RICE UNIVERSITY

AWE: Attention Word Embedding

By

Shashank Sonkar

A THESIS SUBMITTED
IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE

Master of Science

APPROVED, THESIS COMMITTEE

Richard Baraniuk (Aug 31, 2020 14:21 ADT)

Richard G. Baraniuk

Victor E. Cameron Professor of Electrical and Computer Engineering

Santiago Segarra (Aug 31, 2020 13:27 CDT)

Santiago Segarra

W. M. Rice Trustee Assistant Professor of Electrical and Computer Engineering

Anshumali Shrivastava

Assistant Professor of Computer Science

HOUSTON, TEXAS

September 2020

ABSTRACT

AWE: Attention Word Embedding

by

Shashank Sonkar

Word embedding models learn semantically rich vector representations of words and are widely used to initialize natural processing language (NLP) models. The popular continuous bag-of-words (CBOW) model of word2vec learns a vector embedding by masking a given word in a sentence and then using the other words as a context to predict it. A limitation of CBOW is that it equally weights the context words when making a prediction, which is inefficient, since some words have higher predictive value than others. We tackle this inefficiency by introducing the *Attention Word Embedding* (AWE) model, which integrates the attention mechanism into the CBOW model. We also propose AWE-S, which incorporates subword information. We demonstrate that AWE and AWE-S outperform the state-of-the-art word embedding models both on a variety of word similarity datasets and when used for initialization of NLP models.

Acknowledgments

I would like to thank my advisor Dr. Richard G. Baraniuk for his guidance, trust in my abilities, and providing me freedom to explore my interests in the field of natural language processing. I would also like to thank Dr. Anshumali Shrivastava and Dr. Santiago Segarra for being on my thesis committee. Many thanks to my OpenStax mentor, Dr. Andrew E. Waters, for his time to discuss and brainstorm countless ideas, being patient with my novice writing skills, and his help to improve them immensely. I would also like to thank my lab mate, Jack Wang, for being my unofficial mentor and his continuous support through the last three years. Thanks to Yu Zhu and Aditya Desai for providing critical feedback on various aspects of this project and helping me navigate smoothly through the turbulent life of a graduate student. At last, I would like to thank my family and friends for their endless encouragement.

Contents

Illustrations

Tables

Chapter 1

Introduction

Word embedding models learn vector representations of words such that words that are semantically related are close to each other in the vector space. Popular word embedding models include word2vec [1,2], GloVe [3], and fastText [4]. Word embedding models are often used to initialize deep learning models for various natural language processing (NLP) tasks such as machine translation [5], part-of-speech tagging [6], and sentiment analysis [7], leading to improved performance over training the models from scratch.

The core idea underlying the training methodology of all the above embedding models is that "a word is characterized by the company it keeps" [8]. For instance, the continuous bag-of-words (CBOW) model of word2vec predicts a randomly selected word in a sentence using the other words in the sentence as a context. The context words are typically treated equally by these models. However, clearly some words in the context will be more predictive of the masked word than others, and intuitively, should be weighted more heavily.

A promising remedy that treats each context word differently is the *attention mechanism* [9], which enables learning the relative importance of input features for the prediction of the desired output. Attention has become the de facto standard and state-of-the-art in modern NLP for a range of different tasks [10, 11, 12]. However, to date, there have been only a few attempts to employ it to improve the performance and interpretability of word embedding models [13, 14].

1.1 Contributions

In this thesis, we propose the *Attention Word Embedding* (AWE), a new word embedding model that integrates the attention mechanism into the CBOW model of word2vec.

AWE consists of two components. First, AWE uses a variant of self-attention [9] to narrow down relevant words in the context for the prediction of the masked word. In contrast to prior work [13, 14], the attention weights are a function of both the context words and the target/masked word. Second, AWE embeds the context words and the masked word representations in a shared subspace, since both representations embody the meaning of the word. Note that this is in sharp contrast with CBOW, which learns two different embeddings for each word, one when the word is present in the context and another when it is masked. We also propose a new initialization scheme for the embedding matrices in AWE, tailored to the masked word prediction task used for training AWE.

We also introduce a variant of AWE, AWE-S, which leverages subword information to enrich the word vectors. The idea to use subwords is inspired from fastText [4], which defines subwords of a word as its character n-grams. Incorporating subword information is particularly valuable for improving word vector representations for rare and unseen words. The morphological structures of the word, such as prefixes, suffixes, and stems help to unravel the underlying meaning of words. For instance, the meaning of the word *unknowingly* can be deduced using its character n-grams, such as *un*, *know*, *ing*, and *ly*.

Our experimental evaluations show the superior performance of AWE and AWE-S than many existing word embedding models in both language modeling tasks and a number of downstream NLP applications including natural language inference,

sentence semantic relatedness, and paraphrase detection. Furthermore, we analyze and interpret the role of the attention weights in AWE, which provide interesting insights into the workings of the attention mechanism.

Chapter 2

Background

2.1 Word2Vec

Word2vec is one of the standard word embedding models [1, 2]. There are two architectures proposed for word2vec: CBOW and Skip-Gram. We focus on the CBOW model in this work.

CBOW predicts a masked word using its context (*fill in the blanks* model). For each word, it learns two vectors to represent the two roles that a word can perform - first, when it is present in the context of the masked word, and second, when it is masked.

Let $U = [\boldsymbol{u}_1, ..., \boldsymbol{u}_N]^T \in \mathbb{R}^{N \times D}$ and $V = [\boldsymbol{v}_1, ..., \boldsymbol{v}_N]^T \in \mathbb{R}^{N \times D}$ where N is the size of the vocabulary, and D is the size of the word vector. U models the first role and is used to calculate the context vector, $\boldsymbol{c}$, given by

$$\boldsymbol{c} = \sum_{i \in [-b,b]-\{0\}} \boldsymbol{u}_{w_i}, \tag{2.1}$$

where b is the size of the context window and w_i is the index of each word (w_0 is the index of the masked word; the rest are the indices of the context words). V models the second role and learns the masked word vector for w_0, given by $\boldsymbol{v}_{w_0}$. The probability p of w_0 to occur in the context of $\{w_{-b}, .., w_{-1}, w_1, .., w_b\}$ is given by

$$p(w_0 | W_{[-b,b]-\{0\}}) \propto \exp \boldsymbol{v}_{w_0}^T \boldsymbol{c}. \tag{2.2}$$

The Skip-Gram model of word2vec is similar to CBOW. The key difference is that

it predicts the context words using the masked word.

2.2 FastText

FastText exploits the morphology of the words to learn a richer semantic representation of the words as compared to word2vec [4]. A set of words like {*exploits, exploited, exploitation*} are different manifestations of the same word *'exploit'*. These words share a common underlying meaning which can be deciphered from their morphology. A word embedding algorithm can take advantage of this fact to model the meaning of the words in a more effective way.

To capture this rich information embedded in the structure of the words, fastText breaks a word into its character n-grams. Character n-gram set for an example word say *'misfit'* is given by {*<mis, isf, sfi, fit>, <misfit>*} for $n = 3$. Special symbols $<$ and $>$ mark the beginning and end of words. Also, the word itself is added to its character n-gram set. Embedding of a word is subsequently represented as the sum of embeddings of its character n-grams. The skipgram objective of word2vec is used to learn both the word and n-gram embeddings.

Let $U = [\boldsymbol{u}_1, ..., \boldsymbol{u}_M]^T \in \mathbb{R}^{M \times D}$ and $V = [\boldsymbol{v}_1, ..., \boldsymbol{v}_N]^T \in \mathbb{R}^{N \times D}$ where M is the size of dictionary of n-grams, N is the size of the vocabulary, and D is the size of the n-gram/word vector. Let w_m be the masked word which we shall predict using one of the context words, say w_c. Let the set S_{w_m} contain the indices of character n-grams of w_m. Then, the probability p of w_m to occur in the context of w_c is given by

$$\boldsymbol{c} = \sum_{e_j \in S_{w_m}} \boldsymbol{u}_{e_j}\,, \qquad p(w_m|w_c) \propto \exp \boldsymbol{v}_{w_c}^T \boldsymbol{c}\,.$$

Note that the word is represented as the sum of its character n-gram embeddings only when it is present in the context. When the word is masked, its embedding is

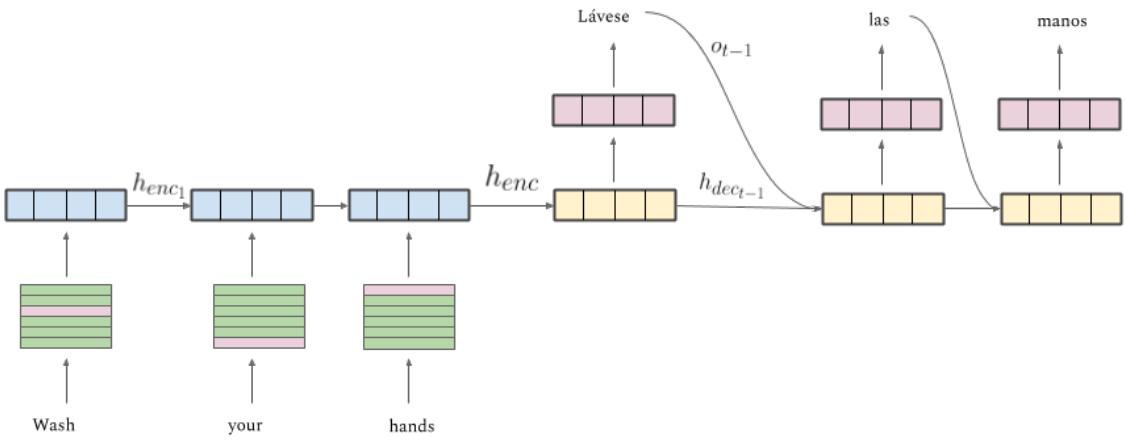

Figure 2.1 : A Simple RNN-based Encoder Decoder Sequence to Sequence Models.

not decomposed into character n-gram embeddings.

2.3 Attention

The attention mechanism is ubiquitous in NLP architectures. One of the first applications of the attention mechanism was in the sequence to sequence neural machine translation (NMT) models [5, 15]. A sequence to sequence NMT model takes as input a sequence of symbols (words or character n-grams) in a source language and predicts the corresponding sequence of symbols in a target language (see Figure 2.1) [16, 17]. These NMT models use variants of recurrent neural networks (RNNs) [18], like Long Short-Term Memory (LSTM) network [19], to tackle the arbitrary length of input and output sequences.

Let $\{x_1, x_2, ..., x_n\}$ be an input sequence to a standard RNN encoder-decoder based sequence to sequence model, and $\{y_1, y_2, ..., y_m\}$ be the corresponding output sequence. At any timestep t, the encoder RNN takes in input the hidden state at timestep $t-1$

and the input sequence at timestep t. The hidden state at timestep t evolves following the equation

$$h_t^e = \text{RNN}_{enc}(W_{enc}x_t, h_{t-1}^e), \tag{2.3}$$

where $W_{enc} \in \mathbb{R}^{N\times D}$ is the word embedding matrix for words in the source language, N is the size of vocabulary of the source language, D is the embedding dimension, and $x_t \in \mathbb{R}^N$ is an one-hot encoded vector which is 1 at the index corresponding to the word at that index, and 0 otherwise. The input to the decoder at timestep t is the output word by the decoder at timestep $t-1$ and the hidden state at timestep $t-1$. The dynamics of the decoder is governed by the following two equations.

$$o_{t-1} = \text{softmax}(h_{t-1})\,, \qquad h_t^d = \text{RNN}_{dec}(W_{dec}o_t, h_{t-1}^d)\,,$$

where $W_{dec} \in \mathbb{R}^{M\times D}$ is the word embedding matrix for words in the target language, M is the size of vocabulary of the target language, D is the embedding dimension, and $o_t \in \mathbb{R}^M$ is the output of the decoder which is the result of a softmax layer.

From (2.3), one can observe that RNN_{enc} compresses the entire input sequence into a fixed dimensional vector, h_n, which is the input to RNN_{dec}. It is difficult to encode the entire sequence information into a single vector [20]. The attention mechanism helped to overcome this limitation of the standard RNN encoder-decoder based sequence to sequence NMT models by allowing the decoder to have access to all the hidden states of the encoder. Equipped with the attention mechanism, the decoder can choose which of the encoder hidden states to extract the most information from which will help it to predict the next token.

Let's take the example depicted in figure 2.2 to elucidate the point. After the words *Lávese* and *las* have been predicted by RNN_{dec}, the decoder can use the attention mechanism to *attend* more on third hidden state of RNN_{enc} that recently had *hands*

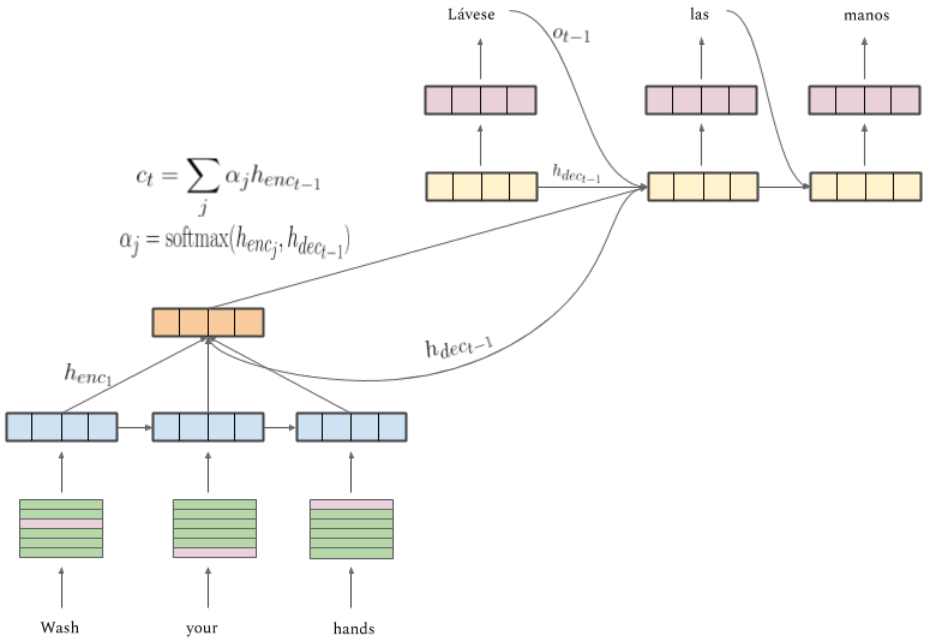

Figure 2.2 : A Simple RNN-based Encoder Decoder Sequence to Sequence Models with the Attention mechanism.

as the input, since it is the most important word in the input sequence that can help the decoder to predict the next word *manos*, the Spanish translation for *hands*.

At any given timestep, say t, the decoder uses its current hidden state (which contains the information of all the words translated so far) to calculate attention weights (given by α_j) to calculate a weighted combination of the encoder hidden states (given by c_{t-1}).

$$\alpha_j = \text{softmax}(H_{enc}, h^d_{t-1}) \,, \qquad c_{t-1} = \sum_{j=1}^{n} \alpha_j h^d_{t-1},$$

where j^{th} row of H_{enc} is h^e_j. With the information of all the words translated and the attention weights, c_{t-1} will ideally have significant contributions from the hidden states that are the most useful to predict the next output token. c_{t-1} is passed as an

additional input to RNN_{dec} at timestep t given by the following equation

$$h_t^d = \text{RNN}_{dec}(W_{dec}o_t, h_{t-1}^d, c_{t-1}).$$

Many NLP architectures have pitched the use of the attention mechanism using the above motivation that it helps the neural network to attend to relevant features. Some studies have probed the trained models to substantiate this claim, and find that it works as expected [21, 22]. However, some studies suggest otherwise and their experiments found that the attention weights are rather poor indicators of feature importance [23, 24]. Thus, the quest to answer why the attention mechanism is so effective is still ongoing and an area of active research.

Chapter 3

The Attention Word Embedding (AWE) Model and its Subword Variant

3.1 The AWE Model

We now explain AWE, our proposed word embedding model that incorporates the attention mechanism. AWE augments the CBOW model of word2vec with the attention mechanism in two different ways. First, we introduce two new matrices, a key matrix, $K \in \mathbb{R}^{N \times D'}$ and a query matrix, $Q \in \mathbb{R}^{N \times D'}$, where N is the size of the vocabulary and $D' \in \mathbb{Z}$. With the attention mechanism, the context vector $\boldsymbol{c}$ is not modeled as a simple sum in (2.1) but rather as a *weighted* sum of context word vector embeddings

$$\boldsymbol{c_{\mathbf{attn}}} = \sum_{i \in [-b,b]-\{0\}} a_{w_i} \boldsymbol{u}_{w_i} \, ,$$

where a_{w_i} is the attention weight of each context word vector $\boldsymbol{u}_{w_i}$ calculated using the key matrix K and the query matrix Q

$$a_{w_i} = \exp\left(\boldsymbol{k_{w_0}}^T \boldsymbol{q_{w_i}} \right) \, ,$$

Second, we share the weights between the context word embedding matrix and the masked word embedding matrix in our model, i.e., we set $U = V$. Sharing weights is a natural and intuitive choice, since both matrices embed the meaning of a word in its vector representation, and the meaning of the word remains the same irrespective of it occurring in the context window, or as the masked word. In CBOW, the choice is to

have two separate matrices, U and V, is justified as it leads to increase in performance. However, in AWE, the intuitive choice to have U same as V works better and adds interpretability to the model. On top of that, it has an added advantage. The number of parameters in AWE are much less as compared to CBOW even though AWE has one more matrix than CBOW. The reason is that the key matrix $K \in \mathbb{R}^{N \times D'}$ and the query matrix $Q \in \mathbb{R}^{N \times D'}$ are much smaller as compared to the value matrix, $V = [\boldsymbol{u}_1, ..., \boldsymbol{u}_D]^T \in \mathbb{R}^{N \times D}$. In our experiments, we set $D' = 50$ and $D = 500$.

3.1.1 Weight Initialization Techniques

Weight initialization is critical to the training and performance of deep neural networks [25]. Several methods have been proposed for initializing the weight matrices which include random initialization, standard normal initialization, Xavier initialization [26], and Kaiming initialization [27].

We propose a new initialization scheme for initializing the key and the query matrices which assigns an equal, non-zero weight to all words in the context at the start of the training. The weights for both the matrices are drawn according to

$$W \sim \mathcal{N}(\frac{1}{\sqrt{D'}}, 0.01),$$

where $\mathcal{N}$ is the normal distribution. It is easy to verify that for any masked word, say w_m, and any word in the context of the masked word, say w_c, the initial attention weight, given by $\boldsymbol{k}_{w_m}^T \boldsymbol{q}_{w_c}$, is centered around 1. Thus, in the beginning AWE mimics the CBOW method of word2vec which assigns equal weight to each context word. As the training progresses, AWE learns the relevance of each of the context words for the prediction of the masked word. For the value matrix, we initialize the weights using $\mathcal{N}(0, 0.1)$.

3.2 AWE-S: A Subword Variant of AWE

We also propose AWE-S, a variant of AWE that leverages *subword* information. A subword of a word is a part of the word, e.g., its character n-grams or its lemmas. Subword information has been used in a number of word embedding models including fastText [4] and BERT [11] to improve the expressiveness of rare word representations. As an illustration, learning a robust representation for a word like *awing*, which occurs rarely in a corpus, is challenging due to a dearth of examples. To circumvent the scarcity of samples, a word embedding model can exploit the fact that *awe* is the verb lemma form of *awing* to learn a better embedding for *awing*.

In AWE-S, the embedding of each word is given by the sum of embeddings of all of its subwords, regardless of whether the word is a context word or the masked word in the prediction task, i.e.,

$$\tilde{\boldsymbol{u}}_{\boldsymbol{w}} = \sum_{e_j \in S_w} \boldsymbol{u}_{e_j} \,,$$

where S_w is the subword set of the word w. The context vector $\boldsymbol{c}$ in AWE-S is given by

$$\boldsymbol{c}_{\mathbf{attn}} = \sum_{i \in [-b,b]-\{0\}} a_{w_i} \left(\sum_{e_j \in S_{w_i}} \tilde{\boldsymbol{u}}_{e_j} \right) \,,$$

and the probability of the masked word w_0 to occur in the context of $\{w_{-b}, .., w_{-1}, w_1, .., w_b\}$ is given by

$$p(w_0 | W_{[-b,b]-\{0\}}) \propto \exp \tilde{\boldsymbol{u}}_{\boldsymbol{w_0}}{}^T \boldsymbol{c}_{\mathbf{attn}} \,. \tag{3.1}$$

Note that in fastText, the context vector of (2.1) and the probability of w_0 to occur in the context of $\{w_{-b}, .., w_{-1}, w_1, .., w_b\}$ is given by

$$\boldsymbol{c} = \sum_{i \in [-b,b]-\{0\}} \sum_{e_j \in S_{w_i}} \boldsymbol{u}_{e_j} \,, \qquad p(w_0 | W_{[-b,b]-\{0\}}) \propto \exp \boldsymbol{v}_{w_0}^T \boldsymbol{c} \,. \tag{3.2}$$

The computations in AWE-S may appear to be similar those in fastText. However, observe that, in fastText, a word is not represented as the sum of the embeddings of its subwords when it is masked. Also, contrary to fastText, which uses character n-grams to construct the subword set for a word, AWE-S uses the noun, adjective, and verb lemmas of the word to construct the subword set. This limits the size of the subword set significantly as compared to using character n-grams, which helps to reduce the number of learnable parameters in AWE-S.

Chapter 4

Experiments

We demonstrate the superior performance of AWE and AWE-S against CBOW, Skip-Gram, GloVe, and fastText on a variety of datasets, which broadly fall into two main categories:

1. **Word similarity tasks.** We use this task to evaluate the performance of word embedding models themselves. The datasets for this task contain word pairs and a semantic similarity score associated with the pairs. The scores were annotated by human subjects. These are the most widely used evaluation methods for word embedding models [28]. We test AWE's performance using spearman correlation score on eight such datasets - MEN [29], WS353 [30], WS353R [31], WS353S [31], SimLex999 [32], RW(RareWords) [33], RG65 [34], and MTurk [35].

2. **Downstream NLP tasks.** These tasks evaluate performance on important NLP applications including natural language inference, semantic entailment, semantic relatedness, and paraphrase detection. Because the conventional application of word embedding models has been to initialize NLP models, we use these tasks to assess the quality of our word embeddings for initializing NLP models for downstream applications. Datasets include SNLI [36], SICK-E [37], SICK-R [37], STS Benchmark [38], and MRPC [39]. Unlike datasets for the previous task, all the datasets here have a training set.

Model	MEN	WS353	WS353R	WS353S	SimLex999	RW(RareWords)	RG65	MTurk
AWE	**0.771**	**0.672**	**0.609**	**0.773**	0.377	**0.440**	**0.836**	**0.683**
CBOW	0.717	0.591	0.478	0.720	**0.383**	0.396	0.791	0.659
SG	0.738	0.664	0.607	0.729	0.371	0.440	0.769	0.662
GloVe	0.695	0.553	0.492	0.683	0.324	0.340	0.763	0.621

Table 4.1 : Spearman correlation on word similarity datasets for AWE, CBOW, Skip-Gram, and GloVe.

Model	MEN	WS353	WS353R	WS353S	SimLex999	RW(RareWords)	RG65	MTurk
AWE-S	**0.771**	0.675	0.610	**0.769**	**0.386**	**0.463**	**0.819**	**0.692**
FastText	0.761	**0.691**	**0.642**	0.745	0.385	0.457	0.795	0.685

Table 4.2 : Spearman correlation on word similarity datasets for FastText and AWE-S, which use additional subword information.

4.1 Experimental Setup

We train AWE, CBOW, Skip-Gram, GloVe, and fastText on the wikipedia dataset ($\sim$ 17 GB in size), with vocabulary consisting of one million words. Dimensions of word embeddings for CBOW, Skip-Gram, GloVe, and fastText is 500. For AWE and AWE-S, $K, Q \in \mathbb{R}^{N\times 50}$, and $V \in \mathbb{R}^{N\times 500}$, where N is the size of the vocabulary. Widely-used parameter settings (context window size is 5, number of negative samples is 5, number of training epochs is 5) are used to train CBOW, Skip-Gram, and fastText. AWE and AWE-S are also trained for the same number of epochs. GloVe is trained for 100 epochs and parameters are set to those recommended in the paper (x-max is 100 and context window size is 10). We used open-source word embedding evaluation tool kits for assessing the quality of the models [40, 41].

Model	SNLI (Acc)	SICK-E (Acc)	SICK-R (Pearson)	STS Benchmark (Pearson)	MRPC (F1)
AWE	**65.23***	76.05*	0.783*	0.648*	**81.70**†
AWE-S	65.21*	76.05*	**0.785***	**0.651***	81.49†
CBOW	65.07*	**76.23***	0.781*	0.600*	81.26†
SG	64.45*	75.04*	0.780*	0.632*	81.12†
FastText	64.98*	75.06*	0.782*	0.586*	80.88†
GloVe	64.18*	75.91*	0.782*	0.643*	80.60†

Table 4.3 : Evaluation results of AWE and its subword variant, AWE-S, against other word embedding methods for initializing models for downstream tasks. Accuracy is reported for SNLI dataset and SICK-E, Pearson correlation is reported for SICK-R and STS Benchmark, and F1 score is reported for MRPC. '$\star$', '$*$' and '$\dagger$' denote accuracy, pearson correlation, and F1 score respectively.

4.2 Numerical Results

We report the performance statistics of AWE, AWE-S, and the competing word embedding models on word similarity datasets and downstream NLP tasks in Tables 4.1, 4.2, and 4.3.

Table 4.1 shows performance of AWE, CBOW, Skip-Gram, and GloVe on word similarity datasets. We use spearman correlation between cosine similarity and human-annotated relatedness scores as the metric to measure the performance. AWE performs significantly better on all of the datasets except SimLex999.

Table 4.2 compares performance of AWE-S vs fastText. Both algorithms incorporate additional subword information into their architecture. FastText uses character n-grams, while AWE-S uses the lemma forms of the words. AWE wins against FastText on six out of the eight datasets.

Table 4.3 reports performance of AWE and AWE-S against CBOW, Skip-Gram, GloVe, and fastText on supervised downstream tasks. A logistic regression classifier is trained to classify sentences using pre-trained word embeddings. For these initialization

applications, AWE provides improvement in performance as compared to other models over all datasets except SICK-E.

4.2.1 Interpretation of the attention mechanism

Studies have shown that the attention mechanism helps a neural network to attend to relevant features in the input [21, 22]. Thus, motivating the integration of the attention mechanism in CBOW. However, some studies argue otherwise and show that the attention weights are poor indicators of feature importance [23, 24]. This differing opinion thus necessitates an investigation as to why attention works in the case of AWE. We visually analyze the attention weights to investigate if the attention mechanism models the importance of a context word for the masked word prediction in AWE. The investigation revealed two key findings.

First, no matter the masked word, the attention weight of the masked word and a highly frequent word is typically high. In Table 4.4 and 4.5, the attention weights for highly frequent words like *for, and, the, has,* and *a* (words that have cells with dark gray background) are quite high as compared to other words in the sentence. Note that even though the attention weight is large, the similarity between word vectors is very close to zero, thus not affecting the probability of prediction of the masked word.

Second, if we leave out these highly frequent words from the set of context words, we observe that the attention weights focus on more informative words in the context for the prediction of the masked word. In Table 4.4, for each masked word, the attention weights corresponding to context words that are most attended to for its prediction (excluding frequent words) has been highlighted with bold numerals and a light gray background. For more examples, please refer to Table 4.5.

Also, inspired by [23], we randomly permute the attention weights and observe the

Mask \ Sentence		professor	scolded	students	for	playing	games
professor	attention weight	-	1.112	**1.666**	29.096	1.125	0.171
	word vector similarity	-	0.005	0.381	-0.003	0.037	-0.228
playing	attention weight	1.392	0.932	0.641	2190.342	-	**2.278**
	word vector similarity	0.037	0.048	0.116	0.001	-	0.975

Mask \ Sentence		mother	and	child	crossing	the	road
child	attention weight	**2.527**	746.334	-	0.55	0.073	0.827
	word vector similarity	1.076	0	-	-0.017	-0.005	-0.22
road	attention weight	0.353	728.901	0.553	**1.86**	1734.955	-
	word vector similarity	-0.05	0	-0.22	0.961	0	-

Mask \ Sentence		my	elementary	school	has	a	skating	rink
skating	attention weight	0.243	0.434	**1.384**	39.814	274.934	-	**1.354**
	word vector similarity	-0.111	0.062	0.206	-0.004	-0.001	-	3.636
school	attention weight	0.138	**0.972**	-	21.477	1066.489	0.369	0.626
	word vector similarity	-0.011	2.595	-	-0.002	0	0.206	0.101

Table 4.4 : Interpretation of attention. Attention weight between the masked word and the context word is given by $e^{(\boldsymbol{k}_{\text{masked word}})^T \boldsymbol{q}_{\text{context word}}}$, while the word vector similarity between the masked word and the context word is given by $e^{(\boldsymbol{u}_{\text{masked word}})^T \boldsymbol{u}_{\text{context word}}}$. Highly frequent words are highlighted with a dark gray background. For each masked word, the attention weights corresponding to context words that are most attended to (excluding highly frequent words) are highlighted with **bold** numerals and light gray background.

change in AWE's model performance. This idea to manipulate the attention weight distribution to study the attention mechanism dynamics is quite popular [22, 24]. This experiment quantifies the change in behavior of the AWE model if it were to focus on a different set of context words to predict the masked word. We find that over the entire dataset, the prediction loss for the masked word drops by 26% if we shuffle

Sentence / Mask		colored	leaves	make	autumn	beautiful						
autumn	att. wt.	0.905	**2.828**	0.069	-	0.561						
	sim	-0.006	0.562	-0.116	-	0.157						
		soldier	returned	home	after	the	battle					
soldier	att. wt.	-	1.593	0.861	0.044	480.195	**2.379**					
	sim	-	0.065	-0.002	-0.108	-0.001	0.67					
		extra	oil	helps	to	add	flavor	to	food			
oil	att. wt.	1.202	-	0.446	599.004	0.587	**1.864**	599.004	**2.965**			
	sim	0.274	-	-0.142	-0.001	-0.071	0.744	-0.001	0.973			
		please	open	the	-	to	chapter	for	reading	today		
book	att. wt.	1.559	0.104	2090.525		0.65	**2.74**	31.606	**4.858**	0.42		
	sim	0.042	-0.292	0.001		-0.004	0.952	-0.004	0.529	0.088		
		watch	is	a	small	clock	carried	or	worn	by	a	person
watch	att. wt.	-	1.977	32.681	0.388	**3.47**	1.855	714.149	1.575	159.371	32.681	1.29
	sim	-	-0.005	-0.004	-0.188	0.921	0.223	0	0.28	-0.002	-0.004	0.097
		phone	is	a	communication	device	that	permits	users	to	conduct	conversation
phone	att. wt.	-	5.54	755.948	**1.735**	**2.634**	176.562	0.902	1.205	567.022	755.948	**4.791**
	sim	-	-0.003	0	0.716	1.19	-0.002	0.002	0.508	-0.001	0	0.869

Table 4.5 : More examples of attention weights between the masked word and the context words. Attention weight (att. wt.) between the masked word and context word is given by $e^{(\boldsymbol{k}_{\text{masked word}})^T \boldsymbol{q}_{\text{context word}}}$, while the word vector similarity (sim.) between the masked word and context word is given by $e^{(\boldsymbol{u}_{\text{masked word}})^T \boldsymbol{u}_{\text{context word}}}$. Highly frequent words are highlighted with a dark gray background. For each masked word, the attention weights corresponding to context words that are most attended to (excluding highly frequent words) are highlighted with **bold** numerals and a light gray background.

the attention weights. Thus, we can conclude that the attention mechanism in AWE focuses on informative context words for the prediction of the masked word. This is consistent with our preceding findings through the visual analysis of the attention weights (Table 4.4 and 4.5).

4.2.2 Role of subword information

Subwords play a critical function in the robust learning of infrequent words. For instance, the word *happiest* may not occur as frequently as the word *happy*, but the model can learn more about the word *happiest* if it were to supplement it with the information of the word *happy* as well. This is the reason why fastText and AWE-S perform significantly better than other embedding methods on the RareWords dataset (Table 4.1 and Table 4.2). FastText relates the two words *happy* and *happiest* through the n-gram *'happ'*. On the other hand, AWE-S relates the word *happiest* to *happy* through its verb lemma form, i.e, *'happy'*.

4.2.3 Application of AWE's distributional loss

Deep NLP models use an input embedding matrix to project a word's one-hot vector to a low-dimensional dense representation, which is then fed into the model. We propose to enrich this dense word vector representation with contextual information using a distributional loss term.

First, in addition to the input embedding matrix, which we refer to as the value matrix $V \in \mathbb{R}^{N \times D}$, we use two extra matrices: a key matrix $K \in \mathbb{R}^{N \times D}$ and a query matrix $Q \in \mathbb{R}^{N \times D}$. N is the size of the vocabulary and D is the embedding dimension. Let $\{w_0, w_1, ..., w_n\}$ be the input sequence to a deep NLP model. Let w_i be an arbitrary word in the input sequence with its context given by the sequence $\{w_{i-b}, .., w_{i-1}, w_{i+1}, .., w_{i+b}\}$, where b is the size of the context window. As previously stated, for w_i, we use its value embedding $\boldsymbol{v}_{w_i}$ as input to the model. Second, we add a new loss term to the model which enforces the input vector representation of w_i, given by $\boldsymbol{v}_{w_i}$, to be close to the weighted vector representation of its context. The loss

for w_i is given by

$$l_{w_i} = 1 - \text{sigmoid}\Big(\boldsymbol{v}_{w_i}.\Big(\sum_{j \in [i-b, i+b]-\{i\}} a_{w_j} \boldsymbol{v}_{w_j}\Big)\Big)\,,$$

where,

$$a_{w_j} = \text{softmax}\big(\boldsymbol{k}_{w_i}^T \boldsymbol{q}_{w_j}\big)\,.$$

This loss is inspired from the distributional hypothesis, which dictates that a word is known through its context [42]. It is interesting to note that this loss can be added to any NLP model and requires no additional training data. The idea is similar to the training paradigm of word2vec and AWE, which learn word embeddings using a variant of this loss.

To test the efficacy of this loss, we train a transformer model [9] on WMT17 English to German news translation dataset*. The best trained transformer model gives a perplexity score of 7.17, however, we achieve a perplexity score of 6.56 (lower is better), with the embedding scheme augmented with the new distributional loss term.

*http://data.statmt.org/wmt17/translation-task/

Chapter 5

Future Work

Word2vec is an incredibly powerful algorithm to pre-process any sequential data. It embeds the tokens that constitute the sequences in a meaningful vector space. This makes it broadly applicable to many domains other than NLP. It has found applications in learning representations of graphs [43], embedding biological sequences [44], modeling students' question response patterns [45], and even in recommendation of your next meal [46].

As our experiments demonstrate the superior performance of AWE for various downstream NLP tasks, we want to explore the generalizability of AWE across different domains as well. Not only can it be effective in improving the performance of the models [43, 44, 45, 46], but also the inbuilt attention mechanism can provide many interesting insights.

Let us consider one concrete example from the educational domain to elucidate the idea. One of the most important challenges in education is to predict a student's future performance based on his or her past performance. The problem is called knowledge tracing and several models have been proposed to tackle the problem [47,48,49]. These models work on time series data that consists of students' question response patterns, given by $X_i = \{(q_t^i, a_t^i)\}_{t=1}^{T}$, where q_t^i denotes the question answered by student i at time t, and a_t^i is the assessment of that response (either correct or incorrect). Recently, our prior work [45] showed that most deep learning based knowledge tracing methods perform better if the input to these methods, which are the question response patterns

((q, a) tuples), are initialized through fastText, rather than initializing them randomly.

Using AWE, instead of fastText, can prove to be useful in this setting. AWE has a key advantage over fastText. AWE, like fastText, can reveal which questions are similar. But, the interpretability provided by the attention mechanism inbuilt in AWE can also be used to trace the question in the performance record (context) whose success or failure correlates strongly with the success or failure for the current question. Therefore, one can use the attention weights to sketch out the pre-requisite graph over the set of questions with reasonable accuracy.

Chapter 6

Conclusion

In this work, we have proposed AWE and AWE-S, which perform significantly better than CBOW, Skip-Gram, GloVe, and fastText on a variety of datasets across a diverse set of tasks. The simple setting of masked word prediction in AWE also helped us visually analyze and interpret how the attention mechanism works. Our analysis revealed that the attention mechanism can figure out the context words that are most relevant for predicting the masked word. These promising results suggest that AWE and AWE-S can further be applied to domains other than NLP to not only boost the learning model's performance, but also to gain key insights through the interpretability offered by the inbuilt attention mechanism.

Bibliography

[1] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *1st International Conference on Learning Representations, ICLR 2013* (Y. Bengio and Y. LeCun, eds.), 2013.

[2] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, pp. 3111–3119, 2013.

[3] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proc. Conference on Empirical Methods in Natural Language Processing*, pp. 1532–1543, Oct. 2014.

[4] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, 2017.

[5] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *3rd International Conference on Learning Representations, ICLR 2015,* (Y. Bengio and Y. LeCun, eds.), 2015.

[6] W. Ling, C. Dyer, A. W. Black, and I. Trancoso, "Two/too simple adaptations of word2vec for syntax problems," in *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1299–1304, 2015.

[7] W. Wang, S. J. Pan, D. Dahlmeier, and X. Xiao, "Recursive neural conditional random fields for aspect-based sentiment analysis," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 616–626, 2016.

[8] J. R. Firth, "A synopsis of linguistic theory 1930-55.," *Studies in Linguistic Analysis (special volume of the Philological Society)*, vol. 1952-59, pp. 1–32, 1957.

[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, pp. 5998–6008, 2017.

[10] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, "Self-attention generative adversarial networks," in *Proceedings of the 36th International Conference on Machine Learning*, Proceedings of Machine Learning Research, pp. 7354–7363, 2019.

[11] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, 2019.

[12] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. Le, and R. Salakhutdinov, "Transformer-XL: Attentive language models beyond a fixed-length context," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, (Florence, Italy), pp. 2978–2988, 2019.

[13] W. Ling, Y. Tsvetkov, S. Amir, R. Fermandez, C. Dyer, A. W. Black, I. Trancoso,

and C.-C. Lin, "Not all contexts are created equal: Better word representations with variable attention," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1367–1372, 2015.

[14] T. Schick and H. Schütze, "Attentive mimicking: Better word embeddings by attending to informative contexts," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 489–494, 2019.

[15] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," in *Empirical Methods in Natural Language Processing (EMNLP)*, (Lisbon, Portugal), pp. 1412–1421, Association for Computational Linguistics, Sep. 2015.

[16] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. Advances in Neural Information Processing Systems*, pp. 3104–3112, Dec. 2014.

[17] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, (Doha, Qatar), pp. 1724–1734, Association for Computational Linguistics, Oct. 2014.

[18] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of the national academy of sciences*, vol. 79, no. 8, pp. 2554–2558, 1982.

[19] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, pp. 1735–1780, Nov. 1997.

[20] R. J. Mooney, "Semantic parsing: Past, present, and future," in *Presentation slides from the ACL Workshop on Semantic Parsing*, 2014.

[21] S. Wiegreffe and Y. Pinter, "Attention is not not explanation," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 11–20, 2019.

[22] S. Vashishth, S. Upadhyay, G. S. Tomar, and M. Faruqui, "Attention interpretability across nlp tasks," *arXiv preprint arXiv:1909.11218*, 2019.

[23] S. Jain and B. C. Wallace, "Attention is not Explanation," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 3543–3556, 2019.

[24] S. Serrano and N. A. Smith, "Is attention interpretable?," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 2931–2951, July 2019.

[25] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *International conference on machine learning*, pp. 1139–1147, 2013.

[26] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, 2010.

[27] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

[28] A. Bakarov, "A survey of word embeddings evaluation methods," *arXiv preprint arXiv:1801.09536*, 2018.

[29] E. Bruni, N.-K. Tran, and M. Baroni, "Multimodal distributional semantics," *Journal of Artificial Intelligence Research*, vol. 49, pp. 1–47, 2014.

[30] L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman, and E. Ruppin, "Placing search in context: The concept revisited," in *Proceedings of the 10th international conference on World Wide Web*, pp. 406–414, 2001.

[31] E. Agirre, E. Alfonseca, K. Hall, J. Kravalova, M. Paşca, and A. Soroa, "A study on similarity and relatedness using distributional and WordNet-based approaches," in *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 19–27, 2009.

[32] F. Hill, R. Reichart, and A. Korhonen, "Simlex-999: Evaluating semantic models with (genuine) similarity estimation," *Computational Linguistics*, vol. 41, no. 4, pp. 665–695, 2015.

[33] M.-T. Luong, R. Socher, and C. D. Manning, "Better word representations with recursive neural networks for morphology," in *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pp. 104–113, 2013.

[34] H. Rubenstein and J. B. Goodenough, "Contextual correlates of synonymy," *Communications of the ACM*, vol. 8, no. 10, pp. 627–633, 1965.

[35] K. Radinsky, E. Agichtein, E. Gabrilovich, and S. Markovitch, "A word at a time: computing word relatedness using temporal semantic analysis," in *Proceedings of the 20th international conference on World wide web*, pp. 337–346, 2011.

[36] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning, "A large annotated corpus for learning natural language inference," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 632–642, 2015.

[37] M. Marelli, S. Menini, M. Baroni, L. Bentivogli, R. Bernardi, R. Zamparelli, *et al.*, "A sick cure for the evaluation of compositional distributional semantic models.," in *LREC*, pp. 216–223, 2014.

[38] D. Cer, M. Diab, E. Agirre, I. Lopez-Gazpio, and L. Specia, "SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation," in *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pp. 1–14, 2017.

[39] B. Dolan, C. Quirk, and C. Brockett, "Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources," in *Proceedings of the 20th international conference on Computational Linguistics*, p. 350, Association for Computational Linguistics, 2004.

[40] S. Jastrzebski, D. Leśniak, and W. M. Czarnecki, "How to evaluate word embeddings? on importance of data efficiency and simple supervised tasks," *arXiv preprint arXiv:1702.02170*, 2017.

[41] A. Conneau and D. Kiela, "SentEval: An evaluation toolkit for universal sentence representations," in *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[42] Z. S. Harris, "Distributional structure," *Word*, vol. 10, no. 2-3, pp. 146–162, 1954.

[43] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 855–864, 2016.

[44] K. K. Yang, Z. Wu, and F. H. Arnold, "Machine-learning-guided directed evolution for protein engineering," *Nature methods*, vol. 16, no. 8, pp. 687–694, 2019.

[45] S. Sonkar, A. E. Waters, A. S. Lan, P. J. Grimaldi, and R. G. Baraniuk, "qdkt: Question-centric deep knowledge tracing," 2020.

[46] R. Yunus, O. Arif, H. Afzal, M. F. Amjad, H. Abbas, H. N. Bokhari, S. T. Haider, N. Zafar, and R. Nawaz, "A framework to estimate the nutritional value of food in real time using deep learning techniques," *IEEE Access*, vol. 7, pp. 2643–2652, 2018.

[47] A. T. Corbett and J. R. Anderson, "Knowledge tracing: Modeling the acquisition of procedural knowledge," *User modeling and user-adapted interaction*, vol. 4, no. 4, pp. 253–278, 1994.

[48] A. S. Lan, A. E. Waters, C. Studer, and R. G. Baraniuk, "Sparse factor analysis for learning and content analytics," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1959–2008, 2014.

[49] C. Piech, J. Bassen, J. Huang, S. Ganguli, M. Sahami, L. J. Guibas, and J. Sohl-Dickstein, "Deep knowledge tracing," in *Advances in neural information processing systems*, pp. 505–513, 2015.