Proceedings of the 12th Conference on Language Resources and Evaluation (LREC 2020), pages 4774–4780
Marseille, 11–16 May 2020

Representation Learning for Unseen Words by Bridging Subwords to Semantic Networks

Yeachan Kim[1], Kang-Min Kim[2], SangKeun Lee[2]
Artificial Intelligence Research Institute (AIRI)[1], Korea University, Republic of Korea[2]
yeachan@airi.kr, {kangmin89, yalphy}@korea.ac.kr

Abstract

Pre-trained word embeddings are widely used in various fields. However, the coverage of pre-trained word embeddings only includes words that appeared in corpora where pre-trained embeddings are learned. It means that the words which do not appear in training corpus are ignored in tasks, and it could lead to the limited performance of neural models. In this paper, we propose a simple yet effective method to represent out-of-vocabulary (OOV) words. Unlike prior works that solely utilize subword information or knowledge, our method makes use of both information to represent OOV words. To this end, we propose two stages of representation learning. In the first stage, we learn subword embeddings from the pre-trained word embeddings by using an additive composition function of subwords. In the second stage, we map the learned subwords into semantic networks (e.g., WordNet). We then re-train the subword embeddings by using lexical entries on semantic lexicons that could include newly observed subwords. This two-stage learning makes the coverage of words broaden to a great extent. The experimental results clearly show that our method provides consistent performance improvements over strong baselines that use subwords or lexical resources separately.
Keywords: Lexicon, Semantics, Knowledge Representation

1. Introduction

Word embeddings have shown to be effective for improving many natural language processing (NLP) tasks, such as machine translation, question answering. Recently, contextualized word embeddings such as BERT (Devlin et al., 2019) have largely improved the performance of NLP tasks compared to static embeddings such as word2vec (Mikolov et al., 2013), GloVe (Pennington et al., 2014). However, static embeddings are still frequently used to various studies for sentence embeddings (Yang et al., 2019; Almarwani et al., 2019) and even the other domains such as the extraction of interaction between drugs in the biomedical field (Sun et al., 2019). In particular, pre-trained word embeddings, which are learned at large corpora, are frequently used and they are publicly available on the web.

Despite of its usefulness and easy-accessibility, there is a major drawback to using pre-trained word embeddings. They are blind to out-of-vocabulary (OOV) words[1] that are not included in the pre-defined vocabulary. This limitation could ignore significant words in tasks and lead to the limited performance of neural models. Unfortunately, this is more evident in real-world scenarios since words in natural language follow a Zipfian distribution whereby some words are frequent but most are rare.

There are two mainstreams to handle such OOV words. The first is to take the pre-trained word embeddings and generalize them to all word entries using subwords (Pinter et al., 2017; Kim et al., 2018; Zhao et al., 2018). These approaches effectively handle surface-variation of words such as inflected forms and typos. However, this is only feasible when the roots of words are existed in a vocabulary. It therefore has a difficulty in representing single-morpheme words (*e.g.,* galaxy, honeymoon) or domain-specific words (*e.g.,* musculoaponeurotic). The second approach is to utilize lexical resources (Pilehvar and Collier, 2017; Bahdanau et al., 2017; Prokhorov et al., 2019) such as WordNet (Miller, 1995), PPDB (Ganitkevitch et al., 2013). Although these resources could cover a wide-range of lexicons and contain domain-specific words, they have a weakness for the representations of morphologically complex words (*e.g.,* uneventfulness) or the endocentric compound words (*e.g.,* **black**jacket, **red**jacket).

In this paper, we present a novel method that utilizes both subword and knowledge information to represent unseen words. This hybrid approach makes the two information complement each other in a useful way. It means that subword information could learn the single-morpheme information from knowledge and, in other hand, knowledge information could mitigate their limitation by learning morphological information from subwords. To this end, we propose two stages of representation learning. We first generalize pre-trained word embeddings using subword embeddings. We then map the learned subword embeddings into semantic knowledge networks and re-train the subword embeddings through a pair-wise learning between lexical entries in the semantic networks.

To verify the efficacy of our methodology, we conduct both an intrinsic and an extrinsic task which are word similarity and sentence classification, respectively. In particular, we apply our method to various languages which are rooted from different language families since subword information is a language-dependent characteristic. The experimental results clearly show that the proposed methodology works quite well and provide consistent performance improvements over strong baselines.

Our goal is not to achieve new state-of-the-art results on NLP tasks but to put the current state of the art method, which used static embeddings, on a more solid footing by enabling it to represent unseen words effectively. Therefore, we believe that a number of existing works can benefit

[1]In this paper, we use unseen words and out-of-vocabulary words interchangeably to represent words that do not exist in pre-trained word embeddings.

from our work.
The remainder of this paper is organized as follows. In Section 2, we discuss related works. In Section 3, we describe the proposed method. We report our performance evaluation results and analyze our methodology in detail in Sections 4 and 5, respectively. Finally, we conclude this paper in Section 6.

2. Related Works

In this section, we review some of works that mainly focus on representing unseen words in pre-trained word embedding.

2.1. Subword Information for Unseen Words

To represent unseen words in pre-trained word embeddings, several works tried to utilize subword information in words. For example, Luong et al. (2013) utilizes morphemes as an atomic unit, and learns the word representations from recursive neural networks (Socher et al., 2013). Similarly, Pinter et al. (2017) and Kim et al. (2018) used bi-directional long-short term memory (LSTM) networks and convolutional neural networks (CNN), respectively, on character representations to represent OOV words. Unlike these works that use specific neural networks, Zhao et al. (2018) proposed a simple method to represent unseen words using a simple additive composition function based on character n-gram embeddings. The aforementioned works are trained to minimize the distance between the generated embeddings and pre-trained word embeddings.

2.2. Lexical Resource for Unseen Words

On the other hand, there are several works that utilized the lexical knowledge encoded in dictionaries, ontologies, or other lexical resources to represent unseen words. For example, Pilehvar and Collier (2017) applies a link analysis algorithm to WordNet graph, and have proved that lexical resources are quite useful at representing rare and unseen words. Bahdanau et al. (2017) utilizes the definitions of words in WordNet to induce embeddings for unseen words. They feed the definitions about rare and unseen words to LSTMs and used the output representations as the embeddings for unseen words. Recently, Prokhorov et al. (2019) introduces a method that learns the alignment between graph embeddings of semantic networks and pre-trained word embeddings.

2.3. Contextualized Representation for Unseen words

To represent novel or unseen words which are not existed in vocabulary, several works proposed methods that utilize contextual information (Herbelot and Baroni, 2017; Khodak et al., 2018). These methods usually focus on representing novel meaning words or disambiguating meaning of words. However, the representations from these models are not reusable, and it is only applicable when context information is given. Compared to these contextualized methods, the above two approaches (i.e., utilizing subword or knowledge) including our method are different in that the generated embeddings are reusable and do not require context information. In our paper, we more focus on the static methods than contextualized ones.

3. Methodology

The proposed methodology has two stages to represent unseen words. Figure 1 describes the main idea of our work. The procedures are straightforward. We first learn the subword embeddings from pre-trained word embeddings (*e.g.,* word2vec, GloVe) through an additive composition function. We then map the learned embeddings to nodes in semantic networks (e.g., WordNet) and re-train the subword embeddings for each word pair which is newly observed in the semantic networks.

3.1. Learning subword embeddings from pre-trained word embeddings

There are several ways to represent subwords, such as characters (Pinter et al., 2017; Kim et al., 2018), character n-grams (Bojanowski et al., 2017; Zhao et al., 2018), morphemes (Luong et al., 2013) and byte-pair encodings (Devlin et al., 2019). We opt for character n-grams as the smallest units in our method since utilizing character n-grams is simple yet effective and does not require any tools compared to byte-pair encodings and morphemes [2].
To generate word representations based on character n-grams, we use an additive composition function since this simple function works quite well with the character n-grams. Before applying this function to words, we pad a special token to the side of an input word to differentiate prefixes and suffixes. For example, given an input word *underground*, we add special tokens to the input word as *__underground__*. It allows us to differentiate *un* tokens which are located in the first and the intermediate position. Based on these subwords, we make a bag of character n-grams to represent words as follows:

$$e_w = \frac{1}{N} \sum_{g \in C(w)} \mathbb{I}[g \in V_s] z_g \qquad (1)$$

where $C(w)$ is a set of character n-grams for a word w and z_g is a subword embedding for g. $\mathbb{I}[p]$ is an indicator function that returns 1 if p is true and 0 otherwise, V_s is a vocabulary for character n-grams and N is the number of character n-grams in V_s. For example, given a word *magical* and n-gram size is 3, $C(w)$ contains *_ma, mag, agi, gic, ica, cal, al_*. The training objective is to minimize the distance between the generated embeddings and pre-trained word embeddings for the same words. We use squared euclidean distance as an objective function and this is as follows:

$$L = \sum_{w \in \mathbf{V}} \|\hat{e}_w - e_w\|_2^2 \qquad (2)$$

where V_w is a vocabulary of pre-trained word embeddings, and $\hat{e}_w$ and e_w are a pre-trained embedding and a generated embedding for a word w, respectively.

3.2. Bridging subword embeddings to semantic networks

Learning subword embeddings from pre-trained word embeddings allows us to cover a wide range of words. However, it is only feasible when the roots of words exist in

[2]When we use morphemes as our smallest units, we obtained the similar performance with character n-grams.

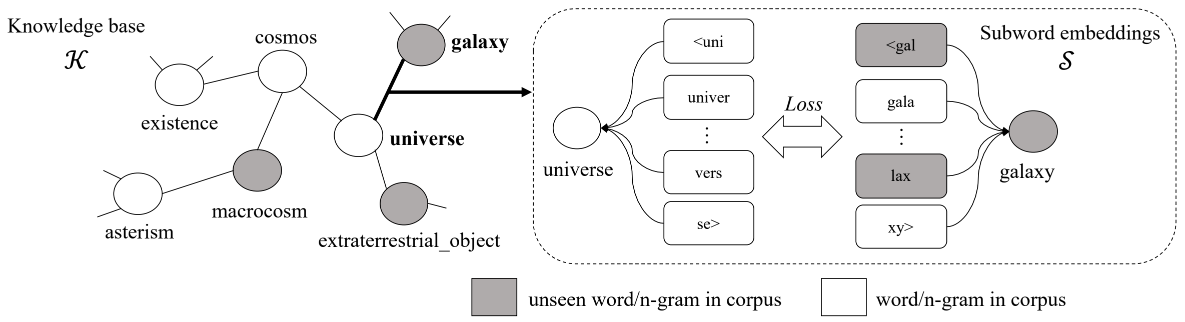

Figure 1: Learning phase of the proposed models. We map the learned subword embeddings to the semantic networks (Upper networks in Figure). We then re-learn the subword embeddings from the pairs in semantic networks.

vocabulary. It, therefore, has a weakness in representing single morpheme words or domain-specific words in specific fields. To mitigate such limitations, we further widen the coverage of words by bridging subwords to semantic networks. The rationale behind using semantic networks is that the networks usually include a lot of rare words and domain-specific terms which are usually absent in a training corpus of pre-trained word embeddings (Prokhorov et al., 2019).

Semantic networks can be viewable as a graph *G = (V, E)*, where *V* is a set of vertices that correspond to words or concepts, and E is a set of edges that indicate semantic relationships between vertices in *V*. We opt for WordNet (Miller, 1995) as our semantic networks because this is the *de-facto* standard lexical resource in a natural language community. WordNet contains around 120K groups of synonyms, referred to as synsets, which are connected to each other by means of around 200K lexical-semantic relations, such as hypernymy and meronymy.

To map the subwords into the semantic networks, we first choose word pairs from the networks which have a direct relationship. Similar to the first stage, we split each word of pairs into the same units (i.e., character n-grams) to match the learned subword embeddings with words in semantic networks. These units contain not only observed subwords from the first stage but also unseen subwords which are newly observed in semantic networks. The words in pair are reconstructed by subword embeddings through the additive composition function in Eq. 1. We then learn the semantic pairs based on generated embeddings. Given a pair w_1 and w_2, the embeddings are optimized through a margin-based loss as follows:

$$\begin{aligned} L = \min_{\theta} \sum & \max(0, \delta - cos(e_{w_1}, e_{w_2}) + cos(e_{w_1}, t_1)) \\ & + \max(0, \delta - cos(e_{w_1}, e_{w_2}) + cos(e_{w_2}, t_2)) \end{aligned} \tag{3}$$

where e_{w_1} and e_{w_2} are the generated embeddings of w_1 and w_2, respectively, and θ is the parameters in our model. δ is the margin, and we found that 0.4 leads the best performance. t_1 and t_2 are the generated embeddings which are sampled from mini-batch. The sampled embeddings t_i play the role of negative samples for the training. There are two choices for sampling negative samples. These are *random* sampling, which simply samples random words, and *max-sim* sampling, which samples most similar words to the w_1 and w_2, in mini-batch. We use *max-sim* since we empirically found that *max-sim* performs better than *random* sampling.

4. Experiments

In this section, we describe the evaluation results on both an intrinsic and an extrinsic task which are word similarity and sentence classification, respectively. In particular, we conduct a word similarity task on multiple languages which are rooted from different language families since subword features are language-dependant. We denote our method as Knowledge-enhanced Subword embedding (KeSub).

4.1. Experimental Settings

Embeddings The proposed method starts from pre-trained word embeddings. In the experiments, we use word2vec (Mikolov et al., 2013) for English which is publicly available[3] and the most popular word embeddings in NLP communities. These embeddings contains 3M words with 300 dimension. For non-Enlglish languages, we use polyglot (Al-Rfou et al., 2013) embeddings which support multilinguality. The polyglot embeddings include roughly 100K words with 128 dimension. In these pre-trained word embeddings, we remove noise words such as web URLs, special characters since we empirically found that such words interfere the subword training.

Lexical Resources As an lexical resources for English, we use WordNet. For non-English languages, We utilize open WordNet projects (Bond and Foster, 2013). These languages contain Chinese, German and Thai which come from Sino-Tibetan, Indo-European and Tai-Kadai, respectively. For Chinese, we used the translated Chinese Open WordNet (Wang and Bond, 2013) which is constructed by several Chinese WordNet. We employ GermaNet (Hamp and Feldweg, 1997) for German. For Thai, we employed thai WordNet (Thoongsup et al., 2009) which is constructed by using a bi-lingual dictionary.

Hyper-parameters We use the range of character n-grams from 2 to 5. We exclude subwords that appear less than 5 times. The batch size of all models is 64, and the models are optimized with 0.01 learning rate through Adam (Kingma and Ba, 2014) Optimizer. We chose the above parameters by validating English RareWord dataset (Luong et al., 2013).

[3]https://code.google.com/archive/p/word2vec/

Model	RW (en)		ZH-240 (zh)		GUR-350 (de)		SL999 (th)	
	ρ	r	ρ	r	ρ	r	ρ	r
Original	45.3	41.9	30.5	31.6	22.6	22.0	15.5	26.3
MIMICK (Pinter et al., 2017)	50.7	48.3	31.4	32.5	28.7	29.4	19.8	28.3
LSTM-Def (Bahdanau et al., 2017)	48.6	45.7	31.3	32.3	24.1	23.9	16.3	27.7
BoS (Zhao et al., 2018)	50.5	48.4	32.6	35.9	36.7	36.4	16.1	30.4
GWR (Kim et al., 2018)	51.6	49.1	32.4	33.6	32.5	33.7	20.5	32.7
KeSub (Ours)	**52.4**	**50.5**	**39.9**	**43.5**	**37.8**	**37.6**	**22.2**	**34.2**

Table 1: Word similarity results on entire datasets (Spearman's rho (ρ) x 100 and Pearson's rho (r) x 100 correlation). Best results are highlighted in boldface.

Initialization	Model	TREC	SST	MR	CR
0 %	Original	88.0	**43.1**	75.7	77.8
	+ KeSub	**89.1**	42.6	**77.2**	**78.4**
25 %	Original	84.6	37.3	72.9	75.1
	+ KeSub	**87.5**	**42.0**	**75.4**	**77.6**
50 %	Original	71.4	32.1	68.9	69.7
	+ KeSub	**87.6**	**41.6**	**72.6**	**76.0**
75 %	Original	59.4	28.5	65.5	67.2
	+ KeSub	**85.1**	**39.6**	**72.9**	**74.3**

Table 2: Sentence classification results (Test Accuracy) on four datasets. The values of the first column indicate the percentage of dropped words from the pre-trained word embeddings. The dropped words are randomly chosen.

Model	TREC	SST	MR	CR
MIMICK (Pinter et al., 2017)	79.3	34.5	65.1	68.9
LSTM-def (Bahdanau et al., 2017)	74.6	33.3	60.6	70.2
BoS (Zhao et al., 2018)	82.6	36.1	69.1	70.0
GWR (Kim et al., 2018)	78.6	35.1	66.4	71.0
KeSub	**85.3**	**37.8**	**71.4**	**76.2**

Table 3: Sentence classification results (Test Accuracy) on four datasets when we drop all task words from the pre-trained word embeddings and generate such words after each model converges. Best results are highlighted in boldface.

Baselines In all tasks, we compare our method with strong baselines which are MIMICK (Pinter et al., 2017), LSTM-def (Bahdanau et al., 2017), BoS (Zhao et al., 2018) and GWR (Kim et al., 2018). The details are as follows:

- **MIMICK** (Pinter et al., 2017): This is a character-based model that learns a function from characters to word embeddings for representing words. For pre-trained word embeddings, we use the same embeddings that our method used. In our experiments, we compare the best MIMICK model, which uses pre-trained word embeddings for vocabulary words and generated embeddings for OOV words. We follow the hyper-parameter settings in the original paper.
- **LSTM-def** (Bahdanau et al., 2017): This is a model that learns OOV words from WordNet (Miller, 1995) definition. For learning, we used two-layered bi-directional LSTM over the definition of WordNet and set the number of hidden states as 256. To obtain the definition for non-English words, we used the language resources which are stated above.
- **BoS** (Zhao et al., 2018): This is a subword-based embedding method that uses character n-grams to reconstruct pre-trained word embeddings similar to (Pinter et al., 2017). We set the hyper-parameter settings following the original paper.
- **GWR** (Kim et al., 2018): This is a character-based model that learns to relate character embeddings to pre-trained word embeddings through the character-level CNNs and highway networks. The settings we used are the same as MIMICK, and we follow the hyper-parameter settings in the original paper.

4.2. Word Similarity

Settings We first perform a word similarity task to evaluate the ability to capture the semantic similarity between words. The performance of the task is measured by computing correlations between human judgments and the cosine similarities between word pairs. As a dataset, we used RW (Luong et al., 2013) for evaluating English vectors since this dataset is a benchmark[4] to evaluate the representations of rare or OOV words. For Chinese, we used ZH-240 (Chen et al., 2016). For German, we used GUR-350 (Zhang et al., 2015), and we evaluate Thai vectors on SL-999 (Netisopakul et al., 2019). As an evaluation metric, we use two correlation metrics which are Spearman's rho and Pearson's rho.

Results Table 1 shows the overall results of the word similarity task. It shows that the methods that utilize subwords usually perform better than the models with knowledge resources in both of the metrics. Among them, our method outperforms such strong baselines by a large margin. Compared to BoS (Zhang et al., 2015) which uses the same units (i.e., character n-grams) with our method, KeSub shows consistently better performance. It means that

[4]WS-353 and SL-999 are quite useful datasets for evaluating word vectors, but these datasets contain very frequent words and only contain a few rare words (Luong et al., 2013). To mainly verify the quality of OOV word representations, we chose the RW dataset.

utilizing knowledge and subword together makes the model robust and generate well-representations for words. For the other languages, we observe that KeSub yields consistent performance improvements across languages.

4.3. Sentence Classification

Settings Sentence classification is the task of classifying a sentence into pre-defined classes such as sentiment, question type. The datasets for sentence classification are TREC (Li and Roth, 2002), SST (Socher et al., 2013), MR (Pang and Lee, 2004) and CR (Hu and Liu, 2004). These datasets have 6 (question type), 5 (sentiment), 2 (sentiment) and 2 (sentiment) classes, respectively.[5]

For a text classification model, we used stacked LSTM models which are used in (Zhang et al., 2015) as a baseline. It feeds the word embeddings in sequence, and use a averaged hidden states to represent sentences and classify the representations using a softmax layer. The hidden dimension is 450, and two-layered stacked LSTMs are used. To fairly compare our methods with others, we freeze the pre-trained word embeddings during a training step, and we only fine-tune other parameters.

Results To confirm how the proposed model well represents unseen words in the task, we adapt the evaluation strategy of (Prokhorov et al., 2019). In short, we drop the pre-trained embeddings for X% of the words which appear in the datasets. We then replace such embeddings with the generated embeddings from the comparison models.

We first verify that how well our method recovers the performance of the original embeddings. Table 2 shows the evaluation results on different initialization percentages. As you can see from the table, the performance of the original embeddings is significantly decreased as we drop more words in pre-trained word embeddings. On average, the original performance is dropped about 23.4% performance in terms of test accuracy when we drop 75% words in pre-trained word embeddings. However, we observe that the proposed model recovers the original performance quite well even at the highest initialization percentage. KeSub only lose 4.8% performance when we drop 75% pre-trained word embeddings during a training step for subword learning.

We compare our method with strong baselines on 100% initialization. It means that all words in the datasets are dropped from the pre-trained word embeddings when we train each method including ours. Table 3 shows the overall results. As can be seen from the table, again, our method outperforms other baselines by a large margin. In particular, the proposed model achieves as much as 8.1% improvement on average in test accuracy compared to strong baselines in the CR dataset. These results shows that our method works quite well in a downstream task.

5. Analysis

In this section, we analyze our method, denoted as KeSub, both in quantitatively and qualitatively. We first conduct an ablation study to confirm that utilizing both subword and knowledge information is indeed useful compared to using independent information (Section 5.1). We then explore the quality of the generated representations for OOV words through a nearest neighbor search (Section 5.2). We lastly apply our method to domain specific fields to verify whether our method works well in specific domains (Section 5.3).

Model	RW (en)
KeSub	52.4
KeSub **w/o** Knowledge	48.6
KeSub **w/o** Subword	50.4

Table 4: Word similarity results on RW dataset (Spearman's rho (ρ)) when we use each information independently.

5.1. Ablation study

KeSub utilizes both subword and knowledge information to represent unseen words. In this subsection, we analyze how the performance of our method is changed when we get rid of each information in training. Here, we evaluate the performance of word similarity task on the RW dataset since it contains a variety of unseen words compared to other datasets. Table 4 shows the comparison results. As can be seen from the table, we can see that using each information independently shows lower performance than that of utilizing both information. However, when we add the other information to the models which are trained using only one information, the performance is largely increased. It means that our method is quite useful to represent unseen words and both information complement each other in a useful way.

5.2. Nearest neighbors

In this subsection, we explore the quality of representations for unseen words through their nearest neighbors. Here, we use the same settings with the word similarity task and choose words which have different type of unseen words (i.e., morphologically complex words, endocentric compound words, typos). To find the nearest neighbors, we first generate the representation for each word. We then calculate a cosine similarity between words in vocabulary and choose top-4 words in similarity scores. Table 5 shows the nearest neighbor words about four unseen words. We draw following findings: i) KeSub represents variant words well (hologrammatic, hologram, holograms) and neighbors have its variation (n't, wern't, havn't, don't). ii) For the compound words, KeSub catches semantically related words in neighbors (applejuice, applesauce, applejack) . iii) KeSub correctly represent typos as appropriate words (inperfect, imperfect).

5.3. Domain-specific setting

We lastly confirm whether our method works well in other domain. To this end, we perform medical word similarity tasks since the coverage of words in a medical domain is highly extensive. We use the Mayo (Pakhomov et al., 2011) and the UMN (Liu et al., 2012) datasets to evaluate the performance on this domain. We use the medical

[5] 5 sentiments includes (very negative, negative, neutral, positive, very positive), 2 sentiments includes (negative, positive).

Word	hologrammatic	applejuice	inperfect	n't
KeSub	holographic	applesauce	imperfect	wern't
	holography	juicer	pluperfect	arn't
	holograms	juices	imperfectly	havn't
	hologram	applejack	imperfection	don't

Table 5: Nearest neighbor words from original embeddings and compressed embedding which are generated from our model.

Dataset	Model	ρ	r
UMN	Original (word2vec)	15.0	13.6
	KeSub	**27.4**	**30.0**
Mayo	Original (word2vec)	10.3	12.3
	KeSub	**13.1**	**15.9**

Table 6: Word similarity results on medical dataset (Spearman's rho (ρ) x 100 and Pearson's rho (r) x 100 correlation). Best results are highlighted in boldface.

subject headings (MeSH) as semantic networks for training. Here, we use the same settings with the word similarity task. The evaluation results are shown in Table 6. As can be seen from the table, we can see that our method works well in other domains. Although the domain where the pre-trained word embeddings are trained is not quite related to the medical domain, the performance is largely increased. In particular, the performance in the UMN dataset is increased as much as 82% in terms of Spearman's correlation. This shows that our method provides domain specialization advantage in that it can be used to generate embeddings not only for morphologically complex forms but also for domain-specific terms.

6. Conclusion

In this paper, we have presented a method, denoted as knowledge-enhanced subword embedding (KeSub), that effectively employs both subwords and lexical resources to represent unseen words. The proposed method has two-stages for learning representations. We first learn the subword embeddings from pre-trained word embeddings, and then map the learned embeddings to semantic networks. On the networks, the model learns newly appeared subwords from the ontologies. In the performance evaluation, we have shown that the proposed method works quite well in both an intrinsic and an extrinsic task, which are word similarity and sentence classification, respectively. We have also found that subword and knowledge information complement each other and our method provides strong performance on the specialized domains such as biological fields. We believe that the existing works can benefit from our work since there are a number of works that used static embeddings to various domains. As a future work, we plan to apply our method to other tasks such as named entity recognition using ProBase (Wu et al., 2012) which has a number of entity information as a form of semantic networks.

7. Bibliographical References

Al-Rfou, R., Perozzi, B., and Skiena, S. (2013). Polyglot: Distributed word representations for multilingual nlp. *Proceedings of the International Conference on Computational Natural Language Learning (CoNLL).*

Almarwani, N., Aldarmaki, H., and Diab, M. (2019). Efficient sentence embedding using discrete cosine transform. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and the International Joint Conference on Natural Language Processing (EMNLP-IJCNLP).*

Bahdanau, D., Bosc, T., Jastrzebski, S., Grefenstette, E., Vincent, P., and Bengio, Y. (2017). Learning to compute word embeddings on the fly. *arXiv preprint arXiv:1706.00286.*

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics (TACL).*

Bond, F. and Foster, R. (2013). Linking and extending an open multilingual wordnet. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL).*

Chen, Y., Mou, L., Xu, Y., Li, G., and Jin, Z. (2016). Compressing neural language models by sparse word representations. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL).*

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL).*

Ganitkevitch, J., Van Durme, B., and Callison-Burch, C. (2013). Ppdb: The paraphrase database. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL).*

Hamp, B. and Feldweg, H. (1997). Germanet-a lexical-semantic net for german. In *Automatic information extraction and building of lexical semantic resources for NLP applications.*

Herbelot, A. and Baroni, M. (2017). High-risk learning: acquiring new word vectors from tiny data. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP).*

Hu, M. and Liu, B. (2004). Mining and summarizing customer reviews. In *Proceedings of the Conference on Knowledge Discovery and Data mining (KDD).*

Khodak, M., Saunshi, N., Liang, Y., Ma, T., Stewart, B., and Arora, S. (2018). A la carte embedding: Cheap but effective induction of semantic feature vectors. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL).*

Kim, Y., Kim, K.-M., Lee, J.-M., and Lee, S. (2018). Learning to generate word representations using subword information. In *Proceedings of the International Conference on Computational Linguistics (COLING).*

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Li, X. and Roth, D. (2002). Learning question classifiers. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Liu, Y., McInnes, B. T., Pedersen, T., Melton-Meaux, G., and Pakhomov, S. (2012). Semantic relatedness study using second order co-occurrence vectors computed from biomedical corpora, umls and wordnet. In *Proceedings of the ACM SIGHIT International Health Informatics Symposium*.

Luong, M.-T., Socher, R., and Manning, C. D. (2013). Better word representations with recursive neural networks for morphology. *Proceedings of the International Conference on Computational Natural Language Learning (CoNLL)*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Proceedings of the International Conference on Neural Information Processing Systems (NeurIPS)*.

Miller, G. A. (1995). Wordnet: a lexical database for english. *Communications of the ACM*.

Netisopakul, P., Wohlgenannt, G., and Pulich, A. (2019). Word similarity datasets for thai: Construction and evaluation. *arXiv preprint arXiv:1904.04307*.

Pakhomov, S. V., Pedersen, T., McInnes, B., Melton, G. B., Ruggieri, A., and Chute, C. G. (2011). Towards a framework for developing semantic relatedness reference standards. *Journal of biomedical informatics*.

Pang, B. and Lee, L. (2004). A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Pennington, J., Socher, R., and Manning, C. (2014). Glove: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Pilehvar, M. T. and Collier, N. (2017). Inducing embeddings for rare and unseen words by leveraging lexical resources. *Proceedings of the European Chapter of the Association for Computational Linguistics (EACL)*.

Pinter, Y., Guthrie, R., and Eisenstein, J. (2017). Mimicking word embeddings using subword rnns. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Prokhorov, V., Pilehvar, M. T., Kartsaklis, D., Lio, P., and Collier, N. (2019). Unseen word representation by aligning heterogeneous lexical semantic spaces. *Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI)*.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A., and Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Sun, X., Dong, K., Ma, L., Sutcliffe, R. F. E., He, F., Chen, S., and Feng, J. (2019). Drug-drug interaction extraction via recurrent hybrid convolutional neural networks with an improved focal loss. *Entropy*, 21(1):37.

Thoongsup, S., Robkop, K., Mokarat, C., Sinthurahat, T., Charoenporn, T., Sornlertlamvanich, V., and Isahara, H. (2009). Thai wordnet construction. In *Proceedings of the Workshop on Asian Language Resources*.

Wang, S. and Bond, F. (2013). Building the chinese open wordnet (cow): Starting from core synsets. In *Proceedings of the Workshop on Asian Language Resources*.

Wu, W., Li, H., Wang, H., and Zhu, K. Q. (2012). Probase: A probabilistic taxonomy for text understanding. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, pages 481–492. ACM.

Yang, Z., Zhu, C., and Chen, W. (2019). Parameter-free sentence embedding via orthogonal basis. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 638–648.

Zhang, X., Zhao, J., and LeCun, Y. (2015). Character-level convolutional networks for text classification. In *Proceedings of the International Conference on Neural Information Processing Systems (NeurIPS)*.

Zhao, J., Mudgal, S., and Liang, Y. (2018). Generalizing word embeddings using bag of subwords. *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.