Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI-20)
Survey Track

Incorporating Extra Knowledge to Enhance Word Embedding

Arpita Roy and **Shimei Pan**
University of Maryland, Baltimore County (UMBC)
{arpita2, shimei}@umbc.edu

Abstract

Word embedding, a process to automatically learn the mathematical representations of words from unlabeled text corpora, has gained a lot of attention recently. Since words are the basic units of a natural language, the more precisely we can represent the morphological, syntactic and semantic properties of words, the better we can support downstream Natural Language Processing (NLP) tasks. Since traditional word embeddings are mainly designed to capture the semantic relatedness between co-occurred words in a predefined context, it may not be effective in encoding other information that is important for different NLP applications. In this survey, we summarize the recent advances in incorporating extra knowledge to enhance word embedding. We will also identify the limitations of existing work as well as point out a few promising future directions.

1 Introduction

Word embedding, a process to automatically transform the words in a vocabulary into dense vectors of real numbers in a continuous embedding space, has gained much attention and popularity as the derived word vectors often encode important syntactic and semantic information that is useful for Natural Language Processing (NLP).

Word embedding was shown to boost the performance of many downstream NLP tasks such as named entity recognition [Fang *et al.*, 2016], relation extraction [Wang *et al.*, 2014], sentiment analysis [Tang *et al.*, 2014], text classification [Liu *et al.*, 2018] and question answering [Cheng *et al.*, 2015]

Since traditional word embeddings mainly capture the semantic relatedness between co-occurred words in a predefined context, they suffer from certain limitations. For example, although it is relatively easy to infer semantic relatedness between words based on word embeddings (e.g., to infer that kitten and cat are related is easy), it is often much harder to derive specific relationship types based on word embedding (e.g., to know that a kitten is a juvenile cat is difficult). Thus, the semantic relations explicitly encoded in a knowledge base can be useful to enrich and enhance word embeddings. Moreover, the same word may have multiple senses. Traditional word embedding is unable to discriminate among the different meanings of a word. Moreover, due to data sparsity, traditional word embedding also has difficulty producing reliable embeddings for rare words.

To address these issues, a large body of recent research has emerged to incorporate extra morphological, syntactic, semantic and domain knowledge to enhance word embedding. This technique can be used to (1) improve the quality of the word embedding learned from text; (2) add domain knowledge to facilitate domain-specific NLP; (3) overcome data sparsity when the text in the target domain is small or the target words are rare. In this survey, we summarize the recent advances in this research area. We include representative work on enhancing both static word embedding (e.g., Word2Vec) where a fixed embedding vector is learned for each word and dynamic/contextual word embedding (e.g., BERT), where the embedding vector for each word varies with its context.

2 Overview

Given the space limit, we define the scope of this survey quite narrowly to include only methods on incorporating knowledge into neural network based word embedding, as they are frequently the most widely used methods with the state-of-the-art performance. We exclude models that learn embeddings for larger text units (e.g., learning embeddings for phrases or documents). In addition, we only focus on English and exclude cross-lingual word embedding models. All the papers in the survey are published in the last ten years at the top NLP and AI venues. Table 1 is an overview of these papers. In the following, we summarize each paper based on the source and the type of knowledge that is added to word embedding, the downstream applications that employ these embeddings. We also explain and categorize the typical methods for knowledge injection into word embedding in Section 5.

3 Knowledge Sources

In this section, we describe various knowledge resources used in these papers to enrich word embeddings.

Wikipedia. This is the largest multilingual encyclopedia consisting of numerous articles. There are 6,017,385 articles

Paper	Knowledge Source	Knowledge Type	Application
[Luong *et al.*, 2013]	Morfessor	Morphological	N.A.
[Xu *et al.*, 2014]	Freebase, WordRep	Categorical, Relational	Topic Prediction
[Bian *et al.*, 2014]	Morfessor, Longman Dictionaris, WordNet, Freebase	Morphological, Categorical, Relational	Word Completion
[Wang *et al.*, 2014]	Freebase	Knowledge graph	Triplet Classification, Relation Extraction
[Tang *et al.*, 2014]	Annotation	Categorical	Sentiment Classification
[Liu *et al.*, 2015]	Wordnet	Relational	Sentence Completion, NER, Synonym Selection
[Cheng *et al.*, 2015]	Probase, LDA	Categorical	Paraphrase Detection, Q&A Relatedness Classification
[Faruqui *et al.*, 2015]	PPDB, Wordnet, Framenet	Relational	Syntactic Relations, Synonym Selection, Sentiment Analysis
[Zhou *et al.*, 2015]	Metadata	Categorical	Question retrieval
[Jauhar *et al.*, 2015]	Wordnet	Relational	Synonym Selection, Contextual similarity
[Fang *et al.*, 2016]	Freebase	Knowledge graph	Entity Disambiguation
[Tissier *et al.*, 2017]	dictionary.com, Oxford, Cambridge, Collins	Textual	Text classification
[Ling *et al.*, 2017]	UMLS	Relational	Biomedical IR
[Cao *et al.*, 2017]	Wikipedia	Knowledge Graph	Entity linking
[Mancini *et al.*, 2017]	BabelNet, WordNet	Categorical	Sense Clustering, Word and Sense Interconnectivity
[Zeng *et al.*, 2017]	Social graph	Relational	Sentiment Classification
[Liu *et al.*, 2018]	Wordnet	Relational	Text Classification, Query Expansion
[Jiang *et al.*, 2018]	AoA rating, Proficiency test etc.	Categorical	Readability Assessment
[Glavaš and Vulić, 2018]	WordNet, Roget's Thesaurus	Relational	Language Transfer,Text Simplification, Dialog State Tracking
[Roy *et al.*, 2019]	MAEC specification and Human annotation	Textual	Malware Attribute Prediction
[Peters *et al.*, 2019]	Wordnet, Wikipedia	Textual, Knowledge graph	Relation extraction, Words in Context, Entity typing
[Zhang *et al.*, 2019]	Wikidata	Knowledge graph	Entity Typing, Relation Classification,

Table 1: Summary of knowledge enhanced word embedding

written in English. Wikipedia articles are all linked or cross-referenced. Both the text descriptions in Wikipedia and the graph created from interlinked Wikipedia pages are used as extra knowledge to enhance word embedding.

WordNet[1] and BabelNet[2]. Wordnet is a lexical database encoding semantic relations between words. Words are grouped into sets of cognitive synonyms called synsets, each expressing a distinct concept. Synsets are interlinked via semantic relations such as hyperonymy, hyponymy, meronymy, troponyms and antonymy. For instance, the synset [plant, flora, plant life] is linked to the synset [organism] via a hypernym relation. It is also linked to the synset [phytoplankton] via a hyponym relation. WordNet version 2.1 contains 155327 words and 117597 synsets. BabelNet is created by integrating WordNet with Wikipedia and other resources and maintaining the structure of WordNet.

Freebase and Probase. Freebase is a large collection of structured data. It comprises important semantic knowledge, especially the information about entities and relations (e.g., entity categories and belonging-to or is-a relations). For example, (France, capital, Paris) represents a relation in Freebase where "capital" denotes the relation between the entity "France" and the entity "Paris". As of January 2014, Freebase contained around over 40 million entities and 2 billion relations. Similarly, Probase [Wu *et al.*, 2012] is a probabilistic taxonomy that contains concept-instance pairs connected in a hierarchical structure. For example, (vegetable, potato) is concept-instance pair and (food $\rightarrow$ vegetable) represents a concept hierarchy. It contains 2.7 million concepts.

Domain Dictionary. Domain dictionaries such as UMLS (Unified Medical Language System) [Bodenreider, 2004] and MAEC (Malware Attribute Enumeration and Characterization) contain useful domain knowledge that may be missing from the general word embedding. UMLS is the largest biomedical dictionary, containing more than 100 biomedical vocabularies, 13 million concepts and thousands of relations. For example, the concept of fever with the UMLS concept identifier "C0015967", is represented in nearly one hundred component vocabularies. Disease-treatment, disease-finding, disease-prevention are some of the relationship types in UMLS. MAEC has specified a set of standard malware action attributes for cybersecurity [Kirillov *et al.*, 2011].

Knowledge from Other NLP Tools. Morfessor is a tool that automatically splits a word into roots, prefixes, and suffixes (e.g. recycle=re(prefix)+cycle(root), kindness=kind(root)+ness(suffix)). Latent Dirichlet allocation (LDA) is a generative model that is used to learn a set of topics (latent themes) from unlabeled documents. A word sense disambiguation tool can also be used to annotate each word token with a specific sense tag.

Other Knowledge Sources. PPDB containing paraphrase pairs of English, general English dictionaries such as Oxford, Cambridge and Collins dictionaries, human annotations about the properties of words or relations between words as well as social graphs are also used in the surveyed papers.

[1]https://wordnet.princeton.edu/

[2]https://babelnet.org/

4 Knowledge Type

Since the knowledge injecting methods are shaped by the type of knowledge that is added, in this section, we categorize the basic knowledge elements that can be incorporated into word embedding.

Textual Knowledge. This type of knowledge can be found in Wikipedia as well as the definitions of words in a general or domain-specific dictionary. Textual descriptions often contain words that are highly relevant to the target word. One frequently used method to incorporate textual knowledge in word embedding is to first identify relevant and informative keywords from the text[Roy *et al.*, 2019; Tissier *et al.*, 2017] and then adapt the word embedding objective so that the embeddings of the target word will be close to the embeddings of the identified keywords.

Morphological Knowledge. Information about the basic elements of a word such as syllables, roots, or affix (prefix and suffix) is found in morphological knowledge. Leveraging morphological knowledge can facilitate the identification of semantically related words (e.g. words with the same root). As a result, it allows the system to generalize and capture the meanings of rare, out of vocabulary (OOV) and morphologically complex words accurately. To incorporate morphological knowledge, each morpheme is considered as the basic unit of representation and a vector representation is first learned for each unique morpheme. Then word embeddings are learned by composing the embeddings of all the morphemes within a word.

Categorical Knowledge. Semantic and syntactic properties of words are encoded in categorical knowledge. The semantic category of a word may include its concept, semantic type or semantic role while the syntactic category may include its part of speech tag and syntactic role. To integrate categorical knowledge in word embedding, we can either add an objective to minimize the distance between the word vectors that belong to the same category [Xu *et al.*, 2014; Zhou *et al.*, 2015] or increase the probability of predicting the target category of a given word [Roy *et al.*, 2019; Cheng *et al.*, 2015].

Relational Knowledge. This type of knowledge encodes the relationships between words. It is often represented as a triplet: (head H, relation R, tail T). Relational knowledge is injected into embedding by adding a constraint that requires the representation vector of T should be close to the representation vector of H plus the representation vector of R [Xu *et al.*, 2014]. It is also possible that the vector representation of R is not explicitly learned during embedding. In this case, the relation constraints are enforced by incorporating additional training objectives related to R. For example, if R is a synonym relation, the system can add an objective to force the distance between the H vector and the T vector to be small. In contrast, if R is an antonym relation, the system can add a training objective to force the embedding of H and T to stay far apart [Bian *et al.*, 2014; Faruqui *et al.*, 2015; Glavaš and Vulić, 2018; Liu *et al.*, 2015].

Knowledge Graph. Knowledge graph provides a rich source of high quality, human-curated structured knowledge

to enhance word embedding. Knowledge graphs can be leveraged in two ways: (1) joint training with both the word embedding objective (the embeddings of neighboring words are similar) and the graph embedding objective (the embeddings of neighboring nodes are similar) [Cao *et al.*, 2017; Fang *et al.*, 2016; Wang *et al.*, 2014]; or (2) using a pre-trained knowledge graph embedding. Pre-trained knowledge graph embedding can be encoded into word embedding either by changing the objective function of word embedding [Zhang *et al.*, 2019] or via word-to-entity attention mechanism [Peters *et al.*, 2019].

5 Methodology

We can categorize the main methods employed in these papers along four dimensions: 1) static or dynamic embedding, (2) joint optimization or post-processing, 3) solo or coupled embedding and 4) single or multi-sense word embedding

5.1 Static or Dynamic Embedding

Word embedding techniques can be categorized into two classes: (1) Static word embedding and (2) Dynamic/contextual word embedding. Static word embedding only learns a single context-independent representation for each word. This implies that the meaning of a word is the same in the entire text corpus, regardless of its context. The most commonly used static embedding method is Word2Vec [Mikolov *et al.*, 2013]. It employs a feed-forward shallow neural network trained with unlabeled text corpora. Two word2vec models have been proposed: CBOW and skip-gram. In the CBOW model, context words are used to predict a target word and in the Skipgram model, context words are predicted based on the target word. Many of the static embedding methods in the survey aimed at improving Word2Vec with extra Knowledge.

On the other hand, dynamic/contextual word embedding generates a word representation that is a function of its context (e.g., the entire sentence). This allows dynamic embeddings to capture more context-dependent aspects of a word. The most commonly used dynamic embedding methods are ELMo (Embeddings from Language Models) [Peters *et al.*, 2018] and BERT(Bidirectional Encoder Representations from Transformers) [Devlin *et al.*, 2019]. For example, BERT is designed to pre-train deep bidirectional representations from a massive amount of unlabeled text by jointly conditioning on both the left and right context in all layers. It uses a transformer and an attention mechanism to learn contextual word embedding. Many of the dynamic embedding methods in the survey aimed at improving BERT with extra knowledge.

There are several differences between static and dynamic embedding: (1) static embedding is frequently trained using shallow neural networks and dynamic embedding is trained using deep neural networks; (2) the output of a static model is the embedding vectors, not the neural network models themselves. In contrast, the output of dynamic word embedding is the trained neural network models themselves as well as the word vectors dynamically generated by the models based on the context; (3) dynamic embeddings are more computationally expensive to train than static embeddings.

So far, the vast majority of the papers in our survey focuses on improving static word embedding (20 out of 22). Recently the field has shifted toward learning context-sensitive embedding. Some initial efforts have emerged to inject extra knowledge to dynamic word embedding. [Peters *et al.*, 2019; Zhang *et al.*, 2019].

5.2 Joint Optimization or Post Processing

In joint optimization, word embedding is trained with both text and extra knowledge simultaneously. These methods frequently modify the training objective to include both the native word embedding objective and a new objective related to the knowledge. The methods can be further categorized into: (1) adding prior or regularization to the original distributed representation learning objective [Xu *et al.*, 2014; Bian *et al.*, 2014]; (2) extending the original distributed representation learning objective to learn additional embedding [Cheng *et al.*, 2015; Mancini *et al.*, 2017; Roy *et al.*, 2019]; (3) adding rank or hierarchical structure constraints, [Liu *et al.*, 2015; Liu *et al.*, 2018]; (4) combining training objective with knowledge graph embedding objective [Fang *et al.*, 2016; Wang *et al.*, 2014]; and (5) augmenting the input text with extra knowledge [Jiang *et al.*, 2018; Tissier *et al.*, 2017]. These models are tied to the distributional objective and any change of the underlying distributional model induces a change in the entire joint model.

Post-processing-based methods integrate extra knowledge into pre-trained word embedding. Popular post-processing methods such as retrofitting [Faruqui *et al.*, 2015] fine tune the original word embeddings so that they satisfy additional constraints generated from the extra knowledge. Recently, [Peters *et al.*, 2019] proposed a general method to inject multiple knowledge bases into pre-trained models.

The biggest benefit of post-processing over joint optimization is that it can be applied to any pre-trained word embedding models without expensive retraining. Post-processing methods often only locally update word vectors involved in the external constraints, whereas vectors of the other words remain intact. In contrast, joint optimization propagates the influence of the external knowledge to all the words via the joint objective.

5.3 Solo or Coupled Embedding

Solo or coupled embedding specifies whether a system learns word embedding only or word embedding is trained simultaneously with the training of one or more additional embeddings (e.g., sense embedding or semantic type embedding). The majority of the research in our survey only trains word embedding (13 out of 22 use solo embedding).

Coupled embedding can be achieved by augmenting the words in the input text with extra knowledge (e.g., augment a word with its sense or concept annotation). The model learns the vectors for them in the same embedding space [Mancini *et al.*, 2017; Cheng *et al.*, 2015; Roy *et al.*, 2019]. Coupled embedding is also frequently used to combine knowledge graph with word embedding. For example, [Wang *et al.*, 2014] presented one of the initial attempts to encode words and entities in a knowledge graph into a unified vector space. [Fang *et al.*, 2016; Cao *et*

Paper	Static or Dynamic	Joint Optimization or Post Processing	Solo or Coupled Embedding	Single or Multi Senese Embedding
[Luong *et al.*, 2013]	Static	Post Processing & Joint Optimization	Solo	Single
[Xu *et al.*, 2014]	Static	Joint Optimization	Solo	Single
[Bian *et al.*, 2014]	Static	Joint Optimization	Solo	Single
[Wang *et al.*, 2014]	Static	Joint Optimization	Coupled	Single
[Tang *et al.*, 2014]	Static	Joint Optimization	Solo	Single
[Liu *et al.*, 2015]	Static	Joint Optimization	Solo	Single
[Cheng *et al.*, 2015]	Static	Joint Optimization	Coupled	Single
[Faruqui *et al.*, 2015]	Static	Post Processing	Solo	Single
[Zhou *et al.*, 2015]	Static	Joint Optimization	Solo	Single
[Jauhar *et al.*, 2015]	Static	Joint Optimization	Solo	Multi
[Fang *et al.*, 2016]	Static	Joint Optimization	Coupled	Single
[Tissier *et al.*, 2017]	Static	Joint Optimization	Solo	Single
[Ling *et al.*, 2017]	Static	Joint Optimization	Solo	Single
[Cao *et al.*, 2017]	Static	Joint Optimization	Coupled	Single
[Mancini *et al.*, 2017]	Static	Joint Optimization	Solo	Multi
[Zeng *et al.*, 2017]	Static	Joint Optimization	Coupled	Single
[Liu *et al.*, 2018]	Static	Joint Optimization	Solo	Single
[Jiang *et al.*, 2018]	Static	Joint Optimization	Solo	Single
[Glavaš and Vulić, 2018]	Static	Post Processing	Solo	Single
[Roy *et al.*, 2019]	Static	Joint Optimization	Coupled	Single
[Peters *et al.*, 2019]	Dynamic	Post Processing	Coupled	Multi
[Zhang *et al.*, 2019]	Dynamic	Joint Optimization	Coupled	Multi

Table 2: Different methodologies used for knowledge integration

al., 2017] focused on solving the ambiguity of entity mentions that arises from directly integrating entity and knowledge in the same space. Alternatively, [Peters *et al.*, 2019; Zhang *et al.*, 2019] focused on utilizing pre-trained entity embedding to improve the quality of word embedding.

5.4 Single or Multi-Sense Word Embedding

Single sense embedding ignores word polysemy and conflates all the meanings or senses of a word into a single representation. This deficiency hampers the effectiveness of word embedding and hurts the performance of downstream tasks.

To alleviate this problem, [Jauhar *et al.*, 2015] uses external knowledge from a sense inventory to generate static sense-specific word embeddings. [Mancini *et al.*, 2017] learns both word and sense embedding in a unified vector space. Recently, given a word, dynamic/contextual embedding produces different embedding vectors for the same word in different contexts. The main difference between these two approaches is that (1) dynamic embedding can learn word polysemy from text on its own while static multi-sense embedding requires external knowledge to identify the multiple senses of a word (e.g., with the help of a word sense disambiguation tool); (2) static multi-sense embedding assigns a fixed number of predefined sense embeddings to each word while in dynamic embedding, the number of vectors that can be associated with a word is, in principle infinite. Empirically, multi-sense embedding has significantly better performance over single sense embedding [Jauhar *et al.*, 2015; Mancini *et al.*, 2017].

5.5 Methodology Discussion

Dynamic word embedding techniques like BERT outperform static embedding techniques on many NLP tasks. However, due to the high computational cost associated with jointly optimizing the objective of dynamic word embedding and that related to the new knowledge, it would be very difficult for people who do not have access to powerful computational resources to apply the joint optimization approach. Since joint

optimization is a more principled way to add new knowledge than post-processing, we may have missed opportunities to explore better but more expensive joint optimization-based approaches to enhance dynamic word embedding. In contrast, since the required resources needed to train a static embedding is much lower, there has been a wide range of methods proposed to enhance static word embeddings.

6 Evaluation Methods

In this section, we present the typical evaluation methods for assessing the quality of word embedding. We categorize them into intrinsic or extrinsic evaluation.

Intrinsic Evaluation. This is a generic evaluation of the quality and coherence of a vector space, independently from their performance in downstream applications. Different properties can be intrinsically tested, with semantic similarity being traditionally viewed as the most straightforward measure of the quality of word embedding. Cosine similarity is the most widely used metric for computing the similarity between word embeddings. Based on this metric, semantically similar words would have higher cosine similarity. The evaluation datasets consist of word pairs with similarity and relatedness scores assigned by humans. Spearman's test is used to compute the correlations between the similarity scores computed by the models and those assigned by humans. The majority of the papers we surveyed evaluated their models using the word similarity task. The second most common intrinsic evaluation task is analogical reasoning. Given two pairs of words with the same relation (e.g. man:woman :: king:queen), a model is considered to have answered the analogy question correctly if the nearest representation to $Embd(\text{“}man\text{”}) - Embd(\text{“}woman\text{”}) + Embd(\text{“}king\text{”})$ is $Embd(\text{“}queen\text{”})$. Among the papers we surveyed, [Xu *et al.*, 2014; Bian *et al.*, 2014; Wang *et al.*, 2014; Cao *et al.*, 2017] reported model performance using this task.

Extrinsic Evaluation. This type of evaluation aims at assessing the quality of word embedding with downstream tasks. In addition to intrinsic evaluation, extrinsic evaluation is necessary to understand the effectiveness of different embedding techniques in real-world applications. Some of the most common NLP tasks used for extrinsic evaluation include synonym selection [Liu *et al.*, 2015; Jauhar *et al.*, 2015; Ling *et al.*, 2017], name entity recognition [Liu *et al.*, 2015], named entity disambiguation [Fang *et al.*, 2016; Cao *et al.*, 2017], entity typing [Peters *et al.*, 2019; Zhang *et al.*, 2019] relation extraction [Peters *et al.*, 2019; Zhang *et al.*, 2019; Wang *et al.*, 2014], text classification [Liu *et al.*, 2018; Tissier *et al.*, 2017], sentiment analysis [Faruqui *et al.*, 2015; Tang *et al.*, 2014], question retrieval [Zhou *et al.*, 2015] and question answering [Cheng *et al.*, 2015].

7 Limitations and Future Directions

Based on this survey, we have identified some limitations in the current research. We also point out a few future research directions aiming at addressing these issues.

Currently, there is a limited understanding of the impact of adding new knowledge to word embedding. For example, word embeddings trained to capture distributional semantic are well known to exhibit seemingly linear behavior (e.g. the embeddings of analogy “woman is to queen as man is to king” and “work is to works as teach is to teaches”). The compositionality of word embedding is also an important topic. When extra knowledge is added to word embeddings using various knowledge injection methods, it is unclear whether the process might disturb the original embedding space and hurt some of the desired properties. Important questions such as “whether the linear behavior between semantically and syntactically related words are preserved in the new embedding space after knowledge injection?” and “how injecting new knowledge may impact the compositionality of word embedding?” are left unanswered. To address this, we may want to focus more on theory-guided knowledge integration methods so that we have a clear understanding of the impact of these methods on the learned embeddings. This issue can also be solved if we can interpret the meaning of word embeddings. Since the learned embeddings are in a high dimensional latent space that cannot be intuitively understood by humans, novel embedding visualization techniques may shed light on how different types of knowledge and injection methods may impact the embedding space. We may also use simulations to train word embeddings with synthesized text to enhance our understanding of the impact of various methods.

There are also insufficient efforts on incorporating diverse types of Knowledge in word embedding. Much of the work in this survey focuses on adding linguistic knowledge (e.g., morphological, syntactic and semantic knowledge). There has not been many efforts on incorporating other types of knowledge such as First Order Logic (FOL). Since logic is a powerful way to represent human knowledge, it is important that the knowledge expressed as FOL can be systematically incorporated into word embedding.

In addition to logic, there has not been much research on incorporating corpus-level constraints into word embedding. Corpus-level constraints can be used to specify certain requirements (e.g., to ensure that the learned word embeddings are unbiased). Previous research has shown that word embedding trained on human-generated text exhibits human biases [Bolukbasi *et al.*, 2016]. It would be useful if we can add fairness as a corpus-level constraint to regularize the embedding training. Although it is possible to de-bias word embedding using other methods such as vector projection, encoding fairness as corpus-level constraints will give us more flexibility to plug in different fairness metrics.

So far, there is also little attention paid to embedding efficiency. Learning word embedding, especially contextual embedding (e.g., BERT) requires massive amount of data and computational power. Adding extra knowledge (e.g., a large knowledge base) to train a joint embedding only exacerbates this problem. Due to its high data and infrastructure demand, there are few options for people without access to the required resources can do to explore different knowledge injection options. So far, most work on integrating knowledge into word embedding has focused on improving model accuracy. More investigations are needed to balance the trade-off between accuracy and efficiency.

References

[Bian *et al.*, 2014] Jiang Bian, Bin Gao, and Tie-Yan Liu. Knowledge-powered deep learning for word embedding. In *ECML PKDD*. Springer, 2014.

[Bodenreider, 2004] Olivier Bodenreider. The unified medical language system (umls): integrating biomedical terminology. *Nucleic acids research*, 32(suppl_1):D267–D270, 2004.

[Bolukbasi *et al.*, 2016] Tolga Bolukbasi, Kai-Wei Chang, James Y Zou, Venkatesh Saligrama, and Adam T Kalai. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In *NIPS*, 2016.

[Cao *et al.*, 2017] Yixin Cao, Lifu Huang, Heng Ji, Xu Chen, and Juanzi Li. Bridge text and knowledge by learning multi-prototype entity mention embedding. In *ACL*, 2017.

[Cheng *et al.*, 2015] Jianpeng Cheng, Zhongyuan Wang, Ji-Rong Wen, Jun Yan, and Zheng Chen. Contextual text understanding in distributional semantic space. In *CIKM*, 2015.

[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*, 2019.

[Fang *et al.*, 2016] Wei Fang, Jianwen Zhang, Dilin Wang, Zheng Chen, and Ming Li. Entity disambiguation by knowledge and text jointly embedding. In *CoNLL*, 2016.

[Faruqui *et al.*, 2015] Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard Hovy, and Noah A. Smith. Retrofitting word vectors to semantic lexicons. In *ACL*, 2015.

[Glavaš and Vulić, 2018] Goran Glavaš and Ivan Vulić. Explicit retrofitting of distributional word vectors. In *ACL*, 2018.

[Jauhar *et al.*, 2015] Sujay Kumar Jauhar, Chris Dyer, and Eduard Hovy. Ontologically grounded multi-sense representation learning for semantic vector space models. In *NAACL*, 2015.

[Jiang *et al.*, 2018] Zhiwei Jiang, Qing Gu, Yafeng Yin, and Daoxu Chen. Enriching word embeddings with domain knowledge for readability assessment. In *COLING*, 2018.

[Kirillov *et al.*, 2011] Ivan Kirillov, Desiree Beck, Penny Chase, and Robert Martin. Malware attribute enumeration and characterization, 2011.

[Ling *et al.*, 2017] Yuan Ling, Yuan An, Mengwen Liu, Sadid A Hasan, Yetian Fan, and Xiaohua Hu. Integrating extra knowledge into word embedding models for biomedical nlp tasks. In *IJCNN*, 2017.

[Liu *et al.*, 2015] Quan Liu, Hui Jiang, Si Wei, Zhen-Hua Ling, and Yu Hu. Learning semantic word embeddings based on ordinal knowledge constraints. In *ACL-IJCNLP*, 2015.

[Liu *et al.*, 2018] Qian Liu, Heyan Huang, Guangquan Zhang, Yang Gao, Junyu Xuan, and Jie Lu. Semantic structure-based word embedding by incorporating concept convergence and word divergence. In *AAAI*, 2018.

[Luong *et al.*, 2013] Minh-Thang Luong, Richard Socher, and Christopher D Manning. Better word representations with recursive neural networks for morphology. In *CoNLL*, 2013.

[Mancini *et al.*, 2017] Massimiliano Mancini, Jose Camacho-Collados, Ignacio Iacobacci, and Roberto Navigli. Embedding words and senses together via joint knowledge-enhanced training. In *CoNLL*, 2017.

[Mikolov *et al.*, 2013] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. 2013.

[Peters *et al.*, 2018] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *CoRR*, abs/1802.05365, 2018.

[Peters *et al.*, 2019] Matthew E. Peters, Mark Neumann, IV RobertLLogan, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A. Smith. Knowledge enhanced contextual word representations. In *EMNLP-IJCNLP*, 2019.

[Roy *et al.*, 2019] Arpita Roy, Youngja Park, and Shimei Pan. Predicting malware attributes from cybersecurity texts. In *NAACL*, 2019.

[Tang *et al.*, 2014] Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. Learning sentiment-specific word embedding for twitter sentiment classification. In *ACL*, 2014.

[Tissier *et al.*, 2017] Julien Tissier, Christophe Gravier, and Amaury Habrard. Dict2vec : Learning word embeddings using lexical dictionaries. In *EMNLP*, 2017.

[Wang *et al.*, 2014] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph and text jointly embedding. In *EMNLP*, 2014.

[Wu *et al.*, 2012] Wentao Wu, Hongsong Li, Haixun Wang, and Kenny Q Zhu. Probase: A probabilistic taxonomy for text understanding. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, pages 481–492, 2012.

[Xu *et al.*, 2014] Chang Xu, Yalong Bai, Jiang Bian, Bin Gao, Gang Wang, Xiaoguang Liu, and Tie-Yan Liu. Rc-net: A general framework for incorporating knowledge into word representations. In *CIKM*, 2014.

[Zeng *et al.*, 2017] Ziqian Zeng, Yichun Yin, Yangqiu Song, and Ming Zhang. Socialized word embeddings. In *IJCAI*, 2017.

[Zhang *et al.*, 2019] Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. ERNIE: Enhanced language representation with informative entities. In *ACL*, 2019.

[Zhou *et al.*, 2015] Guangyou Zhou, Tingting He, Jun Zhao, and Po Hu. Learning continuous word embedding with metadata for question retrieval in community question answering. In *ACL-IJCNLP*, 2015.