Graph-based Dynamic Word Embeddings

Yuyin Lu[1] , **Xin Cheng**[2] , **Ziran Liang**[1] , **Yanghui Rao**[1*]
[1]School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou, China
[2]School of Mathematical Sciences, Shanghai Jiao Tong University, Shanghai, China
luyy37@mail2.sysu.edu.cn, chengxin_19@aliyun.com, liangzr5@mail2.sysu.edu.cn,
raoyangh@mail.sysu.edu.cn

Abstract

As time goes by, language evolves with word semantics changing. Unfortunately, traditional word embedding methods neglect the evolution of language and assume that word representations are static. Although contextualized word embedding models can capture the diverse representations of polysemous words, they ignore temporal information as well. To tackle the aforementioned challenges, we propose a graph-based dynamic word embedding (GDWE) model, which focuses on capturing the semantic drift of words continually. We introduce word-level knowledge graphs (WKGs) to store short-term and long-term knowledge. WKGs can provide rich structural information as supplement of lexical information, which help enhance the word embedding quality and capture semantic drift quickly. Theoretical analysis and extensive experiments validate the effectiveness of our GDWE on dynamic word embedding learning.

1 Introduction

Learning changes of language over time is critical to language understanding [McMahon, 1994]. An indication of language evolution is the change of vocabulary and semantics, which arises with the generation of new words and the meaning drift of old words. Taking the word "*apple*" as an example, it referred to a kind of fruit in the past, but it often refers to a technology company nowadays. Although traditional word embedding models such as Word2Vec [Mikolov *et al.*, 2013b] and PPMI-SVD [Levy and Goldberg, 2014] provide an efficient way for modeling word meaning, they neglect temporal information and generate static word embeddings. While contextual word embedding models make impressive progress in various time-invariant natural language processing (NLP) tasks, their performance degrade on a new corpus far from the training period [Lazaridou *et al.*, 2021].

Due to the importance of learning language development and the limitation of prior word embedding models, dynamic word embedding learning has attracted much attention in recent years. Dynamic word embedding models aim at learning word embeddings in different conditions. Since the change of language over time is one of the most common conditions, learning dynamic word embeddings often refers to learning time-specific word embeddings. The key idea of early dynamic word embedding models [Kulkarni *et al.*, 2015; Hamilton *et al.*, 2016] is to align word embedding spaces pretrained on every time slice separately, called "alignment after training" (i.e., 2-step) method. Later, Yao *et al.* [2018] proposed to jointly train the word embedding spaces over all time slices, so as to avoid the alignment process. However, the above approach needs to acquire the full corpus before training, which fails to meet the real-time requirement in practice. Instead of learning a word embedding space for each time slice, Bamler and Mandt [2017] proposed a probability model to incrementally fine-tune a word embedding space. Unfortunately, it may forget long-term semantics and fail to capture new meanings with limited contextual information. Additionally, several studies developed dynamic contextual word embeddings by introducing temporal information to model components [Hofmann *et al.*, 2021] or input texts [Rosin *et al.*, 2022; Dhingra *et al.*, 2022]. But their high costs for training hinder them from adapting to rapidly changing conditions.

In this work, we propose a graph-based dynamic word embedding (GDWE) model[1], which continuously updates a word embedding space because such a mechanism is more consistent with the development of language [Bamler and Mandt, 2017]. To tackle the aforementioned challenges, we introduce word-level knowledge graphs (WKGs) to encode and store past word co-occurrence knowledge, since semantic drift is often reflected by the change of co-occurring words [Kutuzov *et al.*, 2018]. Besides, the word co-occurrence graph has valuable structural information and can be adapted to support efficient streaming updates [Spitz and Gertz, 2018]. It is worth noting that language changes more significantly in a longer period. For example, the word "*you*" in modern English was written as "*thee*" in the Middle Ages, and it is often abbreviated as "*u*" in informal writing these years. To distinguish characteristics of language at different time stages, we construct and maintain WKGs which store long-term and short-term knowledge separately. Then, we utilize past knowledge encoded in both short-term and long-

[*]The corresponding author.

[1]Our code and supplementary materials are available in public at: https://github.com/luyy9apples/GDWE.

term WKGs to learn better dynamic word embeddings at present. Furthermore, to improve the efficiency of GDWE, we train on all texts only once. The auxiliary co-occurrence information retrieved from WKGs help GDWE avoid the risk of underfitting in online learning, and balance well between efficiency and effectiveness. We summarize the contributions of our work as follows:

- We introduce a short-term and multiple long-term WKGs to encode word co-occurrence information for capturing semantic drift over time.
- We propose a graph-based dynamic word embedding model named GDWE, which updates a time-specific word embedding space efficiently.
- We theoretically prove the correctness of using WKGs to assist dynamic word embedding learning and verify the effectiveness of GDWE by cross-time alignment, text stream classification, and qualitative analysis.

2 Related Works

Different from static word representations, dynamic word embeddings can capture semantic drift within language. Since the cost functions of most word embedding models are rotation-invariant, word embedding spaces trained on different time slices are not in the same latent semantic space [Yao *et al.*, 2018], thus the geometric relations among word embeddings of different time slices lack semantic information.

Early dynamic word embedding models usually adopt a 2-step method. First, they pre-train a word embedding space for each time slice using static word embedding models like Word2Vec [Mikolov *et al.*, 2013b]. Then they align these temporal embedding spaces. Specifically, Kulkarni *et al.* [2015] achieved alignment by keeping embeddings of semantics-invariant words (i.e., anchor words) the same in each time slice. Similarly, Hamilton *et al.* [2016] used an orthogonal transformation to align embeddings. Different from the above 2-step approach, Yao *et al.* [2018] proposed a model called DW2V that jointly trains time-specific word embedding spaces by matrix factorization. DW2V avoids the alignment process and shares semantic information across time slices. Recently, Gong *et al.* [2020] considered to learn transformations from a general embedding space to temporal embedding spaces. Following the idea of fine-tuning word embeddings incrementally [Kim *et al.*, 2014; Kaji and Kobayashi, 2017], Bamler and Mandt [2017] proposed a probabilistic model using approximate Bayesian inference to learn dynamic word embeddings continuously.

On the other hand, there are some works [Hu *et al.*, 2019; Giulianelli *et al.*, 2020] employing contextual word embeddings [Devlin *et al.*, 2019] to investigate semantic drift. Since the contextual word embeddings used before are not dynamic, Hofmann *et al.* [2021] proposed a dynamic contextual word embedding model, which transforms pre-trained BERT embeddings into time-specific word embeddings through temporal neural networks. Apart from introducing time-specific model components, Rosin *et al.* [2022] and Dhingra *et al.* [2022] modified input texts by prepending time tokens. Unfortunately, the results of some shared tasks [Schlechtweg *et al.*, 2020; Basile *et al.*, 2020] indicated that contextual embeddings perform quite limited on unsupervised lexical semantic change (LSC) detection. Although contextual embedding based methods first top the leaderboard in the LSC task with label supervision and external linguistic resources, these different results among the LSC tasks may lie in the difference between models rather than that between contextual and traditional embeddings [Kutuzov and Pivovarova, 2021]. Besides, contextual embedding models require high computational resources for pre-training or fine-tuning, which remains an obstacle for adapting them to dynamic conditions.

3 Proposed Method

3.1 Problem Definition

For the convenience of the description, we first define the continuous learning paradigm of dynamic word embeddings. As presented in [Hofmann *et al.*, 2021], the training corpus for dynamic word embeddings is a text stream in which new documents constantly appear. While a word embedding space is updated continuously in training, we split the text stream $\mathcal{D}$ into N time slices based on timestamps for the convenience of testing. Let $\mathcal{D}^{(i)}$ denote the document set containing all the texts in the i-th time slice. The goal of dynamic word embedding models is to obtain a word embedding space U_i and a context embedding space V_i, which form a snapshot of the continuously updated embedding space in the i-th time slice.

3.2 Model Architecture

Our GDWE continuously updates a word embedding space, where the coordinate axis remains unchanged. Therefore, GDWE can avoid the aforementioned alignment process. Additionally, we introduce WKGs to help our model capture semantic drift with limited lexical information. WKGs not only hold past word co-occurrence information, but also contain structural information such as common neighbors. To improve the efficiency of WKGs, we divide $\mathcal{D}^{(i)}$ into a sequence of fixed-size mini-batches $\{\mathcal{B}_1^{(i)}, \ldots, \mathcal{B}_{M_i}^{(i)}\}$ according to timestamps, where M_i denotes the number of mini-batches in the i-th time slice. Then we update and utilize WKGs by mini-batches, which approximates continuous learning when the mini-batch size is relatively small.

Figure 1 illustrates the architecture of our GDWE, which contains the following modules: (1) Construct and update the short-term WKG. To capture short-term knowledge, we first construct the current WKG $\mathcal{G}_j^{(i)}$ for $\mathcal{B}_j^{(i)}$. Then, we cumulatively update a short-term WKG $\mathcal{G}_{s,<j}^{(i)}$ based on $\mathcal{G}_j^{(i)}$; (2) Construct and update long-term WKGs. After training all mini-batches, we obtain a WKG $\mathcal{G}_{s,<M_i}^{(i)}$ which is considered as a long-term WKG $\mathcal{G}_l^{(i)}$ containing representative knowledge of $\mathcal{D}^{(i)}$. Then we update the long-term WKGs set $\mathcal{G}_L^{(i+1)}$ using $\mathcal{G}_l^{(i)}$; (3) Retrieve knowledge from WKGs. When training on $\mathcal{B}_j^{(i)}$, we retrieve word relevance knowledge from the short-term WKG $\mathcal{G}_{s,<j}^{(i)}$ and long-term WKGs $\mathcal{G}_L^{(i)}$ of past time slices; (4) Update dynamic word embeddings. Both the auxiliary knowledge and lexical information

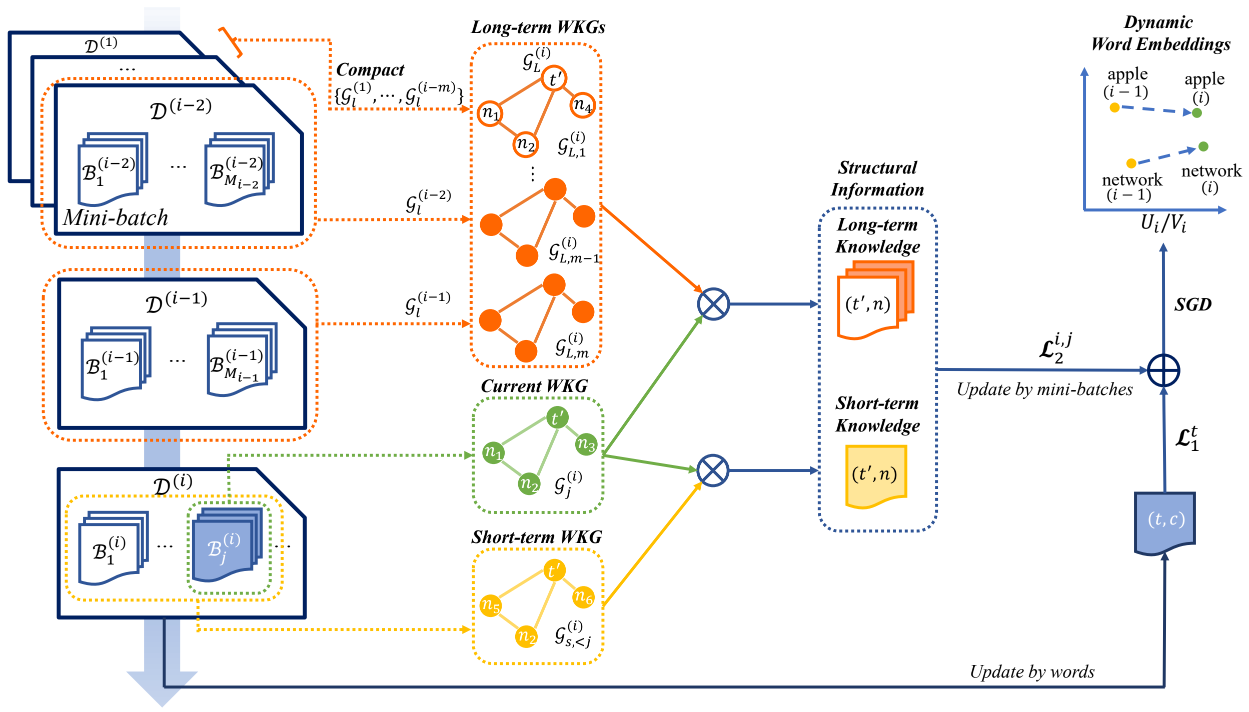

Figure 1: Model architecture of GDWE, in which, $\mathcal{G}_L^{(i)} = \{\mathcal{G}_{L,1}^{(i)}, \ldots, \mathcal{G}_{L,m}^{(i)}\}$ is the set of compacted long-term WKGs for the i-th time slice, where m is the size of $\mathcal{G}_L^{(i)}$. Besides, $\mathcal{G}_l^{(i)}$ denotes the long-term WKG for $\mathcal{D}^{(i)}$, while $\mathcal{G}_j^{(i)}$ and $\mathcal{G}_{s,<j}^{(i)}$ are the current WKG and the short-term WKG for $\mathcal{B}_j^{(i)}$, respectively. Finally, (t, c) denotes a target word and its context word, and (t', n) denotes a node and its neighbor in a WKG.

of $\mathcal{B}_j^{(i)}$ are adopted to update U_i and V_i using the stochastic gradient descent (SGD) optimization method.

3.3 Training Strategy

In this section, we explain the training strategy of our GDWE in detail, including the update of WKGs and dynamic word embeddings primarily.

Definition of WKGs. A WKG is a weighted and undirected graph where each node represents a unique word. If two words co-occurred in the original text, then there is an edge between them. And the edge weight indicates their co-occurrence frequency. For a clearer explanation, we denote a WKG as $\mathcal{G}(\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is nodes set and $\mathcal{E}$ is edges set.

Update the short-term WKG. Constructing lexical graphs can provide richer information than sequential data of plain texts, but it may mutually introduce noise [Zuckerman and Last, 2019]. Therefore, we denoise the short-term WKG in pre-processing. Intuitively, word pairs that co-occurred frequently are more likely to reappear in the future, and their semantic correlation tends to be accurate. Accordingly, we propose a rule-based method for updating the short-term WKG. For each edge, we calculate a credit score based on its weight:

$$score(e) = \frac{g(w_e)}{g(w_{max})}, \tag{1}$$

where w_e is the weight of edge e and w_{max} is the maximum edge weight in the WKG. $g(x) = \ln x$ is a smoothing function. The denominator regularizes the edge score to

Algorithm 1 Short-term WKG update algorithm

Input: Mini-batches $\{\mathcal{B}_1, \cdots, \mathcal{B}_M\}$;
Hyper-parameters: Thresholds δ_e and δ_d;
Output: Short-term WKG $\mathcal{G}_{s,<j}$, where $j = \{1, \cdots, M\}$.

```
1: Let $\mathcal{G}_{s,<0} = (\emptyset, \emptyset)$.
2: for $j = 1$ to $M$ do
3:     Construct the current WKG $\mathcal{G}_j$ of $\mathcal{B}_j$.
4:     $\mathcal{G}_{s,<j} \leftarrow$ UpdateWKG($\mathcal{G}_j, \mathcal{G}_{s,<(j-1)}, \delta_e, \delta_d$).
5:     yield short-term WKG $\mathcal{G}_{s,<j}$.
6: end for
```

$[0, 1]$ and makes $score(e)$ adaptive to WKGs with different scales. Later, we only reserve edges with a score higher than a threshold δ_e. Since a word with few neighbors may provide little information for other words, we use a threshold δ_d to weed out nodes with low degrees. Algorithm 1 shows the update process of the short-term WKG.

Update long-term WKGs. For preventing capacity saturation, we cannot save WKGs of all past time slices. Moreover, the past knowledge may become “outdated” and turn into noise as time evolves. Therefore, for the i-th time slice, we only save long-term WKGs of its previous $(m-1)$ time slices and one long-term WKG obtained by compacting WKGs of the $[1, i-m]$ time slices using Algorithm 2, which retains longer-term knowledge. The update process of long-term WKGs is shown in Algorithm 3.

Algorithm 2 UpdateWKG($\mathcal{G}_a, \mathcal{G}_b, \delta_e, \delta_d$)

```
1: Union two WKGs: $\mathcal{G}(\mathcal{V}, \mathcal{E}) \leftarrow \mathcal{G}_a \bigcup \mathcal{G}_b$.
2: for each $e \in \mathcal{E}$ do
3:     Calculate $score(e)$ by Equation (1).
4: end for
5: $\mathcal{E}_u \leftarrow \{e \in \mathcal{E} | score(e) > \delta_e\}$.
6: $\mathcal{V}_u \leftarrow \{n \in \mathcal{V} | deg(n, \mathcal{G}) > \delta_d\}$.
7: return updated WKG $\mathcal{G}_u = (\mathcal{V}_u, \mathcal{E}_u)$.
```

Algorithm 3 Long-term WKGs update algorithm

Input: Document streams $\{\mathcal{D}^{(1)}, \cdots, \mathcal{D}^{(N)}\}$ and their corresponding mini-batch sizes $\{M_1, \cdots, M_N\}$;
Hyper-parameters: The number of long-term WKGs m; thresholds δ_e and δ_d;
Output: Long-term WKGs $\mathcal{G}_L^{(i)} = \{\mathcal{G}_{L,1}^{(i)}, \cdots, \mathcal{G}_{L,K_i}^{(i)}\}$, where $i = \{2, \cdots, N\}$ and $K_i = |\mathcal{G}_L^{(i)}|$.

```
1: For the 1st time slice: $\mathcal{G}_L^{(1)} \leftarrow \{\}$.
2: for $i = 2$ to $N$ do
3:     Construct a WKG $\mathcal{G}_l^{(i-1)} \leftarrow \mathcal{G}_{s,<M_{i-1}}^{(i-1)}$ by Algorithm
       1 with input of $\mathcal{D}^{(i-1)} = \{\mathcal{B}_1^{(i-1)}, \cdots, \mathcal{B}_{M_{i-1}}^{(i-1)}\}$.
4:     $\mathcal{G}_{L,a}^{(i)} \leftarrow \mathcal{G}_{L,a}^{(i-1)}$ where $a = \{1, \cdots, |\mathcal{G}_L^{(i-1)}|\}$.
5:     Push $\mathcal{G}_l^{(i-1)}$ into $\mathcal{G}_L^{(i)}$: $\mathcal{G}_{L,|\mathcal{G}_L^{(i-1)}|+1}^{(i)} \leftarrow \mathcal{G}_l^{(i-1)}$.
6:     if $|\mathcal{G}_L^{(i-1)}| + 1 > m$ then
7:         $\mathcal{G}_{L,2}^{(i)} \leftarrow$UpdateWKG($\mathcal{G}_{L,1}^{(i)}, \mathcal{G}_{L,2}^{(i)}, \delta_e, \delta_d$).
8:         Pop $\mathcal{G}_{L,1}^{(i)}$ from $\mathcal{G}_L^{(i)}$: $\mathcal{G}_{L,b}^{(i)} \leftarrow \mathcal{G}_{L,(b+1)}^{(i)}$, where $b =$
           $\{1, \cdots, m\}$.
9:     end if
10:    yield long-term WKGs $\mathcal{G}_L^{(i)}$.
11: end for
```

Retrieve knowledge from WKGs. The key idea of utilizing WKGs is to extract word co-occurrence knowledge from short-term and long-term WKGs as supplement of lexical information. Specifically, considering a word that exists in both the current WKG (denoted as $\mathcal{G}$) and one of the short-term and long-term WKGs (denoted as $\mathcal{G}_k \in \{\mathcal{G}_s, \mathcal{G}_L\}$), we sample its direct neighbors in $\mathcal{G}_k$ as auxiliary information. The sampling probability of each neighbor is defined as follows:

$$p(n|t, \mathcal{G}_k) \propto \frac{w_{n,t}}{deg(n, \mathcal{G}_k)}, \tag{2}$$

where $w_{n,t}$ is the weight of the edge between the target word t and its neighbor word n, and $deg(n, \mathcal{G}_k)$ denotes the degree of n in $\mathcal{G}_k$. The sampling strategy is based on the assumption that a higher edge weight implies a closer correlation between two words. Besides, we conduct a penalty based on degree, since a word has a lot of neighbors may contain limited semantic information, such as articles and personal pronouns.

However, the above neighbor sampling method cannot distinguish whether the meaning of the target word has drifted. So there is a risk of "replaying" neighbor words related to the target word's old meaning and hindering the model from capturing semantic drift. According to the distributional hypothesis, words which often have the same neighboring words tend to be semantically similar. Therefore, we estimate the similarity between the meanings of a target word in different time slices using the classic Jaccard's coefficient as follows:

$$\mathrm{sim}(t_{\mathcal{G}}, t_{\mathcal{G}_k}) = \frac{|\mathcal{N}(t, \mathcal{G}) \cap \mathcal{N}(t, \mathcal{G}_k)|}{|\mathcal{N}(t, \mathcal{G}) \cup \mathcal{N}(t, \mathcal{G}_k)|}, \tag{3}$$

where $\mathcal{N}(t, \mathcal{G})$ denotes neighbors of the target word t in a WKG $\mathcal{G}$. The numerator represents the common neighbors' number of t in $\mathcal{G}$ and $\mathcal{G}_k$, and the denominator is a regularization term. With a similarity higher than a threshold δ_s, the neighbors of t in $\mathcal{G}_k$ are used as supplementary information.

Update dynamic word embeddings. Inspired by the incremental skip-gram with negative sampling (ISGNS) model [Kaji and Kobayashi, 2017], we define the loss function for updating word embeddings continually with word co-occurrence information as follows:

$$\mathcal{L}_1^t = -\sum_{c \in \mathcal{C}_t} [\psi_{t,c}^+ + k \mathbb{E}_{s \sim q_t(s)} \psi_{t,s}^-], \tag{4}$$

where $\psi_{t,x}^{\pm} = \ln \sigma(\pm \mathbf{u}_t (\mathbf{v}_x)^T)$. c is a context word of the target word t in a sliding window $\mathcal{C}_t$. $\mathbf{u}_t \in U$ denotes the word embedding of the target word t, while $\mathbf{v}_c \in V$ denotes the embedding of the context word c. As we apply the adaptive negative sampling algorithm, s is a negative sample drawn from a smoothed uniform distribution (i.e., *noise distribution*) with sampling probability $q_t(s) \propto f(s)^\beta$, where $f(s)$ is the frequency of word s occurred before word t in the text stream and $\beta \in [0, 1]$ is a smoothing factor. k is the number of negative samples. $\sigma(\cdot)$ is the sigmoid function.

As mentioned earlier, we retrieve knowledge from short-term and long-term WKGs as supplementary samples. For $\mathcal{B}_j^{(i)}$, the loss function using WKGs is defined as follows:

$$\begin{aligned}\mathcal{L}_2^{i,j} = -\textstyle\sum_{\mathcal{G}_k} \sum_{t'} d \mathbb{E}_{n \sim p(n|t', \mathcal{G}_k)} [\psi_{t',n}^+ \\ + k \mathbb{E}_{s \sim q_{i,j}(s)} \psi_{t',s}^-],\end{aligned} \tag{5}$$

where $\mathcal{G}_k \in \{\mathcal{G}_{s,<j}^{(i)}, \mathcal{G}_L^{(i)}\}, t' \in (\mathcal{V}_j^{(i)} \cap \mathcal{V}_k)$, and $\{\mathcal{G}_{s,<j}^{(i)}, \mathcal{G}_L^{(i)}\}$ denotes the short-term and long-term WKGs of $\mathcal{B}_j^{(i)}$. The target word t' satisfies $\mathrm{sim}(t'_{\mathcal{G}_j^{(i)}}, t'_{\mathcal{G}_k}) \geq \delta_s$, and n is a neighbor word of t' in $\mathcal{G}_k$ sampled with probability $p(n|t', \mathcal{G}_k)$. d is the number of sampled neighbors. In summary, the total loss function of our model can be written as follows:

$$\mathcal{L} = \sum_{t \in \mathcal{D}} \mathcal{L}_1^t + \sum_{i=1}^{N} \sum_{j=1}^{M_i} \alpha \mathcal{L}_2^{i,j}, \tag{6}$$

where α is a hyper-parameter for balancing contributions of lexical data and supplementary information from WKGs.

4 Theoretical Analysis

In this section, we theoretically prove the correctness of our WKG-based loss function $\mathcal{L}_2$, which utilizes word co-occurrence knowledge from WKGs to update dynamic word embeddings. First, we prove that for a complete WKG $\mathcal{G}(\mathcal{V},\mathcal{E})$ constructed on a corpus $\mathcal{D} = \{\mathcal{D}^{(1)},\ldots,\mathcal{D}^{(N)}\}$, it contains all the word co-occurrence information of $\mathcal{D}$ under general conditions. Then, we prove that $\mathcal{L}_2$ is a form of the loss function of SGNS when we apply SGNS to $\mathcal{G}$.

Theorem 1. *If $\delta_e = \delta_d = 0$ and $m = 1$, then a WKG $\mathcal{G}$ constructed on the complete corpus $\mathcal{D}$ encodes all word co-occurrence information of $\mathcal{D}$.*

Proof. According to Algorithms 1 and 2, for the update process of short-term WKG, we have

$$\begin{aligned}\mathcal{G}^{(i)}_{s,<M_i} &= \textbf{UpdateWKG}(\mathcal{G}^{(i)}_{M_i}, \mathcal{G}^{(i)}_{s,<(M_i-1)}, 0, 0)\\ &= \mathcal{G}^{(i)}_{M_i} \cup \mathcal{G}^{(i)}_{M_i-1} \cup \cdots \cup \mathcal{G}^{(i)}_{1}.\end{aligned}$$

Similarly, for the long-term WKG, we have

$$\begin{aligned}\mathcal{G}^{(i)}_{L,1} &= \textbf{UpdateWKG}(\mathcal{G}^{(i-1)}_{L,1}, \mathcal{G}^{(i-1)}_{L,2}, 0, 0)\\ &= \mathcal{G}^{(1)}_{l} \cup \mathcal{G}^{(2)}_{l} \cup \cdots \cup \mathcal{G}^{(i-m+1)}_{l}, \text{ where } \mathcal{G}^{(i)}_{l} = \mathcal{G}^{(i)}_{s,<M_i}.\end{aligned}$$

Since $\mathcal{G}^{(i)}_{L} = [\mathcal{G}^{(i)}_{L,1},\ldots,\mathcal{G}^{(i)}_{L,m}]$, we have $\mathcal{G} = \mathcal{G}^{(N)}_{L,1} = \mathcal{G}^{(1)}_{l} \cup \cdots \cup \mathcal{G}^{(N)}_{l}$ when $m = 1$, thus Theorem 1 is proved. □

Theorem 2. *If $\delta_e = \delta_d = 0$ and $m = 1$, then $\mathcal{L}_2$ is a form of the loss function of SGNS when we apply SGNS to a WKG.*

Proof. According to Theorem 1, the loss function $\mathcal{L}_2$ can be rewritten as follows:

$$\mathcal{L}_2 = -\sum_{t\in\mathcal{V}} d\mathbb{E}_{n\sim p(n|t,\mathcal{G})}[\psi^{+}_{t,n} + k\mathbb{E}_{s\sim q(s)}\psi^{-}_{t,s}].$$

If we don't penalize neighbors based on their degrees, i.e., $p(n|t,\mathcal{G}) = \frac{w_{t,n}}{Z}$, where $Z = \sum_{n'\in\mathcal{N}(t,\mathcal{G})} w_{t,n'}$, then we have

$$\mathcal{L}_2 = -\sum_{t\in\mathcal{V}} \frac{d}{Z} \sum_{n\in\mathcal{N}(t,\mathcal{G})} w_{t,n}[\psi^{+}_{t,n} + k\mathbb{E}_{s\sim q(s)}\psi^{-}_{t,s}].$$

Corollary 1. *According to Theorem 1, we have: (1) $\mathcal{V}$ is the same as the vocabulary $\mathcal{V}_{\mathcal{D}}$ of $\mathcal{D}$; (2) The weight of an edge between t and n in $\mathcal{G}$ is equal to their co-occurrence times in $\mathcal{D}$, as $w_{t,n} = \#(t,n)$; (3) The neighbors of a target word t in $\mathcal{G}$ is equal to all context words of t in $\mathcal{D}$, as $\mathcal{N}(t,\mathcal{G}) = \mathcal{C}_t = \cup\{\mathcal{C}_{pos}|\mathcal{D}[pos] = t\}$, where $\mathcal{D}[pos]$ is the pos-th word in $\mathcal{D}$.*

Clearly, we have $Z = \sum_{c\in\mathcal{C}_t} \#(t,c)$, thus

$$\mathcal{L}_2 = -\sum_{t\in\mathcal{V}_{\mathcal{D}}} \frac{d}{Z} \sum_{c\in\mathcal{C}_t} \#(t,c)[\psi^{+}_{t,c} + k\mathbb{E}_{s\sim q(s)}\psi^{-}_{t,s}].$$

The loss function of SGNS [Mikolov *et al.*, 2013a] is defined as follows:

$$\begin{aligned}\mathcal{L}_{SGNS} &= -\sum_{i=1}^{|\mathcal{D}|}\sum_{c\in\mathcal{C}_i}[\psi^{+}_{i,c} + k\mathbb{E}_{s\sim q(s)}\psi^{-}_{i,s}]\\ &= -\sum_{t\in\mathcal{V}_{\mathcal{D}}}\sum_{c\in\mathcal{C}_t} \#(t,c)[\psi^{+}_{t,c} + k\mathbb{E}_{s\sim q(s)}\psi^{-}_{t,s}].\end{aligned}$$

In practice, d is set to a small constant and $\frac{d}{Z}$ approximates the sub-sampling strategy of SGNS. Thus, $\mathcal{L}_2$ is a form of $\mathcal{L}_{SGNS}$ when we apply SGNS to a WKG. □

Task	Dataset	#Train	#Val	#Test	#Label
Cross-time Alignment	**NYT**	-	2,205	8,823	-
Text Stream Classification	**NYT**	47,423	6,759	12,531	7
	NYT (low)	32,475	4,624	9,260	7
	Arxiv	1.062,296	151,748	303,506	8

Table 1: Statistics of datasets.

5 Experiments

5.1 Datasets and Experimental Setting

We employ two diachronic datasets to compare the performance of different dynamic word embedding models: **(1) NYT**[2]: A collection of news from New York Times with 99,872 articles. We set the size of the time slice to one year, which generates 27 time slices from 1990 to 2016. **(2) Arxiv**[3]: A collection of abstracts from papers published on the Arxiv website from 2007 to 2021. There are 1,952,261 documents over 15 yearly time slices. Since GDWE learns dynamic word embeddings online, we apply the Misra-Gries algorithm to build a dynamic vocabulary by following Kaji and Kobayashi [2017], in which, the maximum size of the dynamic vocabulary is set to 100,000. Since matrix factorization based baselines have high memory costs for storing data matrices, they build a static vocabulary by filtering words that occurred less than 200 times according to Gong *et al.* [2020]. The static vocabulary sizes of NYT and Arxiv are 21,435 and 26,313, respectively.

To evaluate the alignment quality of temporal embedding spaces, we use the testing set constructed by Yao *et al.* [2018] on NYT to conduct a cross-time alignment experiment. We also perform text stream classification on NYT and Arxiv, which evaluates the effectiveness of dynamic word embedding models on diachronic downstream tasks. To validate the role of WKGs in learning word embeddings with limited co-occurrence information, we further perform text stream classification by considering words with an occurrence frequency lower than 100 on the NYT dataset. Specifically, we use the most popular topics and academic subjects as the classification labels for documents in NYT and Arxiv, respectively. The statistics of all datasets are shown in Table 1.

The baseline models include **TW2V** [Kulkarni *et al.*, 2015] and **AW2V** [Hamilton *et al.*, 2016], which both adopt the 2-step approach. We also employ **DW2V** [Yao *et al.*, 2018] and **CW2V** [Gong *et al.*, 2020], which jointly process all time slices for comparison. Besides, **DCWE** [Hofmann *et al.*, 2021] is a baseline which introduces temporal information to pre-trained BERT embeddings. We also conduct ablation experiments to validate the effect of each component in WKGs. Specifically, GDWE w/o WKG refers to a model that only uses $\mathcal{L}_1$ to update dynamic word embeddings[4]. GDWE w/o JC refers to a model with $\delta_s = 0$, which uses neighbors from all past WKGs without filtering knowledge that disagrees with the current WKG estimated by Equation (3).

[2]https://www.dropbox.com/s/nifi5nj1oj0fu2i/data.zip?dl=0

[3]https://www.kaggle.com/Cornell-University/arxiv

[4]Note that GDWE w/o WKG is theorectically similar to ISGNS and [Bamler and Mandt, 2017].

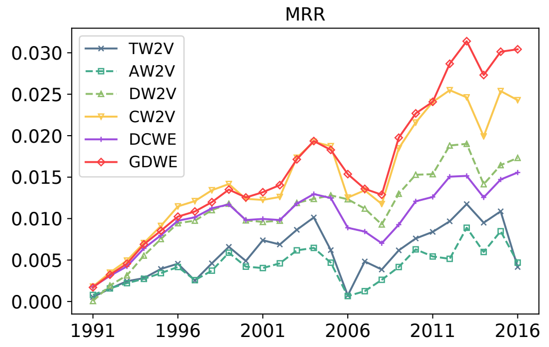

Figure 2: MRR of cross-time alignment evaluation.

Model	MRR	MP@1	MP@3	MP@5	MP@10
TW2V	0.151	0.107	0.175	0.207	0.260
AW2V	0.111	0.072	0.132	0.160	0.211
DW2V	0.289	0.253	0.309	0.346	0.356
CW2V	0.392	0.314	0.453	0.491	0.541
DCWE	0.265	0.255	0.272	0.276	0.291
GDWE w/o WKG	0.402	0.343	0.442	0.479	0.532
GDWE w/o JC	0.400	0.338	0.437	0.481	0.543
GDWE w/o $\mathcal{G}_L$	0.408	0.348	0.447	0.488	0.544
GDWE w/o $\mathcal{G}_s$	0.415	0.354	0.450	0.496	0.552
GDWE	**0.422**	**0.356**	**0.464**	**0.512**	**0.571**

Table 2: Overall results of cross-time alignment evaluation. The performance of our GDWE is significantly better than all baselines ($p < 0.05$ in t-test), except for the baseline of CW2V and four ablation models of GDWE before 2012.

GDWE w/o $\mathcal{G}_L$ and GDWE w/o $\mathcal{G}_s$ are models without using long-term WKGs $\mathcal{G}_L$ or short-term WKG $\mathcal{G}_s$, respectively.

For hyper-parameter settings of GDWE and baselines, we uniformly set the window size to 5, the subsampling ratio to 10^{-4}, the number of negative samples to 10. The embedding size of DCWE is 768, which is equal to that of pre-trained BERT embeddings. For GDWE and other baselines, the embedding size is set to 50. We use grid search to determine the best values of other hyper-parameters for GDWE and most of baselines, while DW2V, CW2V, and DCWE use the optimal hyper-parameter settings reported in their original papers. To ensure fair comparisons and affordable time costs, we train DCWE for 7 and 1 epochs on NYT and Arxiv, respectively, and we adopt embeddings generated by the dynamic component of DCWE as its results.

5.2 Cross-time Alignment Evaluation

In Yao *et al.* [2018], each testing sample contains two words sharing similar semantics at different time slices. For example, *bush-1990* and *obama-2008* build up a test sample that represents the president of the United States in different periods. We evaluate whether *obama-2008* is among the top-20 closest neighbors of *bush-1990* in the word embedding space of 2008, using MRR and MP@K as metrics.

As Figure 2 shows, the performance of most models enhanced gradually as time passed, except for TW2V and AW2V. While WKGs continuously accumulated knowledge over time, GDWE achieved the best results and widened the gap in effectiveness as compared with the top-performing baseline of CW2V. Table 2 presents the overall results of embedding space alignment, indicating that GDWE has the most notable ability to capture semantic drift, while the poor results of 2-step models (i.e., TW2V and AW2V) revealed their limitations. Besides, CW2V outperformed other baselines, because it explicitly learned the transformation from the general embedding space to time-specific embedding spaces. Moreover, DCWE obtained poor results, indicating that it is difficult for pre-trained BERT embeddings to capture semantic drift over time. For ablation models of GDWE, the results of GDWE w/o JC and GDWE w/o WKG were close, which reveals the importance of distinguishing polysemous target words in WKGs using Jaccard's coefficient. Moreover, long-term WKGs play more important roles in boosting the performance than the short-term WKG, which agrees with the fact that languages often develop slowly in a long time.

δ_e	δ_d	MRR	MP@1	MP@3	MP@5	MP@10
0.05	1	0.363	0.331	0.416	0.455	0.507
0.1	2	0.375	0.327	0.426	0.473	0.537
0.2	4	**0.422**	**0.356**	**0.464**	**0.512**	**0.571**

Table 3: Overall results of cross-time alignment evaluation with different values of δ_e and δ_d.

For completeness, we here conduct supplementary ablation analysis. First, we analyze the influence of the number of long-term WKGs (i.e., m). We set m to different values and fix other hyper-parameters. The results indicated that the effect of GDWE reached a peak (MRR = 0.422) when $m = 5$, while it obtained the worst value (MRR = 0.403) when there is only one long-term WKG (i.e., $m = 1$). Since the long-term WKG fuses all past semantic information, it may fail to distinguish words that hold different meanings among different time slices. Furthermore, when m keeps increasing to values over 5, the effect of GDWE dropped due to the incorporation of outdated information.

To validate the effect of denoising WKGs in training GDWE, we also construct and update WKGs with relaxed constraints. Specifically, we set δ_e to low values of 0.1 and 0.05, which results in reserving more low-weighted edges in WKGs. Besides, δ_d is set to 2 or 1, which means that WKGs contain nodes with few neighbors. As shown in Table 3, the weight-based edge filtering and degree-based node filtering methods are effective in denoising WKGs. It also validates the importance of denoising in lexical knowledge graphs.

5.3 Text Stream Classification

We follow Hofmann *et al.* [2021] to conduct text stream classification. First, we adapt a two-layer FNN classifier by processing the documents of each new coming time slice. Then we use the updated classifier to predict the labels of documents in the current, past, and future testing sets. For NYT and Arxiv, we use an attention-based Bi-LSTM model to obtain document embeddings with the input of pre-trained dynamic word embeddings [Yang *et al.*, 2016]. For NYT (low), we use the average of embeddings of all low-frequency words in a document as its representation. Figure 3 shows the model performance in terms of accuracy and macro-F1.

Different from the cross-time alignment evaluation, this task validates the alignment of anchor words over time im-

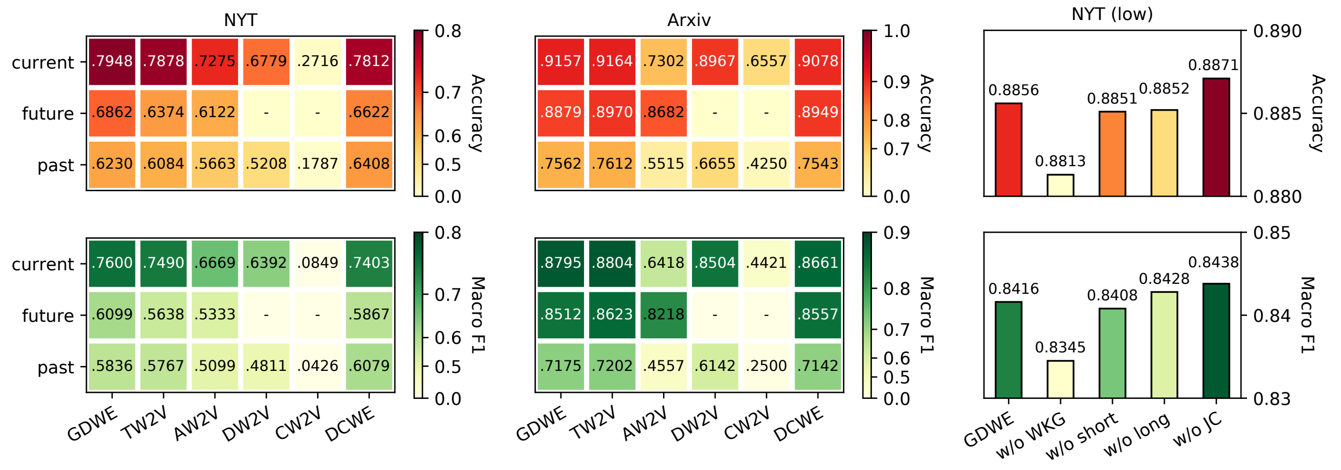

Figure 3: Results on text stream classification. "-" means that the result can not be obtained by the corresponding model.

plicitly, because most of the words in a document may be semantics-invariant. For this reason, CW2V and DW2V which performed well in cross-time alignment, obtained the worst results in this task. Conversely, TW2V and AW2V showed their advantages in the alignment of anchor words. Though DCWE performed poorly in the unsupervised cross-time alignment evaluation even with adequate training, it is quite competitive in this supervised text stream classification task using similar computational time with other baselines, which is consistent with the results in prior LSC tasks [Schlechtweg *et al.*, 2020; Basile *et al.*, 2020; Kutuzov and Pivovarova, 2021]. The different results of DCWE in these two tasks indicate that contextual embedding models can significantly improve the effect of NLP downstream tasks, but their performance depend on label supervision and abundant training samples due to their massive model parameters. Since there may be fewer semantic changes occurring in academic papers than in news, the differences between the performance of GDWE, TW2V, and DCWE are slight on Arxiv. Notably, GDWE performed quite well on the current and future documents of NYT, owing to its outstanding ability of modeling language evolution.

Furthermore, we explore the influence of WKGs on learning embeddings for low-frequency words. As the results of current documents in NYT (low) show, WKGs boosted the performance of text stream classification that only utilized words with an occurrence frequency lower than 100. Since GDWE w/o JC adopted a "high risk" strategy and used knowledge from all past WKGs, it achieved the best performance in learning high-quality embeddings with limited contextual information. Therefore, WKGs can alleviate the underfitting problem of online learning to a certain extent.

5.4 Trending Words Detection

As an important application of dynamic word embeddings and WKGs, mining time-specific trending words from a text stream of real-time news like NYT can assist public opinion detection. Following Saeed *et al.* [2019], we capture the boost of trending words through the heartbeat graph $\mathcal{G}_h^{(i)}$ built from WKGs of the current and previous time slices (denoted as $\mathcal{G}_l^{(i)}$ and $\mathcal{G}_l^{(i-1)}$). Then we assume that a trending word tends to encounter significant semantic drift and have a high centrality in $\mathcal{G}_h^{(i)}$. Specifically, we use the distance between the current and previous embeddings of a word t to measure its degree of semantics change, which is defined as follows:

$$SC(t) = \parallel \mathbf{e}_t^{(i)} - \mathbf{e}_t^{(i-1)} \parallel_2^2, \tag{7}$$

where $\mathbf{e}_t^{(i)}$ is the embedding of t in the i-th time slice. Besides, we estimate the centrality of t by summing up the positive weights of its connected edges in $\mathcal{G}_h^{(i)}$ as follows:

$$C(t) = \sum_{n \in \mathcal{N}(t, \mathcal{G}_h^{(i)})} \log_2(\mathbf{e}_t^{(i)}(\mathbf{e}_n^{(i)})^T + 1) \times w_{n,t}, \tag{8}$$

where $n \in \mathcal{N}(t, \mathcal{G}_h^{(i)})$ is a neighbor of t in $\mathcal{G}_h^{(i)}$ with $\mathbf{e}_t^{(i)}(\mathbf{e}_n^{(i)})^T > 0$ and $w_{n,t}$ is the weight of the edge between n and t. Then, the trending score of t is defined as follows:

$$TS(t) = SC(t) \times C(t). \tag{9}$$

Figure 4 illustrates top-25 trending words and their connections for NYT news in 2001. The trending words detection approach based on WKGs effectively discovered that the 9/11 terrorist attack was the most concerning social event in 2001. Moreover, details of the event were captured, including the terrorists ("*osama bin laden*"), the terrorist organization ("*qaeda*"), and the occurrence time ("*sept*"). In addition, there was a breakthrough in human embryonic stem cell research in 2001 according to trending words. Finally, the result of trending words detection also includes words with general meanings such as entertainment-related words ("*singer*").

5.5 Lexical Semantic Drift

To validate the ability of GDWE to capture lexical semantic drift qualitatively, we describe the changing semantics of representative target words "*apple*" in NYT and "*network*" in Arxiv through their neighbor words at different time slices.

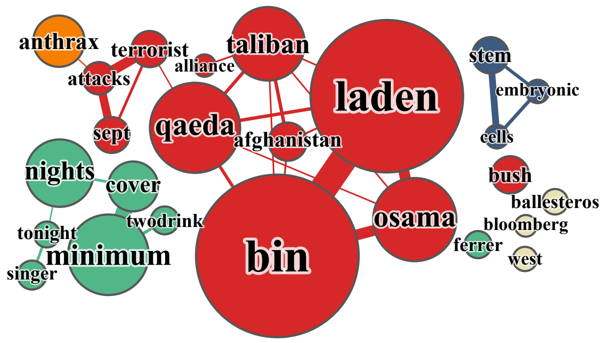

Figure 4: Top-25 trending words and their connections for NYT news in 2001 detected from WKGs. The diameters of nodes and the thickness of edges are proportional to their weights.

Year	Neighbor words of "*apple*" in NYT				
	1	2	3	4	5
1993	blueberry	chocolate	tart	mascarpone	pie
2008	imac	chips	pc	itunes	server
2010	ipone	chips	server	conagra	itunes
2011	chips	blackberry	3g	android	ipad
2016	microsoft	android	blackberry	samsung	google

Year	Neighbor words of "*network*" in Arxiv				
	1	2	3	4	5
2010	wireless	peertopeer	connectivity	p2p	internet
2014	overlay	multiplex	vehicular	peertopeer	sensor
2017	overlay	neural	feedforward	dnn	generative
2018	neural	convolutional	feedforward	cnn	dnn
2019	neural	gcn	convolutional	cnn	capsule
2021	neural	convolutional	dnn	cnn	gcn

Table 4: Top-5 neighbor words of target words "*apple*" and "*network*" in different years, where different noun forms of "*network*" are normalized for clarity.

As shown in Table 4, the neighbors of "*apple*" gradually drifted from dessert-related words to electronic technology-related words. Additionally, GDWE effectively revealed the products and technologies that attracted attention in different periods. For example, personal computers ("*imac*") were the main products of the Apple company in 2008. In 2010 and 2011, the hot spot shifted to portable devices ("*iphone*" and "*ipad*") and their key technologies, including wireless network technology ("*3g*") and operating systems ("*android*"). And the neighbor words of "*apple*" in 2016 changed to other technology companies (e.g., "*microsoft*" and "*google*").

For the target word "*network*", its semantics gradually drifted from communication networks to neural networks (NNs) with the development of computer science. Specifically, its neighbor words in 2010 and 2014 were mainly technical terms related to communication networks, including "*p2p*", "*internet*", and "*overlay*". In recent years, NNs have gained widespread attention and GDWE can discover the development of NN architectures over time, from feedforward NNs and generative networks in 2017 to convolutional NNs in 2018, and then to graph convolutional NNs in 2019.

6 Efficiency Analysis

6.1 Time Complexity

In this part, we analyze the time complexity of our GDWE, especially on the update and utilization of WKGs. For a WKG $\mathcal{G}$, we use V and E to denote the numbers of its nodes and edges, respectively. Besides, D denotes the average degree of each node in $\mathcal{G}$. According to Algorithm 2, it takes a merge of two WKGs, a traverse of all nodes, and a traverse of edges to update a WKG, which costs $\mathcal{O}(2E+V)$, and so as the time complexity of updating a short-term WKG or a long-term WKG. Moreover, according to Equations (2) and (3), the time complexity of sampling neighbors of a word is $\mathcal{O}(2dD)$, where d is the number of samples. So it takes $\mathcal{O}(2dDV)$ to retrieve word co-occurrence information from a WKG.

6.2 Runtime Comparison

Here, we compare the runtime of our GDWE and baselines by Intel(R) Xeon(R) Silver 4214R CPU @ 2.40GHz. Let's take the training process of each model on NYT as an example. Among all dynamic word embedding models, DW2V is the fastest one, taking about 10 minutes to converge, but it performed poorly in the cross-time alignment and text stream classification tasks. GDWE w/o WKG only needed about 31 minutes for training. Because GDWE requires extra time costs to update and retrieve knowledge in WKGs, it took about 44 minutes to train. With a rational cost of time, GDWE achieved better results than GDWE w/o WKG in many experiments, that is, GDWE better balances the effectiveness and efficiency of learning dynamic word embeddings. Because both AW2V and TW2V need to pre-train static word embeddings on each time slice, which brings an extra time cost, they cost about 73 and 152 minutes for training, respectively. Finally, although CW2V and DCWE performed competitively in time-specific tasks, their time costs are quite high, which were about 18.3 hours and 13.6 hours, respectively.

7 Conclusion

In this study, we propose a GDWE model to learn dynamic word embeddings continuously, where WKGs are introduced to encode and update long-term and short-term word co-occurrence knowledge. To help GDWE capture semantic drift even with limited contexts, we retrieve knowledge from WKGs to update dynamic word embeddings. Theoretical analysis and extensive experiments on cross-time alignment, text stream classification, and qualitative analysis validate the effectiveness of our model. Since how to apply contextual word embeddings to dynamic conditions more properly and efficiently remains to be studied, we will further explore the differences of contextual and traditional embeddings in the temporal scenario. Besides, as WKG provides an effective way to memorize past word co-occurrence information and distinguish the change of word sense based on common neighbors, it has the potential to be combined with contextual embedding models by developing time-specific self-supervised tasks using temporal co-occurrence knowledge.

Acknowledgements

We are grateful to the reviewers for their constructive comments and suggestions. This work has been supported by the National Natural Science Foundation of China (61972426).

References

[Bamler and Mandt, 2017] Robert Bamler and Stephan Mandt. Dynamic word embeddings. In *ICML*, pages 380–389, 2017.

[Basile *et al.*, 2020] Pierpaolo Basile, Annalina Caputo, Tommaso Caselli, Pierluigi Cassotti, and Rossella Varvara. Diacr-ita @ EVALITA2020: overview of the EVALITA2020 diachronic lexical semantics (diacr-ita) task. In *EVALITA*, 2020.

[Devlin *et al.*, 2019] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, pages 4171–4186, 2019.

[Dhingra *et al.*, 2022] Bhuwan Dhingra, Jeremy R Cole, Julian Martin Eisenschlos, Daniel Gillick, Jacob Eisenstein, and William W Cohen. Time-aware language models as temporal knowledge bases. *TACL*, 10:257–273, 2022.

[Giulianelli *et al.*, 2020] Mario Giulianelli, Marco Del Tredici, and Raquel Fernández. Analysing lexical semantic change with contextualised word representations. In *ACL*, pages 3960–3973, 2020.

[Gong *et al.*, 2020] Hongyu Gong, Suma Bhat, and Pramod Viswanath. Enriching word embeddings with temporal and spatial information. In *CoNLL*, pages 1–11, 2020.

[Hamilton *et al.*, 2016] William L. Hamilton, Jure Leskovec, and Dan Jurafsky. Diachronic word embeddings reveal statistical laws of semantic change. In *ACL*, pages 1489–1501, 2016.

[Hofmann *et al.*, 2021] Valentin Hofmann, Janet B. Pierrehumbert, and Hinrich Schütze. Dynamic contextualized word embeddings. In *ACL-IJCNLP*, pages 6970–6984, 2021.

[Hu *et al.*, 2019] Renfen Hu, Shen Li, and Shichen Liang. Diachronic sense modeling with deep contextualized word embeddings: An ecological view. In *ACL*, pages 3899–3908, 2019.

[Kaji and Kobayashi, 2017] Nobuhiro Kaji and Hayato Kobayashi. Incremental skip-gram model with negative sampling. In *EMNLP*, pages 363–371, 2017.

[Kim *et al.*, 2014] Yoon Kim, Yi-I Chiu, Kentaro Hanaki, Darshan Hegde, and Slav Petrov. Temporal analysis of language through neural language models. In *LTCSS@ACL*, pages 61–65, 2014.

[Kulkarni *et al.*, 2015] Vivek Kulkarni, Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. Statistically significant detection of linguistic change. In *WWW*, pages 625–635, 2015.

[Kutuzov and Pivovarova, 2021] Andrey Kutuzov and Lidia Pivovarova. Rushifteval: A shared task on semantic shift detection for russian. In *Dialogue*, 2021.

[Kutuzov *et al.*, 2018] Andrey Kutuzov, Lilja Øvrelid, Terrence Szymanski, and Erik Velldal. Diachronic word embeddings and semantic shifts: A survey. In *COLING*, pages 1384–1397, 2018.

[Lazaridou *et al.*, 2021] Angeliki Lazaridou, Adhiguna Kuncoro, Elena Gribovskaya, Devang Agrawal, Adam Liska, Tayfun Terzi, Mai Gimenez, Cyprien de Masson d'Autume, Tomáš Kočiský, Sebastian Ruder, Dani Yogatama, Kris Cao, Susannah Young, and Phil Blunsom. Mind the gap: Assessing temporal generalization in neural language models. In *NeurIPS*, pages 29348–29363, 2021.

[Levy and Goldberg, 2014] Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In *NIPS*, pages 2177–2185, 2014.

[McMahon, 1994] April M. S. McMahon. *Understanding language change*. Cambridge University Press, 1994.

[Mikolov *et al.*, 2013a] Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *ICLR (Workshop Poster)*, 2013.

[Mikolov *et al.*, 2013b] Tomás Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119, 2013.

[Rosin *et al.*, 2022] Guy D Rosin, Ido Guy, and Kira Radinsky. Time masking for temporal language models. In *WSDM*, pages 833–841, 2022.

[Saeed *et al.*, 2019] Zafar Saeed, Rabeeh Ayaz Abbasi, Muhammad Imran Razzak, and Guandong Xu. Event detection in twitter stream using weighted dynamic heartbeat graph approach [application notes]. *IEEE Computational Intelligence Magazine*, 14(3):29–38, 2019.

[Schlechtweg *et al.*, 2020] Dominik Schlechtweg, Barbara McGillivray, Simon Hengchen, Haim Dubossarsky, and Nina Tahmasebi. Semeval-2020 task 1: Unsupervised lexical semantic change detection. In *SemEval@COLING*, pages 1–23, 2020.

[Spitz and Gertz, 2018] Andreas Spitz and Michael Gertz. Exploring entity-centric networks in entangled news streams. In *WWW (Companion Volume)*, pages 555–563, 2018.

[Yang *et al.*, 2016] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alexander J. Smola, and Eduard H. Hovy. Hierarchical attention networks for document classification. In *NAACL-HLT*, pages 1480–1489, 2016.

[Yao *et al.*, 2018] Zijun Yao, Yifan Sun, Weicong Ding, Nikhil Rao, and Hui Xiong. Dynamic word embeddings for evolving semantic discovery. In *WSDM*, pages 673–681, 2018.

[Zuckerman and Last, 2019] Matan Zuckerman and Mark Last. Using graphs for word embedding with enhanced semantic relations. In *TextGraphs@EMNLP*, pages 32–41, 2019.