REL: An Entity Linker Standing on the Shoulders of Giants

Johannes M. van Hulst
Radboud University
mick.vanhulst@gmail.com

Faegheh Hasibi
Radboud University
f.hasibi@cs.ru.nl

Koen Dercksen
Radboud University
koen.dercksen@ru.nl

Krisztian Balog
University of Stavanger
krisztian.balog@uis.no

Arjen P. de Vries
Radboud University
a.devries@cs.ru.nl

ABSTRACT

Entity linking is a standard component in modern retrieval system that is often performed by third-party toolkits. Despite the plethora of open source options, it is difficult to find a single system that has a modular architecture where certain components may be replaced, does not depend on external sources, can easily be updated to newer Wikipedia versions, and, most important of all, has state-of-the-art performance. The REL system presented in this paper aims to fill that gap. Building on state-of-the-art neural components from natural language processing research, it is provided as a Python package as well as a web API. We also report on an experimental comparison against both well-established systems and the current state-of-the-art on standard entity linking benchmarks.

KEYWORDS

Entity Linking; Toolkit; Entity Disambiguation; NER

ACM Reference format:
Johannes M. van Hulst, Faegheh Hasibi, Koen Dercksen, Krisztian Balog, and Arjen P. de Vries. 2020. REL: An Entity Linker Standing on the Shoulders of Giants. In *Proceedings of Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, China, July 25–30, 2020 (SIGIR '20),* 4 pages.
DOI: 10.1145/3397271.3401416

1 INTRODUCTION

Entity linking (EL) refers to the task of recognizing mentions of specific entities in text and assigning unique identifiers to them from an underlying knowledge repository [2]. The problems of entity recognition and disambiguation have traditionally been studied in the natural language processing (NLP) community. It was also them who first recognized the utility of Wikipedia as a large-scale knowledge repository to disambiguate against [4, 6]. This line of work has been quickly followed up by information retrieval (IR) researchers [17, 18]. Over the past years, entity linking has become a standard component in modern retrieval systems, and has been leveraged in a range of tasks, including document ranking [26], entity retrieval [11], knowledge base population [3], and query recommendation [23]. Since entity linking is not the main focus of these works, it is commonly performed by some third-party toolkit, with the resulting annotations being utilized in downstream processing. Some of the most prominent toolkits used for this purpose include DBpedia Spotlight [16], TAGME [7], WAT [21], and FEL [19].

Existing toolkits fall short in a number of areas. Some are unmaintained [19]; others are meant for short text and inefficient for long text [12]; some rely on external sources like web search engines [5]. Typically, they are shipped with a specific Wikipedia version that has become dated, causing difficulties when attempting to update to a recent Wikipedia [5, 21]. An issue that is often not addressed is the lack of speed (throughput). Most importantly, none of the default open source entity linkers incorporate recent progress made in the NLP community on neural network-based approaches [14]. With this work, we aim to close that gap and remedy all of these problems by introducing an efficient, up-to-date entity linker that has a modular architecture to ease, e.g., updates of external resources like Wikipedia.

We present REL[1] (which stands for Radboud Entity Linker), an open source toolkit for entity linking. REL stands on the shoulders of giants and is an ensemble of multiple methods and packages from the state-of-the-art natural language processing research. REL has been developed with the following design considerations:

- Use *state-of-the-art* approaches for entity disambiguation (ED) [8, 15] and named entity recognition (NER) [1], ensuring it is on par with the state-of-the-art on end-to-end entity linking [14].
- Use a *modular architecture* with mention detection (using a NER approach) and entity disambiguation components. Specifically, separating mention detection from entity disambiguation enables us to choose an NER method appropriate for the context in which entity linking is employed (i.e., optimizing for recall vs. throughput).
- Design for sufficient *throughput*; reporting 700 ms for an average document of 300 words. Notably, most of the time is used for NER, which could be changed to a more efficient option.
- Develop a *lightweight* solution that can be deployed on an average laptop/desktop machine; it does not need much RAM, and, importantly, it does not need a GPU.
- Train on a recent Wikipedia dump (2019-07) and ensure easy *updates* to new Wikipedia versions (all necessary scripts included).

REL is available at http://tiny.cc/RadboudEL under a MIT license, can be deployed as a Python package, or used via a restful API.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
SIGIR '20, Virtual Event, China
© 2020 ACM. 978-1-4503-8016-4/20/07…$15.00
DOI: 10.1145/3397271.3401416

[1] REL in Dutch means mayhem, interference, or disturbance; and, it is easily recognized to abbreviates 'relatie' (relation in English).

2 ENTITY LINKING IN REL

In this section, we present the entity linking method underlying REL. We follow a standard entity linking pipeline architecture [2], consisting of three components: (i) mention detection, (ii) candidate selection, and (iii) entity disambiguation.

2.1 Mention Detection

In the mention detection step, we aim to detect all text spans that can be linked to entities. These text spans, referred to as *mentions*, are obtained by employing a Named Entity Recognition (NER) tool. NER taggers detect entity mentions in text and annotate them with (coarse-grained) entity types [2]. We employ Flair [1], a state-of-the-art NER based on contextualized word embeddings. Flair takes the input to be a sequence of characters and passes it to a bidirectional character-level neural language model to generate a contextual string embedding for each word. These embeddings are then utilized in a sequence labeling module to generate tags for NER.

Using a NER method for mention detection enables us to strike a balance between precision and recall. Another approach, which may result in high recall, is matching all n-grams (up to a certain n) in the input text against a rich dictionary of entity names [2, 10]. In REL, the mention detection component can easily be replaced by another NER tagger such as spaCy[2] or by a dictionary-based approach.

2.2 Candidate Selection

For each text span detected as a mention, we select up to $k_1 + k_2$ (=7) candidate entities (following [8]). The k_1 (=4) candidate entities are selected from the top ranked entities based on the mention-entity prior $p(e|m)$, for a given entity e and a mention m. To compute this prior, we sum up hyperlink counts from Wikipedia and from the CrossWikis corpus [25] to estimate probability $P_{\text{Wiki}}(e|m)$. A uniform probability $P_{\text{YAGO}}(e|m)$ is also extracted from YAGO dictionary [13]. These two probabilities are combined into the final $P(e|m)$ prior as $\min(1, P_{\text{Wiki}}(e|m) + P_{\text{YAGO}}(e|m))$ [8].

The other k_2 (=3) candidate entities are chosen based on their similarity to the context of the mention. This similarity score is obtained by $\mathbf{e}^T \sum_{w \in c} \mathbf{w}$, where c is n-word ($n = 50$) context surrounding mention m and $\mathbf{w}$ and $\mathbf{e}$ are entity and word embedding vectors. This score is computed for k (=30) entities with the highest $P(e|m)$ prior and the top-k_2 entities are added to the list of candidate entities [8].

In REL, we use Wikipedia2Vec word and entity embeddings [28] to estimate the similarity between an entity and a mention's local context. Wikipedia2Vec jointly learns word and entity embeddings from Wikipedia text and link structure, and is available as an open source library [27]. The hyper-parameters k_1, k_2, k, and n are set based on the recommended values in [8, 15].

2.3 Entity Disambiguation

In the entity disambiguation step, we link mentions to their corresponding entities in the knowledge graph (here: Wikipedia). Entity disambiguation in REL is based on the Ment-norm method proposed by Le and Titov [15]. Given an input document D, the entity linking decisions are made by combining local compatibility (which includes prior importance and contextual similarity) and coherence with the other entity linking decisions in the document:

$$E^* = \underset{E \in C_1 \times \ldots \times C_n}{\operatorname{argmax}} \sum_{i=1}^{n} \psi(e_i, c_i) + \sum_{i \neq j} \phi(e_i, e_j, D) \,, \tag{1}$$

where C_i denotes the set of candidate entities for mention m_i and $E = \{e_1, .., e_n\}$. The coherence score between entity e_i and its local context c_i is computed by the function $\psi(e_i, c_i)$ as defined in [8], and the coherence between all entity linking decisions is captured by the function $\phi(e_i, e_j, D)$. Le and Titov [15] compute the ϕ function by incorporating relations between mentions of a document. Assuming K latent relations, ϕ is calculated as:

$$\phi(e_i, e_j, D) = \sum_{k=1}^{K} \alpha_{ijk} \mathbf{e}_i^T \mathbf{R}_k \mathbf{e}_j \,, \tag{2}$$

where $\mathbf{e}_i, \mathbf{e}_j \in \mathbb{R}^d$ are the embeddings of entities e_i, e_j (using the same embeddings as in the candidate selection step), $\mathbf{R}_k$ is a diagonal matrix, and α_{ijk} is a normalized score defined as:

$$\alpha_{ijk} = \frac{1}{Z_{ijk}} \exp\left\{ \frac{f^T(m_i, c_i) \mathbf{D}_k f(m_j, c_j)}{\sqrt{d}} \right\} , \tag{3}$$

where $\mathbf{D}_k \in \mathbb{R}^{d \times d}$ is a diagonal matrix, and function f is a single-layer neural network that maps mention m_i and its context c_i to a d-dimensional vector. Z_{ijk} is a normalization factor over j and is computed as:

$$Z_{ijk} = \sum_{\substack{j'=1 \\ j' \neq i}}^{n} \exp\left\{ \frac{f^T(m_i, c_i) \mathbf{D}_k f(m_j, c_j)}{\sqrt{d}} \right\} . \tag{4}$$

The optimization of Eq. (1) is performed using max-product loopy belief propagation (LBP), and the final score for an entity of a mention is obtained by a two-layer neural network that combines $P(e|m)$ with max-marginal probability of an entity for a given document. The training of the model, referred to as the *ED model* henceforth, is performed using max-margin loss. To estimate posterior probabilities of the linked entities, we fit a logistic function over the final scores obtained by the neural model [22].

3 IMPLEMENTATION AND USAGE

Next, we describe the implementation details and usage of REL.

3.1 Implementation Details

Memory and GPU usage. One of the design requirements of REL is being lightweight, such that it can be deployed on an average machine. To minimize memory requirements, we store Wikipedia2Vec entity and word embeddings, GloVe embeddings, and an index of pre-computed $P(e|m)$ values (i.e., a surface form dictionary) in a SQLite3[3] database. Using SQLite, we are able to minimize memory usage for our API to 1.8GB if the user chooses to not preload embeddings. REL also does not require GPU during inference. The

[2] https://spacy.io/

[3] https://www.sqlite.org/index.html

Figure 1: Example API input and output for entity linking.

```
INPUT:
 {"text": "Belgrade 1996-08-30 Result in an international
       basketball tournament on Friday: Red Star ( Yugoslavia )
       beat Dinamo ( Russia) 92-90 ( halftime 47-47 )."}
```

```
OUTPUT:
[
[0, 8, 'Belgrade', 'Belgrade', 0.91, 0.98, 'LOC', ],
[80, 8, 'Red Star', 'KK_Crvena_zvezda', 0.36, 0.99, 'ORG'],
[91, 10, 'Yugoslavia', 'Yugoslavia', 0.8, 0.99, 'LOC'],
[109, 6, 'Dinamo', 'FC_Dinamo_Bucuresti', 0.7, 0.99, 'ORG'],
[118, 6, 'Russia', 'Russia', 0.85, 0.99, 'LOC']
]
```

neural model used for entity disambiguation is a feed-forward network and does not require heavy CPU/GPU usage. Training of Wikipedia2Vec embeddings, however, requires high memory and is done more efficiently using a GPU.

REL components. REL has a modular architecture, with separate components for mention detection, entity disambiguation, and the generation of the $P(e|m)$ index. The mention detection component is based on the Flair package[4] and can be easily replaced by another mention detection approach. The disambiguation component is implemented using PyTorch and based on the source code of [15].[5] The generation of the $P(e|m)$ index is based on the source code of [8] and involves the parsing of Wikipedia, the CrossWikis corpus, and YAGO. Any of these may be either removed completely, or replaced by different corpora; using the resulting $P(e|m)$ index in the package instead.

ED Training. For the entity disambiguation method, we used the AIDA-train dataset for training and AIDA-A for validation. We use the Adam optimizer and reduce the learning rate from 1^{-3} to 1^{-4} once the F1-score of the validation set reaches 0.88 (following [15]).

Embeddings. The entity and word embeddings used for selecting candidate entities are trained on a Wikipedia 2019-07 dump using the Wikipedia2Vec package.[6] Following [9], we set the *min-entity-count* parameter to zero and used the Wikipedia link graph during training. For the entity disambiguation model, we used GloVe embeddings [20] as suggested in [15].

3.2 Usage

REL can be used as a Python package deployed on a local machine, or as a service, via a restful API.

To use REL as a package, our GitHub repository contains step-by-step tutorials on how to perform end-to-end entity linking, and on how to (re-)train the ED model. We provide scripts and instructions for deploying REL using a new Wikipedia dump; this helps REL users to keep up-to-date with emerging entities in Wikipedia, and enables researchers to deploy REL for any specific Wikipedia version that is required for a downstream task.

The API is publicly available. Given an input text, depicted in Fig. 1 (Top), the API returns a list of mentions, each with (i) the start position and length of the mention, (ii) the mention itself, (iii) the linked entity, (iv) the confidence score of ED, and (vi) the confidence score and type of entity from the mention detection step (if available); see Fig. 1 (Bottom). Alternatively, a user can use the API for entity disambiguation only, by submitting an input text and a list of spans (specified with start position and length).

[4]https://github.com/flairNLP/flair
[5]https://github.com/lephong/mulrel-nel
[6]https://wikipedia2vec.github.io/wikipedia2vec

Table 1: EL strong matching results on the GERBIL platform.

Macro F1 **Micro F1**	AIDA-B	MSNBC	OKE-2015	OKE-2016	N3-Reuters-128	N3-RSS-500	Derczynski	KORE50
DBpedia	52.0	42.4	42.0	41.4	21.5	26.7	33.7	29.4
Spotlight	57.8	40.6	44.4	43.1	24.8	27.2	32.2	34.9
WAT	70.8	62.6	53.2	51.8	45.0	**45.3**	**44.4**	37.3
	73.0	64.5	56.4	53.9	49.2	**42.3**	38.0	49.6
SOTA NLP	**82.6**	73.0	56.6	47.8	45.4	43.8	43.2	26.2
	82.4	72.4	61.9	52.7	**50.3**	38.2	34.1	35.2
REL (2014)	81.3	**73.2**	61.5	**57.5**	**46.8**	35.9	38.1	**60.1**
	83.3	**74.4**	**64.8**	**58.8**	49.7	34.3	**41.2**	**61.6**
REL (2019)	78.6	71.1	**61.8**	57.4	45.7	36.2	38.0	50.1
	80.5	72.4	63.1	58.3	49.9	35.0	41.1	50.7

Table 2: ED results on the GERBIL platform.

Macro F1 **Micro F1**	AIDA-B	MSNBC	OKE-2015	OKE-2016	N3-Reuters-128	N3-RSS-500	Derczynski	KORE50
DBpedia	53.7	43.6	30.4	43.0	41.8	42.6	50.3	48.7
Spotlight	56.1	42.1	35.8	43.1	43.4	34.6	43.3	52.3
WAT	79.8	79.7	62.2	0.0	59.2	62.8	70.4	52.4
	80.5	78.8	64.9	0.0	63.1	63.9	69.5	62.2
SOTA NLP	83.8	88.5	**73.2**	**76.7**	**63.4**	**66.6**	**65.3**	52.4
	83.0	86.2	**74.0**	**78.1**	**67.3**	**68.6**	**65.4**	60.8
REL (2014)	**85.5**	**89.6**	65.5	72.0	59.8	61.0	61.9	**61.9**
	86.6	**88.5**	65.8	72.2	64.9	62.8	62.1	**64.6**
REL (2019)	82.9	86.3	64.0	67.0	58.2	61.7	62.3	54.4
	84.0	85.8	64.3	67.3	64.9	64.1	62.0	54.0

4 EVALUATION

We compare REL with a state-of-the-art end-to-end entity linking [14], referred to as SOTA NLP, and two popular well-established entity linking systems: (i) DBpedia-spotlight [16] and (ii) WAT [21], the updated version of TagMe [7]. We report the results for two versions of our system. The first one, denoted as *REL (2014)*, is based on the original implementation of [15] for ED. It uses Wikipedia 2014 as the reference knowledge base and employs entity embeddings provided by [8] for candidate selection. The second version of our system, denoted as *REL (2019)*, is based on Wikipedia 2019-07 and uses Wikipedia2Vec embeddings; cf. Section 3.

Table 3: Local ED results as reported in [15]

Micro F1	AIDA-B	ACE2004	Aquaint	CLUEWEB	MSNBC	Wikipedia
MulRel-NEL [15]	93.1	89.9	88.3	77.5	93.9	78.0
REL (2014)	92.8	89.7	87.4	77.6	93.5	78.7
REL (2019)	89.4	85.3	84.1	71.9	90.7	73.1

Table 4: Efficiency of REL (in seconds) for 50 documents from AIDA-B with > 200 words, which is 323 (± 105) words and 42 (± 19) mentions per document.

	Time MD	Time ED
With GPU	0.44±0.22	0.24±0.08
Without GPU	2.41±1.24	0.18±0.09

We use the GERBIL platform [24] for evaluation, and report on micro and macro InKB F1 scores for both EL and ED. Table 1 shows the strong matching results for EL, where strong refers to the requirement of exactly predicting the gold mention boundaries. We first note that REL outperforms the well-established entity linking toolkits (DBpedia Spotlight and WAT) by a large margin. Comparing with SOTA NLP, we observe that REL (2019) outperforms (or performs on par with) SOTA NLP on half of the datasets. The ED results in Table 2 also show consistent and significant improvements of REL over the two well-established toolkits. SOTA NLP, however, obtains better results than REL for all, except three datasets. For both EL and ED results, we observe that REL (2014) achieves better results compared to REL (2019). This can be attributed to the different embeddings used for candidate selection: the recall of candidate entities chosen by their similarity to the context of the mentions is lower in REL (2019) when compared to REL (2014).

For a reference comparison, we also report the results of the ED method (referred to as MulRel-NEL) as reported in [15]; see Table 3. The micro F1 score reported in this table is computed locally and by matching ED results against the original datasets. The results show that REL (2014) and MulRel-NEL scores are almost identical, which attests to the repeatability of [15]. Again, we observe a decrease in performance when comparing REL (2019) to REL (2014), just like in Table 2.

Finally, we report on the runtime efficiency of REL in Table 4. Specifically, we measure efficiency on a random sample of 50 documents (with a minimum length of 200 words) taken from AIDA-B. The experiments were run on a laptop with Intel i7 CPU (2.80GHz), 16GB RAM, and an NVIDIA Geforce GTX 1050 (4GB) GPU. The results show that detecting the mentions takes considerably more time than ED, and is done more efficiently using GPU. The ED time, however, is less affected by the GPU usage. This indicates that the overall efficiency of REL can be improved by replacing MD with a more efficient NER approach.

5 CONCLUSION

We have introduced the Radboud Entity Linker (REL), an open source toolkit for entity linking. REL builds on state-of-the-art neural components from natural language processing research, and is provided as a Python package and as a web API. Currently, REL is optimized for annotating documents and short texts. In the future, we plan to train REL on a large corpus of annotated queries and make it available for the task of entity linking in queries as well.

REFERENCES

[1] Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual String Embeddings for Sequence Labeling. In *Proc. of COLING '18.* 1638–1649.
[2] Krisztian Balog. 2018. *Entity-Oriented Search.* The Information Retrieval Series, Vol. 39. Springer.
[3] Krisztian Balog, Heri Ramampiaro, Naimdjon Takhirov, and Kjetil Nørvåg. 2013. Multi-step Classification Approaches to Cumulative Citation Recommendation. In *Proc. of OAIR '13.* 121–128.
[4] Razvan Bunescu and Marius Paşca. 2006. Using Encyclopedic Knowledge for Named Entity Disambiguation. In *Proc. of EACL '06.* 9–16.
[5] Marco Cornolti, Paolo Ferragina, Massimiliano Ciaramita, Stefan Rüd, and Hinrich Schütze. 2018. SMAPH: A Piggyback Approach for Entity-Linking in Web Queries. *ACM Trans. Inf. Syst.* 37, 1 (2018).
[6] Silviu Cucerzan. 2007. Large-Scale Named Entity Disambiguation Based on Wikipedia Data. In *Proc. of EMNLP-CoNLL '07.* 708–716.
[7] Paolo Ferragina and Ugo Scaiella. 2010. TAGME: On-the-fly Annotation of Short Text Fragments (by Wikipedia Entities). In *Proc. of CIKM '10.* 1625–1628.
[8] Octavian-Eugen Ganea and Thomas Hofmann. 2017. Deep Joint Entity Disambiguation with Local Neural Attention. In *Proc. of EMNLP '17.* 2619–2629.
[9] Emma Gerritse, Faegheh Hasibi, and Arjen P. de Vries. Graph-Embedding Empowered Entity Retrieval. In *Proc. of ECIR '20.* 97–110.
[10] Faegheh Hasibi, Krisztian Balog, and Svein Erik Bratsberg. 2015. Entity Linking in Queries: Tasks and Evaluation. In *Proc. of ICTIR '15.* 171–180.
[11] Faegheh Hasibi, Krisztian Balog, and Svein Erik Bratsberg. 2016. Exploiting Entity Linking in Queries for Entity Retrieval. In *Proc. of ICTIR '16.* 209–218.
[12] Faegheh Hasibi, Krisztian Balog, Darío Garigliotti, and Shuo Zhang. 2017. Nordlys: A Toolkit for Entity-Oriented and Semantic Search. In *Proc. of SIGIR '17.* 1289–1292.
[13] Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust Disambiguation of Named Entities in Text. In *Proc. of EMNLP '11.* 782–792.
[14] Nikolaos Kolitsas, Octavian-Eugen Ganea, and Thomas Hofmann. 2018. End-to-End Neural Entity Linking. In *Proc. of CoNLL '18.* 519–529.
[15] Phong Le and Ivan Titov. 2018. Improving Entity Linking by Modeling Latent Relations between Mentions. In *Proc. of ACL '18.* 1595–1604.
[16] Pablo N Mendes, Max Jakob, Andrés García-Silva, and Christian Bizer. 2011. DBpedia Spotlight: Shedding Light on the Web of Documents. In *Proc. of I-Semantics '11.* 1–8.
[17] Rada Mihalcea and Andras Csomai. 2007. Wikify! - Linking Documents to Encyclopedic Knowledge. In *Proc. of CIKM '07.* 233–242.
[18] David Milne and Ian H Witten. 2008. Learning to Link with Wikipedia. In *Proc. of CIKM '08.* 509–518.
[19] Aasish Pappu, Roi Blanco, Yashar Mehdad, Amanda Stent, and Kapil Thadani. 2017. Lightweight Multilingual Entity Extraction and Linking. In *Proc. of WSDM '17.* 365–374.
[20] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global Vectors for Word Representation. In *Proc. of EMNLP '14.* 1532–1543.
[21] Francesco Piccinno, Paolo Ferragina, and Dipartimento Informatica. 2014. From TagME to WAT : a new Entity Annotator Categories and Subject Descriptors. (2014), 55–62.
[22] John Platt. 2000. Probabilities for SV Machines. In *Advances in Large-Margin Classifiers.* 61–73.
[23] Ridho Reinanda, Edgar Meij, and Maarten de Rijke. 2015. Mining, Ranking and Recommending Entity Aspects. In *Proc. of SIGIR '15.* 263–272.
[24] Michael Röder, Ricardo Usbeck, and Axel-Cyrille Ngonga Ngomo. 2018. GERBIL–Benchmarking Named Entity Recognition and Linking Consistently. *Semantic Web* 9, 5 (2018), 605–625.
[25] Valentin I. Spitkovsky and Angel X. Chang. 2012. A Cross-Lingual Dictionary for English Wikipedia Concepts. In *Proc. of LREC'12.* 3168–3175.
[26] Chenyan Xiong, Jamie Callan, and Tie-Yan Liu. 2017. Word-Entity Duet Representations for Document Ranking. In *Proc. of SIGIR '17.* 763–772.
[27] Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. 2018. Wikipedia2Vec: An Optimized Tool for Learning Embeddings of Words and Entities from Wikipedia. *arXiv preprint 1812.06280* (2018).
[28] Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. 2016. Joint learning of the embedding of words and entities for named entity disambiguation. In *Proc of CoNLL '16.* 250–259.