Wikipedia2Vec: An Efficient Toolkit for Learning and Visualizing the Embeddings of Words and Entities from Wikipedia

Ikuya Yamada[1,2] ikuya@ousia.jp
Akari Asai[3] akari@cs.washington.edu
Jin Sakuma[4] jsakuma@tkl.iis.u-tokyo.ac.jp
Hiroyuki Shindo[5,2] shindo@is.naist.jp
Hideaki Takeda[6] takeda@nii.ac.jp
Yoshiyasu Takefuji[7] takefuji@sfc.keio.ac.jp
Yuji Matsumoto[2] matsu@is.naist.jp

[1]Studio Ousia [2]RIKEN AIP [3]University of Washington [4]The University of Tokyo
[5]Nara Institute of Science and Technology [6]National Institute of Informatics [7]Keio University

Abstract

The embeddings of entities in a large knowledge base (e.g., Wikipedia) are highly beneficial for solving various natural language tasks that involve real world knowledge. In this paper, we present Wikipedia2Vec, a Python-based open-source tool for learning the embeddings of words and entities from Wikipedia. The proposed tool enables users to learn the embeddings efficiently by issuing a single command with a Wikipedia dump file as an argument. We also introduce a web-based demonstration of our tool that allows users to visualize and explore the learned embeddings. In our experiments, our tool achieved a state-of-the-art result on the KORE entity relatedness dataset, and competitive results on various standard benchmark datasets. Furthermore, our tool has been used as a key component in various recent studies. We publicize the source code, demonstration, and the pretrained embeddings for 12 languages at https://wikipedia2vec.github.io.

1 Introduction

Entity embeddings, i.e., vector representations of entities in knowledge base (KB), have played a vital role in many recent models in natural language processing (NLP). These embeddings provide rich information (or *knowledge*) regarding entities available in KB using fixed continuous vectors. They have been shown to be beneficial not only for tasks directly related to entities (e.g., entity linking (Yamada et al., 2016; Ganea and Hofmann, 2017)) but also for general NLP tasks (e.g., text classification (Yamada and Shindo, 2019), question answering (Poerner et al., 2019)). Notably, recent studies have also shown that these embeddings can be used to enhance the performance of state-of-the-art contextualized word embeddings (i.e., BERT (Devlin et al., 2019)) on downstream tasks (Zhang et al., 2019; Peters et al., 2019; Poerner et al., 2019).

In this work, we present *Wikipedia2Vec*, a Python-based open source tool for learning the embeddings of words and entities easily and efficiently from Wikipedia. Due to its scale, availability in a variety of languages, and constantly evolving nature, Wikipedia is commonly used as a KB to learn entity embeddings. Our proposed tool jointly learns the embeddings of words and entities, and places semantically similar words and entities close to one another in the vector space. In particular, our tool implements the word-based skip-gram model (Mikolov et al., 2013a,b) to learn word embeddings, and its extensions proposed in Yamada et al. (2016) to learn entity embeddings. Wikipedia2Vec enables users to train embeddings by simply running a single command with a Wikipedia dump file as an input. We highly optimized our implementation, which makes our implementation of the skip-gram model faster than the well-established implementation available in gensim (Řehůřek and Sojka, 2010) and fastText (Bojanowski et al., 2017).

Experimental results demonstrated that our tool achieved enhanced quality compared to the existing tools on several standard benchmarks. Notably, our tool achieved a state-of-the-art result on the entity relatedness task based on the KORE dataset. Due to its effectiveness and efficiency, our tool has been successfully used in various downstream NLP tasks, including entity linking (Yamada et al., 2016; Eshel et al., 2017; Chen et al., 2019), named entity recognition (Sato et al., 2017; Lara-Clares and Garcia-Serrano, 2019), question answering (Yamada et al., 2018b; Poerner et al., 2019), knowledge graph completion (Shah et al., 2019), paraphrase detection (Duong et al., 2019), fake news detection (Singh et al., 2019), and text classification (Yamada and Shindo, 2019).

We also introduce a web-based demonstration of our tool that visualizes the embeddings by plotting them onto a two- or three-dimensional space

Proceedings of the 2020 EMNLP (Systems Demonstrations), pages 23–30

using dimensionality reduction algorithms. The demonstration also allows users to explore the embeddings by querying similar words and entities.

The source code has been tested on Linux, Windows, and macOS, and released under the Apache License 2.0. We also release the pretrained embeddings for 12 languages (i.e., English, Arabic, Chinese, Dutch, French, German, Italian, Japanese, Polish, Portuguese, Russian, and Spanish).

The main contributions of this paper are summarized as follows:

- We present Wikipedia2Vec, a tool for learning the embeddings of words and entities easily and efficiently from Wikipedia.
- Our tool achieved a state-of-the-art result on the KORE entity relatedness dataset, and performed competitively on the various benchmark datasets.
- We present a web-based demonstration that allows users to explore the learned embeddings.
- We publicize the code, demonstration, and the pretrained embeddings for 12 languages at https://wikipedia2vec.github.io.

2 Related Work

Many studies have recently proposed methods to learn entity embeddings from a KB (Hu et al., 2015; Li et al., 2016; Tsai and Roth, 2016; Yamada et al., 2016, 2017, 2018a; Cao et al., 2017; Ganea and Hofmann, 2017). These embeddings are typically based on conventional word embedding models (e.g., skip-gram (Mikolov et al., 2013a)) trained with data retrieved from a KB. For example, Ristoski et al. (2018) proposed RDF2Vec, which learns entity embeddings using the skip-gram model with inputs generated by random walks over the large knowledge graphs such as Wikidata and DBpedia. Furthermore, a simple method that has been widely used in various studies (Yaghoobzadeh and Schutze, 2015; Yamada et al., 2017, 2018a; Al-Badrashiny et al., 2017; Suzuki et al., 2018) trains entity embeddings by replacing the entity annotations in an input corpus with the unique identifier of their referent entities, and feeding the corpus into a word embedding model (e.g., skip-gram). Two open-source tools, namely Wiki2Vec[1] and Wikipedia Entity Vectors,[2] have implemented this method. Our proposed tool is based on Yamada et al. (2016), which extends this idea by using neighboring entities connected by internal hyperlinks of Wikipedia as additional contexts to train the model. Note that we used the RDF2Vec and Wiki2Vec as baselines in our experiments, and achieved enhanced empirical performance over these tools on the KORE dataset. Additionally, there have been various relational embedding models proposed (Bordes et al., 2013; Wang et al., 2014; Lin et al., 2015) that aim to learn the entity representations that are particularly effective for knowledge graph completion tasks.

```
$ wget https://dumps.wikimedia.org/enwiki/latest/
    enwiki-latest-pages-articles.xml.bz2
$ wikipedia2vec train enwiki-latest-pages-articles.
    xml.bz2 MODEL_FILE
```

Figure 1: Shell commands to train embeddings from the latest English Wikipedia dump.

```
>>> from wikipedia2vec import Wikipedia2Vec
>>> model = Wikipedia2Vec.load(MODEL_FILE)
>>> model.get_entity_vector("Scarlett Johansson")
memmap([-0.1979,  0.3086, ..., ], dtype=float32)
>>> model.get_word_vector("tokyo")
memmap([ 0.0161, -0.0332, ..., ], dtype=float32)
>>> model.most_similar(model.get_entity("Python (
    programming language)"))[:3]
[(<Word python>, 0.7265),
 (<Entity Ruby (programming language)>, 0.6856),
 (<Entity Perl>, 0.6794)]
```

Figure 2: An example that uses the Wikipedia2Vec embeddings on a Python interactive shell.

3 Overview

Wikipedia2Vec is an easy-to-use, optimized tool for learning embeddings from Wikipedia. This tool can be installed using the Python's `pip` tool (`pip install wikipedia2vec`). Embeddings can be learned easily by running the `wikipedia2vec train` command with a Wikipedia dump file[3] as an argument. Figure 1 shows the shell commands that download the latest English Wikipedia dump file and run training of the embeddings based on this dump using the default hyper-parameters.[4] Furthermore, users can easily use the learned embeddings. Figure 2 shows the example Python code that loads the learned embedding file, and obtains the embeddings of an entity *Scarlett Johansson* and a word *tokyo*, as well as the most similar words and entities of an entity *Python*.

[1] https://github.com/idio/wiki2vec

[2] https://github.com/singletongue/WikiEntVec

[3] The dump file can be downloaded at Wikimedia Downloads: https://dumps.wikimedia.org

[4] The *train* command has many optional hyper-parameters that are described in detail in the documentation.

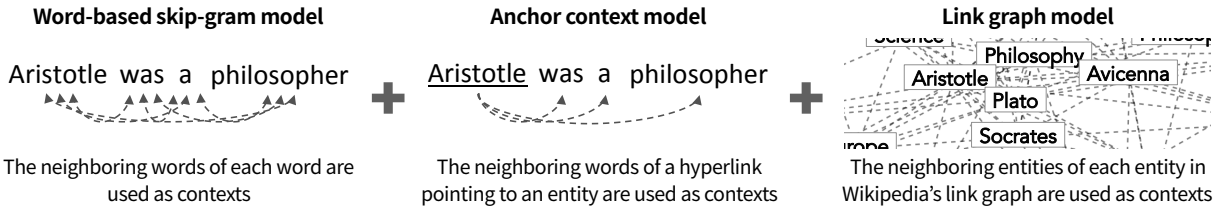

Figure 3: Wikipedia2Vec learns embeddings by jointly optimizing word-based skip-gram, anchor context, and link graph models.

3.1 Model

Wikipedia2Vec implements the conventional skip-gram model (Mikolov et al., 2013a,b) and its extensions proposed in Yamada et al. (2016) to map words and entities into the same d-dimensional vector space. The skip-gram model is a neural network model with a training objective to find embeddings that are useful for predicting context items (i.e., words or entities in this paper) given each item. The loss function of the model is defined as:

$$\mathcal{L}_s = -\sum_{o_i \in O} \sum_{o_c \in C_{o_i}} \log P(o_c|o_i), \qquad (1)$$

where O is a set of all items (i.e., words or entities), C_o is the set of context items of o, and the conditional probability $\log P(o_c|o_i)$ is defined using the following softmax function:

$$P(o_c|o_i) = \frac{\exp({\mathbf{V}_{o_i}}^\top \mathbf{U}_{o_c})}{\sum_{o \in O} \exp({\mathbf{V}_{o_i}}^\top \mathbf{U}_o)}, \qquad (2)$$

where $\mathbf{V}_o \in \mathbb{R}^d$ and $\mathbf{U}_o \in \mathbb{R}^d$ denote the embeddings of item o in embedding matrices $\mathbf{V}$ and $\mathbf{U}$, respectively.

Our tool learns the embeddings by jointly optimizing the three skip-gram-based sub-models described below (see also Figure 3). Note that the matrices $\mathbf{V}$ and $\mathbf{U}$ contain the embeddings of both words and entities.

Word-based Skip-gram Model Given each word in a Wikipedia page, this model learns word embeddings by predicting the neighboring words of the given word. Formally, given a sequence of words $w_1, w_2, ..., w_N$, the loss function of this model is defined as follows:

$$\mathcal{L}_w = -\sum_{i=1}^{N} \sum_{-c \leq j \leq c, j \neq 0} \log P(w_{i+j}|w_i), \qquad (3)$$

where c is the size of the context words, and $P(w_{i+j}|w_i)$ is computed based on Eq.(2).

Anchor Context Model This model aims to place similar words and entities close to one another in the vector space using hyperlinks and their neighboring words in Wikipedia. From a given Wikipedia page, the model extracts the referent entity and surrounding words (i.e., previous and next c words) from each hyperlink in the page, and learns embeddings by predicting surrounding words given each entity. Consequently, the loss function of this model is defined as follows:

$$\mathcal{L}_a = -\sum_{(e_i, Q) \in A} \sum_{w_c \in Q} \log P(w_c|e_i), \qquad (4)$$

where A denotes a set of all hyperlinks in Wikipedia, each containing a pair of a referent entity e_i and a set of surrounding words Q, and $P(w_c|e_i)$ is computed based on Eq.(2).

Link Graph Model This model aims to learn entity embeddings by predicting the neighboring entities of each entity in the Wikipedia's link graph–an undirected graph whose nodes are entities and the edges represent the presence of hyperlinks between the entities. We create an edge between a pair of entities if the page of one entity has a hyperlink to that of the other entity, or if both pages link to each other. The loss function of this model is defined as:

$$\mathcal{L}_e = -\sum_{e_i \in E} \sum_{e_o \in C_{e_i}} \log P(e_o|e_i), \qquad (5)$$

where E is the set of all entities in the vocabulary, and C_e is the neighboring entities of entity e in the link graph, and $P(e_o|e_i)$ is computed by Eq.(2).

Finally, we define the loss function of our model by linearly combining the three loss functions described above:

$$\mathcal{L} = \mathcal{L}_w + \mathcal{L}_a + \mathcal{L}_e \qquad (6)$$

The training is performed by minimizing this loss function using stochastic gradient descent. We use

negative sampling (Mikolov et al., 2013b) to convert the softmax function (Eq.(2)) into computationally feasible ones. The resulting matrix $\mathbf{V}$ is used as the learned embeddings.

3.2 Automatic Generation of Hyperlinks

Because Wikipedia instructs its contributors to create a hyperlink only at the first occurrence of the entity name on a page, many entity names do not appear as hyperlinks. This is problematic for our anchor context model because it uses hyperlinks as a source to learn the embeddings.

To address this problem, our tool automatically generates hyperlinks using a mention-entity dictionary that maps entity names (e.g., "apple") to its possible referent entities (e.g., *Apple Inc.* or *Apple_(food)*) (see Section 4 for details). Our tool extracts all words and phrases from a Wikipedia page and converts each into a hyperlink to an entity if either the entity is referred to by a hyperlink on the same page, or there is only one referent entity associated with the name in the dictionary.

4 Implementation

Our tool is implemented in Python and most of its code is compiled into C++ using Cython (Behnel et al., 2011) to optimize the run-time performance.

As described in Section 3.1, our link graph and anchor context models are based on the hyperlinks in Wikipedia. Because Wikipedia contains numerous hyperlinks, it is challenging to use them efficiently. To address this, we introduce two optimized components–link graph matrix and mention-entity dictionary–that are used during training.

Link Graph Matrix During training, our link graph model needs to obtain numerous neighboring entities of an entity in a large link graph of Wikipedia. To reduce latency, this component stores the entire graph in the memory using the binary sparse matrix in the compressed sparse row (CSR) format, in which its rows and columns represent entities and its values represent the presence of hyperlinks between corresponding entity pairs. Because the size of this matrix is typically small, it can easily be stored on the memory.[5] Note that given a row index in the CSR matrix, the time complexity of obtaining its non-zero column indices (corresponding to the neighboring entities of the entity that corresponds to the row index) is $O(1)$.

[5]The size of the matrix of English Wikipedia is less than 500 megabytes with our default hyper-parameter settings.

Mention-entity Dictionary A mention-entity dictionary is used to generate hyperlinks described in Section 3.2. The dictionary maps entity names to their possible referent entities and is created based on the names and their referent entities obtained from all hyperlinks in Wikipedia. Our tool extracts all words and phrases from a Wikipedia page that are included in the dictionary containing a large number of entity names. To implement this in an efficient manner, we use the Aho–Corasick algorithm, which is an efficient string search algorithm using finite state machine constructed from all entity names. After detecting the words and phrases in the dictionary, our tool converts them to hyperlinks based on heuristics described in Section 3.2.

The embeddings are trained by simultaneously iterating over pages in Wikipedia and entities in the link graph in a random order. The texts and hyperlinks in each page are extracted using the mwparserfromhell MediaWiki parser.[6] We do not use semi-structured data such as tables and infoboxes. We also generate hyperlinks using the mention-entity dictionary. We store the embeddings as a float matrix in a shared memory and update it using multiple processes. Linear algebraic operations required to learn embeddings are implemented using C functions in Basic Linear Algebra Subprograms (BLAS).

Additionally, our tool uses a tokenizer to detect words from a Wikipedia page. The following four tokenizers are currently implemented in our tool: (1) the multi-lingual ICU tokenizer[7] that implements the unicode text segmentation algorithm (Davis, 2019), (2) a simple rule-based tokenizer that splits the text using white space characters, (3) the Jieba tokenizer[8] for Chinese, and (4) the MeCab tokenizer[9] for Japanese and Korean.

5 Experiments

We conducted experiments to compare the quality and efficiency of our tool with those of the existing tools. To evaluate the quality of the entity embeddings, we used the KORE entity relatedness dataset (Hoffart et al., 2012). The dataset consists of 21 entities, and each entity has 20 related entities with scores assessed by humans. Following past work, we reported the Spearman's rank correlation co-

[6]https://github.com/earwig/mwparserfromhell

[7]http://site.icu-project.org

[8]https://github.com/fxsjy/jieba

[9]https://taku910.github.io/mecab

Name	Score
Ours	**0.71**
Ours (w/o link graph model)	0.61
Ours (w/o hyperlink generation)	0.69
RDF2Vec (Ristoski et al., 2018)	0.69
Wiki2vec	0.52

Table 1: The results of Wikipedia2Vec and the baseline entity embeddings on the KORE dataset.

	SEM	SYN	SL	WS	Time
Ours	**0.79**	0.68	**0.40**	0.71	276min
Ours (w/o link graph model)	0.77	0.67	0.39	0.70	170min
Ours (w/o hyperlink generation)	**0.79**	0.67	0.39	**0.72**	211min
Ours (word-based skip-gram)	0.75	0.67	0.36	0.70	154min
gensim (Řehůřek and Sojka, 2010)	0.75	0.67	0.37	0.70	197min
fastText (Bojanowski et al., 2017)	0.63	**0.70**	0.37	0.69	243min

Table 2: The results of Wikipedia2Vec and the baseline word embeddings on the word analogy and word similarity datasets.

efficient between the gold scores and the cosine similarity between the entity embeddings. We used two popular entity embedding tools, RDF2Vec (Ristoski et al., 2018) and Wiki2vec, as baselines.

We also evaluated the quality of the word embeddings by employing two standard tasks: (1) a word analogy task using the semantic subset (SEM) and syntactic subset (SYN) of the Google Word Analogy data set (Mikolov et al., 2013a), and (2) a word similarity task using two standard datasets, namely SimLex-999 (SL) (Hill et al., 2015) and WordSim-353 (WS) (Finkelstein et al., 2002). Following past work, we reported the accuracy for the word analogy task, and the Spearman's rank correlation coefficient between the gold scores and the cosine similarity between the word embeddings for the word similarity task. As baselines for these tasks, we used the skip-gram model (Mikolov et al., 2013a) implemented in the gensim library 3.6.0 (Řehůřek and Sojka, 2010) and the extended skip-gram model implemented in the fastText tool 0.1.0 (Bojanowski et al., 2017). We used WikiExtractor[10] to create the training corpus for baselines. To the extent possible, we used the same hyperparameters to train our models and the baselines.[11]

We also reported the time required for training using our tool and the baseline word embedding tools. Note that the training of RDF2Vec and Wiki2vec tools are implemented using gensim.

We conducted experiments using Python 3.6 and OpenBLAS 0.3.3 installed on the c5d.9xlarge instance with 36 CPU cores deployed on Amazon Web Services. To train our models and the baseline word embedding models, we used the April 2018 version of the English Wikipedia dump.

5.1 Results

Table 1 shows the results of our models and the baseline entity embedding models of the KORE dataset.[12] *w/o link graph model* and *w/o hyperlink generation* are the results of ablation studies disabling the link graph model and automatic generation of hyperlinks, respectively.

Our model successfully outperformed the RDF2Vec and Wiki2vec models and achieved a state-of-the-art result on the KORE dataset. The results also indicated that the link graph model and automatic generation of hyperlinks improved the performance of the KORE dataset.

Table 2 shows the results of our models with the baseline word embedding models on the word analogy and word similarity datasets. We also tested the performace of the word-based skip-gram model implemented in our tool by disabling the link graph and anchor context models.

Our model performed better than the baseline word embedding models on the SEM dataset, as well as on both word similarity datasets. This demonstrates that the semantic signals of entities provided by the link graph and anchor context models are beneficial for improving the quality of word embeddings. Additionally, the feature of the automatic generation of hyperlinks did not generally contribute to the performance on these datasets.

Our implementation of the word-based skip-gram model was substantially faster than gensim and fastText. Furthermore, the training time of our full model was comparable to that of the baseline word embedding models.

6 Interactive Demonstration

We developed a web-based interactive demonstration that enables users to explore the embeddings of words and entities learned by our proposed tool (see Figure 4). This demonstration enables users to visualize the embeddings onto a two- or three-dimensional space using three dimensionality reduction algorithms, namely t-distributed stochastic

[10] https://github.com/attardi/wikiextractor

[11] We used the following settings: $dim\_size = 500$, $window = 5$, $negative = 5$, $iteration = 5$

[12] We obtained the results of the RDF2Vec and Wiki2vec models from Ristoski et al. (2018).

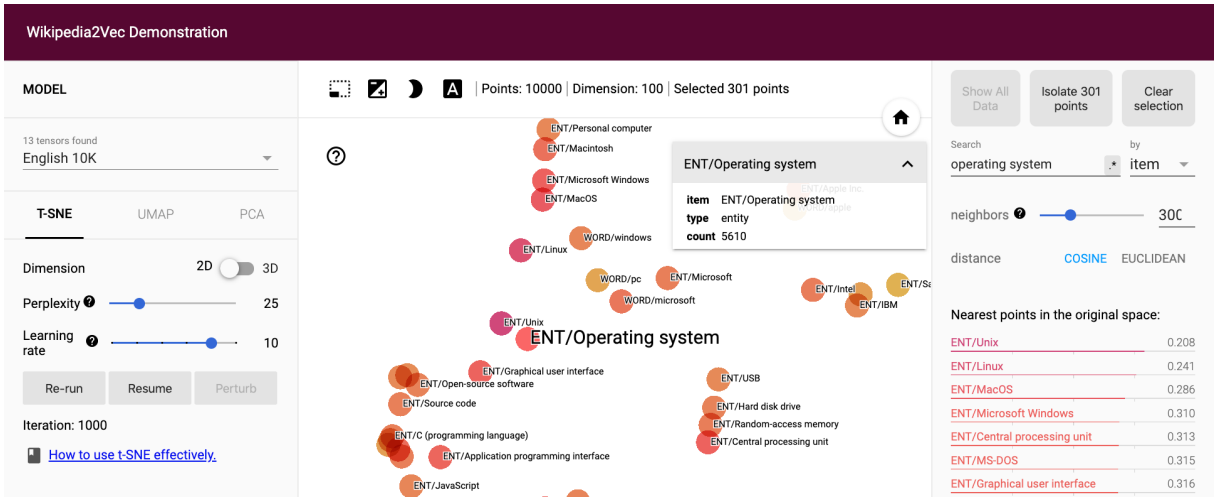

Figure 4: The screenshot of our web-based demonstration. Users can select the target embeddings (top left), configure the dimensionality reduction algorithm (bottom left), explore the visualized embeddings (center), and query similar words and entities based on an arbitrary word or an entity (right).

neighbor embedding (t-SNE) (Maaten and Hinton, 2008), uniform manifold approximation and projection (UMAP) (McInnes et al., 2018), and principal component analysis (PCA). Users can move around the visualized embedding space by dragging and zooming using the mouse. Moreover, the demonstration also allows users to explore the embeddings by querying similar items (words or entities) of an arbitrary item.

We used the pretrained embeddings of 12 languages released with this paper as the target embeddings. Furthermore, we also provided the English embeddings trained without the link graph model to allow users to qualitatively investigate how the link graph model affects the resulting embeddings.

Our demonstration is developed by extending the TensorFlow Embedding Projector.[13] The demonstration is available at `https://wikipedia2vec.github.io/demo`.

7 Use Cases

The embeddings learned using our proposed tool have already been used effectively in various recent studies. Poerner et al. (2019) have recently demonstrated that by combining BERT with the entity embeddings trained by our tool outperforms BERT and knowledge-enhanced contextualized word embeddings (i.e., ERNIE (Zhang et al., 2019)) on unsupervised question answering and relation classification tasks, without any computationally expensive additional pretraining of BERT. Yamada et al. (2018b) developed a neural network-based question answering system based on our tool, and won a competition held by the NIPS 2017 conference. Sato et al. (2017), Chen et al. (2019), and Yamada and Shindo (2019) achieved state-of-the-art results on named entity recognition, entity linking, and text classification tasks, respectively, based on the embeddings learned by our tool. Furthermore, Papalampidi et al. (2019) proposed a neural network model of analyzing the plot structure of movies using the entity embeddings learned by our tool. Other examples include entity linking (Yamada et al., 2016; Eshel et al., 2017), named entity recognition (Lara-Clares and Garcia-Serrano, 2019), paraphrase detection (Duong et al., 2019), fake news detection (Singh et al., 2019), and knowledge graph completion (Shah et al., 2019).

8 Conclusions

In this paper, we present Wikipedia2Vec, an open-source tool for learning the embeddings of words and entities easily and efficiently from Wikipedia. Our experiments demonstrate the superiority of the proposed tool in terms of the quality of the embeddings and the efficiency of the training compared to the existing tools. Furthermore, our tool has been effectively used in many recent state-of-the-art models, which indicates the effectiveness of our tool on downstream tasks. We also introduce a web-based interactive demonstration that enables users to explore the learned embeddings. The source code and the pre-trained embeddings for 12 languages are released with this paper.

[13] `https://projector.tensorflow.org`

References

Mohamed Al-Badrashiny, Jason Bolton, Arun Tejasvi Chaganty, Kevin Clark, Craig Harman, Lifu Huang, Matthew Lamm, Jinhao Lei, Di Lu, Xiaoman Pan, and others. 2017. TinkerBell: Cross-lingual Cold-Start Knowledge Base Construction. In *Text Analysis Conference*.

Stefan Behnel, Robert Bradshaw, Craig Citro, Lisandro Dalcin, Dag Sverre Seljebotn, and Kurt Smith. 2011. Cython: The Best of Both Worlds. *Computing in Science & Engineering*, 13(2):31–39.

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.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating Embeddings for Modeling Multi-relational Data. In *Advances in Neural Information Processing Systems 26*, pages 2787–2795.

Yixin Cao, Lifu Huang, Heng Ji, Xu Chen, and Juanzi Li. 2017. Bridge Text and Knowledge by Learning Multi-Prototype Entity Mention Embedding. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1623–1633.

Haotian Chen, Sahil Wadhwa, Xi David Li, and Andrej Zukov-Gregoric. 2019. YELM: End-to-End Contextualized Entity Linking. *arXiv preprint arXiv:1911.03834v1*.

Mark Davis. 2019. Unicode Text Segmentation. *Unicode Technical Reports*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Phuc H. Duong, Hien T. Nguyen, Hieu N. Duong, Khoa Ngo, and Dat Ngo. 2019. A Hybrid Approach to Paraphrase Detection. In *Proceedings of 2018 5th NAFOSTED Conference on Information and Computer Science*, pages 366–371.

Yotam Eshel, Noam Cohen, Kira Radinsky, Shaul Markovitch, Ikuya Yamada, and Omer Levy. 2017. Named Entity Disambiguation for Noisy Text. In *Proceedings of the 21st Conference on Computational Natural Language Learning*, pages 58–68.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2002. Placing Search in Context: The Concept Revisited. *ACM Transactions on Information Systems*, 20(1):116–131.

Octavian-Eugen Ganea and Thomas Hofmann. 2017. Deep Joint Entity Disambiguation with Local Neural Attention. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2619–2629.

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. SimLex-999: Evaluating Semantic Models with Genuine Similarity Estimation. *Computational Linguistics*, 41(4):665–695.

Johannes Hoffart, Stephan Seufert, Dat Ba Nguyen, Martin Theobald, and Gerhard Weikum. 2012. KORE: Keyphrase Overlap Relatedness for Entity Disambiguation. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, pages 545–554.

Zhiting Hu, Poyao Huang, Yuntian Deng, Yingkai Gao, and Eric Xing. 2015. Entity Hierarchy Embedding. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1292–1300.

Alicia Lara-Clares and Ana Garcia-Serrano. 2019. LSI2 UNED at eHealth-KD Challenge 2019: A Few-shot Learning Model for Knowledge Discovery from eHealth Documents. In *Proceedings of the Iberian Languages Evaluation Forum*.

Yuezhang Li, Ronghuo Zheng, Tian Tian, Zhiting Hu, Rahul Iyer, and Katia Sycara. 2016. Joint Embedding of Hierarchical Categories and Entities for Concept Categorization and Dataless Classification. In *Proceedings of the 26th International Conference on Computational Linguistics*, pages 2678–2688.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning Entity and Relation Embeddings for Knowledge Graph Completion. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, pages 2181–2187.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing Data using t-SNE. *Journal of machine learning research*, 9(Nov):2579–2605.

Leland McInnes, John Healy, and James Melville. 2018. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. *arXiv preprint arXiv:1802.03426v1*.

Tomas Mikolov, Greg Corrado, Kai Chen, and Jeffrey Dean. 2013a. Efficient Estimation of Word Representations in Vector Space. In *Proceedings of the 2013 International Conference on Learning Representations*, pages 1–12.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013b. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119.

Pinelopi Papalampidi, Frank Keller, and Mirella Lapata. 2019. Movie Plot Analysis via Turning Point Identification. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1707–1717.

Matthew E. Peters, Mark Neumann, Robert Logan, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A. Smith. 2019. Knowledge Enhanced Contextual Word Representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 43–54.

Nina Poerner, Ulli Waltinger, and Hinrich Schütze. 2019. BERT is Not a Knowledge Base (Yet): Factual Knowledge vs. Name-Based Reasoning in Unsupervised QA. *arXiv preprint arXiv:1911.03681v1*.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50.

Petar Ristoski, Jessica Rosati, Tommaso Di Noia, Renato De Leone, and Heiko Paulheim. 2018. RDF2Vec: RDF Graph Embeddings and Their Applications. *Semantic Web*, 10(4):721–752.

Motoki Sato, Hiroyuki Shindo, Ikuya Yamada, and Yuji Matsumoto. 2017. Segment-Level Neural Conditional Random Fields for Named Entity Recognition. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 97–102.

Haseeb Shah, Johannes Villmow, Adrian Ulges, Ulrich Schwanecke, and Faisal Shafait. 2019. An Open-World Extension to Knowledge Graph Completion Models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33:3044–3051.

Iknoor Singh, Deepak P, and Anoop K. 2019. On the Coherence of Fake News Articles. *arXiv preprint arXiv:1906.11126v1*.

Masatoshi Suzuki, Koji Matsuda, Satoshi Sekine, Naoaki Okazaki, and Kentaro Inui. 2018. A Joint Neural Model for Fine-Grained Named Entity Classification of Wikipedia Articles. *IEICE Transactions on Information and Systems*, E101.D(1):73–81.

Chen-Tse Tsai and Dan Roth. 2016. Cross-lingual Wikification Using Multilingual Embeddings. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 589–598.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge Graph and Text Jointly Embedding. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1591–1601.

Yadollah Yaghoobzadeh and Hinrich Schutze. 2015. Corpus-level Fine-grained Entity Typing Using Contextual Information. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 715–725.

Ikuya Yamada and Hiroyuki Shindo. 2019. Neural Attentive Bag-of-Entities Model for Text Classification. In *Proceedings of the 23rd Conference on Computational Natural Language Learning*, pages 563–573.

Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. 2016. Joint Learning of the Embedding of Words and Entities for Named Entity Disambiguation. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, pages 250–259.

Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. 2017. Learning Distributed Representations of Texts and Entities from Knowledge Base. *Transactions of the Association for Computational Linguistics*, 5:397–411.

Ikuya Yamada, Hiroyuki Shindo, and Yoshiyasu Takefuji. 2018a. Representation Learning of Entities and Documents from Knowledge Base Descriptions. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 190–201.

Ikuya Yamada, Ryuji Tamaki, Hiroyuki Shindo, and Yoshiyasu Takefuji. 2018b. Studio Ousia's Quiz Bowl Question Answering System. In *The NIPS '17 Competition: Building Intelligent Systems*, pages 181–194.

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. ERNIE: Enhanced Language Representation with Informative Entities. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1441–1451.