NEURAL WORD EMBEDDINGS FOR SENTIMENT ANALYSIS

DUYGU ANALİZİ İÇİN SİNİRSEL SÖZCÜK ÖZ YERLEŞİKLERİ

BEHZAD NADERALVOJOUD

PROF. DR. EBRU SEZER

Supervisor

Submitted to

Graduate School of Science and Engineering of Hacettepe University

as a Partial Fulfillment to the Requirements

for the Award of the Degree of Doctor of Philosophy

in Computer Engineering

2020

To my lovely wife and son...

ABSTRACT

NEURAL WORD EMBEDDINGS FOR SENTIMENT ANALYSIS

Behzad NADERALVOJOUD

Doctor of Philosophy,**Computer Engineering Department**

Supervisor: Prof. Dr. Ebru SEZER

July 2020, 100 pages

Most pre-trained word embeddings are achieved from context-based learning algorithms trained over a large text corpus. This leads to learning similar vectors for words that share most of their contexts, while expressing different meanings. Therefore, the complex characteristics of words cannot be fully learned by using such models. One of the natural language processing applications that suffers from this problem is sentiment analysis. In this task, two words with opposite sentiments are not distinguished well by using common pre-trained word embeddings.

This thesis addresses this problem and proposes two empirically effective approaches to learn word embeddings for sentiment analysis. The both approaches exploit sentiment lexicons and take into account the polarity of words in learning word embeddings. While the first approach encodes the sentiment information of words into existing pre-trained word embeddings, the second one builds synthetic sentimental contexts for embedding models along with other semantic contexts.

The word embeddings obtained from both approaches are evaluated on several sentiment classification tasks using Skip-gram and GloVe models. Results show that both approaches improve state-of-the-art results using basic deep learning models over sentiment analysis benchmarks.

Keywords: Word Embedding, Word Semantics, Sentiment Analysis, Deep Learning, Machine Learning

ÖZET

DUYGU ANALİZİ İÇİN SİNİRSEL SÖZCÜK ÖZ YERLEŞİKLERİ

Behzad NADERALVOJOUD

Doktora, **Bilgisayar Mühendisliği Bölümü**

Tez Danışmanı: Prof. Dr. Ebru SEZER

Temmuz 2020, 100 sayfa

Ön-eğitimli kelime özyerleşiklerinin çoğu, büyük bir metin derlemi üzerinde eğitilmiş bağlam tabanlı öğrenme algoritmalarından elde edilmektedir. Bu durum, benzer bağlamlarda sıkça yer alan ancak farklı anlamlar taşıyan kelimeler için benzer vektörlerin öğrenilmesine yol açmaktadır. Bu nedenle, kelimelerin karmaşık özellikleri bu modeller kullanılarak tam olarak öğrenilememektedir. Bu sorundan etkilenen doğal dil işleme uygulamalarından birisi de duygu analizidir. Bu görevde, zıt duygulara sahip iki kelime, ön-eğitimli kelime özyerleşikleri kullanılarak iyi ayırt edilmemektedir.

Bu tez, bu sorunu çözmeyi hedeflemektedir ve çözüm için duygu analizine özel kelime özyerleşikleri öğrenmek amacıyla ampirik olarak etkili iki yaklaşım önermektedir. Her iki yaklaşım da duygu sözlüklerini kullanmaktadır ve kelime özyerleşiklerini öğrenirken kelimelerin duygu eğilimlerini dikkate almaktadır. İlk yaklaşım, kelimelerin duygu bilgisini mevcut ön-eğitimli kelime özyerleşiklerine kodlarken, ikincisi diğer anlamsal bağlamlarla birlikte modelleri eğitmek için sentetik duygusal bağlamlar oluşturmaktadır.

Her iki yaklaşımdan elde edilen kelime özyerleşikleri, Skip-gram ve GloVe modelleri kullanılarak çeşitli duygu sınıflandırma görevlerinde değerlendirilmiştir. Sonuçlar, her iki yaklaşımın da duygu analizi referans veri kümeleri üzerinde derin öğrenme modelleri kullanarak elde edilen en başarılı sonuçları geçtiğini göstermektedir.

Anahtar Kelimeler: Kelime Özyerleşiği, Kelime Anlamları, Duygu Analizi, Derin Öğrenme, Makine Öğrenme

ACKNOWLEDGEMENTS

I would not have been able to complete this thesis without guidance of my advisor Prof. Dr. Ebru SEZER. First and foremost, I would like to thank her for her advice, encouragement, guidance and endless supports during my PhD. In addition, I would like to thank my thesis committee for revising it and providing useful help and feedback.

My deepest gratitude goes to my mom and dad for all of their sage advice, and especially my lovely wife whose supports and what she have been doing for me cannot be described by any words. I could not have achieved this success without her unlimited support, patience and encouragements.

Finally, thank God for all that He has done and all that He will do, and for allowing me to do this thesis right here.

CONTENTS

Page

ABSTRACT i
ÖZET iii
ACKNOWLEDGEMENTS v
CONTENTS vi
FIGURES ix
TABLES xi
1. Introduction 1
1.1. Motivation 1
1.2. Thesis Objectives 5
1.3. Thesis Outline 6
2. Background 7
2.1. Distributional Semantics 7
2.2. Distributed Word Representation 8
2.3. Neural Word Embedding Models 10
2.4. Neural Sentiment Classification Models 26
3. Related Work 31
3.1. Sentiment Specific Word Embedding Models 31
3.2. Word Embedding Refinement Approaches 33
3.3. Gap in Research 37
4. Refinement of Pretrained Word Embeddings 39
4.1. Sentiment Lexicons 40
4.2. Senti-Embedding Model 42
4.3. Sentiment Aware Word Embedding 44
5. Senti-Contextualized Learning Approach 47
6. Experiments 51
6.1. Experimental Setting 51
6.2. Empirical Results 55

6.3. Discussion and Evaluation 62
7. Conclusions & Future Work 67
7.1. Conclusions 67
7.2. Future Work 68
REFERENCES 69
APPENDIX 1 - Article Published from Thesis 77
APPENDIX 2 - Thesis Originality Report 78
CURRICULUM VITAE 80

FIGURES

Page

1.1. Position of positive and negative words obtained from Word2vec (a) and GloVe (b) pre-trained word embeddings 2
1.2. Sentence modeling using MaxLSTM-CNN model proposed in [1] 4

2.1. Neural network language model proposed in [2] 10
2.2. C&W general neural network model for NLP tasks [3] 11
2.3. Overall architecture of CBOW and Skip-gram models 14
2.4. Cross-lingual CCA-based word vector projection proposed in [4] 19
2.5. 1ToN+ method proposed in [5] 24
2.6. LSTM vs Tree-LSTM model proposed in [6] 27
2.7. MT-LSTM model proposed in [7] 27
2.8. Attention pooling approach proposed in [8] 28

3.1. Neural network models proposed in [9] for learning sentiment-specific word embedding 32
3.2. Framework proposed in [10] with CNN and SentiNet for encoding sentiment knowledge into pre-trained word embedding 34
3.3. Sentiment-based ranking for word 'good' using the method proposed in [11] for refining pre-trained word embedding 35

4.1. The overall structure of the refinement approach 39
4.2. Summary of refinement approach 40
4.3. Summary of generating sentiment lexicon 41
4.4. Neural network model to learn senti-embeddings 44
4.5. Principle components vs original coordinates (image source [12]) 45
4.6. Projection of data points on principle components (Eigenvectors) 46

5.1. Example of context words for embedding learning models 47

5.2. Details of the negative sampling in the Skip-gram model 48
5.3. Architecture of the Skip-gram model 49

6.1. Results of the LSTM model on different ratios between the semantic and sentimental size of SAWEs (denoted in the semantic:sentiment form) using the PCA method. Figures (a) and (b) respectively show the results obtained from Word2vec and GloVe word embeddings 65

TABLES

6.1. Hyper parameters used in each classification model..... 54
6.2. Accuracy of sentiment classification models over SST benchmark using Word2vec and refined word embeddings 55
6.3. Accuracy of sentiment classification models over SST benchmark using GloVe and refined word embeddings 56
6.4. State-of-the-art results on the SST benchmark. The last block shows our results. The numbers shown in parentheses indicate the P value between the corresponding accuracy and our accuracy in the last row. P values less than 0.1 are shown by an underline. 57
6.5. Macro F1 score of sentiment classification models over SemEval-2013 benchmark using Word2vec and refined word embeddings 59
6.6. Macro F1 score of sentiment classification models over SemEval-2013 benchmark using GloVe and refined word embeddings..... 60
6.7. State-of-the-art results on the SemEval-2013 benchmark. The last block shows our results. The numbers shown in parentheses indicate the P value between the corresponding macro-F1 and our macro-F1 scores in the last row. P values less than 0.1 are shown by an underline. 61
6.8. Results of the Word2vec model with senti-contextualized approach over SST and SemEval datasets 62
6.9. Results of the GloVe model with senti-contextualized approach over SST and SemEval datasets..... 62
6.10. Paired t-test results of SAWEs with Word2vec and GloVe pre-trained embeddings over the accuracies of four classification algorithms in the SST benchmark 63
6.11. Paired t-test results of SAWEs with Word2vec and GloVe pre-trained embeddings over the macro-F1 scores of four classification algorithms in the SemEval-2013 benchmark..... 63

6.12. Performance of the BiLSTM model using SAWEs(PCA100) obtained from single and multiple lexicons 66
6.13. Paired *t*-test results (*P* values) of SAWEs obtained from senti-contextualized learning approach with common embedding models over four classification algorithms 66

1. Introduction

1.1. Motivation

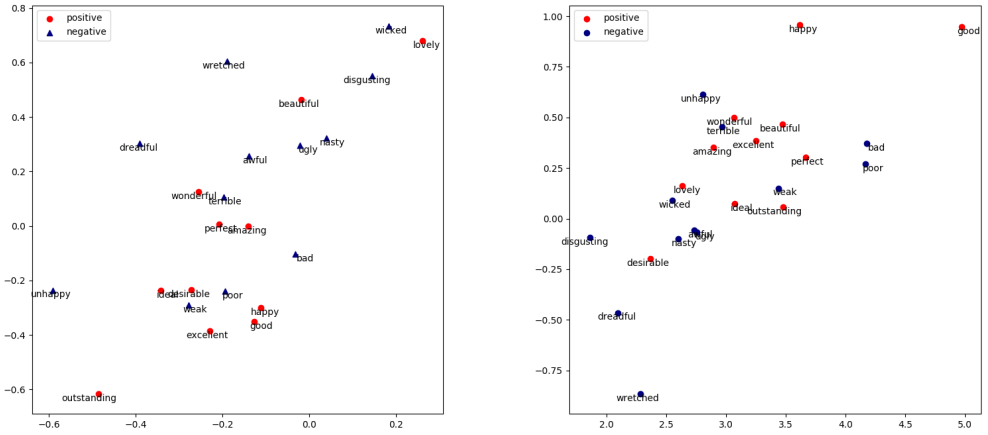
Pre-trained word embeddings play an important role in various NLP tasks because of their efficacy for semantic representation of words. They represent words in continuous low-dimensional vectors in which the semantic and syntactic information of words are embedded in the vector space. This kind of representation along with deep learning has brought significant improvements to most of NLP tasks, such as Part-Of-Speech (POS) tagging [13], Named Entity Recognition (NER) [14], Semantic Role Labeling (SRL) [3], Syntactic Parsing [15], Sentiment Analysis [16], etc.

Here word embeddings feed deep learning models to extract high level (task-specific) features automatically from raw text. As word embeddings provide a semantic representation, the features extracted by deep learning methods are more related to semantic compositions. For example, they are used to build a semantic representation for a sentence or document. In this context, the objective of word embedding models is to provide continuous representation of words for machine and deep learning algorithms in stead of discrete representation. This kind of representation increases the generality of models in deep learning and machine learning since it provides similar vectors for similar words.

However, the main question is that when are two words similar? or how can we say two words are similar where they may express several linguistic properties?

Here our main problem arises from this fact: word embedding models learn similar vectors for sentimentally opposite words. Figure 1.1. shows the position of positive and negative words generated by PCA method on Word2vec and GloVe pre-trained word embeddings trained over 100B and 6B tokens, respectively. As can be seen, positive and negative words are not distinguished from each other and there is no sentimental relationships between word-vectors.

This problem is in the nature of learning word embeddings because all these models learn word vectors based on Harris's hypothesis [17]; This hypothesis says two words are similar if they occur in similar contexts.



(a) Word2vec-100B (b) GloVe-6B

Figure 1.1. Position of positive and negative words obtained from Word2vec (a) and GloVe (b) pre-trained word embeddings

According to Harris's hypothesis, most of the pre-trained word embeddings are achieved from context-based learning algorithms trained over a large corpus. This approach learns word-vectors by predicting context words using a neural network model. In this case, all the semantic and syntactic characteristics of words are derived from the context words. Therefore, words with similar contexts would have similar vectors indicating that those words possess similar linguistic features.

However, this may not always be considered an advantage in sentiment analysis because two words with similar contexts may have opposite sentiments. This reduces the discriminating power of word embeddings when used in sentiment analysis. For example, the two following sentences, which express opposite sentiments, cannot be clearly distinguished by the context-based word embeddings:

1. "This film is very good"
2. "This film is very bad"

Yu et al. [11] have carried out an experiment on the word embeddings obtained from CBOW model [18] and showed that approximately 30% of the top 10 semantically similar words had

opposite sentiment polarities. Overall, this demonstrates that not all linguistic characteristics of words can be learned by context-based algorithms.

According to the idea behind word embedding learning algorithms, we cannot learn vectors that distinguish the sentiment polarity of words well because, in most cases, they share similar context. Therefore, the main objectives of this thesis are encoding sentiment aspects of words into pre-trained word vectors, and learning sentimental properties of words together with semantic aspects using word embedding learning algorithms. We call this kind of word embedding *Sentiment Aware Word Embedding* (SAWE).

Here, our motivation question is that "why is the sentiment aware word embedding important for sentiment analysis? In other words, "why is common word embedding not enough for sentiment analysis?

To answer these questions, the architectures of sentiment classification models should be considered. For example, Figure 1.2. shows the model proposed in [1] for representing sentence by multiple word embeddings. As can be seen, such models require complicated architectures to model a sentence. On the other hand, such models should tune lots of parameters through training that leads to over-fitting of the model.

In contrast to that, we improve the performance of basic deep learning models by relying on pre-trained sentiment-aware word embeddings that include the semantic and sentimental information of words. The use of such embeddings reduces the parameters of the model and resists over-fitting of the model.

Therefore, two main research questions are addressed in this thesis:

1. What kind of word embedding is required for sentiment analysis?
2. How to learn sentiment using context-based word embedding learning algorithms?

To avoid generating similar word embeddings for sentimentally opposite words, two basic approaches have been proposed in recent years:

1. Refining pre-trained word embeddings using sentiment lexicons.
2. Learning word embeddings from labeled corpora in a supervised manner.

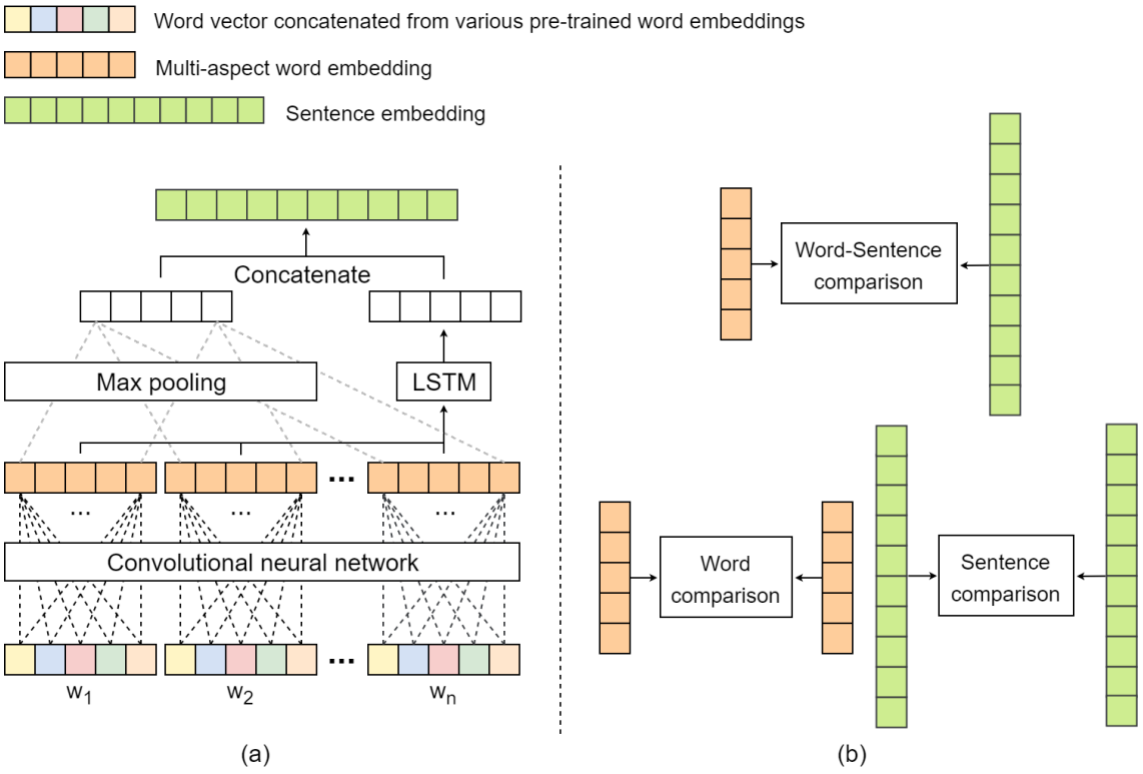


Figure 1.2. Sentence modeling using MaxLSTM-CNN model proposed in [1]

In the first approach, existing word embeddings are refined so that the opposite sentimental words do not appear in the k nearest neighbor of each other. The downside of this approach is that the optimum distance between two opposite sentimental words is unknown. It means that we do not know how far the refined vector of the given word should be away from its opposite one. On the other hand, this refinement may exterminate the semantic relationships between words because a particular set of word vectors is refined and many others remain unchanged.

Alternatively, the latter approach incorporates the unsupervised manner of embedding learning algorithms with a supervised manner using a hybrid loss function. In this approach, the sentiment of a sentence (or any text annotated by its sentiment) is propagated to each context, and word embedding models are trained to predict the context words and their sentiments, simultaneously. The constraint of this method is that it performs well on short texts such as tweets and that it requires a hyperparameter to make an optimum trade-off between the loss functions to predict the context and sentiment.

1.2. Thesis Objectives

This thesis addresses the problems noted above to create sentiment-aware word embeddings so that not only the semantic relations but also the sentimental properties of words are learnt at the same time.

To this end, two types of methods are proposed. In the first approach, we propose a method for refining pre-trained word embeddings so that all the word vectors are refined based on their polarity strengths in a sentiment lexicon. The difference in this approach is that it refines the vectors of all the words in the input vocabulary—including objective (neutral) and subjective words—in the same way that all vectors are transformed into the same distributional space.

When the sentiment information of subjective words is encoded into original vectors, two different distributional spaces are created for refined and unchanged vectors. For this reason, we first learn sentimental embeddings for all input words using a neural network model, and then use them to refine the original vectors. In the second approach, we embed the sentiment information into the context of words instead of using multiple loss functions. In this case, we do not need any annotated data or a hyperparameter to incorporate the loss functions. The key idea is to distinguish the context of two words with opposite sentiments with two synthetic sentimental words.

In recent years, many deep learning methods with complex architectures have been proposed to improve the performance of sentiment classification in different tasks [1, 19–21]. However, our attention is limited to those for which sentiment-specific word embeddings improve performance. This is the main contribution of this thesis, which improves the performance of deep learning models only by relying on the pre-trained sentiment-specific word embeddings to achieve state-of-the-art results.

Additionally, this thesis aims to address the above-noted research questions and to determine the characteristics of word embedding needed for deep sentiment classification models.

1.3. Thesis Outline

The rest of the thesis is organized as follows:

Chapter 2. presents the basic concepts related to word semantics as well as word embedding and deep sentiment classification models. Chapter 3. reviews word embedding learning methods in capturing semantic and sentimental characteristics of words and compares them with our proposed approaches. The word embedding refinement approach is presented in Chapter 4. We describe our senti-contextualized learning approach in Chapter 5. Empirical results and evaluations are presented in Chapter 6. Finally, Chapter 7. presents our conclusions and future works.

2. Background

2.1. Distributional Semantics

Understanding the meaning of words (called the word semantic) is one of the most important challenges in the field of Information Retrieval (IR) and Natural Language Processing (NLP). This can be considered a beginning step in the understanding of the meaning of computer-based human languages.

In general, two types of approaches are used to represent the meaning of words. The first, well-known counting-based method, represents the meaning of words in the Vectors Space Model (VSM) [22] in which each vector represents the semantic and syntactic information of a particular word by counting the co-occurrence of that word with other words as its context. Indeed, this approach is inspired by Harris's Distribution Hypothesis [17], which states that words occurring in similar contexts have a similar meaning. This hypothesis is the foundation of the Distribution Semantic Models (DSM).

In distributional semantics, the first step is to summarize the co-occurrence statistics of words in a word-context matrix obtained by sliding a fixed size window over a big text. Here, each word is represented by a vector with a vocabulary length that is sparse and contains redundant information. In the second step, the word-context matrix can optionally be factored into a low dimensional space using a dimensional reduction technique such as Singular Value Decomposition (SVD).

Aside from SVD, there are several alternatives for matrix smoothing such as Nonnegative Matrix Factorization (NMF) [23], Probabilistic Latent Semantic Indexing (PLSI) [24], Iterative Scaling (IS) [25], Kernel Principal Components Analysis (KPCA) [26], Latent Dirichlet Allocation (LDA) [27], and Discrete Component Analysis (DCA) [28].

All this results in compact/dense real-valued vectors for all words. These vectors can be thought of representing each word in the *d-dimensional* Euclidean space. In this space, words that are closer to each other are more similar. Here, the cosine similarity is typically used to measure the similarity of word vectors. Although this approach has produced state-of-the-art results on different tasks of word similarity [29, 30], it is more expensive in terms of memory and time. In fact, this approach cannot be applied to a large text corpus with billions of words.

To deal with this problem, different types of approaches have been proposed in the literature, such as Locality Sensitive Hashing (LSH) [31], and Hadoop MapReduce [32].

LSH is a general technique to define functions that map vectors into short signatures or fingerprints, so that two similar vectors are likely to have similar fingerprints where their fingerprints are calculated by summing the index vectors. The goal here is to boost efficiency of memory and time usage by projecting high-dimensional vectors into a low-dimensional subspace. A similar approach is also used in [33] to measure the similarity of words in a cross-lingual task.

Alternatively, Hadoop MapReduce provides a distributed processing technique for efficient processing of large datasets using a parallel distributed algorithm on a cluster. This approach can be used for the implementation of term-document (or word-context) matrix where a large dataset is present [30].

2.2. Distributed Word Representation

The second type of word representation is inspired by the neural network language modeling (NNLM) proposed by Bengio et al. (2003) [2], where a neural network model is used to learn the distributed (continuous) representation for words. This representation is used to compute the probability function for word sequences. The main idea is that a sequence of words that has never been seen before has a high probability if it is composed of words that are similar to the words previously seen in the training set. This is indeed a solution to the problem of dimensionality in statistical language modeling (SLM).

Although the *n-gram* model roughly resolves this problem, it still suffers from the curse of dimensionality. This means that any word sequence observed in the test set may be different from all word sequences seen in the training set. Thus, computing the joint probability of such word sequences poses a problem for such language models.

To deal with this problem, the context is represented as a sequence of word vectors (instead of words) and the probabilities are calculated on the basis of these feature vectors. As similar words are expected to have similar feature vectors, the probabilities computed by these features can be extended to other similar contexts. Therefore, the similar context of word sequences will have similar probabilities.

For example, the sentence “The boy is walking in the woods” in the training set can evoke another sentence such as “A girl was running in a park”. In this example, the semantic and syntactic similarities between the paired words ‘boy-girl’, ‘walking-running’, ‘woods-park’, ‘the-a’, ‘is-was’ can help us generalize the join probability function of the first sentence for other similar sentences. Overall, similarities between words are adopted to obtain a generalization from observed word sequences to new unseen sequences.

The objective here is to learn a neural network model f that predicts the probability of a particular word w_t given its context w_n^{t-1}, i.e. $f(w_t, ..., w_{t-n+1}) = \hat{P}(w_t|w_n^{t-1})$ such that for any choice of n, $\sum_{i=1}^{|V|} f(i, w_t, ..., w_{t-n+1}) = 1$.

The above model is realized by mapping an input of sequence words to a sequence of feature vectors via a projection function C:

$$f(w_t, ..., w_{t-n+1}) = g(i, C(w_{t-1}), ..., C(w_{t-n+1})) \tag{1}$$

Where $C(.)$ maps each word to a continuous vector and $g(.)$ is the neural network model. The continuous vectors of words are indeed parameters of the model that is tuned through the training. Because this neural language model embeds each word into a low dimensional real-valued vectors, it is known as *word embedding learning*. Figure 2.1. shows the details of this neural language model proposed in [2].

As shown above, evaluating the output of the neural language model in a probability distribution requires a normalization over the whole vocabulary set. It would be much more expensive, while computing the gradients of each probability with respect to the parameters of the model.

To deal with this problem many context based neural network models, which use a fixed-size window, were proposed in the literature. These models are called *neural word embedding models* and described in the following sections.

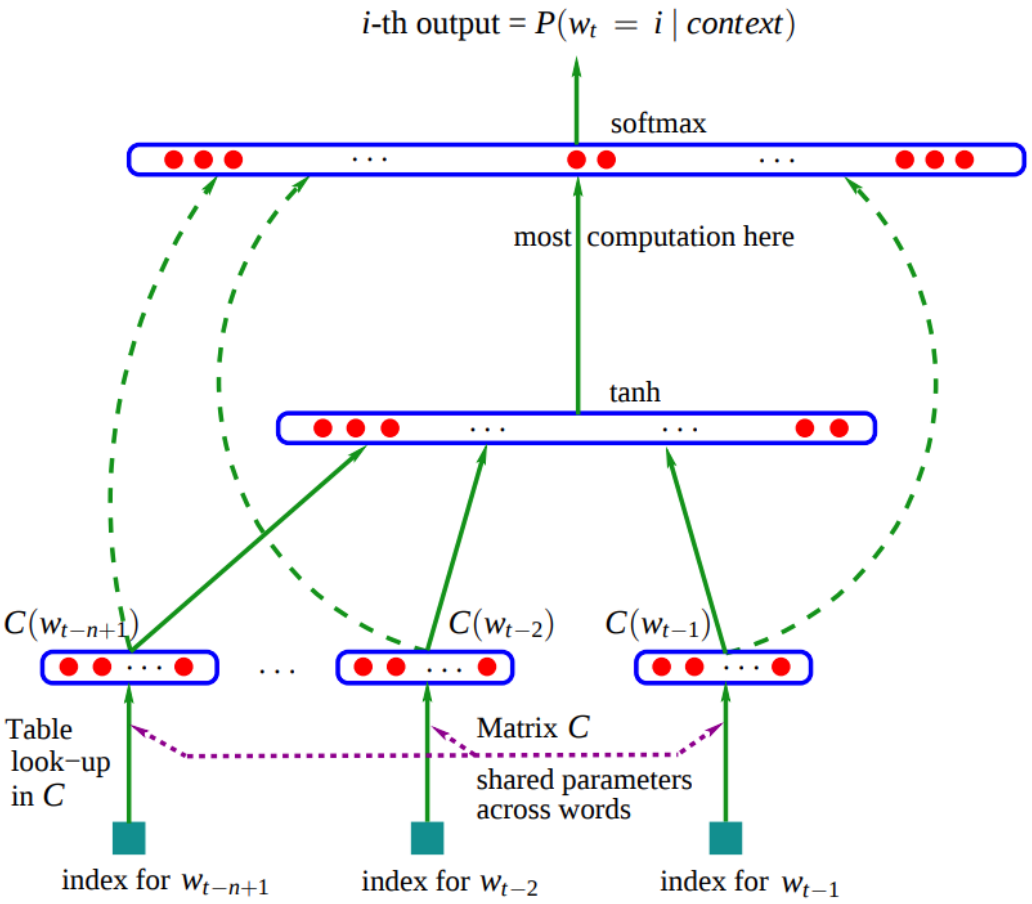


Figure 2.1. Neural network language model proposed in [2]

2.3. Neural Word Embedding Models

2.3.1. C&W Model

In the C&W model, that was proposed by Collobert and Weston (2008) [3], word embeddings are learned through a unified deep neural network model that has been proposed for several NLP tasks. The neural network (NN) architecture they proposed (shown in Figure 2.2.) learns relevant features for each task using very limited prior knowledge without any feature engineering. In their neural network architecture, word embeddings are jointly learned through the network and then used to learn task-specific features in the deep NN layers.

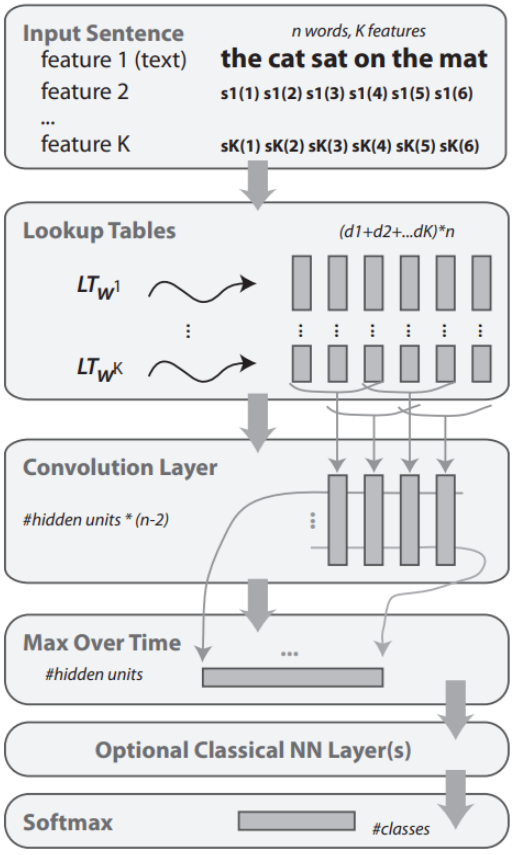


Figure 2.2. C&W general neural network model for NLP tasks [3]

They also adopt their architecture to learn word embeddings separately based on a fixed-size of text window over the English Wikipedia data. In that approach, the neural network is used to perform a binary classification task by which it specifies whether the word in the middle of the input window is related to its context or not. In this classification, a positive sample is the context of the window and the negative sample is the same context in which the word in the middle of the window is replaced by the other words in the vocabulary.

In their model, the hinge loss function is used to train the model as a ranking type cost:

$$\sum_{S}\sum_{w}\max(0, 1 - f(S) + f(S^{w})) \tag{2}$$

where S is window context and w is the all vocabulary words excluding the middle word in the context. It actually computes the noise and the model tries to distinguish the correct context (with the true middle word) from the noisy ones.

Unlike Bengio's model, they used a ranking type loss function by a margin of 1 for training. Their objective was to eliminate the softmax function in the output layer and minimize the error by random negative samples.

2.3.2. vLBL and ivLBL

Mnih and Kavukcuoglu (2013) [34] proposed a scalable variant of the log-bilinear (LBL) model to learn word embeddings using noise contrastive estimation (NCE). The log-bilinear language model (LBL), proposed by Mnih and Hinton (2007) [35], is one of the most effective approaches in the neural probabilistic language modeling (NPLM). This model utilizes the distributed representation of words to predict the distribution of the target word where several preceding, following, or surrounding words are given.

Unlike previous neural probabilistic language models [2], this approach eliminates the hidden layer and predicts the distributed vector of the target word w^t by using a linear function $\odot$ over the context vectors where the context is comprised of n words preceding, following, or surrounding the target word, i.e $h = w_1, ..., w_n$. This linear function can be considered as an element-wise multiplication and shown in equation 3.

$$\hat{q}(h) = \sum_{i=1}^{n} c_i \odot \nu_{w_i} \tag{3}$$

Where c_i denotes the weight vector of the context word w_i and specifies the interaction of each context word ν_{w_i} when predicting the target word. It can be simply considered as a position-dependent weight vector for each context word w_i with respect to target word w^t. The scoring function of the model is defined for the target word w^t when its context h is given, as equation 4:

$$S_\theta(w^t, h) = \hat{q}(h)^T \nu_{w^t} + b_{w^t} \tag{4}$$

Where b_{w^t} is a bias that captures the context-independent effect of the target word. This bias indicates that the prediction of the target word may not completely depend on the context. Although this issue contradicts the philosophy of NPLM—predicting the next word given preceding words—it makes a speed-up at the training stage.

In the above model, the target word is considered to be the middle word in the context and is predicted by the distributed vectors of the surrounding words. In addition, position-dependent weights can be eliminated when constructing a context vector; it can simply be calculated as the average of context word vectors as $\frac{1}{n}\sum_{i=1}^{n}\nu_{w_i}$. The proposed model is called vLBL in the literature.

The second model is actually the inverse of the vLBL model—predicting the context given the target word—because the objective is only to learn word embeddings and not to propose a language model. Hence, it is called ivLBL. This model is proposed based on the assumption that all words in the context are conditionally independent of each other; a *n-word* context should be predicted by modelling the joint probability of all context words as equation 5.

$$P_{\theta}^{w^t}(h) = \prod_{i=1}^{n} P_{\theta,i}^{w^t}(w_i) \tag{5}$$

The scoring function of the model with position-dependent weight vector is defined as the equation 6

$$S_{i,\theta}(w_i, w^t) = \left(c_i \odot \nu_{w^t}\right)^T \nu_{w_i} + b_{w_i} \tag{6}$$

The model without position-dependent weight vector can be defined as equation 7:

$$S_{i,\theta}(w_i, w^t) = {\nu_{w^t}}^T \nu_{w_i} + b_{w_i} \tag{7}$$

Both vLBL and ivLBL models are trained based on NCE method [36] to estimate the log-likelihood gradient. This approach is more effective and simpler than tree structured-based methods like hierarchical softmax and can guarantee the stability. The basic idea is the discrimination of the sample data from the noise data by using binary logistic regression classifier.

2.3.3. CBOW and Skip-gram (Word2vec) Models

The Continuous Bag of Words (CBOW) and Skip-gram models were proposed in [37] to create high-quality, distributed dense vector representation for words using two types of neural network learning models. The general architecture of these two models are shown in Figure 2.3. The most important characteristic of these two models is to be trained efficiently over a massive textual data.

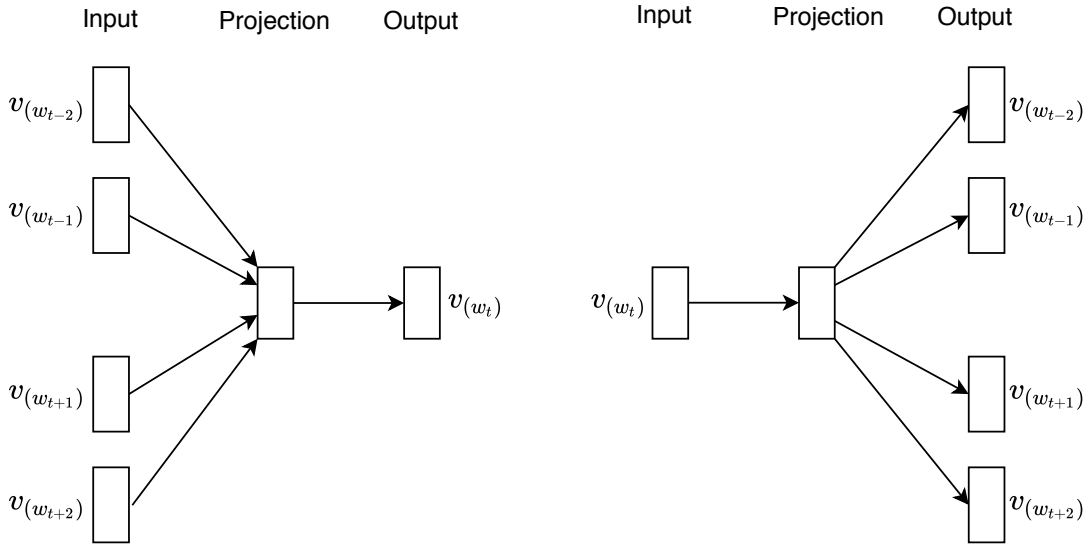


Figure 2.3. Overall architecture of CBOW and Skip-gram models

Indeed, the CBOW model is a neural probabilistic language model that uses a fixed-size context window to predict the target word that is at the center of the context, based on the surrounding words.

The CBOW model calculates the probability of the target word w when the context h is given, i.e. $P(w|h)$, using a log-linear model. This is similar to the feed-forward neural network model where the non-linear hidden layer is excluded.

In this approach, $P(w|h)$ is calculated by the log linear model used in vLBL through the following equations, while position-dependent weights are not considered:

$$\nu_h = \frac{1}{|h|} \sum_{i=1}^{|h|} v_{h_i} \tag{8}$$

$$S_\theta(w, h) = {\nu_h}^T u_w \tag{9}$$

$$P(w|h) = P_\theta^h(w) = \frac{\exp S_\theta(w, h)}{\sum_{i=1}^{|V|} S_\theta(w_i, h)} \tag{10}$$

In the above equations, ν_h is the context vector, v_{h_i} denotes the input vector of the i-*th* word in the context h, u_w is the output vector of the target word w and $S_\theta(w, h)$ is the scoring function of the model when the target word w and the context h is given. In the architecture of the CBOW model two types of vectors are leaned for each word. Therefore, the model includes two parameter matrices namely V and U for input and output words, respectively. In this case, the vectors v_{h_i} and u_w belong to V and U matrices, respectively.

In the second approach, the Skip-gram model predicts or generates the surrounding words when the center word is given.

Here, the model calculates the probability $P(w_{t-m}, ..., w_{t-1}, w_{t+1}, ..., w_{t+m}|w_t)$ where the window size is m. This probability is broken out on the basis of the Naive Bayes assumption, in that all surrounding words are independent when the center word is given. This means that the Skip-gram model calculates the probability of each context word independently of its distance from the target word. Therefore the probability is calculates through the following equations:

$$P(w_{t-m}, ..., w_{t-1}, w_{t+1}, ..., w_{t+m}|w_t) = \prod_{j=0, j\neq m}^{2m} P(w_{t-m+j}|w_t) \tag{11}$$

$$P(w_{t-m+j}|w_t) = \frac{\exp(u_{t-m+j}^T v_t)}{\sum_{k=1}^{|V|} \exp(u_k^T v_t)} \tag{12}$$

In the above equations, like the CBOW model, the vectors v and u are the input and output vectors belonging to matrices V and U, respectively.

As seen in Equations 10 and 12, in order to train both CBOW and Skip-gram models, a normalization over the whole vocabulary set is required in the output layer. It would be much more expensive, while computing the gradients of each probability with respect to the parameters of the model.

To make speed-up in the training, two approaches have been proposed to train the both models. In the first approach, the hierarchical softmax is used in the output layer. In this approach, a binary tree structure is used in the output layer to calculate the probabilities based on the relative probabilities of the child nodes. Compared to normal softmax, this approach decreases the computational cost from $O(|V|)$ to $O(log(|V|))$.

Alternatively, following [34], the second approach proposes a simplified variant of the NCE called *negative sampling* (NEG) that utilizes the unigram distribution to calculate the probability of negative samples (noise distribution). This approach assumes that a good model should be able to make a distinction between data and noise via logistic regression. This idea is used to estimate the noise of each sample data under consideration of k negative samples. Thus, it distinguishes the context words from the noise distribution while k negative samples are considered for each context. Unlike softmax, this approach does not need to know the probability distribution of the noise and only uses the samples, so it has lower training time.

Although NEG improves the speed of training and the quality of word embedding, it is not an optimal solution. In this approach, high-frequency words, which are less informative, would possess a high probability in the negative sampling.

To deal with this issue, Qin et al. [38] proposed a more rational negative sampling strategy using the *tfidf* weighting method to optimize NEG. In this approach, the probability of noise distribution is calculated on the basis of *tfidf* so that the medium-frequency words–which are more informative–possess higher probabilities than the high-frequency words.

2.3.4. GloVe Model

Previous methods for learning the distributed representation of words have achieved great success in capturing the semantic and syntactic relationship between words. These relationships exhibit linear regularities of word vectors which allow the use of vector arithmetic to obtain a word vector from two other specific vectors.

However, models like Skip-gram do not use global corpus statistics as these models are trained on the basis of a local context window. On the other hand, although traditional methods such as LSA effectively leverage the global statistics of the word-context co-occurrence matrix, they do not perform well on the word analogy task. This is natural because there is no semantic linearity between the word vectors in the co-occurrence matrix, and SVD only transforms all vectors into a reduced space in which similarities between vectors are more tangible.

As a solution, the GloVe (Global Vectors) [39] leverages the two previous approaches, i.e Skip-gram and LSA, to learn the semantic and syntactic regularities of word vectors by relying on the global statistical information of the corpus obtained from the word-context matrix. GloVe is a simple model that makes a projection between the word vectors and the conditional probability of their corresponding words in the co-occurrence matrix. Therefore, for target word w_i and any context word $\hat{w}_j$, the GloVe model, following Skip-gram, calculates $P(\hat{w}_j|w_i)$ using co-occurrence matrix $\mathbf{X}$ as:

$$P(\hat{w}_j|w_i) = F({\nu_i}^T \hat{\mu}_j) = \frac{\mathbf{X}_{ij}}{\mathbf{X}_i} \tag{13}$$

Where ν_i and $\hat{\mu}_j$ denote the vectors of w_i and $\hat{w}_j$, $\mathbf{X}_{ij}$ is the number of times word $\hat{w}_j$ occurs in the context of word w_i, and $\mathbf{X}_i = \sum_k \mathbf{X}_{ik}$. If the F function equals $\exp$ then Equation 13 is solved as:

$${\nu_i}^T \hat{\mu}_j = \log \mathbf{X}_{ij} - \log \mathbf{X}_i \tag{14}$$

$${\nu_i}^T \hat{\mu}_j + \log \mathbf{X}_i = \log \mathbf{X}_{ij} \tag{15}$$

In Equation 15, $\log \mathbf{X}_i$ is a scalar depending on the target word w_i, therefore it can be considered a bias b_i for w_i. In order to preserve the symmetry, an additional bias $\hat{b}_j$ is added to

the equation 15 with respect to $\hat{w}_j$. Therefore the equation 15 is solved as:

$$\nu_i{}^T \hat{\mu}_j + b_i + \hat{b}_j = \log \mathbf{X}_{ij} \tag{16}$$

To learn word vectors, the GloVe proposes a weighted least squares regression model using Equation 16. The loss function of this model is defined as a least square problem:

$$J = \sum_{i,j=1}^{V} f(\mathbf{X}_{ij})(\nu_i{}^T \hat{\mu}_j + b_i + \hat{b}_j - \log \mathbf{X}_{ij})^2 \tag{17}$$

In the above loss function, V denotes the size vocabulary set and $f(.)$ is a weighting function to reduce the impact of much more frequent words—which are less informative— in the co-occurrence matrix.

As seen in equation 17, the main difference between GloVe and the previous embedding models, such as Skip-gram, is that the GloVe does not use the online window-based method in training and it does not have the Softmax normalization problem or negative sampling. Instead, the GloVe is trained based on the non-zero elements in the co-occurrence matrix, regardless of the negative samples.

2.3.5. Multilingual Correlation Word Embedding Model

This model—that was proposed by Faruqui and Dyer (2014) [4]—incorporates semantic information in bilingual parallel contexts into monolingual word embeddings using the Canonical Correlation Analysis (CCA). Figure 2.4. shows how cross lingual word vectors are projected using CCA. In this approach, the word vectors obtained from the monolingual corpus are projected into a new word space achieved by their correlations with the corresponding word vectors in the other languages.

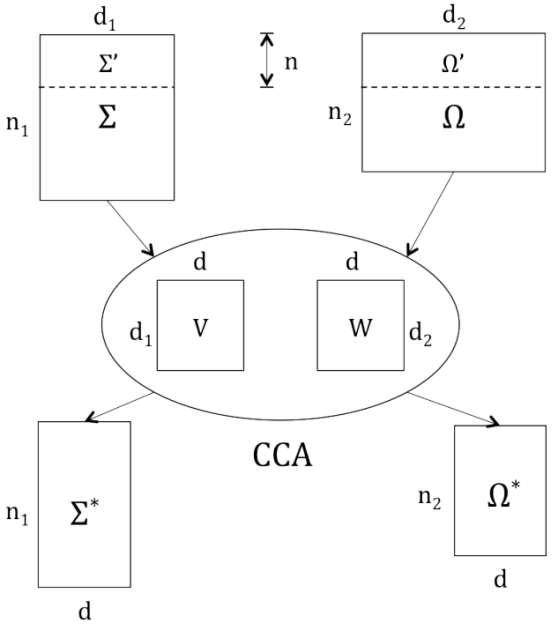


Figure 2.4. Cross-lingual CCA-based word vector projection proposed in [4]

This approach exploits the semantic information of words in the word embeddings of the other language and combines them with the corresponding ones in the target language. Word embeddings obtained from this technique can better distinguish the synonyms and antonyms of a particular word. Hence, it shows a better performance on the word similarity task than the others such as word analogy, i.e. semantic or syntactic relations.

This approach makes a substantial improvement on the performance of monolingual LSA. However, the effect of this technique on the neural network based word embedding models are comparable.

2.3.6. Dependency-Based Word Embedding Model

Dependency-based word embedding model [40] generalizes the Skip-gram model with a dependency-based context instead of a bag-of-word context. In word embeddings, the semantic regularities between words arise from similarities in their contexts; similar words occur in similar contexts based on Harris's distributional hypothesis.

However, the bag-of-word context may not always be considered as a good estimate of the context for a particular word. For example, in the sentence "Turkish police discovered a suspicious substance" the context of word 'discovered' is composed of words 'Turkish', 'police', 'a', and 'suspicious' with a window size of 2. In this example, while 'police' and 'suspicious' can become a good context words for 'discovered', 'Turkish' cannot. Moreover, the word 'substance' could also be placed in the context because it could help to determine some particular semantic relations.

Therefore, a bag-of-word context with a fixed-size window cannot always capture appropriate contexts for a target word, especially when the relevant words are relatively far from the target word. Here, different kinds of contexts produce different word embeddings that can discover a variety of functional regularities between words. Therefore, the semantic regularity of word embeddings depends on the choice of contexts in the training stage.

To deal with this problem, context words are determined based on their syntactic dependency with the target word. Here, the long dependency between target and context words can be captured in such a way that more relevant context words are chosen with respect to the target word.

As a result of this approach, different kinds of semantic similarities can be achieved. While the linear bag-of-word context produces domain-specific similarities, the dependency-based context yields more general similarities. For example, according to [40], while bag-of-word based embeddings specify the similar words for 'Florida' as countries and cities within Florida (meronyms), the dependency-based embeddings indicate that other U.S. states are more similar to Florida. It can be concluded that the bag-of-word contexts generate semantic associations, whereas the dependency-based contexts specify semantic similarities between words.

2.3.7. PPMI Based Word Embedding Model

PPMI based word embedding model—that was proposed by Levy and Goldberg (2014) [41]—is a factorization of the PPMI word-context matrix using a weighted SVD approach, in which the eigenvalue matrix is weighted with an exponent factor [42, 43]. This model, actually, proposes an approach to dealing with the PMI's "Achilles heel" problem—having a bias towards the co-occurrence of rare words—using the context distribution smoothing to modify the PMI. In addition, this model shows that neural embedding models implicitly implement co-occurrence matrix factorization.

In this model, the similarity between two words is assessed based on two types of criteria. Two words are similar:

1. If they have similar contexts (according to the distributional hypothesis)
2. If one word tends to appear in the context of the other word or vice versa

The second one is well suited to sentiment analysis, since two words with opposite sentiments are not more likely to occur in the context of each other when the context is small. To capture both items, this model adds the context vectors to the target word vectors. This approach has already been proposed in the GloVe model.

Based on [41], two other criteria that play an important role in the quality of word embedding are:

1. The weighting of context according to the positional importance (called dynamic context)
2. The sub-sampling of frequent words

Sub-sampling before processing word-context pairs has a strong effect on the performance of embedding models as it implicitly enlarges the size of the window. In this way, some context words that cannot be included in the context of the target word (before sub-sampling) are placed in the context of the target word after sub-sampling.

2.3.8. FastText Word Embedding Model

Most word embedding models, such as Skip-gram, do not take into account word morphology. In such models, all contexts are represented as bag of words without considering the internal structure of words. This factor is important in morphologically rich languages such as Turkish, German and Russian, where different forms of words—generated by affixes—may be very scarce in the training corpus. Therefore, the prior models may have limitations on learning good word embeddings for such kinds of words.

FastText model—proposed by Bojanowski et al. in 2017 [44]—takes into account the morphological information of words and proposes an extension of the Skip-gram model based on character n-grams. In this approach, each target word is represented as a bag of character n-grams to which the model learns vector representations; the word vector is achieved by summing up the character n-gram vectors. As a result of this approach, word vectors can be computed for words never seen in the training corpus.

In the FastText model, Skip-gram with negative sampling is used to learn word vectors of the character n-grams. The objective is to propose an extension of the Skip-gram model that uses the internal structure of words (subword) in the learning of word vectors. This approach has been evaluated on word similarity and analogy tasks in several languages. From the results, word representations learned with this approach outperform those obtained from morphological analysis methods.

Another important characteristic of this model is that very good word embeddings are learned with very small training data. This may be desirable in cases where word embeddings need to be trained from task-specific datasets which are more restricted than general ones.

2.3.9. Meta-Word Embedding Models

Since several types of word embeddings are trained using different neural network (NN) models over different corpora, each can represent different aspects of word semantics in compact, real-valued vectors. This gives a motivation for combining different types of word embeddings and learning meta-word embeddings that encompass all the semantic aspects of words captured by individual ones. The influence of meta embeddings can be discussed in two ways:

- enhancing the distributed representation of words
- covering more words

Yin and Schutze in 2016 [5] proposed four ensemble methods to combine five kinds of word embeddings as follows:

1. CONC: this is achieved by concatenating the individual word embeddings
2. CONC-SVD: this is obtained by applying dimensionality reduction technique (SVD) to the concatenation vectors
3. 1ToN: this employs a neural network model to learn the meta embeddings from individual ones
4. 1ToN+: this extends the previous neural network model to learn meta embeddings for words that do not exist in some individual sets of word embeddings

The objective of learning meta embeddings form individual embeddings is to find a meta word vector so that it can be projected to each individual vector using learned parameters. In other words, a meta embedding is a projection of the combination of all the individual embeddings.

Figure 2.5. illustrates how a meta- embedding is learned from individual embeddings using the 1ToN+ method. In this figure, M_1 to M_5 are the model parameters which realize the word embedding projection.

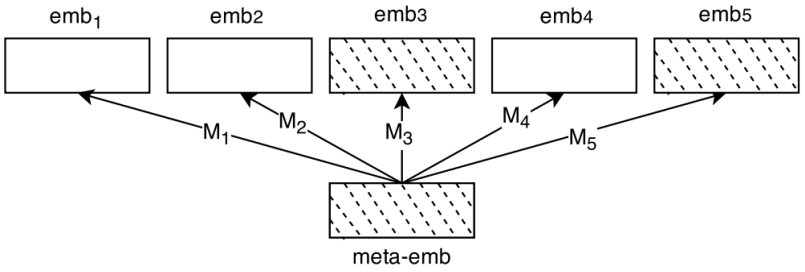


Figure 2.5. 1ToN+ method proposed in [5]

The meta word embeddings proposed in [5] were evaluated on three tasks: word similarity, word analogy, and part-of-speech tagging. The reported results show that the two sets of word embeddings—word2vec and GloVe—have more dominant effects on the meta embeddings than the others. In addition, meta embeddings outperform the word2vec and GloVe in the analogy task as well as the word similarity task in most cases.

In addition, the meta embeddings achieved by the concatenation method perform better than the others on the word similarity task. However, in the POS tagging task, small improvements are observed (mostly one point).

2.3.10. LexVec Model

LexVec model [45, 46] addresses the problems of PMI-SVD, GloVe and SGNS (Skip-Gram with Negative Sampling) methods and proposes a model to learn word embeddings using a weighted factorization on the PPMI matrix. In the PMI-SVD approach, the PMI matrix is unbounded in negative values and has a bias towards infrequent contexts. Therefore, replacing the negative values with zero (PPMI) and smoothing the context distribution improves its performance on word similarity tasks [47]. However, the factorization of the PPMI matrix to a lower-rank representation using SVD cannot perform well on the word similarity and semantic analogy. This is because SVD minimizes L2 error and assigns the same weights to all errors (cells with zero values).

On the other hand, GloVe makes a factorization on the logarithmic word-context matrix in which the loss function of factorization is weighted based on the frequency of co-occurrences: errors arising from frequent co-occurrences are severely penalized than those which are infrequent. Moreover, GloVe is trained on non-zero co-occurrences and does not consider negative samples, i.e word-context pairs with zero value.

In addition, the SGNS method implicitly factorizes the shifted PMI matrix with a neural network model [41].

By considering the problems of each method, LexVec proposes a neural network model to learn word embeddings based on the factorization of the PPMI matrix using a loss function that not only weights errors unequally, but also it takes into account the negative samples with zero co-occurrences. The evaluation results show that this approach achieves the best results on the word similarity task compared to GloVe and SGNS. Best results are achieved by word embeddings generated by sum of word and context vectors as proposed in GloVe.

2.4. Neural Sentiment Classification Models

Sentiment classification task is widely used to evaluate most deep neural network models in the literature. Recently, sentiment analysis has received much attention and many sophisticated models have been proposed to improve its performance. Sentiment analysis is indeed a multifaceted problem that involves several sub-problems. Therefore, the methods proposed in this area address the problems of sentiment analysis from three different aspects [48]: *task-oriented*, *granularity-oriented*, and *methodology-oriented*. This section presents an overall introduction of the most successful deep models in sentiment classification and classifies them in two main groups:

- Algorithmic-level approaches that tailor basic deep learning models to sentiment analysis
- Knowledge-based approaches that incorporate sentiment lexical resources into deep learning models

In the first approach, basic deep learning models are tailored to different sentiment classification tasks: binary, ternary, and fine-grained (multi-class). For example, Kim (2014) [49] proposed three convolutional neural network (CNN) models—namely *static*, *non-static*, and *multichannel*—that achieved successful results in the binary and fine-grained sentiment classification tasks. In this CNN model, word embedding layer is treated with three different settings: (1) word embedding layer is not tuned during training (static) (2) word embeddings are considered to be trainable (non-static) parameters, and (3) both types of embedding layers are used in multiple channels (multichannel). Another CNN model was also proposed in [50] for sentiment classification tasks.

Long Short-Term Memory (LSTM) is another successful model in sentiment analysis. Tai et al. (2015) [6] proposed a variant of the LSTM model to represent a sentence based on its semantic and syntactic tree structure. In fact, this approach modifies the order of the input words and feeds the LSTM model with nodes in the dependency parse tree of the sentence. Figure 2.6. shows the difference between the LSTM model where sequential and tree-structured methods are used. This approach allows the model to better represent the semantics of the sentence so that the problem of polarity shift is addressed in sentiment analysis.

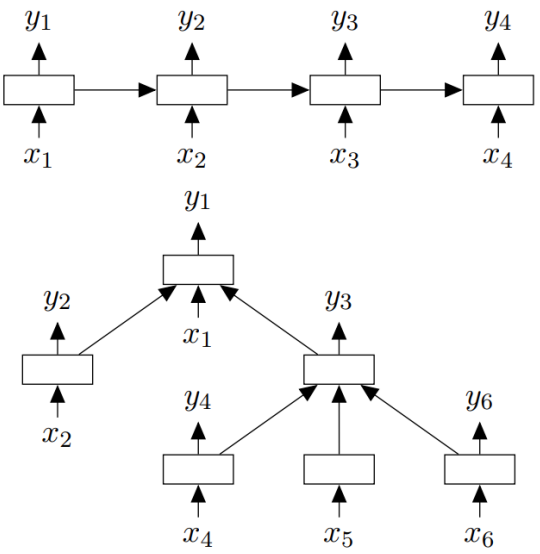


Figure 2.6. LSTM vs Tree-LSTM model proposed in [6]

In the other approach, called Multi-Timescale LSTM (MT-LSTM), the LSTM model is extended to several groups of hidden states in which each group is activated at particular time scales [7]. Therefore, this approach represents long texts better than the LSTM model. Figure 2.7. indicates the overall structure of the MT-LSTM model.

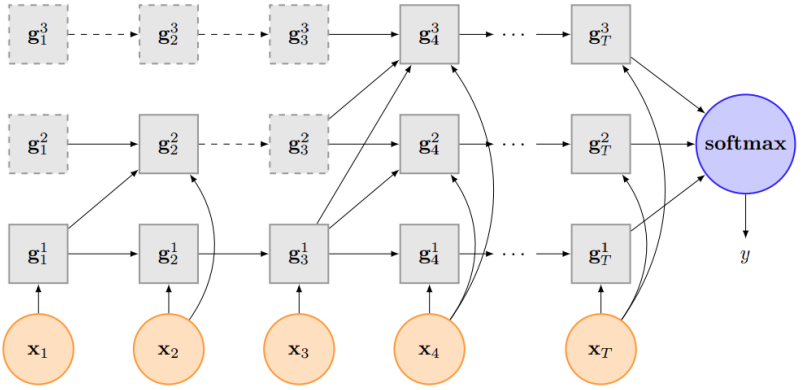


Figure 2.7. MT-LSTM model proposed in [7]

In some other methods, a combination of several models is used in sentiment classification. For instance, [51] uses the Bidirectional-LSTM (BiLSTM) model [52, 53] with a convolution layer and attention mechanism. In this approach, the BiLSTM model is fed by a sequence of features extracted from the convolutional layer and then an attention mechanism is applied to the output of the BiLSTM layer in order to learn the importance of contextual features.

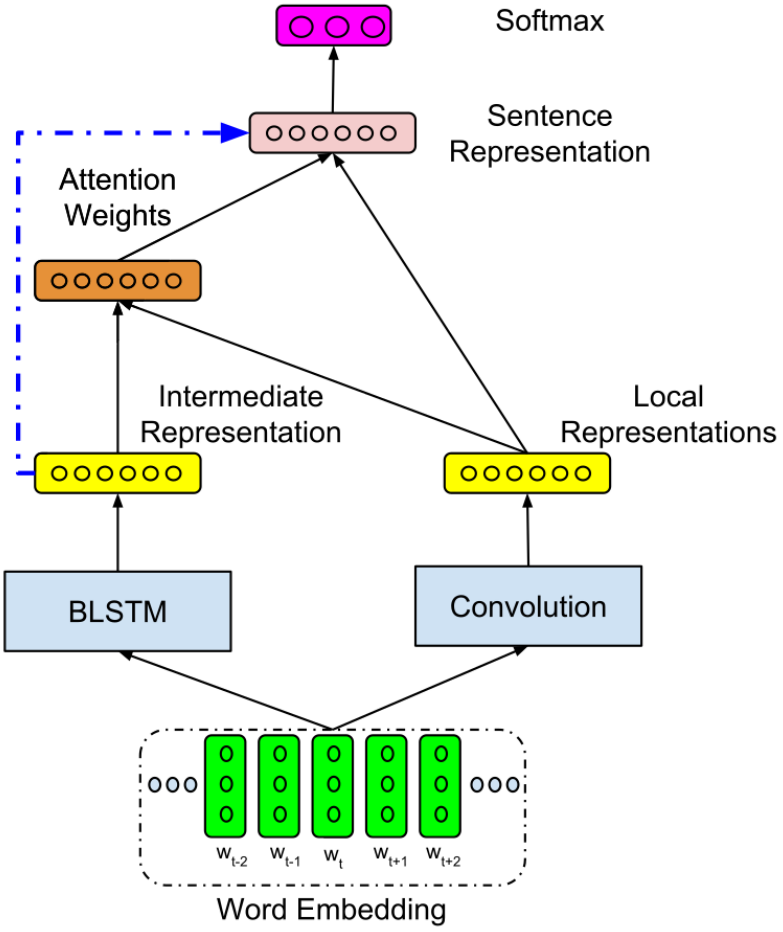


Figure 2.8. Attention pooling approach proposed in [8]

In another similar approach, BiLSTM and CNN models are combined using a novel *attention pooling* method proposed in [8]. In this approach, two types of features, intermediate and local, are gained by BiLSTM and Convolution layer, respectively. These features are used to calculate the attention weights so that the local features of the convolution layer are weighted by these attention weights. As a result, the representation of the sentence is generated by the weighted local features. The summary of this approach can be seen in Figure 2.8.

To evaluate the performance of the proposed sentiment lexicon, we use a lexicon based neural sentiment analysis in which the prior sentiment knowledge of words are modified based on the syntactic structure of the sentence. In fact, the prior sentiment of words in the lexicon can be changed based on the negation, intensification, or semantic structure of terms in the context. For example, the positive sentiment of “good” is shifted to negative in the sentence

"Nobody gives a good performance in the team" by the word "nobody". A similar situation can be seen for the word "great" in the sentence "He was a great liar" and its positive sentiment (based on the lexicon) is shifted to negative. Thus, the use of sentiment lexicons without consideration of the context cannot achieve a satisfying result.

Knowledge-based approach has shown more successful results in sentiment analysis. In this approach, sentiment lexical resources are used in deep learning models, and the prior polarity of words is incorporated into context-dependent features.

For instance, Teng et al. (2016) [54] proposed a context-sensitive lexicon-based method for different sentiment classification tasks. In this approach, the sentiment score of a sentence is computed based on the weighted sum of the polarity values of the subjective words obtained from a sentiment lexicon. These weights are considered as context-dependent features that can modify the prior polarity of words with respect to the context. Therefore, the effects of negation, shifting and intensification can be considered in the sentiment classification task. The overall structure of this model is simply shown in Eq. 18.

$$Score(s) = \sum_{k=1}^{m} \gamma_k^s \times LexScore(t_k) + b^s \tag{18}$$

In Eq. 18, m denotes the number of subjective words in the sentence s, $LexScore(t_k)$ is the polarity score of word t_k in the lexicon, γ_k^s denotes the context-dependent weight of term t_k and b^s is the sentence bias score.

In this model, context-dependent sentiment features are extracted by using BiLSTM. In fact, BiLSTM model maps each sentence word w_t to two *left-to-right* (h_t^L) and *right-to-left* (h_t^R) hidden state vectors. These state vectors are considered as the semantic composition features (or contextual features) in computing the context-dependent sentiment wights γ. The output layer of this model calculates the context-dependent weights for each subjective word and also calculates the sentence bias. The contextual weight of word w_t in the context s is computed through two $tanh$ layers using h_t^L and h_t^R hidden vectors as Eqs. 19 and 20 [54]:

$$p_t^s = tanh(W_{ps}^L h_t^L + W_{ps}^R h_t^R + b_{ps}) \tag{19}$$

$$\gamma_t^s = \frac{\lambda}{m} 2tanh(W_{pw} p_t^s + b_{pw}) \tag{20}$$

where W_{ps}^L, W_{ps}^R , b_{ps}, W_{pw} and b_{pw} are the model parameters. The λ is a scale in the range of [0,1] denoting the importance of the lexicon side of the model.

Similarity, the sentence bias is calculated by using two $tanh$ layers to sentence start and end hidden state vectors. h_s^R and h_e^L as Eqs. 21 and 22 [54]:

$$p_b^s = tanh(W_{pb}^L h_e^L + W_{pb}^R h_s^R + b_{pb}) \tag{21}$$

$$b^s = (1 - \lambda) 2tanh(W_b p_b^s + b_b) \tag{22}$$

where W_{pb}^L, W_{pb}^R , b_{pb}, W_p and b_b are parameters.

The model noted above has also been used in cross-lingual sentiment classification using different sentiment lexicons [55, 56].

More recently, Chen et al. (2019) [21] proposed a similar approach that encodes the sentiment information of words into a gated recurrent neural network [57] using two types of hidden states: (1) sentiment modifier context and (2) sentiment polarity information. In this approach, Like [54], a sentiment modifier context updates the hidden state of the sentiment polarity so that the last sentiment hidden state is considered to be the sentimental representation of the text.

3. Related Work

As the main objective of this thesis is to propose sentiment-aware word embedding, this chapter focuses only on methods that learn word embeddings with semantic and sentimental properties. Therefore, at first, word embedding learning algorithms are discussed where word vectors are learned in a combination of supervised and unsupervised manner. Then, refinement approaches that encode sentiment information into pre-trained word embeddings are described, and finally, the research gap in this area is presented.

3.1. Sentiment Specific Word Embedding Models

The main objective of all embedding models is to encode the semantics of words into dense vectors using context-based neural network models. To encode the sentimental properties of words, two main approaches have been proposed in the literature:

- learning word embeddings in a combination of supervised and unsupervised manner;
- refining pre-trained word embeddings using lexical sentiment resources.

The first approach has been considered in many studies. For instance, Maas et al. (2011) [58] proposes two different models to capture the semantic and sentimental information of words and generates a full model by maximizing the sum of their objective functions. It uses a probabilistic topic model to learn word vectors by computing words' association strength with respect to each latent topic. The sentiment information of words is then learned by using a logistic regression model through a sentiment classification task. However, its results indicate that the word vectors obtained from the semantic model perform better than those obtained from the full model in the sentiment classification.

Later Tang et al. (2014) [9] proposed a similar approach to learning sentiment-specific word embeddings. Unlike [58], it uses the multi-task neural network model proposed in [3] to learn the semantic and sentimental information of words from a mass of tweets. Both models, $C\&W$ and $SSWE_h$ are unified using a weighted-sum of individual loss functions. The unified model ($SSWE_u$) is shown in Figure 3.1.

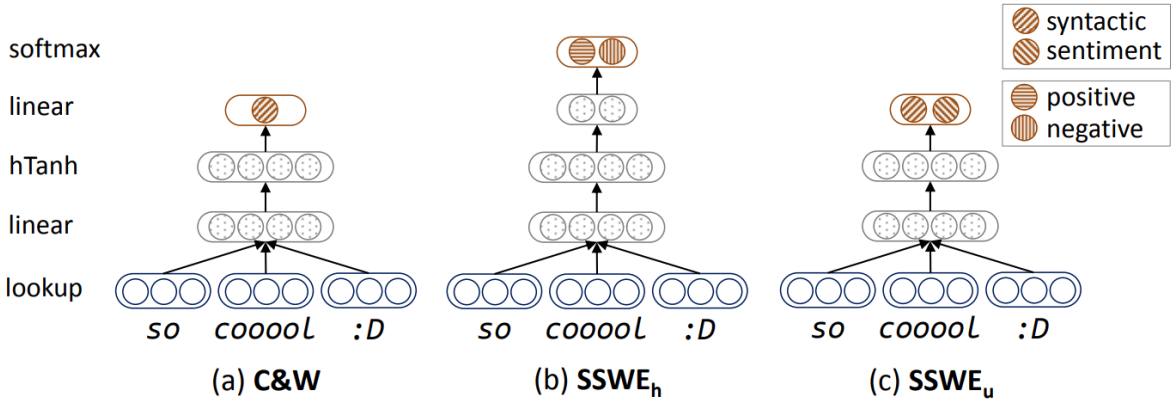


Figure 3.1. Neural network models proposed in [9] for learning sentiment-specific word embedding

The reported results show that the embeddings obtained from the unified model outperform the conventional word embeddings, such as word2vec and C&W, in the twitter sentiment classification.

Learning sentiment-specific word embeddings from tweets has also been proposed in [59]. It adopts a number of neural network models that encode the text sentiment information into word embeddings with tailoring loss functions. It proposes two prediction models to learn contexts of words and their sentiments, and then combines them with a hybrid loss function.

The different side of that work is to exploit two kinds of lexical information, namely *word-word* and *word-sentiment* associations, as a regularizer in the embedding model. The objective of word-word association is to regularize the embedding model for words having semantic relationships in the Urban Dictionary, such as synonym. On the other hand, word-sentiment association builds another regularizer that tunes the embedding model to predict the sentiment of words in addition to predicting their contexts. Their empirical results indicate that the sentiment-specific embeddings are superior to the context-based embeddings in the sentiment classification using SVM.

Zhou et al. (2015) [60] proposed an autoencoder-based approach for learning bilingual sentiment word embeddings. This approach not only captures semantic similarities across languages, but also encodes text sentiment information into bilingual embeddings. Their proposed approach achieved state-of-the-art results in the cross-lingual sentiment classification task.

More recently, Xiong et al. (2018) [61] leveraged sentiment resources and distant supervised information to develop a multi-level embedding learning method. This method employs a parallel asymmetric neural network to model n-gram, word-level sentiment, and tweet-level sentiment in the learning process.

However, encoding the sentimental information of text into word embeddings in a supervised manner requires sentiment annotation of short texts. In addition, a hyperparameter is needed to create an optimum combination of objective functions for adding sentimental properties of words. On the contrary, our embedding learning approach does not require any annotated data or hyperparameter. Instead, it encodes the sentimental information into the context of words and distinguishes the context of two words having opposite sentiments.

3.2. Word Embedding Refinement Approaches

From another point of view, recent studies focus on proposing methods for refining pre-trained word embeddings using lexical sentiment resources. This is a variant of the approach that encodes external semantic knowledge into word embeddings for the benefit of relational information in semantic lexicons, such as Retrofitting [62], Dict2vec [63], and AffectiveSpace [64].

In a similar way, Ye et al. (2018) [10] proposed a method to encode sentiment knowledge into pre-trained word embeddings. It proposes a joint learning method that uses a convolutional neural network for sentiment classification and a feedforward neural network—called SentiNet—for word polarity classification. The proposed framework is shown in Figure 3.2.

The difference between this work and ours is that we do not tune pre-trained word embeddings through a document-level sentiment classification, because this retunes the word embeddings according to the objective of the classification task and eliminates the semantic relationships between word vectors.

In addition, word embedding dramatically increases the size of the training parameters so that the model is over-fitted. On the contrary, our refinement method results in pre-trained sentiment-aware word embedding that can be used in any sentiment classification task.

In another study, Yu et al. (2018) [11] proposed a lexicon-based refinement model that does not require any annotated data and can be applied to any pre-trained word embeddings. It

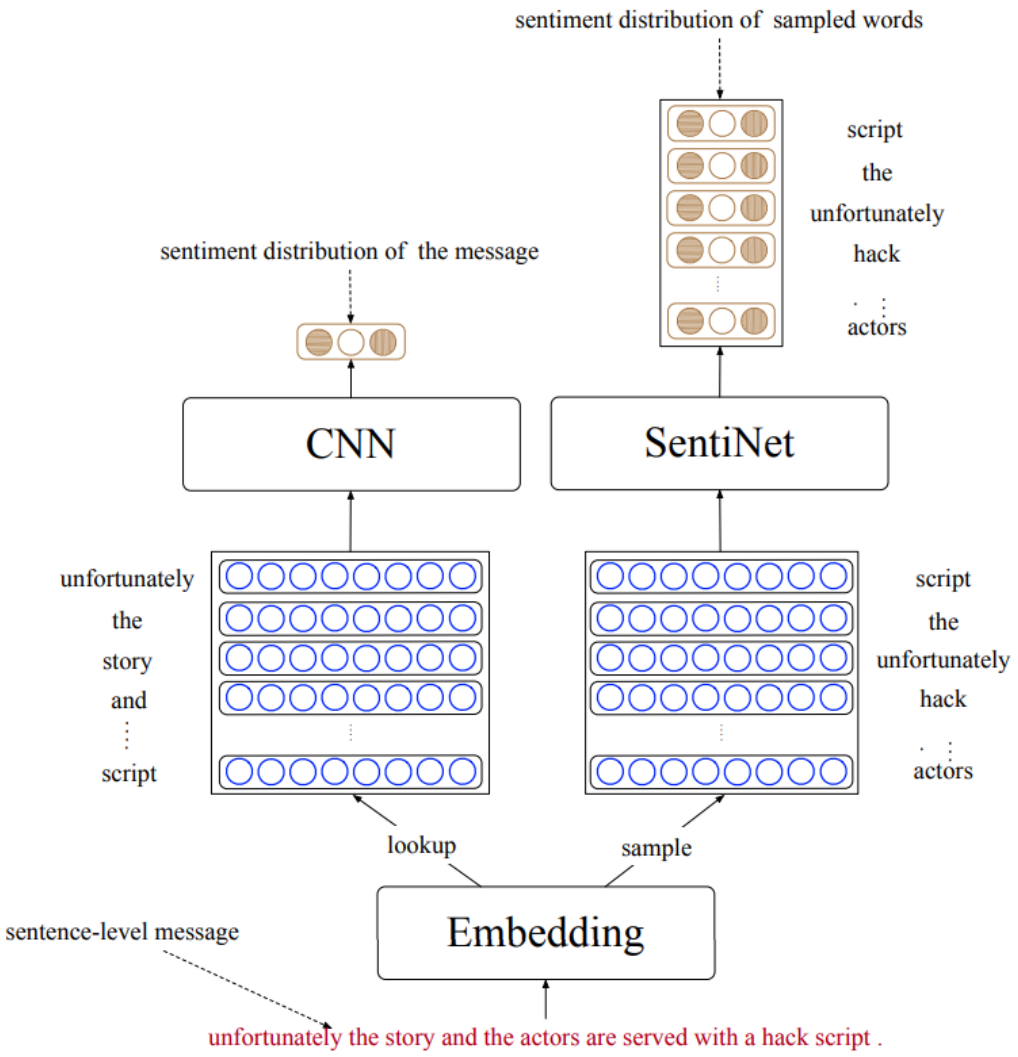
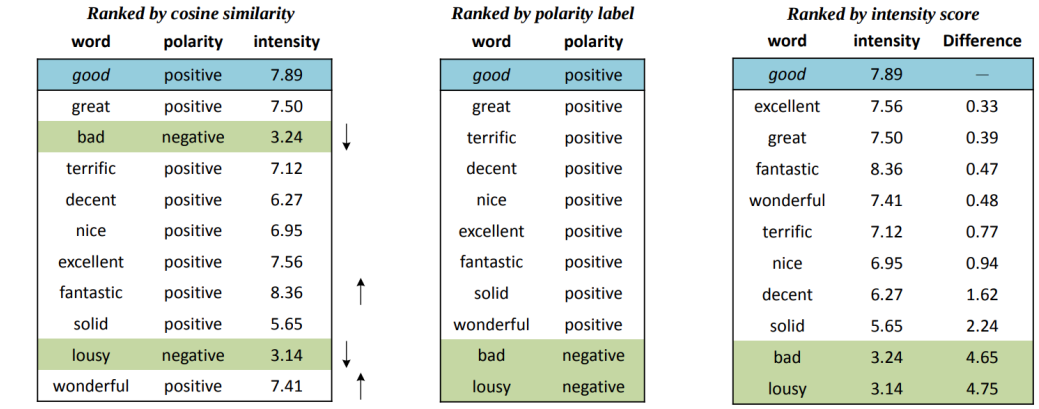


Figure 3.2. Framework proposed in [10] with CNN and SentiNet for encoding sentiment knowledge into pre-trained word embedding

refines the word vectors based on a model that optimizes the vector-distance between any subjective word and its k nearest neighbors according to three goals that state each refined vector should be:

1. closer to the sentimentally similar neighbors
2. far away from dissimilar neighbors
3. not too far away from the original vector

Although the refined embeddings achieved from this approach outperform the original ones in sentiment classification tasks, it cannot determine how far the refined vectors should be away from the original ones. Moreover, this approach calculates the k nearest neighbors by a limited number of words in the sentiment lexicon. Therefore, the words out of the lexicon (but included in the input pre-trained embeddings) are ignored when computing k-nearest neighbors. Figure 3.3. shows the detail of this ranking approach for word 'good'.



Ranked by cosine similarity

word	polarity	intensity
good	positive	7.89
great	positive	7.50
bad	negative	3.24
terrific	positive	7.12
decent	positive	6.27
nice	positive	6.95
excellent	positive	7.56
fantastic	positive	8.36
solid	positive	5.65
lousy	negative	3.14
wonderful	positive	7.41

(a) initial ranking (by cosine similarity)

Ranked by polarity label

word	polarity
good	positive
great	positive
terrific	positive
decent	positive
nice	positive
excellent	positive
fantastic	positive
solid	positive
wonderful	positive
bad	negative
lousy	negative

(b) polarity-based ranking

Ranked by intensity score

word	intensity	Difference
good	7.89	–
excellent	7.56	0.33
great	7.50	0.39
fantastic	8.36	0.47
wonderful	7.41	0.48
terrific	7.12	0.77
nice	6.95	0.94
decent	6.27	1.62
solid	5.65	2.24
bad	3.24	4.65
lousy	3.14	4.75

(c) intensity-based ranking

Figure 3.3. Sentiment-based ranking for word 'good' using the method proposed in [11] for refining pre-trained word embedding

We address these issues in our refinement approach and calculate senti-embeddings for all words in the input pre-trained embeddings. Here our refinement approach can be seen as learning new embeddings from two or more types of pre-trained embeddings. This idea is similar to the meta embedding learning approach proposed in [5].

Since several kinds of word embeddings are trained using various neural network models from different corpora, each one can represent different aspects of word semantics in compact real-value vectors. Therefore, combining them and learning meta word embeddings can encompass all the semantic aspects of words captured by individual models. Inspired by this idea, we first learn senti-embeddings that represent only the sentimental aspects of words, and then combine them with original vectors to obtain new embeddings that represent both semantic and sentimental aspects.

The same idea was also used in [64] to generate an affective knowledge resource (called AffectiveSpace) for emotive reasoning. It represents ConceptNet [65] and WordNet Affcet [66] with a sparse matrix and transforms it into a dense vector space using SVD. The difference

between this work and ours is that such resources are used in a concept-level sentiment analysis that requires a semantic parser to extract concepts from the text. However, such concept level features can be extracted by deep learning models through training. On the other hand, our approach can produce senti-embeddings with any dimension that enables us to obtain sentiment-aware word embeddings with any degree of sentimental density.

3.3. Gap in Research

As stated earlier, pre-trained word embeddings enable deep learning models to better extract high-level and task-specific features through the training. Moreover, the use of pre-trained word embeddings reduces the model parameters dramatically where they are non-trainable. This may resist the overfitting of the model. However, the efficacy of the features extracted for sentiment classification depends on the word vectors.

At this point, adding sentiment information to pre-trained word embeddings may change the semantic relationship between words. To avoid this, most of the approaches in the literature focus on refinement methods that do not transform refined vectors extremely from the original vectors. In fact, they try not to change the distributional space when refining word vectors. This constraint hinders methods from generating word vectors that, together with semantics, represent the sentimental relationship between words more correctly.

On the other hand, imposing the sentiment of the whole text on the word embedding learning algorithms can change the real characteristics of the individual words. Under this condition, the objective of the classification task is embedded in the word vectors and, as a result, all kinds of words, including objective and subjective, are affected by the sentiment of the whole text. Therefore, the learned vectors cannot provide a general-purpose embedding that indicates the semantic association between words and sentiment at the same time.

Overall, we need word vectors that maintain the semantic regularity of words together with sentiment properties. We call this kind of word vectors *sentiment aware word embeddings* (SAWEs) because these embeddings are not specific to sentiment classification alone and can perform well on any NLP task. In this context, refinement and supervised learning approaches are both considered to generate such embeddings.

However sentiment-based contextualized embedding learning algorithms have received less attention in the literature. In fact, instead of using the sentiment of the whole text, the prior polarity of words can be considered in the embedding learning algorithms along with word contexts. This gap is considered in this thesis so that a contextualized learning approach is proposed.

Another gap addressed in this thesis is the lack of empirical studies on the sentiment aware word embeddings obtained from pre-trained embeddings where their distributional space is

different. In other words, instead of refining, new word embeddings—with a new semantic distribution—are generated from pre-trained embeddings that include both the semantic and sentimental aspects of words. Our hypothesis is that adding sentiment information into pre-trained word embeddings by keeping their prior distributional space cannot have much more impact on the sentiment classification. This is because the distributional space of a collection of words is generated by looking at the whole co-occurrence of words in a textual corpus, and any external modification on the individual vectors may ruin that distributional space.

Therefore, the sentimental relationship between words and their semantics cannot be met by modifying vectors individually. Here, we need to know the effect of every change on the whole relationship of words. At this point, two approaches are considered:

1. modifying all vectors homogeneously according to their sentimental polarities
2. learning word vectors according to the context of words and sentimental polarities

Basically, this thesis is accomplished to fill these gaps and show how sentiment aware word embeddings can be generated from pre-trained word embeddings or unsupervised learning methods.

4. Refinement of Pretrained Word Embeddings

As pre-trained word embeddings have shown successful results in the sentiment analysis, we first focus on these embeddings and refine them to improve the performance of sentiment classification models. Figure 4.1. shows the overall structure of the proposed approach to refine pre-trained word embeddings using a sentiment lexicon.

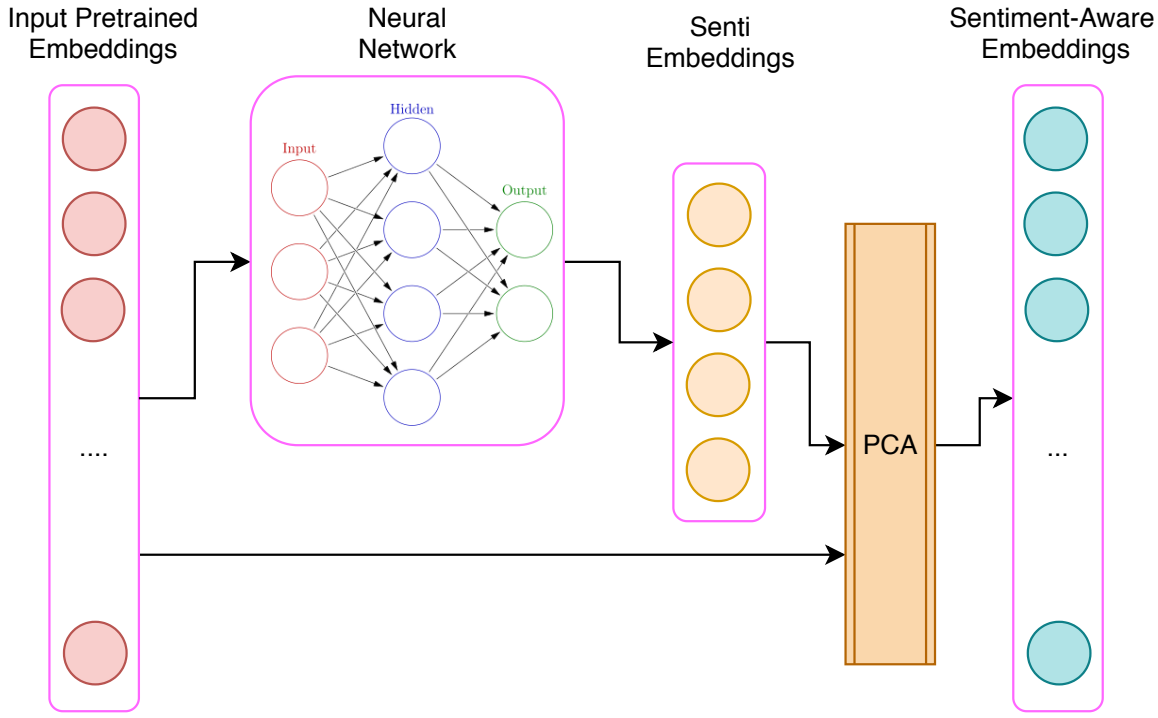


Figure 4.1. The overall structure of the refinement approach

This approach encodes the sentiment information of words into vector representation using two main steps. In the first step, we learn a kind of word vectors based on the sentiment polarity of words (called *senti-embeddings*) using a neural network model. The learned embeddings are incorporated with the original vectors in the second step in order to retain the semantic characteristics of words.

As shown in Figure 4.1., senti-embeddings are learned through the neural network model that predicts the polarity score of the input word vector. The senti-embeddings are extracted from the model and incorporated with the original vectors to generate the final vectors.

However for learning senti-embedding, the prior polarity strength of words is required. To this end, two types of sentiment lexicons are used in the refinement approach. Overall, the

refinement approach can be summarized in three steps as shown in Figure 4.2. The details of each step are presented in the following sections.

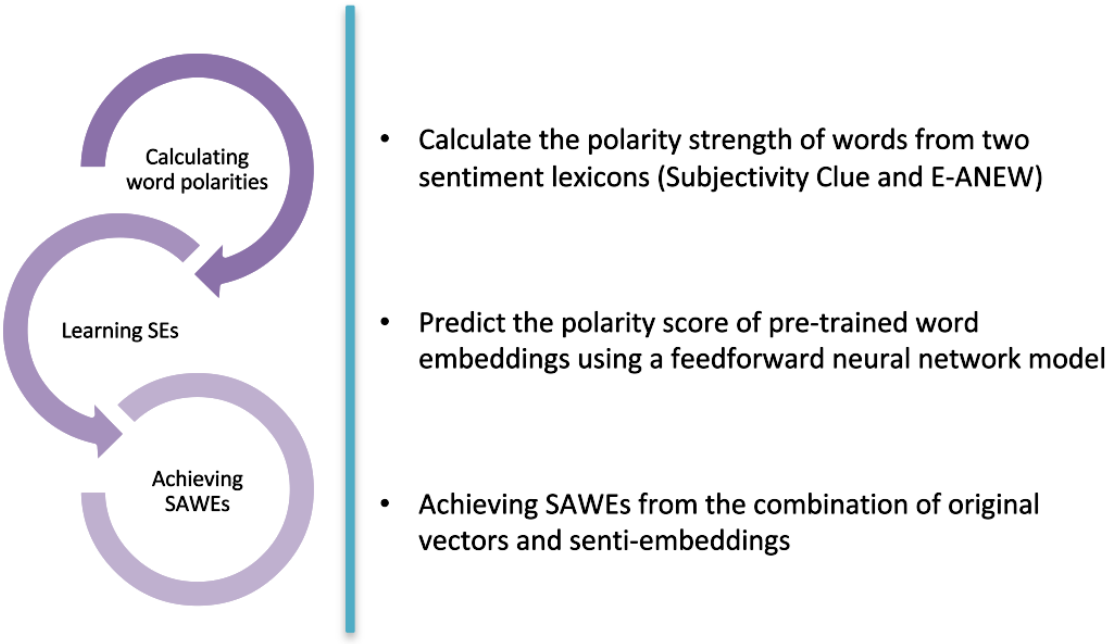


Figure 4.2. Summary of refinement approach

4.1. Sentiment Lexicons

To determine the sentiment polarity of words, we adopt a combination of two sentiment lexicons achieved from manual resources. The first lexicon is the extended version of Affective Norms of English Words (E-ANEW) [67]. This lexicon consists of 13915 words with *3-dimensional* real-valued scores for *valence* (the pleasantness of a stimulus), *arousal* (the intensity of emotion provoked by a stimulus), and *dominance* (the degree of control exerted by a stimulus).

Here, the value of valence indicates the intensity of positive and negative sentiments through values in the range from 1 to 9. While the values of 1 and 9 represent the most negative and the most positive sentiments, respectively, the value of 5 shows the neutral sentiment. Therefore, we consider the value of valence to be the value of polarity.

To enrich the E-ANEW lexicon, we also employ the Subjectivity clue lexicon [68] to extend and refine the word polarities. In this lexicon, 8222 words have been annotated by four types of contextual polarity labels: strongly positive, weakly positive, strongly negative, and weakly negative. In this lexicon, the words that are positive in most contexts are marked strongly positive, and those that may have only certain positive usages are marked weakly positive. In the negative words, the interpretation is similar.

We assign real-valued scores to the polarity of the words in the Subjectivity clue to comply with E-ANEW lexicon. To this end, the values of 8, 6.5, 3.5, and 2 are adopted for strongly positive, weakly positive, weakly negative, and strongly negative, respectively. To select these numbers, we made a comparison between words that are common in both lexicons. Average polarity values (from E-ANEW) were calculated for each group of polarity labels in the subjectivity clue. The numbers were selected on the basis of these averages.

Therefore, the two lexicons are unified by calculating the polarities of words from two sources of sentiment. The averages of the polarity values are considered to be the final scores for common words between two lexicons. In this way, the final scores are more accurate and reliable than the individual scores. As a result of this process, we have achieved a lexicon of 17301 words. Here, all words not included in our final lexicon, but included in the vocabulary set of input embeddings, are considered neutral and possess the polarity of 5. All the process is summarized in Figure 4.3.

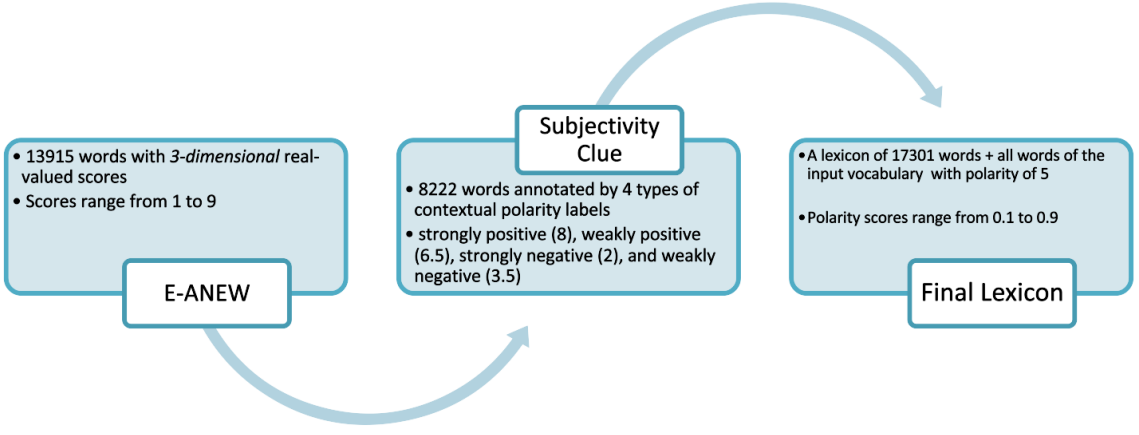


Figure 4.3. Summary of generating sentiment lexicon

As noted above, we refine all words in the vocabulary set for any pre-trained word embedding using a sentiment lexicon. However, the number of words in the vocabulary set is much greater than sentiment lexicons. Therefore, we cannot find the polarity score for all words in the vocabulary set using any sentiment lexicon.

In our approach, words for which no sentiment polarity can be assigned using a lexicon are considered neutral (objective). This assumption is not exactly true because all subjective words may not be included in the given lexicon. Therefore, lexicons with more words can perform better. On the other hand, less accurate lexicons that may assign incorrect polarity scores to words lead our approach to incorrect word embeddings. Therefore, we adopt lexicons generated manually in our study. However, the size of such lexicons is small. At this point, the merger of multiple lexicons can partially alleviate this problem. This can increase the size of the lexicon and improve the polarity score of common words.

4.2. Senti-Embedding Model

To learn senti-embeddings, we propose a simple feedforward neural network model to predict the polarity score of pre-trained word embeddings. Figure 4.4. shows the architecture of the proposed model and illustrates how senti-embeddings are generated from the model. This model consists of three layers: *input*, *hidden* and *output*. Each pre-trained word vector $v \in \mathbb{R}^d$ is given to the input layer of the d neurons in which the input neurons correspond to vector elements. Depending on the size of senti-embeddings, the model generates a $m-unit$ hidden layer with a non-linear $tanh$ activation function as in Equation 23.

$$\mathbf{h}_v^\theta = \tanh(\mathbf{W_h}^T.v + \mathbf{b_h}) \tag{23}$$

In Equation 23, $\mathbf{W_h} \in \mathbb{R}^{d \times m}$ and $\mathbf{b_h} \in \mathbb{R}^m$ are the model parameters where d and m indicate the dimensions of the input embeddings and senti-embeddings, respectively. The output layer calculates the score of the model using sigmoid function that corresponds to the polarity of the given vector. As noted in Section 4.1., the polarity score of input words ranged from 1 to 9. Here we transform them into a range between 0.1 and 0.9 because the model calculates scores between 0 and 1 as in Equation 24:

$$f_v^\theta = \sigma({\mathbf{W_o}}^T.\mathbf{h}_v^\theta + \mathbf{b_o}) \tag{24}$$

In equation 24, $\mathbf{W_o}$ and $\mathbf{b_o}$ are the model parameters of the output layer with the dimension of 1: $\mathbf{W_o} \in \mathbb{R}^{m\times 1}$ and $\mathbf{b_o}$ is a scalar (bias parameter). The model is trained with Adam optimization algorithm using *mean absolute error* loss function with L2 regularization:

$$J(\theta) = \frac{1}{n}\sum_{k=1}^{n} |f_{v_k}^\theta - f_{v_k}^g| + \lambda||\theta||_2^2 \tag{25}$$

In equation 25, while $f_{v_k}^\theta$ indicates the output of the model for input vector v_k, $f_{v_k}^g$ is the gold score of the given vector according to our sentiment lexicon. The model is trained at 200 epochs with the batch size of 1024. After training, the *m-dimensional* senti-embedding is extracted from the output of the hidden layer $\mathbf{h}_v^\theta$ when the input vector is given. However, to produce senti-embeddings for all input vectors, as shown in Figure 4.4., we calculate them from a linear combination of the input matrix $\mathbf{I}$ and hidden layer parameters through Equation 26.

$$SE_I^\theta = \tanh(\mathbf{I}.\mathbf{W_h} + \mu.{\mathbf{b_h}}^T) \tag{26}$$

In Equation 26, $\mathbf{I} \in \mathbb{R}^{n\times d}$ denotes the input matrix containing all pre-trained word embeddings, $\mathbf{W_h}$ and $\mathbf{b_h}$ are the model parameters in the hidden layer, and μ is the all-ones vector with size of n ($\mu \in \mathbb{R}^{n\times 1}$). The vector μ is used to create a matrix with size of $n \times m$ in which each row is a copy of the transpose of vector $\mathbf{b_h}$.

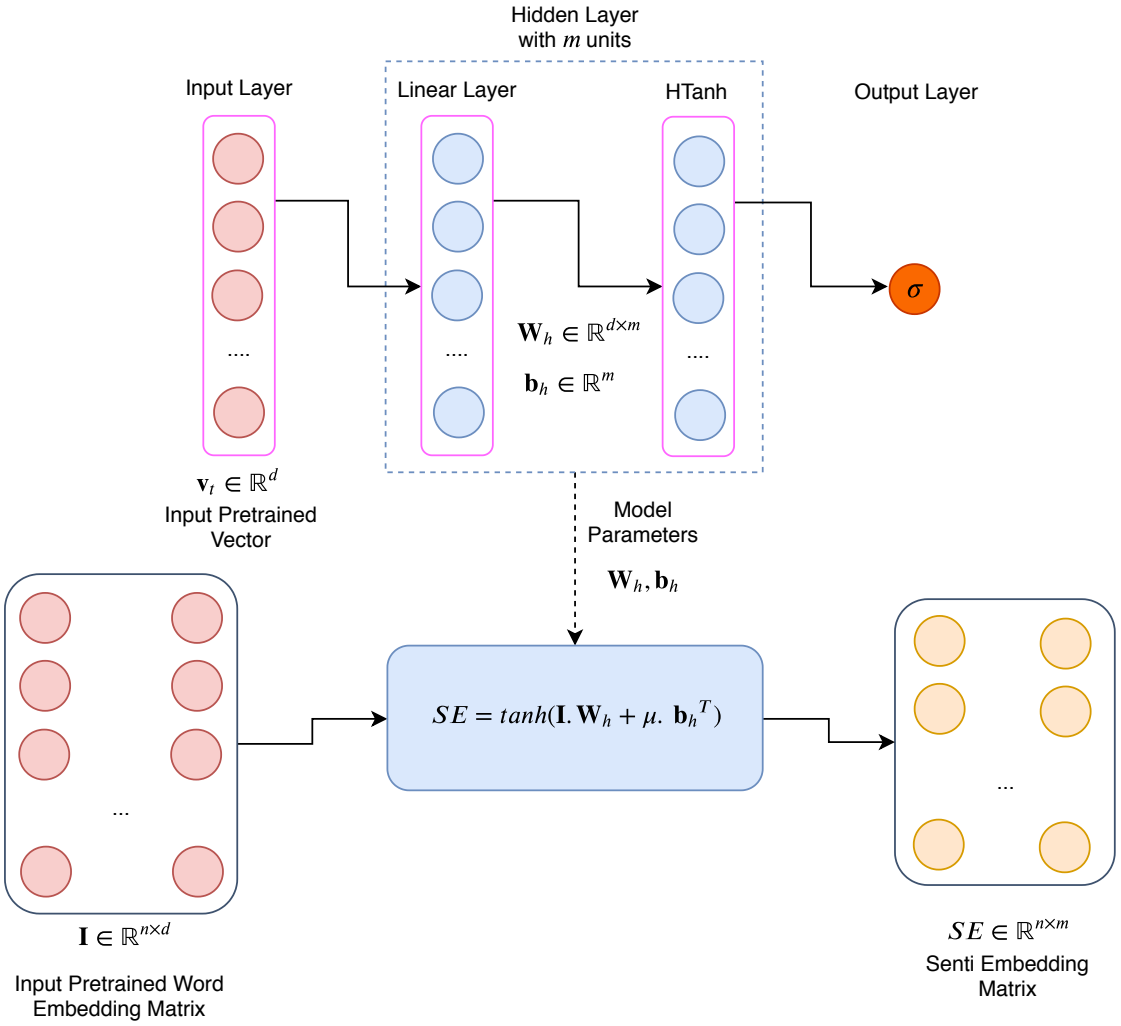


Figure 4.4. Neural network model to learn senti-embeddings

4.3. Sentiment Aware Word Embedding

We concatenate the input pre-trained word vectors with senti-embeddings and reduce the size of the output vectors to the input vectors using the Principal Component Analysis (PCA) method.

$$SAWEs = PCA(\mathbf{I} \oplus \mathbf{SE}) \tag{27}$$

PCA is considered a dimensionality reduction method that reduces the large set of variables to a small set so that most of the information in the large set is preserved. Here, PCA results

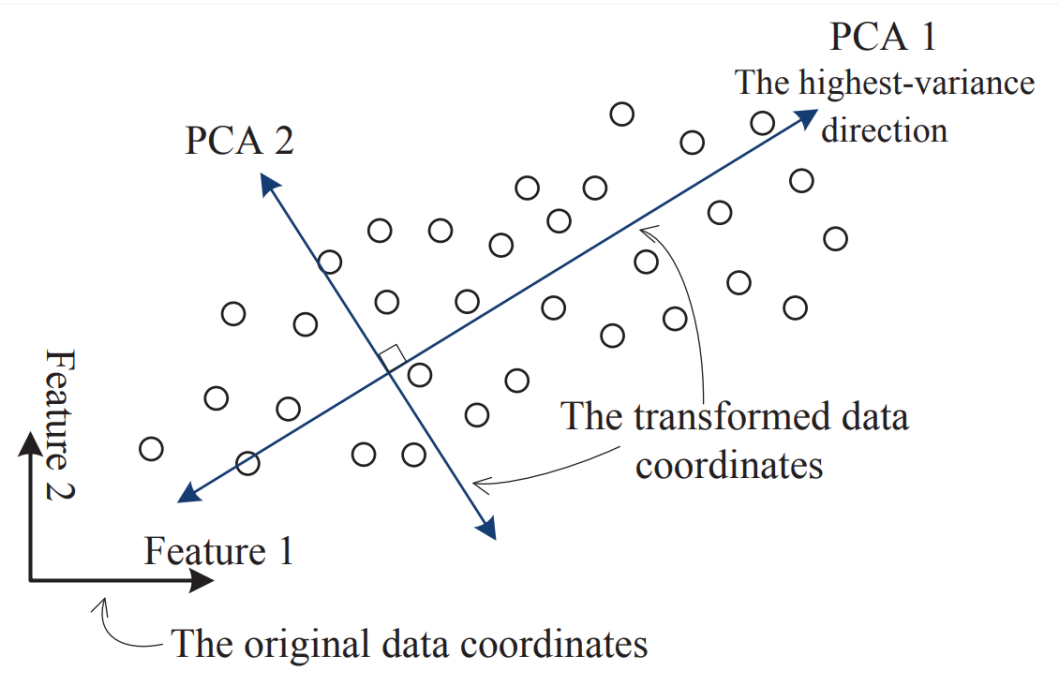


Figure 4.5. Principle components vs original coordinates (image source [12])

in vectors called *sentiment-aware word embeddings* (SAWEs) that simultaneously involve the semantic and sentimental characteristics of words.

PCA, here, does not consider choosing certain features and discarding the others. Rather, it generates some new features that can provide a good overview of our data. Figure 4.5. depicts the PCA components in contrast to original ones.

Therefore, it can be considered as a feature extraction technique that combines input variables in such a way that less important variables are omitted while the most valuable parts remain.

In addition, PCA tries to find properties that allow the original characteristics to be regenerated from the new ones as well as possible. These new characteristics are called principle components (as shown in Figure 4.5.) and PCA calculates the input data projection for these components so that they have the maximum spread or variance.

Figure 4.6. shows the projection of data points on principle components (or Eigenvectors). In this figure, the red lines perpendicular to the Egv. 1 indicate the distance between the projected points and the original data points. Moreover, this distance indicates an error in regenerating the original data from the projected data. Therefore, PCA tries to minimize this error as well as maximize projection variance on the principle components.

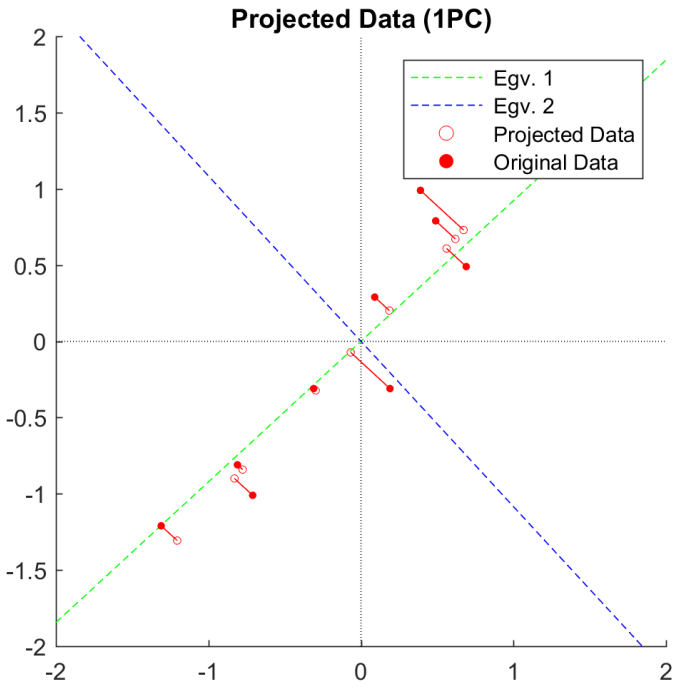


Figure 4.6. Projection of data points on principle components (Eigenvectors)

In Figure 4.6., by the rotation of the Egv. 1, the spread or variation of the data projected is reduced. On the other hand, this increases the distance of the projection and, consequently, the regeneration error.

As a result, PCA looks for components and projections that maximize the variance of projection and minimize regeneration error. These projections are considered to be new features by which the input data is represented.

5. Senti-Contextualized Learning Approach

In the refinement approach, we embedded the prior polarity information of words into word vectors using a neural network model. In the second approach, we learn word embeddings from contexts that not only indicate the contextual relationships between words, but also distinguish the sentimental association between words.

In this approach, we differentiate the context of subjective words through the learning process instead of using sentiment-specific loss functions. This approach is implemented in two different ways for the Skip-gram and GloVe models, each using a different training technique.

In the Skip-gram model, we change the model where training examples are generated. For example, Figure 5.1. shows the context words for the word w_t when the window size is 2. For target word w_t , the Skip-gram model calculates the probability of each context word given the target word, i.e. $P(w_{t-2}|w_t)$. As shown in Figure 5.1., we add a synthetic word into the context of the target word if it is subjective in the sentiment lexicon. In fact, in addition to any context word, the model calculates $P(w_{pos}|w_t)$ or $P(w_{neg}|w_t)$ if the target word is positive or negative.

However, when the model is trained with the negative sampling approach, we avoid selecting negative samples with the same polarity. For example, for the target word w_t with positive polarity, while the model considers (w_{pos}, w_t) as a positive example, $(w_{pos}, \hat{w}_t)$ is considered negative if the word $\hat{w}_t$ does not have a positive polarity (it can be a neutral or negative word). The detail of negative sampling is shown in Figure 5.2.

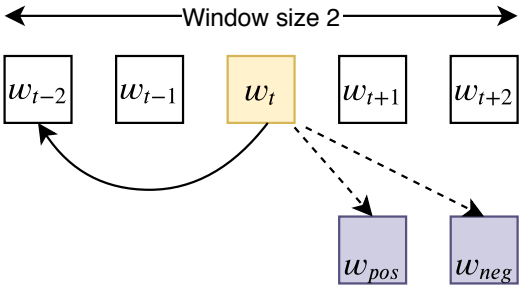


Figure 5.1. Example of context words for embedding learning models

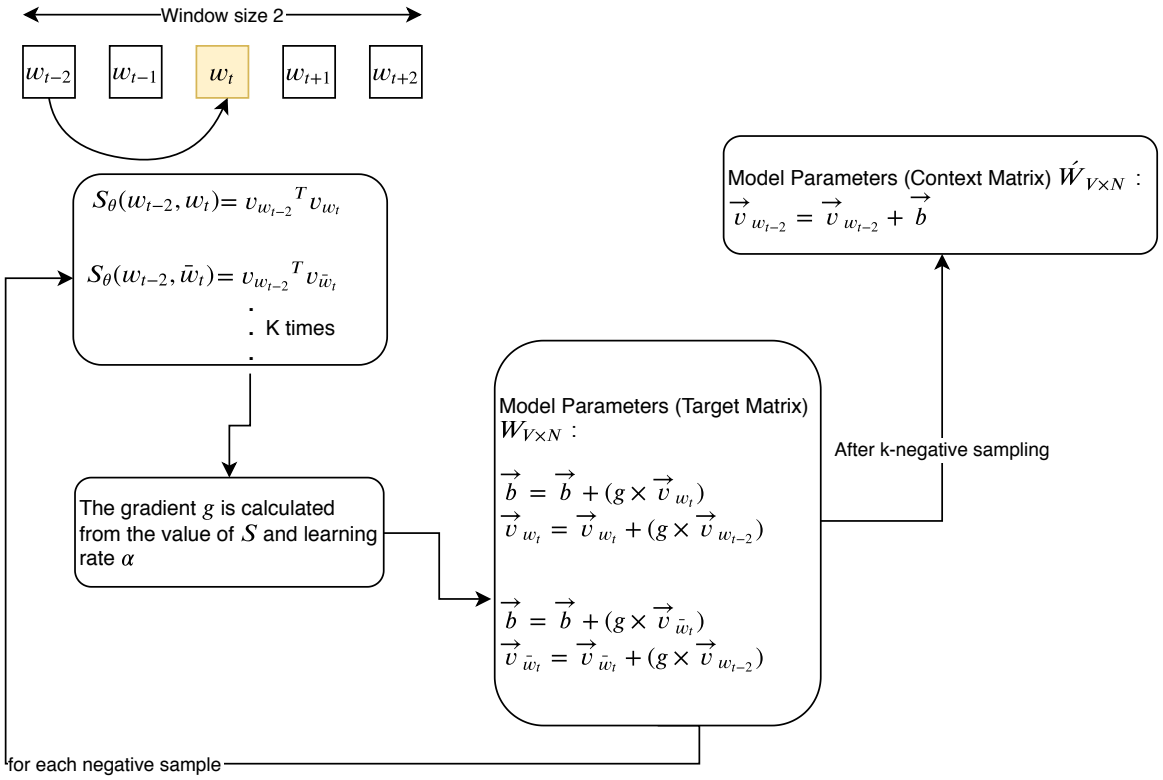


Figure 5.2. Details of the negative sampling in the Skip-gram model

In the above method, the two synthetic words always play the role of context words for any subjective word. From Figure 5.1., we always predict a subjective context when the target word is given, i.e. $P(w_{pos}|w_t)$ or $P(w_{neg}|w_t)$. However, the reverse case is not considered; we do not predict a real subjective word given a synthetic word. In other words, not one of those two synthetic words is considered a target word to predict a subjective word, i.e. $P(w_t|w_{pos})$ or $P(w_t|w_{neg})$.

This is important because two types of vectors are considered model parameters in the Skip-gram model. This means each word has two vectors. According to the architecture of the Skip-gram model shown in Figure 5.3., $\mathbf{W}$ and $\mathbf{W}'$ are the model parameters with respect to target and context words, respectively. As shown in Figure 5.2., all these parameters exert influence on each other throughout the training. Therefore, in the method noted above the context vector of w_t in matrix $\mathbf{W}'$ is not tuned when the word w_{pos} or w_{neg} is given.

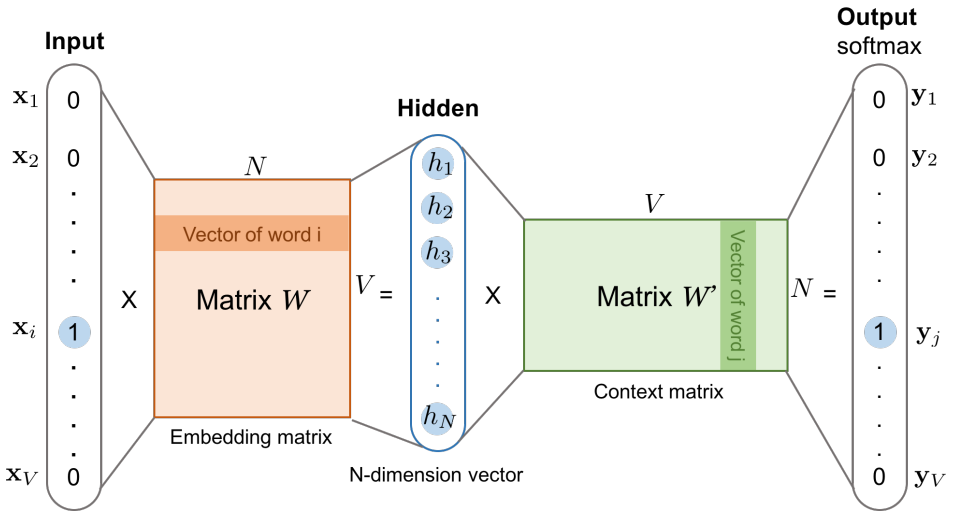


Figure 5.3. Architecture of the Skip-gram model

In the later approach that is used in the GloVe model, we take this issue into account by considering the synthetic words in both target and context positions. To this end, we simply add the synthetic words next to (or before) each subjective word in the training corpus without any change to the learning algorithm. For two examples given in Section 1.1., adding synthetic words makes the following changes in the context of words 'good' and 'bad':

1. "This film is very $< pos >$ good"
2. "This film is very $< neg >$ bad"

As can be seen, adding the synthetic words $< pos >$ and $< neg >$ differentiates the context of word 'good' from word 'bad'.

GloVe factorizes the logarithmic word-context matrix in which the loss function of factorization is weighted based on the frequency of co-occurrences as shown in Equation 28. Therefore, the errors caused by frequent co-occurrences are heavily penalized than the infrequent ones. However, the GloVe is trained from non-zero co-occurrences and does not consider the negative samples (word-context pairs with zero value). This is the difference between training on the GloVe and Skip-gram models.

$$J = \sum_{i,j=1}^{\mathrm{V}} f(\mathrm{X}_{i,j}) \left(\mathrm{w}_i^T \tilde{\mathrm{w}}_j + \mathrm{b}_i + \tilde{\mathrm{b}}_j - \log \mathrm{X}_{i,j} \right)^2 \tag{28}$$

In Equation 28, $X_{i,j}$ denotes the co-occurrence frequency of the words $\mathrm{w_i}$ and $\mathrm{w_j}$ in the word-context matrix, $\mathrm{b_i}$ and $\tilde{\mathrm{b}}_j$ are considered bias in the model with respect to word vectors $\mathrm{w_i}$ and $\mathrm{w_j}$, respectively. In this way, adding synthetic words next to subjective words will make it possible to distinguish between word vectors with opposite sentiments.

This simple approach provides the following advantages:

1. Capturing the sentimental association between words along with their contextual associations.

2. Distinguishing the context of two opposite-sentiment words where they appear in similar semantic contexts.

3. Unlike the previous supervised methods, where all the words in the sentence are affected by the sentiment polarity of the sentence, in our method, the sentiment information only affects the words in the context-window. We have used the default window size of 5 for both Skip-gram and GloVe models in our experiments. A large window size can reduce the effect of synthetic words, and the small size can lead model to learn more sentimental aspects of words.

Using the overall polarity of a sentence or text in the learning of word embeddings may not provide correct information about the individual words. This is because word embeddings focus on the contextual properties of words, while full text information reveals the compositional properties of words. For example, a set of objective words in a sentence may express a particular sentiment. However, this does not change the semantic aspects of the individual words; no sentiment can be expressed by any of those words in combination with other words.

6. Experiments

In this section, we evaluate our proposed sentiment-aware word embeddings in the binary, ternary, and fine-grained sentiment classification. The proposed refinement method is evaluated by two types of pre-trained word embeddings achieved from the Skip-gram (called Word2vec) and GloVe models. The performance of refined word embeddings is compared to conventional embeddings and other state-of- the-art methods. The Skip-gram and GloVe embedding models are also used to evaluate the second approach to learning embeddings from sentiment-aware contexts.

6.1. Experimental Setting

6.1.1. Datasets

Two widely used datasets, SemEval-2013 [69] and SST [70] are employed in our experiment. The SemEval dataset is a set of tweets with positive, negative and neutral labels. This dataset has a standard split of train, dev, and test sets with 9684, 1654 and 3813 tweets, respectively.

On the other hand, the SST (Stanford Sentiment Treebank) dataset is a set of movie reviews with two types of annotations:

1. *binary* with two positive and negative annotations
2. *fine-grained* with five, very positive, positive, neutral, negative, and very negative annotations.

Both types have a standard split of train / dev / test sets:

1. 6920 / 872 / 1821 for binary
2. 8544 / 1101 / 2210 for fine-grained

In most research, these two datasets were widely used as a benchmark for evaluating sentiment classification models. In this study, we compare the performance of sentiment-aware word embeddings with contextual embeddings when used in three types of sentiment classification: binary, ternary and multi-class.

6.1.2. Word embeddings

The word embeddings used in our refinement method include Word2vec (Skip-gram) and GloVe with 300 dimensions pre-trained on GoogleNews and common Crawl 840B, respectively. Based on the refinement method explained in Section 4., we learn $m - dimensional$ senti-embeddings and combine them with the input embeddings using PCA. The result vectors are called SAWEs (Sentiment Aware Word Embeddings).

We also evaluate the performance of SAWEs without the PCA method. In this case, SAWEs are achieved by the concatenation of the input embeddings with senti-embeddings. They are called SAWE(CAT) in the experiments. We set m to 30 and 100, 10% and 33% of the size of the input embeddings, to exert 10% and 33% of the sentimental influence on the input vectors.

This is because most of deep learning methods used in sentiment analysis require contextual information to extract sentiment-specific features from the raw text. This is the reason for the success of deep learning methods in sentiment analysis against shallow learning. This issue has also been addressed in previous refinement methods, so that the refined embeddings should not be far away from the original vectors [11].

In our second approach, the Skip-gram and GloVe models are used to learn word embeddings from a subset of wiki-dump[1] 2018 texts with almost 1B tokens as well as datasets used in this study. For the Skip-gram model, we modified its C implementation to add two synthetic words, namely $< pos >$ and $< neg >$, to the context of subjective words. We used the default parameters in the training:

- window size of 5
- 5 negative samples
- 10 iterations

The Cython implementation was adopted for the Glove model with default settings and the same parameters as the Skip-gram.

[1] https://dumps.wikimedia.org/enwiki/

6.1.3. Classifiers

According to recent studies, four kinds of classifiers have achieved more successful results than the others in sentiment classification: Convolutional Neural Network (CNN), Long Short Term Memory (LSTM), Bidirectional LSTM (BiLSTM), and Logistic Regression (LR) [19, 71, 72]. In addition, these classifiers have already been performed on the SemEval and SST benchmarks. They are therefore used to evaluate the performance of the proposed word embeddings in the experiments. For CNN, we use the static method proposed in [49] with its default settings for both benchmarks. For LSTM and BiLSTM, the methods proposed in [6] are implemented with suggested hyperparameters.

To observe the effect of word embeddings, we consider the embedding layer of all models to be non-trainable. Because the three previous classifiers are sequential (deep) models, we add a bag-of-words (shallow) model using LR to compare the performance of conventional and sentiment-aware word embeddings on both types of models.

In bag-of-words models, phrase or sentence representation is independent of the word order. Therefore, a sentence is represented by a combination of constituent words. To this end, we use the LR model with *min*, *max*, and *average* pooling layers, following [59], to represent the input sentence. This is a simple and effective method in vector-based semantics for learning compositionality.

6.1.4. Implementation details

We implemented all baseline methods using the Keras library with the TensorFlow backend. For the initial learning rate and other hyperparameters, we adopted the default values provided by Keras and those suggested in the original papers cited above. We selected the best model according to the accuracy of the classification in the development set. Moreover, we used the development set to select the best dropout, regularization, batch size, and epoch size in our models. Table 6.1. summarizes the hyperparameters selected for each classifier.

Table 6.1. Hyper parameters used in each classification model.

Parameter Name	LR	CNN	LSTM	BiLSTM
Filter Sizes		3, 4, 5		
# of Filters		100		
Pooling Size		Seq. Len. - FilterSize + 1		
Dropout		0.5		
LSTM Dim.			168	168
Regularization		L2(0.01)	L2(0.01)	L2(0.01)
Optimizer	Adam	Adam	Adam	Adam
Batch Size	50	50	25	25
Epoch	40	40	20	20

6.1.5. Evaluation metrics

For SemEval, we adopt the macro-averaged F1 score over positive, negative, and neutral categories as well as the official evaluation metric that uses the macro-averaged F1 score over positive and negative categories. The macro-averaged F1 score is calculated by the average of the individual F1 scores over categories. The F1 score is known as the harmonic mean of precision (P) and recall (R) and is calculated as:

$$F1 = \frac{2PR}{P+R} \tag{29}$$

$$P = \frac{TP}{TP+FP} \tag{30}$$

$$R = \frac{TP}{TP+FN} \tag{31}$$

Where TP, FP, and FN are true positives, false positives, and false negatives, respectively. Accuracy is used as the evaluation metric for both binary and fine-grained tasks in the SST benchmark. Additionally, each experiment is run 3 times and the average results are reported with a standard deviation within the parentheses.

6.2. Empirical Results

6.2.1. Refinement approach results on the SST benchmark

Tables 6.2. and 6.3. report the results of the SST benchmark. Table 6.2. indicates the performance of SAWEs obtained from the proposed refinement method for Word2vec pre-trained word embeddings using four classification models. As can be seen, in most cases, refined embeddings outperform Word2vec. In the binary task, the accuracy of the LSTM and BiLSTM models increases from 85.8 and 86.3 to 87.4 and 87.2, respectively. The same situation can be observed in the fine-grained task, so that the improvement is more robust in the four classification models, especially in the LSTM (accuracy is improved from 47.6 to 50.6).

Table 6.2. Accuracy of sentiment classification models over SST benchmark using Word2vec and refined word embeddings

Word Embeddings	LR	CNN	LSTM	BiLSTM
	Binary			
Word2vec	79.7 (0.4)	85.1 (0.5)	85.8 (0.7)	86.3 (0.6)
SAWE(CAT30)	80.3 (0.2)	**85.6** (0.2)	**87.4** (0.2)	86.8 (0.4)
SAWE(CAT100)	**81.1** (0.4)	85.0 (0.3)	86.7 (0.4)	86.2 (0.5)
SAWE(PCA30)	80.4 (0.1)	85.1 (0.4)	85.3 (0.6)	86.2 (0.6)
SAWE(PCA100)	80.0 (0.2)	85.3 (0.2)	86.2 (0.4)	**87.2** (0.2)
	Fine-grained			
Word2vec	41.4 (0.4)	44.5 (0.5)	47.6 (0.2)	48.7 (0.4)
SAWE(CAT30)	42.3 (0.2)	46.2 (0.2)	48.5 (0.4)	47.9 (0.6)
SAWE(CAT100)	**42.4** (0.2)	**46.3** (0.4)	48.9 (0.2)	49.1 (0.5)
SAWE(PCA30)	42.0 (0.4)	45.2 (0.2)	48.5 (0.2)	49.8 (0.1)
SAWE(PCA100)	41.5 (0.5)	44.4 (0.6)	**50.6** (0.2)	**49.8** (0.2)

Table 6.3. shows the results of the refined word embeddings of the GloVe model over SST benchmark. The results of binary and fine-grained tasks indicate that the refinement method performs better on the Glove pre-trained word embeddings and achieves state-of-the-art results. For example, SAWE(PCA30) increases the accuracy of the BiLSTM model from 47.2 to 52.1 in the fine-grained task. The performance of the LSTM model improves with all the refined embeddings in the binary task, especially with SAWEs using PCA.

Table 6.3. Accuracy of sentiment classification models over SST benchmark using GloVe and refined word embeddings

Word Embeddings	LR	CNN	LSTM	BiLSTM
	Binary			
GloVe	79.8 (0.2)	84.1 (0.4)	85.4 (0.6)	88.0 (0.2)
SAWE(CAT30)	80.2 (0.2)	85.1 (0.3)	87.9 (0.4)	87.5 (0.4)
SAWE(CAT100)	79.6 (0.3)	85.3 (0.2)	87.7 (0.4)	87.8 (0.4)
SAWE(PCA30)	**80.4** (0.1)	85.2 (0.2)	87.1 (0.5)	**88.5** (0.2)
SAWE(PCA100)	79.7 (0.2)	**85.6** (0.1)	**88.1** (0.3)	88.3 (0.2)
	Fine-grained			
GloVe	40.2 (0.4)	44.2 (0.3)	47.2 (0.3)	47.2 (0.4)
SAWE(CAT30)	41.3 (0.3)	45.3 (0.2)	46.4 (0.5)	49.6 (0.5)
SAWE(CAT100)	**41.9** (0.6)	45.4 (0.3)	**49.2** (0.4)	50.7 (0.7)
SAWE(PCA30)	41.7 (0.3)	**45.5** (0.2)	47.1 (0.3)	**52.1** (0.8)
SAWE(PCA100)	41.3 (0.5)	44.6 (0.3)	49.1 (0.2)	49.9 (0.7)

To evaluate the performance of our sentiment-aware word embeddings, we compare our findings with state-of-the-art results. Table 6.4. shows the results of nine groups of methods (including our method) that have achieved more successful results on the SST benchmark.

In this table, while the first four groups focus on generic algorithmic techniques: (1) CNN [49], (2) Tree-LSTM [6], (3) Multi-Timescale-LSTM (MT-LSTM) [7], and (4) BiLSTM with attention mechanism and convolutional layer (AC-BiLSTM) [51], the latter methods leverage word sentiment knowledge into deep learning models. For example, SAAN [20] and MEAN [73] incorporate sentiment knowledge resources into deep learning models using a sentiment specific attention network mechanism.

In this table, the last three groups focus on sentiment-specific word embeddings and perform deep learning models with this kind of embeddings. According to the results, the BiLSTM model with our proposed sentiment-aware word embeddings (SAWE-GloVe-PCA30) achieves the best result with the accuracy of 52.1 on the fine-grained task and outperforms Tree-LSTM [6], lexicon-based BiLSTM [54], and sentiment specific attention based models [20, 73].

In the binary task, the lexicon-based BiLSTM [54] has achieved the best accuracy of 89.2 using the SentiWordNet [74] lexicon. This method is context-sensitive and modifies the prior polarity values of words with respect to their usage in the context. Therefore, the

Table 6.4. State-of-the-art results on the SST benchmark. The last block shows our results. The numbers shown in parentheses indicate the *P* value between the corresponding accuracy and our accuracy in the last row. *P* values less than 0.1 are shown by an underline.

Method	**Binary**	**Fine**
CNN-static [49]	86.8 (0.0593)	45.5 (0.0000)
CNN-non-static [49]	87.2 (0.1150)	48.0 (0.0031)
CNN-multichannel [49]	88.1 (0.3519)	47.4 (0.0009)
Dependency TreeLSTM [6]	85.7 (0.0058)	48.4 (0.0069)
Constituency TreeLSTM-GloVe-fixed [6]	87.5 (0.1761)	49.7 (0.0548)
Constituency TreeLSTM-GloVe-tuned [6]	88.0 (0.3191)	51.0 (0.2327)
MT-LSTM (S2F) [7]	86.7 (0.0494)	48.9 (0.0165)
MT-LSTM (F2S) [7]	87.2 (0.1150)	49.1 (0.0233)
AC-LSTM [51]	87.6 (0.2004)	48.2 (0.0048)
AC-BiLSTM [51]	88.3 (0.4246)	48.9 (0.0165)
SAAN [20]	-	49.7 (0.0548)
MEAN [73]	-	51.4 (0.3191)
BiLSTM+SD-Lex [54]	88.1 (0.3519)	50.0 (0.0807)
BiLSTM+SWN-Lex [54]	**89.2** (0.2514)	51.1 (0.2514)
BiLSTM+Re(Word2vec) [11]	88.2 (0.3897)	49.6 (0.0484)
BiLSTM+Re(GloVe) [11]	88.6 (0.4641)	49.7 (0.0548)
AFTER+Senti-Net [10]	84.4 (0.0001)	-
DURING+Senti-Net [10]	84.2 (0.0000)	-
BilSTM+SAWE-Word2vec (PCA100) (our model)	87.2 (0.1150)	49.8 (0.0630)
BilSTM+SAWE-GloVe (PCA100) (our model)	88.3 (0.4246)	49.9 (0.0721)
BilSTM+SAWE-GloVe (PCA30) (our model)	88.5	**52.1**

compositional effects, such as negation, intensification, or shifting can be considered better in the classification model.

Despite this, our proposed GloVe-based SAWEs perform well with the BiLSTM model and outperform most of the complex methods such as Tree-LSTM, MT-LSTM, and AC-BLSTM in the binary task. In addition, SAWE-GloVe outperforms other refinement methods, such as those proposed in [10, 11] with the BiLSTM model. Unlike the previous methods such as Tree-LSTM and CNN, which have achieved their best results with trainable embeddings, our BiLSTM model performs well with static refined embeddings. This reveals the effectiveness of the proposed sentiment-aware word embeddings in the sentiment classification.

In order to observe the significance of the improvements, the Z-test with a significance level of 0.1 is applied to the accuracy of each classifier with our best model accuracy (BilSTM SAWE-GloVe(PCA30)). Here, the null hypothesis is that there is no significant difference

between the accuracy of the two classifiers over the test set. In Table 6.4., P values are reported in parentheses next to each classification accuracy. The P values less than 0.1 (shown by underline) fail the null hypothesis and indicate that there is a statistically significant difference between our best model accuracy and other models.

6.2.2. Refinement approach results on the SemEval-2013 benchmark

The performance of the proposed SAWEs is also evaluated on the twitter sentiment classification. Table 6.5. shows the results of Word2vec pre-trained word embeddings and its refined embeddings using four classification models on the SemEval dataset. The results indicate that our proposed SAWEs outperform pre-trained Word2vec embeddings and improve the performance of classification models in both binary and ternary classification tasks. The best results are achieved with the refined embeddings obtained from the concatenation method, so that SAWE(CAT100) is the best in most cases.

Table 6.5. Macro F1 score of sentiment classification models over SemEval-2013 benchmark using Word2vec and refined word embeddings

Word Embeddings	LR	CNN	LSTM	BiLSTM
	Binary			
Word2vec	86.6 (0.5)	89.2 (0.3)	88.1 (0.3)	87.6 (0.5)
SAWE(CAT30)	88.2 (0.3)	90.4 (0.2)	88.9 (0.4)	**89.3** (0.4)
SAWE(CAT100)	88.6 (0.1)	90.2 (0.2)	**89.0** (0.5)	88.2 (0.6)
SAWE(PCA30)	88.3 (0.2)	**90.4** (0.1)	87.7 (0.6)	86.8 (0.8)
SAWE(PCA100)	**89.1** (0.2)	89.8 (0.4)	88.2 (0.3)	88.9 (0.2)
	Ternary			
Word2vec	57.3 (0.6)	63.2 (0.5)	66.4 (0.4)	65.3 (0.5)
SAWE(CAT30)	60.7 (0.3)	64.5 (0.3)	66.7 (0.7)	68.0 (0.4)
SAWE(CAT100)	**60.7** (0.4)	**64.6** (0.3)	**69.0** (0.4)	**69.5** (0.3)
SAWE(PCA30)	59.2 (0.5)	63.4 (0.5)	67.2 (0.4)	66.2 (0.4)
SAWE(PCA100)	59.3 (0.6)	63.2 (0.4)	66.5 (0.5)	66.9 (0.4)

Similar observations can be seen for GloVe in Table 6.6. In the binary task, all SAWEs perform well with all classification models and, in most cases, outperform the GloVe pre-trained embeddings. In the ternary classification, the effect of SAWEs is more remarkable on the performance of the LR and CNN models.

The SemEval benchmark has been widely used in the evaluation of sentiment-specific embedding learning algorithms. Table 6.7. lists the best results of different methods. While the first three blocks indicate the macro-F1 scores of three sentiment-specific learning algorithms, the latter blocks show the results of the best system of Semeval-2013 [75], context-sensitive lexicon-based BiLSTM [54], refined Word2vec and GloVe embeddings with CNN [11], and our proposed SAWEs with CNN, LSTM and BiLSTM models, respectively.

Table 6.6. Macro F1 score of sentiment classification models over SemEval-2013 benchmark using GloVe and refined word embeddings

Word Embeddings	LR	CNN	LSTM	BiLSTM
	Binary			
GloVe	88.8 (0.4)	89.9 (0.5)	88.7 (0.4)	89.2 (0.3)
SAWE(CAT30)	**89.5** (0.1)	**91.0** (0.2)	88.7 (0.5)	89.4 (0.2)
SAWE(CAT100)	89.2 (0.1)	90.6 (0.3)	89.1 (0.3)	**89.9** (0.2)
SAWE(PCA30)	88.3 (0.1)	90.1 (0.3)	**89.8** (0.3)	89.5 (0.3)
SAWE(PCA100)	87.9 (0.1)	90.5 (0.2)	88.9 (0.4)	89.4 (0.1)
	Ternary			
GloVe	60.1 (1.2)	59.5 (1.3)	66.2 (0.5)	68.7 (0.5)
SAWE(CAT30)	63.3 (0.9)	63.3 (0.7)	66.2 (0.2)	**69.9** (0.3)
SAWE(CAT100)	**63.6** (0.4)	63.4 (0.5)	67.2 (0.4)	69.0 (0.4)
SAWE(PCA30)	62.0 (1.1)	61.8 (0.9)	67.3 (0.5)	68.4 (0.4)
SAWE(PCA100)	62.0 (0.8)	**64.0** (0.4)	**67.8** (0.4)	68.5 (0.4)

According to these results, the CNN model achieves the best macro-F1 score of 91.0 for the binary task in combination with our proposed SAWE-GloVe(CAT30). The same word embedding also produces the best macro-F1 score of 69.9 for the ternary task by using the BiLSTM model. The P values obtained from the Z-test with 0.1 significance level have been shown in parentheses and indicate the statistical significance of our best model (BiLSTM+SAWE-GloVe(CAT30)) with others.

Table 6.7. State-of-the-art results on the SemEval-2013 benchmark. The last block shows our results. The numbers shown in parentheses indicate the *P* value between the corresponding macro-F1 and our macro-F1 scores in the last row. *P* values less than 0.1 are shown by an underline.

Method	Binary	Ternary
SE-HyPred-Lex [59]	81.2 (0.0000)	63.3 (0.0000)
SE-HyRank-Lex [59]	84.0 (0.0000)	64.7 (0.0000)
SSWE_u [9]	85.0 (0.0000)	-
SSWE_u+NRC [9]	86.6 (0.0049)	-
SSWE_u+NRC-ngram [9]	86.5 (0.0037)	-
MSWE [61]	85.7 (0.0004)	-
NRC (Top System in SemEval2013)	84.7 (0.0000)	-
BiLSTM+TS-Lex [54]	87.6 (0.0455)	-
BiLSTM+S140-Lex [54]	88.0 (0.0934)	-
CNN+Re(Word2vec) [11]	88.4 (0.1710)	66.5 (0.0017)
CNN+Re(GloVe) [11]	90.5 (0.1378)	68.8 (0.0157)
CNN+SAWE-GloVe(CAT30) (our model)	**91.0** (0.0537)	63.3 (0.0000)
LSTM+SAWE-Word2vec(CAT100) (our model)	89.0 (0.3482)	69.0 (0.0161)
BiLSTM+SAWE-GloVe(CAT30) (our model)	89.4	**69.9**

6.2.3. Empirical results of senti-contextualized learning approach

This section reports the results of the proposed senti-contextualized approach in combination with the Skip-gram and GloVe models. Table 6.8. shows the results of the Skip-gram model over SST and SemEval datasets using four classification models. The results indicate that our proposed approach remarkably improves the performance of all classification models in both benchmarks. For example, in the SemEval dataset, the macro-F1 score of LR is improved from 86.5 to 87.4 and from 51.1 to 58.3 in the binary and ternary tasks, respectively. Improvements can also be seen in the performance of CNN, LSTM, and BiLSTM models.

The effect of the senti-contextualized approach is more remarkable on the GloVe model. Table 6.9. lists the classification performance of different models with two types of word embeddings. From this table, the SAWEs produce state-of-the-art results on the SemEval dataset using the BiLSTM model. We achieved macro-F1 scores of 91.3 and 69.4 for binary and ternary tasks, respectively.

Table 6.8. Results of the Word2vec model with senti-contextualized approach over SST and SemEval datasets

	SST		SemEval	
Word	Binary	Fine	Binary	Ternary
Embedding	Acc		Macro-F1	
LR(Word2vec)	73.9	37.6	86.5	51.1
LR(SAWE)	**78.1**	**39.1**	**87.4**	**58.3**
CNN(Word2vec)	81.5	41.7	88.1	55.9
CNN(SAWE)	**83.0**	**43.7**	**88.7**	**59.2**
LSTM(Word2vec)	81.3	45.7	87.4	65.8
LSTM(SAWE)	**84.0**	**46.7**	**89.8**	**66.4**
BiLSTM(Word2vec)	81.9	44.2	88.8	62.6
BiLSTM(SAWE)	**84.9**	**45.6**	**90.1**	**66.0**

Table 6.9. Results of the GloVe model with senti-contextualized approach over SST and SemEval datasets

	SST		SemEval	
Word	Binary	Fine	Binary	Ternary
Embedding	Acc		Macro-F1	
LR(GloVe)	74.5	38.4	87.6	54.3
LR(SAWE)	**79.8**	**40.3**	**88.5**	**59.5**
CNN(GloVe)	82.1	42.0	88.4	57.4
CNN(SAWE)	**84.3**	**44.6**	**89.0**	**60.7**
LSTM(GloVe)	83.4	45.2	87.6	65.5
LSTM(SAWE)	**85.2**	**46.4**	**89.5**	**66.9**
BiLSTM(GloVe)	84.7	46.1	89.0	67.3
BiLSTM(SAWE)	**87.0**	**47.4**	**91.3**	**69.4**

6.3. Discussion and Evaluation

In this section, we evaluate the performance of the proposed refinement and learning approaches based on the empirical results obtained from the two benchmarks. To this end, the paired *t*-test is applied to the results of the four classification algorithms. Here, our objective is to assess the significance of the improvements and to compare the performance of the four SAWEs with each other and with context-based word embeddings over four classification algorithms.

Table 6.10. shows the *P* values obtained from the paired *t*-test with a significance level of 0.1 over the accuracies of the four classifiers gained by SAWEs and each of Word2vec and GloVe pre-trained word embeddings in the SST benchmark. According to *P* values less than

0.1, while the proposed refinement approach with PCA 30 and 100 significantly improves the performance of GloVe embeddings in both binary and fine-grained tasks, the results are comparable in Word2vec. For instance, the improvement achieved by SAWE(PCA30) is statistically significant only for the fine-grained task, and SAWE(PCA100) provides a statistically significant improvement for the binary task.

Another remarkable point is that while SAWE(CAT100) significantly improves the performance of Word2vec and GloVe embeddings in the fine-grained task, it cannot maintain this superiority in the binary task. Overall, refinement with the PCA method preforms better than the concatenation approach in most cases in the SST benchmark.

Table 6.10. Paired *t*-test results of SAWEs with Word2vec and GloVe pre-trained embeddings over the accuracies of four classification algorithms in the SST benchmark

Proposed word embeddings	*P*-value			
	Word2vec		GloVe	
	Binary	Fine	Binary	Fine
SAWE(CAT30)	**0.0291**	0.1450	0.1720	0.1225
SAWE(CAT100)	0.1280	**0.0155**	0.2330	**0.0120**
SAWE(PCA30)	0.4633	**0.0025**	**0.0998**	**0.0857**
SAWE(PCA100)	**0.0314**	0.1219	**0.0886**	**0.0274**

In Table 6.11., the results of the paired *t*-test can be seen for the SemEval benchmark. Because all *P* values in the ternary task are less than 0.1, all the improvements made by Word2vec and GloVe SAWEs are statistically significant. However, in the binary task, the SAWEs with the concatenation approach perform better than the PCA.

Table 6.11. Paired *t*-test results of SAWEs with Word2vec and GloVe pre-trained embeddings over the macro-F1 scores of four classification algorithms in the SemEval-2013 benchmark

Proposed word embeddings	*P*-value			
	Word2vec		GloVe	
	Binary	Ternary	Binary	Ternary
SAWE(CAT30)	**0.0038**	**0.0349**	**0.0688**	**0.0512**
SAWE(CAT100)	**0.0171**	**0.0083**	**0.0039**	**0.0467**
SAWE(PCA30)	0.2668	**0.0370**	0.2314	**0.0587**
SAWE(PCA100)	**0.0596**	**0.0843**	0.4715	**0.0687**

From both benchmarks, it is observed that our refinement approach performs more consistently with 100-dimensional senti-embeddings than with *30-dimensional* senti-embeddings. However, increasing the size of senti-embeddings does not necessarily improve the performance of the refinement approach.

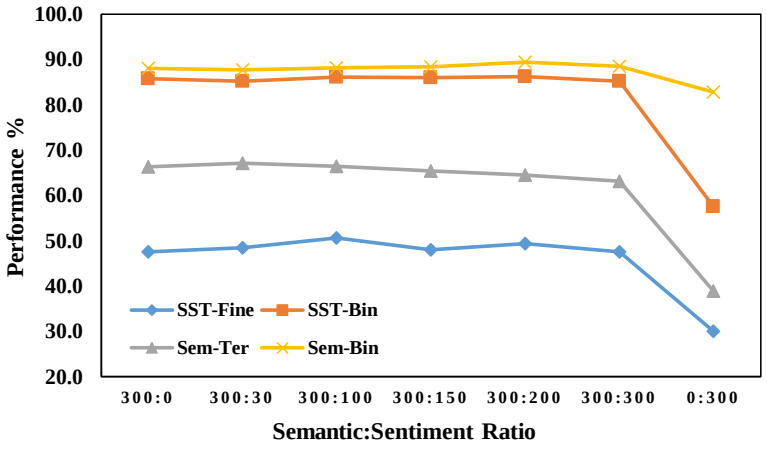
Figure 6.1. shows the results of the LSTM model on different ratios between the semantic and sentimental size of the SAWEs using the PCA method. From Figures 6.1.(a) and 6.1.(b)—respectively showing the results obtained from Word2vec and GloVe—the performance of the LSTM model in most cases decreases as the size of senti-embeddings grows. However, the best result for the binary task is achieved by the ratio of 300:200 in the SemEval dataset.

In addition, Figure 6.1. shows the performance of the LSTM model when SAWEs only include sentimental aspects of words, i.e. *300-dimensional* senti-embeddings denoted by the ratio of 0:300. The results indicate that the performance of the LSTM model drops sharply when only senti-emebddings are used. This means that word embeddings without semantic aspects of words cannot improve the performance of the LSTM model in sentiment classification. This reveals the importance of semantic knowledge in word embeddings, where they are used in deep learning models.

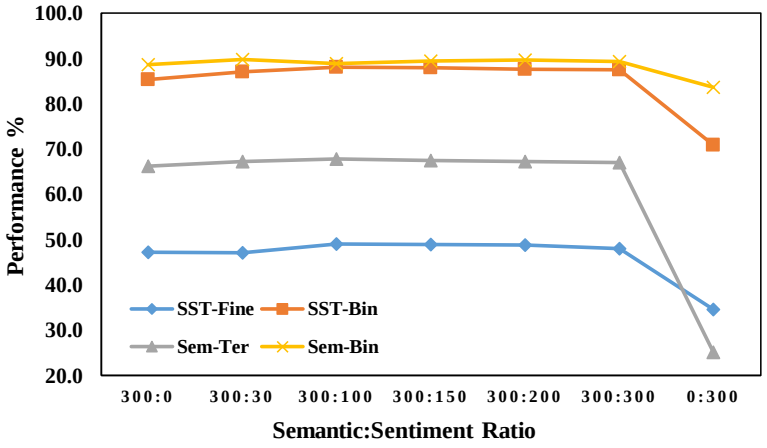
As noted above, the performance of SAWEs with PCA and concatenation approaches is comparable between two benchmarks. This is because the same hyperparameters were used for each classification model in all experiments. Therefore, finding the best hyperparameters with respect to each SAWE and benchmark may achieve more consistent results. Apart from this issue, the impact of SAWEs on the performance of sentiment classification models is remarkable.

Although the results of the PCA and concatenation approaches are comparable, the PCA approach can provide more benefits over the performance of deep learning models. The main advantage of the PCA method is to reduce the dimension of word embeddings; PCA reduces the concatenated dimension of word embeddings to smaller one with new inter-correlated quantitative variables (called principal components) explaining most of the variance in the concatenated dimension.

Therefore, PCA would be able to extract important information from high-dimensional word-embeddings and smooth over noise and provide better representation of data for learning algorithms. In addition, word embeddings are usually used to initialize words in deep learning models. However, the growth of the word embeddings dimension causes a great increase in



(a)



(b)

Figure 6.1. Results of the LSTM model on different ratios between the semantic and sentimental size of SAWEs (denoted in the semantic:sentiment form) using the PCA method. Figures (a) and (b) respectively show the results obtained from Word2vec and GloVe word embeddings

training parameters, which reduces the speed of convergence and the quality of results due to over-fitting.

As discussed in Section 4.1., senti-embeddings are learned from the sentiment lexicon that has been achieved from the merger of two lexicons: E-ANEW and Subjectivity clue. To demonstrate the impact of using multiple lexicons, we carry out an additional experiment and refine word embeddings using the E-ANEW lexicon and the PCA-100 method. Table 6.12. shows the performance of the BiLSTM modelwith two types of SAWEs learned from

single and multiple lexicons. Based on this experiment, the merger of two lexicons achieves the best results in the SST and SemEval benchmarks.

Table 6.12. Performance of the BiLSTM model using SAWEs(PCA100) obtained from single and multiple lexicons

	Lexicons	SST		SemEval	
		Fine	Binary	Ternary	Binary
Word2vec	E-ANEW	48.9 (0.3)	86.4 (0.4)	66.0 (0.6)	88.1 (0.3)
	E-ANEW + Subj. Clue	49.8 (0.2)	87.2 (0.2)	66.9 (0.4)	88.9 (0.2)
GloVe	E-ANEW	48.8 (0.6)	87.7 (0.3)	68.2 (0.3)	88.8 (0.2)
	E-ANEW + Subj. Clue	49.9 (0.7)	88.3 (0.2)	68.5 (0.4)	89.4 (0.1)

To observe the significance of improvements gained by the proposed senti-contextualized approach over four classification algorithms, the paired *t*-test is applied to the two sets of classification results achieved by the common word embedding models (Word2vec and GloVe) and our senti-contextualized approach previously reported in Tables 6.8. and 6.9. The *P* values are obtained at a significance level of 0.05 and reported in Table 6.13. Since all *P* values are less than 0.05, all improvements are statistically significant at 95% confidence level. This demonstrates the effectiveness of the senti-contextualized approach to learning word embeddings for sentiment analysis.

However, our empirical results achieved from the refinement approach are more robust than senti-contextualized because Word2vec and GloVe pre-trained word embeddings have been trained over hundreds of billions of tokens. Nevertheless, our senti-contextualized approach achieves the results as good as the refinement approach in the SemEval dataset despite training on a smaller text corpus.

Table 6.13. Paired *t*-test results (*P* values) of SAWEs obtained from senti-contextualized learning approach with common embedding models over four classification algorithms

Embedding Model	SST		SemEval	
	Binary	Fine	Binary	Ternary
Word2vec-SAWE	0.0071	0.0028	0.0228	0.0378
GloVe-SAWE	0.0185	0.0062	0.0192	0.0183

7. Conclusions & Future Work

7.1. Conclusions

This thesis proposed two empirically effective approaches to create sentiment-aware word embeddings. The first approach refines the pre-trained word embedding of Word2vec and GloVe models based on senti-embeddings learned from a neural network model. The refined embeddings encompass both the semantic and sentimental properties of words simultaneously.

Unlike state-of-the-art refinement methods, we modify input vectors of both subjective and objective words and produce sentiment-aware word embeddings in a different vector space. In this case, all input vectors are transformed into a new distributional space in which the vectors represent the semantic and sentimental aspects of words. Therefore, we do not need to know how far the refined embeddings should be away from the original vectors.

According to our empirical results, the proposed SAWEs improve the performance of classification models in different tasks, particularly in the fine-grained task that is more difficult than the binary. Furthermore, our SAWEs in combination with basic deep learning models outperform state-of-the-art methods over SST and SemEval-2013 benchmarks.

We also found that the proposed refinement approach works well when the sentimental aspects of words are encoded in pre-trained word vectors ranging from 10% to 33%. For example, while the 10% sentimental aspect was the best in some classification models, 33% of sentimental information was more effective in some other cases. However, finding the best hyperparameters for each classification model with respect to each refined embedding may provide more consistent results.

In addition, the proposed senti-contextualized approach to learning sentiment-aware word embeddings has achieved much better results than context-based approaches using Word2vec and GloVe models. Although we evaluated our approach on a small corpus (compared to pre-trained Word2vec and GloVe), it achieved state-of-the-art results on the SemEval benchmark using the GloVe model. This demonstrates the impact of the proposed approach in encoding the sentimental information of words into word embedding models.

7.2. Future Work

In our future work, we aim to use several kinds of word embeddings in deep learning models instead of using single sentiment aware embedding. We are planning to propose multi-channel and multi-stream models that extract different types of features from several word embeddings.

REFERENCES

[1] Nguyen Huy Tien, Nguyen Minh Le, Yamasaki Tomohiro, and Izuha Tatsuya. Sentence modeling via multiple word embeddings and multi-level comparison for semantic textual similarity. *Information Processing & Management*, 56(6):102090, **2019**.

[2] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, **2003**.

[3] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, **2008**.

[4] Manaal Faruqui and Chris Dyer. Improving vector space word representations using multilingual correlation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 462–471. **2014**.

[5] Wenpeng Yin and Hinrich Schütze. Learning word meta-embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1351–1360. **2016**.

[6] Kai Sheng Tai, Richard Socher, and Christopher D Manning. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*, **2015**.

[7] Pengfei Liu, Xipeng Qiu, Xinchi Chen, Shiyu Wu, and Xuanjing Huang. Multi-timescale long short-term memory neural network for modelling sentences and documents. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 2326–2335. **2015**.

[8] Meng Joo Er, Yong Zhang, Ning Wang, and Mahardhika Pratama. Attention pooling-based convolutional neural network for sentence modelling. *Information Sciences*, 373:388–403, **2016**.

[9] Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. Learning sentiment-specific word embedding for twitter sentiment classification. In

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), volume 1, pages 1555–1565. **2014**.

[10] Zhe Ye, Fang Li, and Timothy Baldwin. Encoding Sentiment Information into Word Vectors for Sentiment Analysis. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 997–1007. **2018**.

[11] Liang-Chih Yu, Jin Wang, K Robert Lai, and Xuejie Zhang. Refining word embeddings using intensity scores for sentiment analysis. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(3):671–681, **2018**.

[12] Mohammad Abu Alsheikh, Shaowei Lin, Dusit Niyato, and Hwee-Pink Tan. Machine learning in wireless sensor networks: Algorithms, strategies, and applications. *IEEE Communications Surveys & Tutorials*, 16(4):1996–2018, **2014**.

[13] Tobias Schnabel and Hinrich Schütze. Flors: Fast and simple domain adaptation for part-of-speech tagging. *Transactions of the Association for Computational Linguistics*, 2:15–26, **2014**.

[14] Cicero Nogueira dos Santos and Victor Guimaraes. Boosting named entity recognition with neural character embeddings. *arXiv preprint arXiv:1505.05008*, **2015**.

[15] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537, **2011**.

[16] Cicero Dos Santos and Maira Gatti. Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 69–78. **2014**.

[17] Zellig S Harris. Distributional structure. *Word*, 10(2-3):146–162, **1954**.

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

[19] Tao Chen, Ruifeng Xu, Yulan He, and Xuan Wang. Improving sentiment analysis via sentence type classification using BiLSTM-CRF and CNN. *Expert Systems with Applications*, 72:221–230, **2017**.

[20] Zeyang Lei, Yujiu Yang, and Min Yang. SAAN: A Sentiment-Aware Attention Network for Sentiment Analysis. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 1197–1200. ACM, **2018**.

[21] Chen Chaotao, Zhuo Run, and Ren Jiangtao. Gated recurrent neural network with sentimental relations for sentiment classification. *Information Sciences*, 502:268–278, **2019**.

[22] Gerard Salton, Anita Wong, and Chung-Shu Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, **1975**.

[23] Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, **1999**.

[24] Thomas Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57. **1999**.

[25] Rie Kubota Ando. Latent semantic space: Iterative scaling improves precision of inter-document similarity measurement. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 216–223. **2000**.

[26] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Kernel principal component analysis. In *International conference on artificial neural networks*, pages 583–588. Springer, **1997**.

[27] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, **2003**.

[28] Wray Buntine and Aleks Jakulin. Discrete component analysis. In *International Statistical and Optimization Perspectives Workshop "Subspace, Latent Structure and Feature Selection"*, pages 1–33. Springer, **2005**.

[29] Marco Baroni and Alessandro Lenci. Distributional memory: A general framework for corpus-based semantics. *Computational Linguistics*, 36(4):673–721, **2010**.

[30] Peter D Turney and Patrick Pantel. From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research*, 37:141–188, **2010**.

[31] Andrei Z Broder. On the resemblance and containment of documents. In *Proceedings. Compression and Complexity of SEQUENCES 1997 (Cat. No. 97TB100171)*, pages 21–29. IEEE, **1997**.

[32] Jeffrey Dean and Sanjay Ghemawat. MapReduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, **2008**.

[33] Behrang QasemiZadeh and Laura Kallmeyer. HHU at semeval-2017 task 2: Fast hash-based embeddings for semantic word similarity assessment. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 250–255. **2017**.

[34] Andriy Mnih and Koray Kavukcuoglu. Learning word embeddings efficiently with noise-contrastive estimation. In *Advances in neural information processing systems*, pages 2265–2273. **2013**.

[35] Andriy Mnih and Geoffrey Hinton. Three new graphical models for statistical language modelling. In *Proceedings of the 24th international conference on Machine learning*, pages 641–648. **2007**.

[36] Michael U Gutmann and Aapo Hyvärinen. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *Journal of Machine Learning Research*, 13(Feb):307–361, **2012**.

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

[38] Pengda Qin, Weiran Xu, and Jun Guo. A novel negative sampling based on TFIDF for learning word representation. *Neurocomputing*, 177:257–265, **2016**.

[39] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543. **2014**.

[40] Omer Levy and Yoav Goldberg. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 302–308. **2014**.

[41] Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems*, pages 2177–2185. **2014**.

[42] John A Bullinaria and Joseph P Levy. Extracting semantic representations from word co-occurrence statistics: stop-lists, stemming, and SVD. *Behavior research methods*, 44(3):890–907, **2012**.

[43] Peter D Turney. Domain and function: A dual-space model of semantic relations and compositions. *Journal of Artificial Intelligence Research*, 44:533–585, **2012**.

[44] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, **2017**.

[45] Alexandre Salle, Marco Idiart, and Aline Villavicencio. Matrix factorization using window sampling and negative sampling for improved word representations. *arXiv preprint arXiv:1606.00819*, **2016**.

[46] Alexandre Salle, Marco Idiart, and Aline Villavicencio. Enhancing the lexvec distributed word representation model using positional contexts and external memory. *arXiv preprint arXiv:1606.01283*, **2016**.

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

[48] Lin Yue, Weitong Chen, Xue Li, Wanli Zuo, and Minghao Yin. A survey of sentiment analysis in social media. *Knowledge and Information Systems*, pages 1–47, **2019**.

[49] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, **2014**.

[50] Igor Santos, Nadia Nedjah, and Luiza de Macedo Mourelle. Sentiment analysis using convolutional neural network with fasttext embeddings. In *2017 IEEE Latin American Conference on Computational Intelligence (LA-CCI)*, pages 1–5. IEEE, **2017**.

[51] Gang Liu and Jiabao Guo. Bidirectional LSTM with attention mechanism and convolutional layer for text classification. *Neurocomputing*, 337:325–338, **2019**.

[52] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *IEEE international conference on acoustics, speech and signal processing (icassp), 2013*, pages 6645–6649. IEEE, **2013**.

[53] Haşim Sak, Andrew Senior, and Françoise Beaufays. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *Fifteenth Annual Conference of the International Speech Communication Association*. **2014**.

[54] Zhiyang Teng, Duy Tin Vo, and Yue Zhang. Context-sensitive lexicon features for neural sentiment analysis. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1629–1638. **2016**.

[55] Behzad Naderalvojoud, Behrang Qasemizadeh, and Laura Kallmeyer. GermEval 2017: Shared Task on Aspect-based Sentiment in Social Media Customer Feedback. In *Proceedings of the GermEval 2017: Shared Task on Aspect-based Sentiment in Social Media Customer Feedback*, pages 18–21. Berlin, Germany, **2017**.

[56] Behzad Naderalvojoud, Behrang Qasemizadeh, Laura Kallmeyer, and Ebru Akcapinar Sezer. A cross-lingual approach for building multilingual sentiment lexicons. In *International Conference on Text, Speech, and Dialogue*, pages 259–266. Springer, **2018**.

[57] Duyu Tang, Bing Qin, and Ting Liu. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 1422–1432. **2015**.

[58] Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1*, pages 142–150. Association for Computational Linguistics, **2011**.

[59] Duyu Tang, Furu Wei, Bing Qin, Nan Yang, Ting Liu, and Ming Zhou. Sentiment embeddings with applications to sentiment analysis. *IEEE transactions on knowledge and data Engineering*, 28(2):496–509, **2016**.

[60] Huiwei Zhou, Long Chen, Fulin Shi, and Degen Huang. Learning bilingual sentiment word embeddings for cross-language sentiment classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 430–440. **2015**.

[61] Shufeng Xiong, Hailian Lv, Weiting Zhao, and Donghong Ji. Towards Twitter sentiment classification by multi-level sentiment-enriched word embeddings. *Neurocomputing*, 275:2459–2466, **2018**.

[62] Manaal Faruqui, Jesse Dodge, Sujay K Jauhar, Chris Dyer, Eduard Hovy, and Noah A Smith. Retrofitting word vectors to semantic lexicons. *arXiv preprint arXiv:1411.4166*, **2014**.

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

[64] Erik Cambria, Jie Fu, Federica Bisio, and Soujanya Poria. AffectiveSpace 2: Enabling affective intuition for concept-level sentiment analysis. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*. **2015**.

[65] Robert Speer, Joshua Chin, and Catherine Havasi. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Thirty-First AAAI Conference on Artificial Intelligence*. **2017**.

[66] Carlo Strapparava, Alessandro Valitutti, and Others. Wordnet affect: an affective extension of wordnet. In *Lrec*, volume 4, page 40. Citeseer, **2004**.

[67] Amy Beth Warriner, Victor Kuperman, and Marc Brysbaert. Norms of valence, arousal, and dominance for 13,915 English lemmas. *Behavior research methods*, 45(4):1191–1207, **2013**.

[68] Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*. **2005**.

[69] Preslav Nakov, Zornitsa Kozareva, Alan Ritter, Sara Rosenthal, Veselin Stoyanov, and Theresa Wilson. SemEval-2013 Task 2: Sentiment Analysis in Twitter. *Second Joint Conference on Lexical and Computational Semantics (*SEM)*, 2:312–320, **2013**.

[70] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642. **2013**.

[71] Duyu Tang, Bing Qin, and Ting Liu. Deep learning for sentiment analysis: successful approaches and future challenges. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 5(6):292–303, **2015**.

[72] Asad Abdi, Siti Mariyam Shamsuddin, Shafaatunnur Hasan, and Jalil Piran. Deep learning-based sentiment classification of evaluative text based on Multi-feature fusion. *Information Processing & Management*, 56(4):1245–1259, **2019**.

[73] Zeyang Lei, Yujiu Yang, Min Yang, and Yi Liu. A multi-sentiment-resource enhanced attention network for sentiment classification. *arXiv preprint arXiv:1807.04990*, **2018**.

[74] Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. Sentiwordnet 3.0: an enhanced lexical resource for sentiment analysis and opinion mining. In *Lrec*, volume 10, pages 2200–2204. **2010**.

[75] Saif M Mohammad, Svetlana Kiritchenko, and Xiaodan Zhu. NRC-Canada: Building the state-of-the-art in sentiment analysis of tweets. *arXiv preprint arXiv:1308.6242*, **2013**.

APPENDIX 1 - Article Published from Thesis

Behzad Naderalvojoud and Ebru Akcapinar Sezer. **Sentiment Aware Word Embeddings Using Refinement and Senti-Contextualized Learning Approach.** Neurocomputing, 405: 149–160, **2020**.

APPENDIX 2 - Thesis Originality Report