy, the data efficiency study was restricted to deep learning models. This study was limited to pairwise accuracies as they are based on linguistic ground truth (for every embedding model, the formation of word pairs is supervised by known linguistic relationships). Table 5 shows the pairwise accuracy of deep-transfer-learned Tamil and English vectors on MLP and CNN networks, for every embedding model. Albeit computationally intricate, P.accuracy was chosen in preference to N.accuracy, as it is linguistically verifiable. The loss functions for the same are shown in Figures 9 and 10.

Table 5. Pairwise Accuracy of MLP and CNN Network over various dictionary lengths. P.accuracy was chosen because it is more difficult to achieve than N.accuracy and is linguistically verifiable.

#. of Word Pairs	MLP			CNN		
	Word2Vec	GloVe	FastText	Word2Vec	GloVe	FastText
1000	80.39	88.88	85.14	82.42	90.99	90.94
2000	81.99	87.34	86.71	83.47	91.21	91.05
3000	81.79	87.28	89.69	83.66	91.58	90.99
5000	82.27	88.20	89.82	83.98	92.55	92.36
7000	82.77	88.10	89.57	84.49	89.75	92.21
8000	82.35	87.70	89.77	85.13	90.93	92.48
10,000	82.68	89.70	90.67	86.34	91.38	92.68

Figure 9. Graphical representation of Pairwise Accuracy of MLP Network over various dictionary lengths. The X-axis represents the dictionary sizes, and the Y-axis represents the accuracy.

Figure 10. Graphical representation of Pairwise Accuracy of CNN Network over various dictionary lengths. The X-axis represents the dictionary sizes, and the Y-axis represents the accuracy.

Neighborhood accuracy was also measured across various POS categories. The vocabulary was divided into the top four categories: nouns, verbs, adverbs, and adjectives. Cosine neighborhood accuracies were measured individually, within each category, as showcased in Table 6.

Table 6. Category-based Neighbourhood Percentage Accuracy of Tamil Embeddings.

Category	Embedding Models		
	Word2Vec	GloVe	FastText
Nouns	80.13	91.79	80.97
Verbs	82.19	91.63	77.85
Adjectives	80.34	89.44	78.77
Adverbs	85.97	90.42	80.40

8.2. TSNE Plots and Qualitative Interpretation

t-SNE can be conceived as a two-dimensional projection of the original high dimensional embedding space. The position of the words in the original embedding space is correspondingly brought down to lower dimensions (two dimensions in this case), such that topologically closer (neighboring) words will remain so, even in the lower dimension. This affords an opportunity to perceive the behaviors of different word pairs when the vectors are transformed by TFGE models. The word pairs list used for the t-SNE plot is given in Table 2. The word pairs in Table 2 are semantically similar words in the target language. Semantically similar words in Tamil are chosen to understand semantic preserving property of cross-lingual embedding visually. The t-SNE plot for Word2Vec is presented in Figure 6. Here, Figures 11 and 12 do the same, for GloVe and FastText, respectively. The semantics between the related word pairs, “attracted” and “attractive”, “mother” and “motherhood”, are preserved in the TFGE as well. Figures depict only a small portion of the actual embedding in order to demonstrate the qualitative inferences. t-SNE is computed by a non-convex optimization method, which implies that positions of the embeddings in the t-SNE plot may not be the same every time, when recomputed.

Figure 11. t-SNE visualization of transfer-learned Tamil vectors (MLP and CNN) and original vectors generated by GloVe. (a) Original Embeddings; (b) Embeddings generated by MLP; (c) Embeddings generated by CNN.

Figure 12. t-SNE visualization of transfer learned Tamil vectors (MLP and CNN) and original vectors generated by FastText. (a) Original Embeddings; (b) Embeddings generated by MLP; (c) Embeddings generated by CNN.

9. Usability Evaluation Results

Three NLP use cases were employed to appraise the usability of TFGE vectors introduced in Section 7.3. Discussions in this subsection address their input and results, followed by qualitative assessment of their success. Text summarization tasks involved four textual input files, with 20, 50, 80, and 110 sentences in each. This extractive summarization algorithm was implemented in Scala and runs on the Apache Spark® Framework. This is a data-intensive process. A text file with twenty sentences is transformed into a Cartesian product, pairing each sentence with every other sentence. The size of the Cartesian product is 20×20 , which is used to align the sentence-pairs on word similarity. The algorithm uses SVD to rank sentences in the text, according to the intensities of their topics. It extracts top-ranking sentences and constructs a summary. The maximum size of the summary is the hyper-parameter; a fraction of the total number of sentences. All original embeddings, Word2Vec, GloVe, and FastText, extract the same sentence for summary in all four instances. This shows how the semantics are captured in all of these three embeddings. When the experiment was repeated with the TFGE vector, the same output was observed with TFGE from all the models ($CNN_{W,G,F}$), as may be seen in Table 7. The number of differing sentences was zero for all cases. This qualitatively shows that the semantics, captured by the original embedding, are equivalent to those captured by TFGE; the transfer of relative semantics from original embeddings to TFGEs was empirically asserted. We provided it with randomly generated embeddings to ensure neutrality of the summarization algorithm. Table 8 depicts the anticipated results as to the count of differing sentences compared to the benchmark summary depicted in Table 7. For a summary size of twenty, the random summary differed in 17 sentences from the benchmark summary.

Table 7. Text summarization results of learned embedding models.

Size of Text Document	Size of the Summary		# of Different Sentences
	Original	Transfer Learned Through CNN	
20	2	2	0
50	6	6	0
80	14	14	0
110	20	20	0

Table 8. Text summarization results of random vectors.

Size of Text Document	Size of the Summary		# of Different Sentences
	Original	Random Vectors	
20	2	2	2
50	6	6	6
80	14	14	12
110	20	20	17

The POS Tagger, depicted in Figure 7, takes input from an annotated Tamil corpus of the cEnTam dataset to predict the tag label. Even though the network predicts thirty tags, the accuracies of only four classes—noun, verb, adverb, and adjective—are compared. Table 9 tabulates the class-wise accuracy of these four POS categories trained with different embeddings. The first column provides the accuracies; when trained with random vectors, as expected, they gave very poor results. The original embedding, however, fared much better with average accuracy over four classes of 70%, 71%, and 81% for Word2Vec, GloVe, and FastText, respectively. Linear Mapping performed even better, with an average of over 82%. TFGE has the highest average accuracy of prediction in this whole exercise; they even outperformed the original embedding, on which they were transfer-learned from. They are trained on a very small monolingual target corpus compared to a pre-trained source (<100 billion words). However, TFGE was generated from a transfer process, where the input comes from an embedding space trained with a billion-word corpus (pre-trained embeddings). Some ineffable properties of the pre-trained embeddings may have been concurrently transferred to TFGEs, which outfits them with an upper hand in the POS tagging task. TFGE vectors dominated this usability test.

Table 9. Category-based accuracy for Deep Learned POS Tagger trained on different embedding models and Linear Mapping on Word2Vec.

Category	Embedding Models					
	Random Vectors	Word2Vec	GloVe	FastText	TFGE	Linear Mapping
Noun	0.53	0.65	0.69	0.70	0.86	0.84
Verb	0.52	0.75	0.71	0.86	0.84	0.74
Adverb	0.57	0.65	0.65	0.89	0.91	0.85
Adjectives	0.69	0.74	0.80	0.81	0.86	0.88

Our next use case task is BDI. The authors of [25] used their bilingual embedding to perform BDI on three separate language pairs. We conducted a reverse lookup of the transferred learned vector using the original monolingual embedding to elicit a word translation of the source word. Provided a series of source and target words $\langle w_{s_i}, w_{t_i} \rangle$ as well as their corresponding embeddings (original monolingual embeddings) $\langle wv_{s_i}, wv_{t_i} \rangle$ and transfer learned target embedding $\langle wv_{t_i}^* \rangle$. The correct target word w_{t_i} is identified for each query source word w_{s_i} by finding the target embedding wv_{t_i} that is the closest neighbor to the transition learned/projected target word embedding $wv_{t_i}^*$, where cosine similarity is computed as a measure between the embeddings. The reported highest accuracy is 68.9 over their dictionary size of 1000 words. Reference [55] describes a shared task system, where bilingual pairs are inducted over German–English (de–en) and Tamil–English (ta–en). The study employed cross-lingual (TFGE) embeddings to induct a bilingual dictionary, in both cases. Table 10 summarizes BDI accuracy, derived using TFGE [55]; TFGEs performed very well in the BDI task.

Table 10. Accuracy of German–English and Tamil–English BDI system as reported in [55].

Models	Language Pairs	
	de–en	ta–en
Linear Mapping	73.01	76.05
MLP	80.67	85.52
CNN	85.16	90.33

Empirical Observations on the Properties of Different Embeddings

The t-SNE plots for all the transfer-learned embeddings and the original embeddings were computed using the three reputed embedding algorithms—Word2Vec, GloVe, and FastText [7,28,29]. A close review of key observations inferred from Table 5 and Figures 6, 11, and 12 are summarized hereunder as empirical characteristics.

The Word2Vec algorithm is focused on preserving neighborhood information of each embedding. By far, Word2Vec can best capture and preserve the relational semantics of etymologically proximal words; it preserves ontology in the form of neighborhoods. Neighborhood can only be perceived with availability of profuse data. Table 5 empirically verifies this fact as the accuracy of Word2Vec consistently improves with more data (word pairs) provided. Our cosine neighborhood accuracy is not a direct measure of this neighborhood; this can only be measured through clustering.

Global Vectors (GloVe) embedding algorithm, on the other hand, is designed on co-occurrence relationships. The co-occurrence relation of cross-lingual dictionary words is almost the same, with little room for improvement, when more data are pumped in. In Table 5, the accuracy of transfer-learned vectors using GloVe simply oscillates between some range for a steady increase in the number of word pairs. Since accuracy is a measure using pairwise relation (relative semantics), GloVe has a natural advantage as it accounts for co-occurrence relations as well.

FastText shows behavior similar to, but more accurate than, Word2Vec. Fast Text's ability to account for substring n-grams affords a clear advantage in the case of extremely agglutinative Tamil.

10. Discussions

This paper's empirical studies entailed trained models on word sets as low as 1000 for Tamil, the target language. Even with such a low word count and corresponding vectors, a cross-lingual transfer learning network could be devised that generated reasonably good quality vectors from English words for unknown Tamil words. The generated vectors were assessed over an unseen set of word pairs. The cosine distance obtained over the pair of words using original embeddings and the cosine distance obtained using generated embeddings were compared. The resultant error was used for calculation of accuracy. The pairwise accuracy is expressed as the complement of root-mean-square percentage error (RMSPE). However, the networks were trained on absolute prediction error with the known monolingual target vectors (of Tamil in this case). The trained model is further validated with real NLP tasks for verification of the propriety of the generated embeddings. Summaries from the same text document were generated with the algorithm presented in [53]. The summary generated with the original embeddings and the transfer learned embeddings were identical for all embedding types. The generated embeddings were tested on a POS classification network for BDI over German–English and Tamil–English pairs. In all these cases, the generated embeddings (TFGE) were as effective as the original embeddings. This conclusively proves the aptness of the learned vectors.

11. Conclusions

The primary objective of this investigation was to devise an efficient transfer learning scheme for attainment of cross-lingual word embeddings obviating the need for large monolingual and bilingual corpora. Multiple experiments were conducted, employing

different methodologies, to attain target word vectors for the English–Tamil bilingual pair. Tamil, a popular Asian language, is linguistically similar to many other south Asian languages. We created sufficient corpora for (monolingual and bilingual) Tamil for the evaluation of the proffered methodologies and empirical outcomes. Furthermore, pre-trained Hindi and Chinese embeddings were marshalled to validate the transfer learning model.

Target word vectors were successfully generated with a minimal corpus (monolingual) size of 5000 words, approximately the size of a textbook. Such a modest-sized corpus was apposite for achievement of useful word vectors—89% using GloVe vectors and at least an accuracy of 80% using Word2Vec—with proven cross-validated topological (pairwise and neighborhood) accuracy. The cosines were scaled between $[0, 2]$, and the error was also computed in the same interval. The accuracies obtained were compared with the standard bilingual embedding algorithm BilBOWA [26], which uses a sentence-aligned parallel and comparable corpora. It also considers a minimal word-aligned model to improve the accuracy of the target vectors. This paper’s investigators are convinced that a bilingual model is a compromise between the languages’ semantics. The ineluctable semantic gap between the languages is traded off in the interest of a common vector space. The proffered model is a cross-lingual transfer learning model that takes a source language vector and projects it to the target space, maintaining the semantic integrity of the target language. The deep learning networks essentially learn the semantic gap between the languages and incorporates the source vector into the target vector space. In contrast, the cross-lingual model requires only monolingual corpora in both the languages with a dictionary.

As a reconfirmation of the submitted approach, pre-trained Hindi and Chinese embeddings (Word2Vec) were piped through the tendered model. The model that was trained on Tamil as the target language yielded accuracies of 77% with Hindi and 70% with Chinese. In their own ways, the Hindi and Chinese languages are very distinct from Tamil, as empirically observed from their accuracies. When the models were re-trained with respective languages using a word set of 1000 words, an accuracy of 83% was reported with Hindi, versus an accuracy of 88% observed with Chinese. These findings corroborate that the method put forth is language-independent. We are optimistic that the word embedding strategy will work seamlessly on syntactically similar target languages, foreclosing the prerequisite of re-training on the second target language.

12. Future Work

In the case of other resource-indigent languages, this proposed robust model can generate word embeddings by creation of a minimal bilingual dictionary and just enough monolingual corpus. Furthermore, the paradigm can be applied to other languages with semantics similar to Tamil. Morphology is an important phenomenon that influences word embeddings. Word vectors augmented by a piece of separate morphology information will surely improve in quality. The accuracy computation used in this paper partially reflects the neighborhood preservation characteristics of any embeddings. Nevertheless, this may admittedly not be an accurate measure of the neighborhood; an accurate measure will be to compute a relative clustering coefficient for a set of words that are ontologically related to the word of interest.

Author Contributions: Conceptualization, S.J. and V.K.M.; methodology, S.J. and V.K.M.; software, S.J.; validation, S.J. and V.K.M.; formal analysis, S.J. and V.K.M.; investigation, S.J. and V.K.M.; resources, S.J.; data curation, S.J.; writing—original draft preparation, S.J.; writing—review and editing, V.K.M. and A.W.; supervision, V.K.M., R.S. and S.K.; project administration, V.K.M. and S.K.; funding acquisition, S.J., V.K.M. and A.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Torabi Asr, F.; Zinkov, R.; Jones, M. Querying Word Embeddings for Similarity and Relatedness. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2018; pp. 675–684. [\[CrossRef\]](#)
2. Billah Nagoudi, E.M.; Ferrero, J.; Schwab, D.; Cherroun, H. Word Embedding-Based Approaches for Measuring Semantic Similarity of Arabic-English Sentences. In *Proceedings of the 6th International Conference on Arabic Language Processing*, Fez, Morocco, 11–12 October 2017.
3. Deerwester, S.; Dumais, S.T.; Furnas, G.W.; Landauer, T.K.; Harshman, R. Indexing by latent semantic analysis. *J. Am. Soc. Inf. Sci.* **1990**, *41*, 391–407. [\[CrossRef\]](#)
4. Fellbaum, C. *WordNet: An Electronic Lexical Database*; Language, Speech, and Communication; MIT Press: Cambridge, MA, USA, 1998.
5. Bengio, Y.; Ducharme, R.; Vincent, P.; Janvin, C. A Neural Probabilistic Language Model. *J. Mach. Learn. Res.* **2003**, *3*, 1137–1155.
6. Collobert, R.; Weston, J.; Bottou, L.; Karlen, M.; Kavukcuoglu, K.; Kuksa, P.P. Natural Language Processing (almost) from Scratch. *arXiv* **2011**, arXiv:1103.0398.
7. Pennington, J.; Socher, R.; Manning, C.D. Glove: Global Vectors for Word Representation. In *Proceedings of the EMNLP, Doha, Qatar, 25–29 October 2014*; Volume 14, pp. 1532–1543.
8. Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.; Dean, J. Distributed Representations of Words and Phrases and their Compositionality. *arXiv* **2013**, arXiv:1310.4546.
9. Wang, S.; Zhou, W.; Jiang, C. A survey of word embeddings based on deep learning. *Computing* **2020**, *102*, 717–740. [\[CrossRef\]](#)
10. Treviso, M.V.; Shulby, C.D.; Aluísio, S.M. Evaluating Word Embeddings for Sentence Boundary Detection in Speech Transcripts. *arXiv* **2017**, arXiv:1708.04704.
11. Bansal, M.; Gimpel, K.; Livescu, K. Tailoring Continuous Word Representations for Dependency Parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2014; pp. 809–815.
12. Guo, J.; Che, W.; Wang, H.; Liu, T. Revisiting Embedding Features for Simple Semi-supervised Learning. In *Proceedings of the EMNLP, Doha, Qatar, 25–29 October 2014*.
13. Soliman, A.B.; Eissa, K.; El-Beltagy, S.R. AraVec: A set of Arabic Word Embedding Models for use in Arabic NLP. *Procedia Comput. Sci.* **2017**, *117*, 256–265. [\[CrossRef\]](#)
14. Xu, J.; Liu, J.; Zhang, L.; Li, Z.; Chen, H. Improve Chinese Word Embeddings by Exploiting Internal Structure. In *Proceedings of the HLT-NAACL, San Diego, CA, USA, 17 June 2016*.
15. Upadhyay, S.; Chang, K.; Taddy, M.; Kalai, A.T.; Zou, J.Y. Beyond Bilingual: Multi-sense Word Embeddings using Multilingual Context. *arXiv* **2017**, arXiv:1706.08160.
16. Wolk, K. Contemporary Polish Language Model (Version 2) Using Big Data and Sub-Word Approach. In *Proceedings of the Interspeech 2020, 21st Annual Conference of the International Speech Communication Association, Virtual Event, Shanghai, China, 25–29 October 2020*.
17. Bhattacharya, P.; Goyal, P.; Sarkar, S. Using Word Embeddings for Query Translation for Hindi to English Cross Language Information Retrieval. *arXiv* **2016**, arXiv:1608.01561.
18. Devi, G.R.; Veena, P.; Kumar, M.A.; Soman, K. Entity Extraction for Malayalam Social Media Text Using Structured Skip-gram Based Embedding Features from Unlabeled Data. *Procedia Comput. Sci.* **2016**, *93*, 547–553. [\[CrossRef\]](#)
19. Devi, G.R.; Veena, P.; Kumar, M.A.; Soman, K. *AMRITA_CEN@FIRE 2016: Code-Mix Entity Extraction for Hindi-English and Tamil-English Tweets*; Indian Statistical Institute: Kolkata, India, 2016.
20. Ajay, S.G.; Srikanth, M.; Kumar, M.A.; Soman, K.P. Word Embedding Models for Finding Semantic Relationship between Words in Tamil Language. *Indian J. Sci. Technol.* **2016**, *9*, 1–5. [\[CrossRef\]](#)
21. Yin, Z.; Shen, Y. On the Dimensionality of Word Embedding. In *Advances in Neural Information Processing Systems 31*; Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R., Eds.; Curran Associates, Inc.: Dutchess County, NY, USA, 2018; pp. 895–906.
22. Li, B.; Drozd, A.; Guo, Y.; Liu, T.; Matsuoka, S.; Du, X. Scaling Word2Vec on Big Corpus. *Data Sci. Eng.* **2019**, *4*, 157–175. [\[CrossRef\]](#)
23. Chandar, A.P.S.; Lauly, S.; Larochelle, H.; Khapra, M.M.; Ravindran, B.; Raykar, V.C.; Saha, A. An Autoencoder Approach to Learning Bilingual Word Representations. *arXiv* **2014**, arXiv:1402.1454.
24. Hermann, K.M.; Blunsom, P. A Simple Model for Learning Multilingual Compositional Semantics. *arXiv* **2013**, arXiv:1312.6173.
25. Vulić, I.; Moens, M.F. Bilingual Word Embeddings from Non-Parallel Document-Aligned Data Applied to Bilingual Lexicon Induction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, Beijing, China, 26–31 July 2015; pp. 719–725. [\[CrossRef\]](#)
26. Gouws, S.; Bengio, Y.; Corrado, G. BilBOWA: Fast Bilingual Distributed Representations without Word Alignments. In *Proceedings of the ICML, Lille, France, 6–11 July 2015*; JMLR Workshop and Conference Proceedings; Volume 37, pp. 748–756.
27. Ruder, S.; Vulić, I.; Søgaard, A. A Survey of Cross-Lingual Word Embedding Models. *J. Artif. Int. Res.* **2019**, *65*, 569–630. [\[CrossRef\]](#)

28. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient Estimation of Word Representations in Vector Space. *arXiv* **2013**, arXiv:1301.3781.
29. Bojanowski, P.; Grave, E.; Joulin, A.; Mikolov, T. Enriching Word Vectors with Subword Information. *Trans. Assoc. Comput. Linguist.* **2017**, *5*, 135–146. [\[CrossRef\]](#)
30. Sanjanasri, J.P.; Premjith, B.; Menon, V.K.; Soman, K.P. cEnTam: Creation and Validation of a New English-Tamil Bilingual Corpus. In Proceedings of the 13th Workshop on Building and Using Comparable Corpora, Marseille, France, 11–16 May 2020.
31. Elsis, M.; Tran, M.Q.; Mahmoud, K.; Lehtonen, M.; Darwish, M.M.F. Deep Learning-Based Industry 4.0 and Internet of Things towards Effective Energy Management for Smart Buildings. *Sensors* **2021**, *21*, 1038. [\[CrossRef\]](#)
32. Elsis, M.; Mahmoud, K.; Lehtonen, M.; Darwish, M.M.F. An Improved Neural Network Algorithm to Efficiently Track Various Trajectories of Robot Manipulator Arms. *IEEE Access* **2021**, *9*, 11911–11920. [\[CrossRef\]](#)
33. Le, Q.V.; Mikolov, T. Distributed Representations of Sentences and Documents. *arXiv* **2014**, arXiv:1405.4053.
34. McCann, B.; Bradbury, J.; Xiong, C.; Socher, R. Learned in Translation: Contextualized Word Vectors. *arXiv* **2017**, arXiv:1708.00107.
35. Peters, M.E.; Neumann, M.; Iyyer, M.; Gardner, M.; Clark, C.; Lee, K.; Zettlemoyer, L. Deep contextualized word representations. *arXiv* **2018**, arXiv:1802.05365.
36. Devlin, J.; Chang, M.; Lee, K.; Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv* **2018**, arXiv:1810.04805.
37. Wolf, T.; Debut, L.; Sanh, V.; Chaumond, J.; Delangue, C.; Moi, A.; Cistac, P.; Rault, T.; Louf, R.; Funtowicz, M.; et al. Transformers: State-of-the-Art Natural Language Processing. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, Online, 16–20 November 2020; Association for Computational Linguistics: Stroudsburg, PA, USA, 2020; pp. 38–45.
38. Sazzed, S. Cross-lingual sentiment classification in low-resource Bengali language. In Proceedings of the Sixth Workshop on Noisy User-generated Text (W-NUT 2020), Online, 19 November 2020; Association for Computational Linguistics: Stroudsburg, PA, USA, 2020; pp. 50–60.
39. Zou, A. Learning Cross-Lingual Word Embeddings for Sentiment Analysis of Microblog Posts. Master's Thesis, Princeton University, Princeton, NJ, USA, 2020.
40. Lastra-Díaz, J.J.; Goikoetxea, J.; Taieb, M.A.H.; García-Serrano, A.M.; Benaouicha, M.; Agirre, E. A reproducible survey on word embeddings and ontology-based methods for word similarity: Linear combinations outperform the state of the art. *Eng. Appl. Artif. Intell.* **2019**, *85*, 645–665. [\[CrossRef\]](#)
41. Yuan, M.; Zhang, M.; Van Durme, B.; Findlater, L.; Boyd-Graber, J. Interactive Refinement of Cross-Lingual Word Embeddings. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), Online, 16–20 November 2020; Association for Computational Linguistics: Stroudsburg, PA, USA, 2020; pp. 5984–5996.
42. Pre-Trained, Asian Embeddings. 2019. Available online: <https://github.com/Kyubyong/wordvectors> (accessed on 10 May 2021).
43. Hindi, Dictionary. 2019. Available online: <https://www.shabdkosh.com/dictionary/english-hindi/> (accessed on 10 May 2021).
44. Chinese, Dictionary. 2019. Available online: <http://www.mandarintools.com> (accessed on 10 May 2021).
45. Mikolov, T.; Le, Q.V.; Sutskever, I. Exploiting Similarities among Languages for Machine Translation. *arXiv* **2013**, arXiv:1309.4168.
46. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems 25*; Pereira, F., Burges, C.J.C., Bottou, L., Weinberger, K.Q., Eds.; Curran Associates, Inc.: Dutchess County, NY, USA, 2012; pp. 1097–1105.
47. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. *arXiv* **2015**, arXiv:1512.03385.
48. Jacovi, A.; Shalom, O.S.; Goldberg, Y. Understanding Convolutional Neural Networks for Text Classification. *arXiv* **2018**, arXiv:1809.08037.
49. Bakarov, A. A Survey of Word Embeddings Evaluation Methods. *arXiv* **2018**, arXiv:1801.09536.
50. Sanjanasri, J.P.; Menon, V.K.; Rajendran, S.; Soman, K.P.; Anand Kumar, M. Intrinsic Evaluation for English-Tamil Bilingual Word Embeddings. In *Intelligent Systems, Technologies and Applications*; Springer: Singapore, 2020; pp. 39–51.
51. van der Maaten, L.; Hinton, G. Visualizing Data using t-SNE. *J. Mach. Learn. Res.* **2008**, *9*, 2579–2605.
52. Hill, F.; Reichart, R.; Korhonen, A. SimLex-999: Evaluating Semantic Models With (Genuine) Similarity Estimation. *Comput. Linguist.* **2015**, *41*, 665–695. [\[CrossRef\]](#)
53. Menon, V.; Maniyil, S.; Harikumar, K.; Soman, K. *Semantic Analysis Using Pairwise Sentence Comparison with Word Embeddings*; Springer International Publishing: Cham, Switzerland, 2018; pp. 268–278. [\[CrossRef\]](#)
54. Aldarmaki, H.; Mohan, M.; Diab, M. Unsupervised Word Mapping Using Structural Similarities in Monolingual Embeddings. *Trans. Assoc. Comput. Linguist.* **2018**, *6*, 185–196. [\[CrossRef\]](#)
55. Sanjanasri, J.P.; Menon, V.K.; Soman, K.P. BUCC2020: Bilingual Dictionary Induction using Cross-lingual Embedding. In Proceedings of the 13th Workshop on Building and Using Comparable Corpora, Marseille, France, 11–16 May 2020.