Using Context-to-Vector with Graph Retrofitting to Improve Word Embeddings

Jiangbin Zheng†, Yile Wang†, Ge Wang, Jun Xia,
Yufei Huang, Guojiang Zhao, Yue Zhang, Stan Z. Li*
School of Engineering, Westlake University
Institute of Advanced Technology, Westlake Institute for Advanced Study
{zhengjiangbin,wangyile,wangge,xiajun,huangyufei,
zhaoguojiang,zhangyue,Stan.ZQ.Li}@westlake.edu.cn

Abstract

Although contextualized embeddings generated from large-scale pre-trained models perform well in many tasks, traditional static embeddings (e.g., Skip-gram, Word2Vec) still play an important role in low-resource and lightweight settings due to their low computational cost, ease of deployment, and stability. In this paper, we aim to improve word embeddings by 1) incorporating more contextual information from existing pre-trained models into the Skip-gram framework, which we call *Context-to-Vec*; 2) proposing a post-processing retrofitting method for static embeddings independent of training by employing priori synonym knowledge and weighted vector distribution. Through extrinsic and intrinsic tasks, our methods are well proven to outperform the baselines by a large margin.

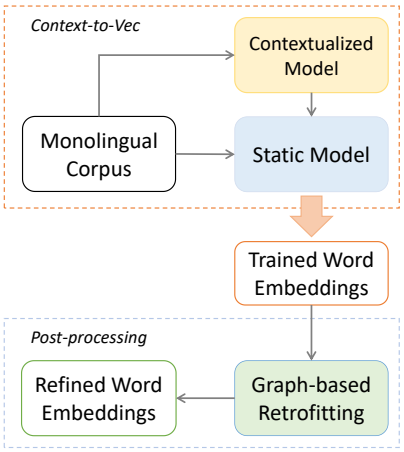

Figure 1: The overall training pipeline of our proposed word embeddings training and post-processing methods. In the Context-to-Vec phase, static word embeddings are trained using contextualized embeddings based on a monolingual corpus. While in the post-processing phase, external knowledge is introduced to fine-tune the word vectors based on the graph topology.

1 Introduction

Contextualized embeddings such as BERT (Devlin et al., 2018) and RoBERTa (Liu et al., 2019) have become the default architectures for most downstream NLP tasks. However, they are computationally expensive, resource-demanding, hence environmentally unfriendly. Compared with contextualized embeddings, static embeddings like Skip-gram (Mikolov et al., 2013) and GloVe (Pennington et al., 2014) are lighter and less computationally expensive. Furthermore, they can even perform without significant performance loss for context-independent tasks like lexical-semantic tasks (e.g., word analogy), or some tasks with plentiful labeled data and simple language (Arora et al., 2020).

Recent work has attempted to enhance static word embedding while maintaining the benefits of both contextualized embedding and static embedding. Among these efforts, one category is the direct conversion of contextualized embeddings to static embeddings (Bommasani et al., 2020). The other category of enhancement is to make use of contextualized embeddings for static embeddings (Melamud et al., 2016). The latter category is a newer paradigm, which we call *Context-to-Vec*. This paradigm not only alleviates the word sense ambiguities from static embedding, but also fuses more syntactic and semantic information in the context within a fixed window.

For the *Context-to-Vec* paradigm, an association between contextualized word vectors and static word vectors is essentially required. In this case, the contextualized signal serves as a source of information enhancement for the static embeddings (Vashishth et al., 2018). However, the existing efforts only consider the contextualized embeddings of center words as the source, which is actually incomplete since the contextualized features for the context words of the center words are ignored.

† Co-first author.
* Corresponding author.

Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 8154 - 8163

In addition, benefiting from the invariance and stability of already trained static embeddings, post-processing for retrofitting word vectors is also an effective paradigm for improving static embeddings. For example, one solution is an unsupervised approach that performs a singular value decomposition to reassign feature weights (Artetxe et al., 2018), but this does not utilize more external knowledge and lacks interpretation. Poor initial spatial distribution of word embeddings obtained from training may lead to worse results. Another common solution is to use a synonym lexicon (Faruqui et al., 2014), which exploits external prior knowledge with more interpretability but does not take into account the extent of spatial distance in the context.

In this work, we unify the two paradigms above within a model to enhance static embeddings. On the one hand, we follow the *Context-to-Vec* paradigm in using contextualized representations of center words and their context words as references for static embeddings. On the other hand, we propose a graph-based semi-supervised post-processing method by using a synonym lexicon as prior knowledge, which can leverage proximal word clustering signals and incorporate distribution probabilities. The overall training pipeline is shown in Fig.1. The pipeline is divided into two separate phases, where the first phase follows the *Context-to-Vec* paradigm by distilling contextualized information into static embeddings, while the second phase fine-tunes the word embeddings based on graph topology. To validate our proposed methods, we evaluate several intrinsic and extrinsic tasks on public benchmarks. The experimental results demonstrate that our models significantly outperform traditional word embeddings and other distilled word vectors in word similarity, word analogy, and word concept categorization tasks. Besides, our models moderately outperform baselines in all downstream clustering tasks.

To our knowledge, we are the first to train static word vectors by using more contextual knowledge in both training and post-processing phases. The code and trained embeddings are made available at https://github.com/binbinjiang/Context2Vector.

2 Related Work

Word Embeddings. For traditional static word embeddings, Skip-gram and CBOW are two models based on distributed word-context pairs (Mikolov et al., 2013). The former uses center words to predict contextual words, while the latter uses contextual words to predict central words. GloVe is a log-bilinear regression model which leverages global co-occurrence statistics of corpus (Pennington et al., 2014); FASTTEXT takes into account subword information by incorporating character n-grams into the Skip-gram model (Bojanowski et al., 2017). While contextualized word embeddings (Peters et al., 2018; Devlin et al., 2018) have been widely used in modern NLP. These embeddings are actually generated using language models such as LSTM and Transformer (Vaswani et al., 2017) instead of a lookup table. This paradigm can generally integrate useful sentential information into word representations.

Context-to-Vec. The fusion of contextualized and static embeddings is a newly emerged paradigm in recent years. For instance, Vashishth et al. (2018) propose SynGCN using GCN to calculate context word embeddings based on syntax structures; Bommasani et al. (2020) introduce a static version of BERT embeddings to represent static embeddings; Wang et al. (2021) enhance the Skip-gram model by distilling contextual information from BERT. Our work also follows this paradigm but introduce more context constraints.

Post-processing Embeddings. Post-processing has been used for improving trained word embeddings. Typically, Faruqui et al. (2014) use synonym lexicons to constrain the semantic range; Artetxe et al. (2018) propose a method based on eigenvalue singular decomposition. Similar to these techniques, our post-processing method is easy for deployment and can be applied to any static embeddings. The difference is that we not only take advantage of the additional knowledge, but also consider the distance weights of the word vectors, overcoming the limitations of existing methods with better interpretability.

3 Proposed Methods

3.1 Embedding Representations

As shown in Fig.2, our proposed framework consists of four basic components. Formally, given a sentence $s = \{w_1, w_2, ..., w_n\}(w_i \in D)$, our objective is to model the relationship between the center word w_i and its context words $\{w_{i-w_s}, ..., w_{i-1}, w_{i+1}, ..., w_{i+w_s}\}$.

Contextualized Embedding Module. To incor-

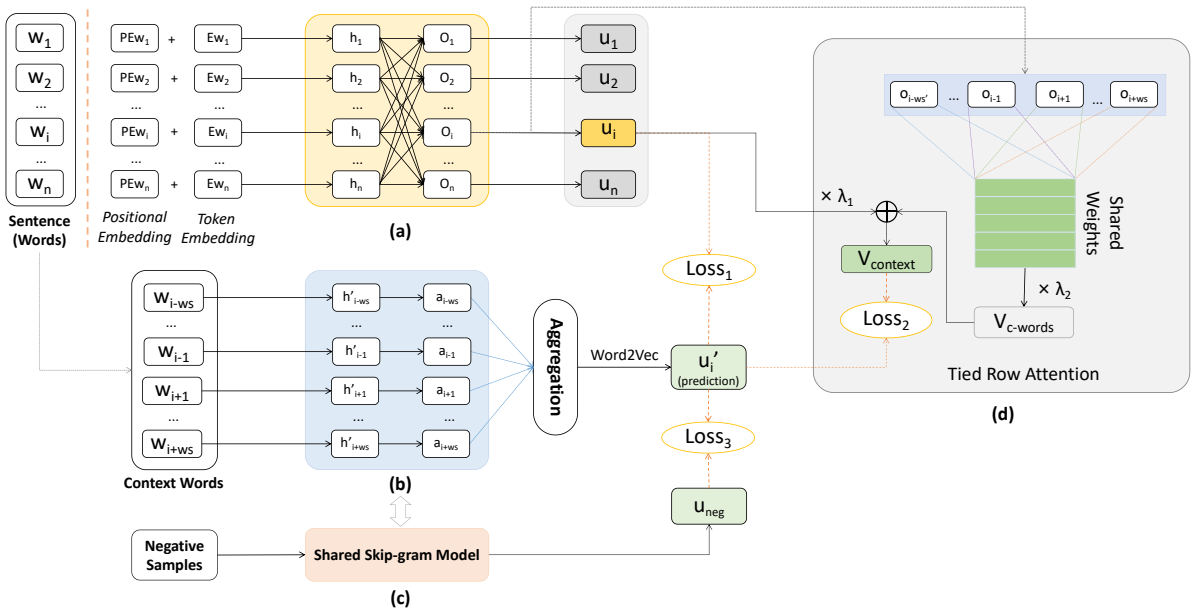

Figure 2: Main framework of our model. (a) **Contextual Embedding Module** generally consists of a pre-trained language model (BERT-like models) that provides the main enhancement information for static embeddings; (b) **Static Embedding Module** is the core component for training word embeddings from scratch and obtaining distilled contextualized information; (c) **Negative Sample Module** collects negative samples randomly and constructs contrast loss to improve the robustness and generalization; (d) **Tied Contextualized Attention Module** is to capture the contextualized embeddings for the context words as supplementary information.

porate contextualized information, an embedding u_i of the center word w_i needs to be generated from a pre-trained language model (Fig.2(a)). Taking the BERT model as an example, the center word w_i is first transformed into a latent vector h_i, then h_i is fed to a bidirectional Transformer for self-attention interaction. Finally, the output representation $o_i \in R^d$ is linearly mapped to $u_i \in R^{d_{emb}}$ through a linear layer as:

$$u_i = W_o \text{Linear}(\text{SA}(h_i)) = W_o o_i, \quad (1)$$

where $W_o \in R^{d_{emb} \times d}$ denotes model parameters, $\text{Linear}(*)$ denotes a linear mapping layer, and $\text{SA}(*)$ denotes self-attention. In practice, the size of o_i is $d = 768$, and the size of u_i is $d_{emb} = 300$. The h_i here is a sum of the *Token Embedding* E_{w_i} and the *Positional Embedding* PE_{w_i} as:

$$h_i = E_{w_i} + PE_{w_i}. \quad (2)$$

Static Embedding Module. The Skip-gram model (Fig.2(b)) is used as the static embedding module. Our method does not directly fit the Skip-gram model by replacing an embedding table, although the original Skip-gram uses an embedding table of center words as the final embedding. Instead, to make the context words predictable and to enable negative sampling from the vocabulary, contextualized representations are used for the center words, while an embedding table of the context words is used for the output static embedding.

3.2 Heuristic Semantic Equivalence

As mentioned above, a key issue for the *Context-to-Vec* paradigm is to bridge the gap between contextualized and static word vectors. To this end, a main intuition is to find key equivalent semantic connections between contextualized vectors and static vectors. We take the following heuristics:

Heuristic 1: *For a given sentence, the contextualized embedding representation of a center word can be semantically equivalent to the static embedding of the center word in the same context.*

According to **Heuristic 1**, in order to model the center word w_i and its context words w_{i+j} (note here that the illegal data that indexes less than 0 or greater than the maximum length are ignored), a primary training target is to maximize the probability of the context words $w_{i+j}(|j| \in [1, w_s])$ in the Skip-gram model:

$$p(w_{i+j}|w_i) = \frac{\exp({u'_{i+j}}^T u_i)}{\sum_{w_k \in D} \exp({u'_k}^T u_i)}, \quad (3)$$

where u_i is the contextualized representation of

the center word, and u'_k is the static embedding from a center word w_k that is generated by a static embedding table with size $d' = 300$.

For **Heuristic 1**, the contextualized word embedding of any center word is essentially used as reference for corresponding static word embedding. Such a source for information enhancement implicitly contains the context of the contextualized embedding, but explicitly ignores the contextual information which is easily accessible. Hence, the proposed:

Heuristic 2: *Inspired by the idea of Skip-gram-like modeling, the contextualized embedding representation for the context words of a center word can be also semantically equivalent to represent the static embedding of the center word.*

To model this semantic relationship, we introduce a **Tied Contextualized Attention** module (Fig.2(d)) for explicitly attending contextual signals, which complements **Heuristic 1** by incorporating more linguistic knowledge into the static embedding. In particular, assume that the center word w_i in the contextualized embedding module corresponds to the contextual vocabulary notated as $\{w_{i-w'_s}, ..., w_{i-1}, w_{i+1}..., w_{i+w'_s}\}$, then the output contextual attention vector can be computed as:

$$\begin{aligned} V_{context} &= \lambda_1 V_{center} + \lambda_2 V_{c-words} \\ &= \lambda_1 o_i^T W_1 + \lambda_2 \tau(U_{1\le|k|\le i+w'_s} \phi(o_k^T W_2)) \\ &= \lambda_1 o_i^T W_1 + \lambda_2 \frac{\phi(\sum_{1\le|k|\le i+w'_s} o_k^T)}{2w'_s} W_2, \end{aligned} \tag{4}$$

where V_{center} denotes the embedding representation of the center word, which is a residual connection here. And $V_{c-words}$ denotes the embedding representations of corresponding context words. ϕ is an optional nonlinear function, $U(*)$ is a merge operation, and τ is an average pooling operation. $W_1 \in R^{d\times d_{emb}}$ and $W_2 \in R^{d\times d_{emb}}$ are trainable parameters, in which W_2 denotes the weight assignment of each context vector.

Since each o_k has similar linguistic properties, the weight W_2 can be shared, and we name this module **Tied Contextualized Attention** mechanism. Therefore, the weighted average of the linear transformation of all context vectors can be reduced to the weighted linear output of the average of all vectors as shown in Eq.4. This weight-sharing mechanism can help speed up calculations.

In practice, to reduce the complexity, the weight parameter λ_1 and λ_2 are the same; the u_i in Eq.1 can be directly used as $V_{context}$; the value of w'_s is the same as that of w_s, e.g., 5.

3.3 Training Objectives

The modular design requires our model to satisfy multiple loss constraints simultaneously, allowing static embeddings to introduce as much contextual information as possible. Given a training corpus with N sentences $s_c = \{w_1, w_2, ..., w_{n_c}\}(c \in [1, N])$, our loss functions can be described as follows.

Semantic Loss. As illustrated in **Heuristic 1**, one of our key objectives is to learn the semantic similarity between the contextualized embedding and the static embedding of the center word. To speed up computation, the inner product of the normalized vectors can be used as the loss L_1:

$$L_1 = -\sum_{c=1}^{N}\sum_{i=1}^{n_c}(\log\sigma((\sum_{1\le|j|\le w_s} u'_{i+j})^T u_i)], \tag{5}$$

where σ is the sigmoid function.

Contextualized Loss. As described in **Heuristic 2**, the contextualized embeddings for the context words of the center word are explicitly introduced to further enhance the static embedding, thus the Contextualized Loss L_2 is expressed as:

$$L_2 = -\sum_{c=1}^{N}\sum_{i=1}^{n_c}(\log\sigma(V_{context}^T u_i)). \tag{6}$$

Contrastive Negative Loss. Negative noisy samples (Fig.2(c)) can improve the robustness and effectively avoid the computational bottleneck. This trick is common in NLP. Our Contrastive Negative Loss L_3 is calculated as:

$$L_3 = \sum_{c=1}^{N}\sum_{i=1}^{n_c}\sum_{m=1}^{k} E_{w_{neg_m}\ P(w)}[\log\sigma(u'^T_{neg_m} u_i)], \tag{7}$$

where w_{neg_m} denotes a negative sample, k is the number of negative samples and $P(w)$is a noise distribution set.

Joint Loss. The final training objective is a joint loss L for multi-tasks as:

$$L = \eta_1 L_1 + \eta_2 L_2 + \eta_3 L_3, \tag{8}$$

where each hyperparameter η_i denotes a weight.

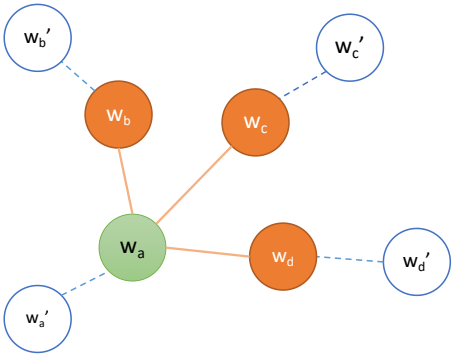

Figure 3: A word graph diagram with edges between related words. The dashed edges indicate the corresponding edge relationships between observed word vectors (white nodes) and inferred word vectors (colored nodes). And the solid edges indicate the relationship between the word (green node) to be refined and its corresponding synonyms (orange nodes).

3.4 Graph-based Post-retrofitting

In the post-processing stage, we propose a new semi-supervised retrofitting method for static word embeddings based on graph topology (Xia et al., 2022; Wu et al., 2021, 2020). This method overcomes the limitations of previously existing work by 1) using a synonym lexicon as priori external knowledge. Since both contextualized embeddings and static embeddings are trained in a self-supervised manner, the word features originate only from within the sequence and no external knowledge is considered; 2) converting the Euclidean distances among words into a probability distribution (McInnes et al., 2018), which is based on the special attributes that the trained static word vectors are mapped in a latent Euclidean space and remain fixed.

Word Graph Representation. Suppose that $V = \{w_1, ..., w_n\}$ is a vocabulary (i.e., a collection of word types). We represent the semantic relations among words in V as an undirected graph (V, E), with each word type as a vertex and edges $(w_i, w_j) \in E$ as the semantic relations of interest. These relations may vary for different semantic lexicons. Matrix Q' represents the set of trained word vectors for $q'_i \in R^{Dim}$, in which q'_i corresponds to the word vector of each word w_i in V.

Our objective is to learn a set of refined word vectors, denoted as matrix $Q = (q_1, ..., q_n)$, with the columns made close to both their counterparts in Q' and the adjacent vertices according to the probability distribution. A word graph with such edge connectivity is shown in Fig.3, which can be interpreted as a Markov random field (Li, 1994).

Retrofitting Objective. To refine all word vectors close to the observed value q'_i and its neighbors q_j ($(i, j) \in E$), the objective is to minimize:

$$\begin{aligned}\Psi(Q) = &\sum_{i=1}^{n}(\alpha_i||q_i - {q'}_i||^2 \\ &+\beta_i \sum_{(i,j)\in E} \gamma_{ij}||q_i - q_j||^2),\end{aligned} \tag{9}$$

where α_i, β_i, and γ_{ij} control the relative strengths of associations, respectively. Since Ψ is convex in Q, we can use an efficient iterative update algorithm. The vectors in Q are initialized to be equal to the vectors in Q'. Assuming that w_i has m adjacent edges corresponding to m synonyms, then we take the first-order derivative of Ψ with respect to a q_i vector and equate it to zero, yielding the following online update:

$$q_i = \alpha_i {q'}_i + \beta_i \frac{\sum_{j:(i,j)\in E} \gamma_{ij} q_j}{m}. \tag{10}$$

By default, α_i and β_i take the same value 0.5, and γ_{ij} can be expressed as:

$$\gamma_{ij} = g(d_{ij}|\sigma, \nu) = C_\nu (1 + \frac{d_{ij}^2}{\sigma\nu})^{-(\nu+1)} \in (0, 1], \tag{11}$$

in which σ is a scale parameter, ν is a positive real parameter, and C_ν is the normalization factor of ν as (the following $\Gamma(*)$ denotes the gamma function):

$$C_\nu = 2\pi (\frac{\Gamma(\frac{\nu+1}{2})}{\sqrt{\nu\pi}\Gamma(\frac{\nu}{2})})^2, \tag{12}$$

and d_{ij} calculates the sum of Euclidean distances of the feature vectors across all dimensions Dim as:

$$d_{ij} = \sqrt{\sum_{k=0}^{Dim} (q_{i_k} - q_{j_k})^2}. \tag{13}$$

Through the above process, the distance distribution is first converted into a probability distribution, and then the original word graph is represented as a weighted graph. This retrofitting method is modular and can be applied to any static embeddings.

4 Experiments

We use Wikipedia to train static embeddings. The cleaned corpus has about 57 million sentences and 1.1 billion words. The total number of vocabularies is 150k. Sentences between 10 and 40 in length were selected during training.

Types	Models	Word Similarity							Analogy	
		WS353	WS353S	WS353R	SimiLex	RW	MEN	RG65	Google	SemEval
Static	Skip-gram	61.0	68.9	53.7	34.9	34.5	67.0	75.2	43.5	19.1
	Skip-gram(context)	53.2	60.9	43.5	32.0	28.0	58.8	69.3	40.6	16.7
	CBOW	62.7	70.7	53.9	38.0	30.0	68.6	72.7	58.4	18.9
	GloVe	54.2	64.3	50.2	31.6	29.9	68.3	61.8	45.3	18.7
	FASTTEXT	68.3	74.6	61.6	38.2	37.3	74.8	80.8	72.7	19.5
	Deps	60.6	73.1	46.8	39.6	33.0	60.5	77.1	36.0	22.9
Contextualized	ELMo$_{token}$	54.1	69.1	39.2	41.7	42.1	57.7	69.6	39.8	19.3
	GPT2$_{token}$	65.5	71.5	55.7	48.4	31.6	69.8	63.2	33.1	21.3
	BERT$_{token}$	57.8	67.3	42.5	48.9	29.5	54.8	66.1	31.7	22.0
	XLNet$_{token}$	62.4	74.4	53.2	48.1	34.0	66.3	68.3	32.6	22.2
	ELMo$_{word}$	45.5	62.1	32.4	40.6	34.6	57.2	60.9	36.4	22.6
	GPT2$_{word}$	30.7	31.4	27.6	26.4	22.5	26.2	10.6	19.9	12.5
	BERT$_{word}$	24.0	31.0	14.1	13.4	10.8	22.0	18.5	25.2	10.1
	XLNet$_{word}$	62.8	69.8	55.5	49.0	29.7	61.7	63.4	31.9	22.5
	ELMo$_{avg}$	58.3	71.3	47.4	43.6	38.4	65.5	66.8	49.1	21.2
	GPT2$_{avg}$	64.5	72.1	59.7	46.9	29.1	68.6	80.0	37.2	21.9
	BERT$_{avg}$	59.4	67.0	49.9	46.8	30.8	66.3	81.2	59.4	20.8
	XLNet$_{avg}$	64.9	72.3	58.0	47.3	27.7	64.1	69.7	30.8	23.2
Context-to-Vec	ContextLSTM	63.5	66.6	57.3	39.3	23.1	66.4	72.6	60.7	20.0
	SynGCN	60.9	73.2	45.7	45.5	33.7	71.0	79.6	58.5	**23.4**
	BERT+Skip-gram	72.8	75.3	66.7	49.4	42.3	76.2	78.6	75.8	20.2
	Ours(preliminary)	76.9	76.7	68.3	54.9	43.5	76.8	84.3	75.6	20.3
	Ours(+post-process)	**78.9**	**77.0**	**70.1**	**55.2**	**44.0**	**77.9**	**85.1**	**76.3**	21.4

Table 1: Results on word similarity and analogy tasks. *Ours(preliminary)*: without post-processing; *Ours (+post-process)*: with post-processing. The best results are bolded, and the second-best underlined.

4.1 Evaluation Benchmarks

We conduct both intrinsic and extrinsic evaluations.

Intrinsic Tasks. We conduct **word similarity** tasks on the WordSim-353 (Finkelstein et al., 2001), SimLex-999 (Kiela et al., 2015), Rare Word (RW) (Luong et al., 2013), MEN-3K (Bruni et al., 2012), and RG-65 (Rubenstein and Goodenough, 1965) datasets, computing the Spearman's rank correlation between the word similarity and human judgments. For **word analogy** task, we compare the analogy prediction accuracy on the Google (Mikolov et al., 2013) dataset. The Spearman's rank correlation between relation similarity and human judgments is compared on the SemEval-2012 (Jurgens et al., 2012) dataset. **Word concept categorization** tasks involves grouping nominal concepts into natural categories. We evaluate on AP (Almuhareb, 2006), Battig (Baroni and Lenci, 2010) and ESSLI (Baroni et al., 2008) datasets. Cluster purity is used as the evaluation metric.

Extrinsic Tasks. The CONLL-2000 shared task (Sang and Buchholz, 2000) is used for **chunking** tasks and F1-score is used as the evaluation metric; OntoNotes 4.0 (Weischedel et al., 2011) is used for **NER** tasks and F1-score is used as the evaluation metric; And the WSJ portion of Penn Treebank (Marcus et al., 1993) is used for **POS tagging** tasks, and token-level accuracy is used as the evaluation metric. These tasks are reimplemented with the open tool NCRF++ (Yang and Zhang, 2018).

4.2 Baselines

As shown in Table 1, baselines are classified into three categories. For the first category (*Static*), static embeddings come from a lookup table. Note here that *Skip-gram(context)* denotes the results from the context word embeddings. For the second category (*Contextualized*), static embeddings come from contextualized word embedding models (i.e., BERT, ELMo, GPT2, and XLNet) for lexical semantics tasks. The models with *_token* use the mean pooled subword token embeddings as static embeddings; The models with *_word* take every single word as a sentence and output its word representation as a static embedding; The models with *_avg* take the average of output over training corpus. For the last category (*Context-to-Vec*), contextualized information is integrated into Skip-gram embeddings. Among these models, ContextLSTM (Melamud et al., 2016) learns the context embeddings by using single-layer bi-LSTM; SynGCN (Vashishth et al., 2018) uses GCN to calculate context word embeddings based on syntax structures; BERT+Skip-gram (Wang et al., 2021) enhances the Skip-gram model by adding context syntactic information from BERT, which is our primary baseline.

Models	AP	Batting	ESSLI(N)	ESSLI(V)	Avg
Skip-gram	63.4	42.8	75.0	62.2	60.8
Skip-gram(context)	57.4	41.6	72.5	66.6	59.5
CBOW	63.2	43.3	75.0	64.4	61.4
Glove	58.0	41.3	72.5	60.0	58.0
FASTTEXT	63.4	44.4	75.0	62.2	61.2
Deps	61.8	41.7	77.5	68.8	62.4
BERT$_{avg}$	55.7	34.7	70.0	64.0	56.1
SynGCN	63.4	42.8	82.5	62.2	62.7
BERT+Skip-gram	64.1	43.8	77.5	66.6	63.0
Ours(preliminary)	65.7	44.0	85.0	70.4	66.3
Ours(+post-process)	**66.4**	**44.2**	**87.5**	**74.1**	**68.1**

Table 2: Results on word concept categorization tasks. The best results are bolded, and the second-best underlined.

Models	CHUNK	NER	POS	Avg
Skip-gram	88.07	83.90	95.12	89.03
GloVe	89.87	89.13	96.52	91.84
BERT$_{avg}$	90.96	84.51	96.80	90.76
SynGCN	91.23	88.75	96.71	82.23
BERT+Skip-gram	91.06	88.98	96.86	92.30
Ours	**91.98**	**89.52**	**96.91**	**92.80**

Table 3: Results on extrinsic tasks. The best results are bolded, and the second-best underlined.

4.3 Quantitative Comparison

Word Similarity and Analogy. Table 1 shows the experimental results of intrinsic tasks. Overall, the models that integrate contextualized information into static embeddings (*Context-to-Vec*) perform better than other types (*Contextualized / Satic*). Our results outperform baselines across the board. To be fair, the backbone of our model here is BERT as that in the main baseline (*BERT+Skip-gram*) (Wang et al., 2021).

Within the *Context-to-Vec* category, our models perform best on all word similarity datasets. Our base model without post-processing obtains an average absolute improvement of about +23.8%(+13.2) and related improvement of +4.4%(+2.9) compared with the main baseline. The performance is further enhanced using post-processing with a +25.6%(+14.2) absolute increase, and a +5.8%(+3.8) relative increase compared with the main baseline, and a +1.4%(+1.0) relative increase compared with our base model (w/o post-processing). It is worth mentioning that the main baseline does not perform better than BERT$_{avg}$ in *Contextualized* group on the RG65 dataset, but our model does make up for their regrets, which indicates that our model is better at understanding contextual correlates of synonymy.

For the word analogy task, our performances are basically equal to the baselines. Overall, we gain the best score (+0.5) on the Google dataset but without a significant improvement. Although we do not gain the best score across all baselines on the SemEval dataset, our model performs better than the main baseline.

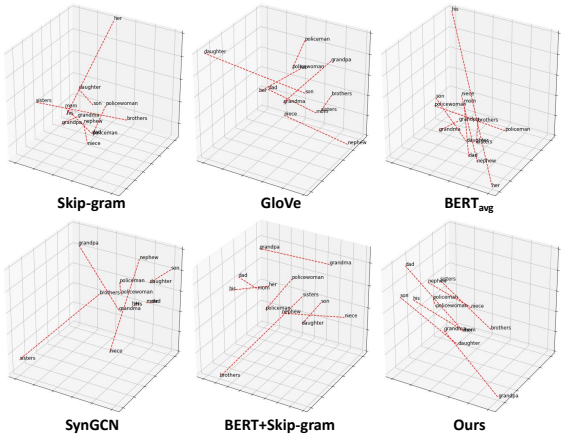

Figure 4: Visualization on word pairs of gender relationship.

For different datasets, especially in word similarity tasks, the improvement of our preliminary model on WS353, SimiLex, RG65 (+4.1, +5.5, and +5.7, respectively) is significantly better than other datasets. For example, the improvement of the main baseline on the WS353R (relatedness) subset and the WS353 set is far greater than that on the WS353S (similarity) subset. While our model bridges their gaps in the WS353 set and also ensures that the performance of WS353S and WS353R is further improved slightly.

Word Concept Categorization. Word concept categorization is another important intrinsic evaluation metric. We use 4 commonly used datasets as shown in Table 2. Overall, our model without post-processing outperforms the baselines by a large margin, giving the best performance and obtaining an average performance gain of +5.2%(+5.1) compared to the main baseline. In particular, the largest increases are observed on the ESSLI(N) (+7.5), ESSLI(V) (+3.8). And with post-processing, our model can obtain better improvements (+3.3 vs. +5.1). The experimental results show the advantage of integrating contextualized and word co-occurrence information, which can excel in grouping nominal concepts into natural categories.

Extrinsic Tasks. Extrinsic tasks reflect the effectiveness of embedded information through downstream tasks. We conduct extrinsic evaluation from chunking, NER, and POS tagging tasks as shown in

Methods	WS353	WS353S	WS353R	SimiLex	RW	MEN	RG65	Avg
w/o retrofitting	76.9	76.7	68.3	54.9	43.5	76.8	84.3	68.8
+Faruqui et al. (2014)	77.2	76.1	69.8	55.0	43.8	76.2	83.5	68.8
+Artetxe et al. (2018)	78.3	75.3	70.0	49.4	42.7	77.4	84.6	68.2
+Ours	**78.9**	**77.0**	**70.4**	**55.2**	**44.0**	**77.9**	**85.1**	**69.8**

Table 4: Comparison on post-processing schemes.

Models	Nearest neighbors of *light*	Nearest neighbors of *while*
Skip-gram	uv, bioluminescence, fluorescent, glare, sunlight, illumination	whilst, recuperating, pursuing, preparing, attenmpting, fending
CBOW	stevenson, initimadation, earle, yellowing, row, kizer	whilst, when, still, although, and, but
GloVe	excluding, justify, orestes, generation, energy, frieze	both, taking, ',' , up, but, after
FASTTEXT	sculpts, baha'i, kinghorn, lick, inputs, minimize	whilst, still, and, meanwhile, instead, though
SynGCN	search, prostejov, preceding, forearms, freewheel, naxos	whilst, time, when, years, months, tenures
BERT+SkipGram	lights, dark, lighter, illumination, glow, illuminating	whilst, whereas, although, conversely, though, meanwhile
Ours	lumière, lumiere, licht, illumination, luminous, lights	whilst, whereas, although, though, despite, albeit

Table 5: Nearest neighbors of words "*light*" and "*while*".

Table 3. We select comparison representatives from the *Static* group, the *Contextualized* group, and the *Context-to-Vec* group, respectively. Although the improvement is not significant compared with the intrinsic evaluations, it can be seen that our performances are better than the baselines, which can prove the superiority of our model. The primary baseline *BERT+Skip-gram* obtains the second-best average score, but does not excel in the chunking task. In contrast, our model not only outperforms all baselines moderately on average, but also performs best in every individual task.

4.4 Ablation and Analysis

Post-processing Schemes. From Table 1, we can initially find that the post-processing method has a positive impact. To further quantitatively analyze, we compare more related methods as shown in Table 4. In this ablation experiment, the comparison baseline is our trained original word vectors (w/o retrofitting), and the other comparison methods include the singularity decomposition-based method (Artetxe et al., 2018), and the synonym-based constraint method (Faruqui et al., 2014). From the results, we can see that other post-processing schemes can improve the word vectors to some extent, but do not perform better in all datasets. However, our proposed post-processing scheme performs the best across the board here, which shows that converting the distance distribution into a probability distribution is more effective.

Nearest Neighbors. To further understand the results, we show the nearest neighbors of the words "*light*" and "*while*" based on the cosine similarity, as shown in Table 5. For the noun "*light*", other methods generate more noisy and irrelevant words, especially static embeddings. In contrast, the *Context-to-Vec* approaches (Ours & BERT+Skip-gram) can capture the key meaning and generate cleaner results, which are semantically directly related to "*light*" literally. For the word "*while*", the static approaches tend to co-occur with the word "*while*", while *Context-to-Vec* approaches return conjunctions with more similar meaning to "*while*", such as "*whilst*", "*whereas*" and "*although*", which demonstrates the advantage of using contextualization to resolve lexical ambiguity.

Word Pairs Visualization. Fig.4 shows the 3D visualization of the gender-related word pairs based on t-SNE (Van der Maaten and Hinton, 2008). These word pairs differ only by gender, e.g., "*nephew* vs. *niece*" and "*policeman* vs. *policewoman*". From the topology of the visualized vectors, the spatial connectivity of the word pairs in Skip-gram and GloVe is rather inconsistent, which means that static word vectors are less capable of capturing gender analogies. In contrast, for vectors based on contextualized embeddings, such as BERT_{avg}, SynGCN, BERT+Skip-gram, and our model, the outputs are more consistent. In particular, our outputs are highly consistent in these instances, which illustrates the ability of our model to capture relational analogies better than baselines and the importance of contextualized information based on semantic knowledge.

5 Conclusion

We considered improving word embeddings by integrating more contextual information from existing pre-trained models into the Skip-gram framework. In addition, based on inherent properties of static embeddings, we proposed a graph-based post-retrofitting method by employing priori synonym knowledge and a weighted distribution probability.

The experimental results show the superiority of our proposed methods, which gives the best results on a range of intrinsic and extrinsic tasks compared to baselines. In future work, we will consider prior knowledge directly during training to avoid a multi-stage process.

Acknowledgements

This work is supported in part by the Science and Technology Innovation 2030 - Major Project (No. 2021ZD0150100) and National Natural Science Foundation of China (No. U21A20427). We thank all the anonymous reviewers for their helpful comments and suggestions.

References

Abdulrahman Almuhareb. 2006. *Attributes in lexical acquisition*. Ph.D. thesis, University of Essex.

Simran Arora, Avner May, Jian Zhang, and Christopher Ré. 2020. Contextual embeddings: When are they worth it? *arXiv preprint arXiv:2005.09117*.

Mikel Artetxe, Gorka Labaka, Inigo Lopez-Gazpio, and Eneko Agirre. 2018. Uncovering divergent linguistic information in word embeddings with lessons for intrinsic and extrinsic evaluation. *arXiv preprint arXiv:1809.02094*.

Marco Baroni, Stefan Evert, and Alessandro Lenci. 2008. Lexical semantics: bridging the gap between semantic theory and computational simulation. In *Proceedings of the ESSLLI Workshop on Distributional Lexical Semantics*. Citeseer.

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

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

Rishi Bommasani, Kelly Davis, and Claire Cardie. 2020. Interpreting pretrained contextualized representations via reductions to static embeddings. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4758–4781.

Elia Bruni, Gemma Boleda, Marco Baroni, and Nam-Khanh Tran. 2012. Distributional semantics in technicolor. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 136–145.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

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

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *Proceedings of the 10th international conference on World Wide Web*, pages 406–414.

David Jurgens, Saif Mohammad, Peter Turney, and Keith Holyoak. 2012. Semeval-2012 task 2: Measuring degrees of relational similarity. In ** SEM 2012: The First Joint Conference on Lexical and Computational Semantics–Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 356–364.

Douwe Kiela, Felix Hill, and Stephen Clark. 2015. Specializing word embeddings for similarity or relatedness. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2044–2048.

Stan Z Li. 1994. Markov random field models in computer vision. In *European conference on computer vision*, pages 361–370. Springer.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Minh-Thang Luong, Richard Socher, and Christopher D Manning. 2013. Better word representations with recursive neural networks for morphology. In *Proceedings of the seventeenth conference on computational natural language learning*, pages 104–113.

Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: The penn treebank.

Leland McInnes, John Healy, and James Melville. 2018. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*.

Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. context2vec: Learning generic context embedding with bidirectional lstm. In *Proceedings of the 20th SIGNLL conference on computational natural language learning*, pages 51–61.

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

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

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Herbert Rubenstein and John B Goodenough. 1965. Contextual correlates of synonymy. *Communications of the ACM*, 8(10):627–633.

Erik F Sang and Sabine Buchholz. 2000. Introduction to the conll-2000 shared task: Chunking. *arXiv preprint cs/0009008*.

Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(11).

Shikhar Vashishth, Manik Bhandari, Prateek Yadav, Piyush Rai, Chiranjib Bhattacharyya, and Partha Talukdar. 2018. Incorporating syntactic and semantic information in word embeddings using graph convolutional networks. *arXiv preprint arXiv:1809.04283*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Yile Wang, Leyang Cui, and Yue Zhang. 2021. Improving skip-gram embeddings using bert. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:1318–1328.

Ralph Weischedel, Sameer Pradhan, Lance Ramshaw, Martha Palmer, Nianwen Xue, Mitchell Marcus, Ann Taylor, Craig Greenberg, Eduard Hovy, Robert Belvin, et al. 2011. Ontonotes release 4.0. *LDC2011T03, Philadelphia, Penn.: Linguistic Data Consortium*.

Chong Wu, Zhenan Feng, Jiangbin Zheng, Houwang Zhang, Jiawang Cao, and Hong Yan. 2020. Star topology convolution for graph representation learning.

Lirong Wu, Haitao Lin, Cheng Tan, Zhangyang Gao, and Stan Z Li. 2021. Self-supervised learning on graphs: Contrastive, generative, or predictive. *IEEE Transactions on Knowledge and Data Engineering*.

Jun Xia, Yanqiao Zhu, Yuanqi Du, and Stan Z Li. 2022. A survey of pretraining on graphs: Taxonomy, methods, and applications. *arXiv preprint arXiv:2202.07893*.

Jie Yang and Yue Zhang. 2018. Ncrf++: An open-source neural sequence labeling toolkit. *arXiv preprint arXiv:1806.05626*.