Local Entity Disambiguation for Web Fragments using Attentional RNNs

Anonymous ACL submission

Abstract

Hello, My name is. what? my name is. who? My name is. what? my name is. who? My name is. what? my name is. who? My name is. what? my name is. who? My name is. what? my name is. who? My name is. what? my name is. who? My name is. what? my name is. who? My name is. what? my name is. who? My name is. what? my name is. who? My name is. what? my name is. who? My name is. what? my name is. who? My name is. what? my name is. who? My name is. what? my name is. who? My name is. what? my name is. who? Lorem ipsum.

1 Introduction

Named Entity Disambiguation (NED) is the task of disambiguating mentions within a fragment of text against a given knowledge base of entities, and linking each mention to its intended entity. It has long been recognized as an important downstream task for many NLP problems such as text categorization (Gabrilovich and Markovitch, 2007) and information retrieval (Dalton et al., 2014).

NED algorithms can be broadly divided into local and global approaches where local algorithms disambiguate mentions by their local context independently from the rest of their parent document, while global approaches assume some affinity among mentions from the same text. This property is typically used for learning a coherency model that enables to simultaneously disambiguate all document related mentions relative to their parent text. Much attention was given recently to global algorithms (Ratinov et al., 2011; Guo and Barbosa, 2014; Pershina et al., 2015) ¡- there's a ton of newer stuff which demonstrate very good performance in many cases of long and coherent documents. However, many entity disambiguation examples do not present a clean and coherent frame text, but rather are given as short and noisy fragments. For example, short textual segments taken from web pages, tweets and other social media data are less appropriate for global disambiguation as they typically contain very messy and erratic content.

Recent applications of Deep Learning have demonstrated Deep Neural Network (DNN) architectures that achieve state of the art results on a wide variety of tasks from image classification (Krizhevsky et al., 2012) to machine translation (Bahdanau et al., 2014). When provided with large training datasets and sufficient computational resources DNNs can learn powerful data representations and model complex feature interactions. We present a Deep Learning based local disambiguation algorithm where Deep Attentional Recurrent Neural Networks (RNN) are trained to rank entities based on a noisy context using large quantities of crowd-sourced data. RNNs are powerful language modeling architectures which we supplement with an attention mechanism that allows the model to learn which parts of context are useful for disambiguation. We incorporate the trained neural model into a full NED system to demonstrate its performance.

We craft a large-scale dataset of short web fragments containing mentions disambiguated into Wikipedia by thousands of website editors. Our dataset is based on a subset of the Wikilinks dataset (Singh et al., 2012) collected by Singh at el. at Google. We demonstrate our model greatly outperforms existing state-of-the-art NED algorithms on this large and noisy data. In addition we examine our algorithm on CoNLL (Hoffart et al.,

2011), a standard NED dataset and show results comparable to other state-out-the-art methods on a smaller and cleaner dataset.

2 Methodology

In this section we introduce our model design, which is founded on an Attentional RNN for context modeling ,and we demonstrate the integration of this component in a complete disambiguation framework. The core concept of this model is to transform the input, which is the mention's context and a single candidate entity, into a feature vector which is in turn used for computing a ranking score for the given entity. We begin by describing the Attentional RNN model. Then, we explain how to integrate this model into a larger DNN that is trained and then used for computing a ranking score. Eventually we detail how to use this model in an overall disambiguation framework.

2.1 Attentional RNN model

RNNs are state of the art.....

The challenge of modeling context is commonly addressed using word embeddings () add all refs from Malmud2014. Yet, in practice many of these studies refer the entire sentence as a simple collection or weighted average of its individual word embeddings, as they completely ignore its sequential structure. RNN are powerful models for learning the representation of wider contextual environments, which enables intelligent embedding of a sentence by gradually passing through its content and predicting some output using the system's history. In practice, it has been shown that copies of the context, which are opposite in direction, can be independently fed into a bidirectional RNN in order to improve prediction performance (**?**; **?**). attention mechanisms are state of the art ....

We first focus on the mechanics of our Attentional RNN component as depicted in Figure 1. The Attentional RNN takes a window of words context as input and transforms it into a fixed length vector of features. It requires an embedded vector representation for a candidate entity to control its attention as well. This component is plugged into a larger DNN model and is trained by SGD and back-propagation along with the larger model. During the training process it learns to extract a useful feature representation of the context (given the candidate entity under consideration) for optimizing the loss function defined by the larger model. In our case predicting a ranking score for the candidate entity given the context. The basic component of our model is a standard RNN unit to process the context input. In our implementation we used standard GRU units (Cho et al., 2014) as these are state-of-the-art RNN models but any other RNN unit can be used as a drop-in replacement.

$$\begin{aligned} h_t &= f_{\Theta_1}(h_{t-1}, v_t) \\ o_t &= g_{\Theta_2}(h_t) \end{aligned} \tag{1}$$

Equation 1 represents the general semantics of an RNN unit. An RNN reads a sequence of vectors $\{v_t\}$ and maintains a hidden state vector $\{h_t\}$. At each step a new hidden state is computed from the previous hidden state and the next input vector by a function f parametrized by learnable parameters Θ_1. The output at each step is computed from the hidden state using a function g that is optionally parametrized by learnable parameters Θ_2.

A straight-forward use of an RNN in our model would be to train the RNN so that given a context as a sequence of word embedding vectors $\{v_t\}$, the last output o_t will be used to predict the ranking score of the candidate entity directly. However ??, ?? have shown attentional models are beneficial for NED which motivated us to incorporate a simple neural attention model. The attention model takes the entire sequence of intermediate output vectors of the RNN $\{o_t\}$ and computes a weighted average of the vectors governed by the equations shown in equation 2

$$\begin{aligned} &a_t \in \mathbb{R}; a_t = r_{\Theta_3}(o_t, c) \\ &a'_t = \frac{1}{\sum_{i=1}^{t} \exp\{a_t\}} \exp\{a_t\} \\ &o_{attn} = \sum_{i=1}^{t} a'_t o_t \end{aligned} \tag{2}$$

The main component in equation 2 is the function r_{Θ_3} which is parametrized by Θ_3 and computes an attention value for each word based on a controller c and the output of the RNN at step t. We then use a softmax to squeeze the attention values such that $\sum_{i=1}^{t} a'_i = 1$ and compute o_{attn}, the output of the attention model. The attention controller in our case is an embedding vector of the candidate entity itself. This allows the attention mechanism to decide how much individual words are important for ranking a given candidate. Our attention controller c is an embedding vector

of the candidate entity and our attention function r is parametrized as a trainable single layer NN as shown in equation 3

$$r_{\Theta_3}(o_t, c) = Ao_t + Bc \quad (3)$$

2.2 Neural Network framework details and training regime AND MIXED NOTES

Our attentional RNN is incorporated in the NN framework as depicted in figure ???. We incorporate two separate RNNs, one for the left context and one for the right context. Note we feed the right context in reverse into RNN. Each RNN outputs a vector and these two vectors are concatenated along with the candidate entity embedding and are fed into a ranking DNN with a single hidden layer of size 300 and a relu non-linearity function, followed by an output layer with 2 neurons and a softmax non linearity. We train the neural network to optimize a cross entropy loss function where the first output neuron is set to predict the probability of the candidate entity being the correct one, and the second output neuron predicts the probability of the candidate entity not being the correct one. We have examined using a more conventional setting with a single output neuron trained using least squares loss, however this setting was found harder to train. We have implemented the entire model using the excellent Keras (Chollet, 2015) over Theano (Team et al., 2016) framework.

2.3 Training word and entity embeddings

Training our model implicitly trains its dictionaries of both word and entities by error back-propagation, however we have found that using pre-trained embeddings as a starting point improve model performance. To this end we have devised a Skip Gram with Negative Sampling (SGNS) (Mikolov et al., 2013) based training procedure that simultaneously trains both word and entity vectors in the same embedded space. We note Yamada at el. (Yamada et al., 2016) have devised a somewhat similar and very successful method however they optimizes directly for disambiguation performance while we optimize for a simple, fast and plausible starting point for our deep model.

We use the word2vecf library by (Levy and Goldberg, 2014a) that is adapted from the original word2vec code[1] but allows to train directly on a dataset made of $(word, context)$ pairs rather then a textual corpus in a long string format. We exploit this and redefine the notion of $context$ from a context word to a context entity id. Specifically we take the corpus of Wikipedia pages as our knowledge base of entities and create a dataset where each page identified by the internal $pageid$ in Wikipedia database is scanned and split into a list of words $\{word_i\}$. We then add the pair $(word_i, pageid)$ for each word in each page in Wikipedia.

As shown by Levy and Goldberg (Levy and Goldberg, 2014b) training embeddings on this dataset using SGNS produces word and entity embedding that implicitly factorize the PPMI matrix of word-entity co-occurrence matrix. This well-known matrix is closely related to the TFIDF word-entity matrix used by (Gabrilovich and Markovitch, 2007) in Explicit Semantic Analysis and was found useful in a wide array of NLP tasks.

Prior to training we filtered words and documents (entities) appearing less then 20 in the dataset. We trained embeddings of length 300 for 10 iterations over the dataset. We used default values for all other parameters.

what was the limit i used for minimum length of articles/minimum freq of words?

words are close to entities they are indicative of. Words appearing in similar entity pages are grouped together. Entities with similar pages are grouped together

show results of the analogies experiment we did indicating semantic structure for the WORD vectors

3 Web-Fragment based NED Dataset

We introduce a new large scale NED dataset for web-fragment crawled from the web. Our dataset is derived from the Wikilinks dataset originally collected at Google by Singh at el. (Singh et al., 2012) for a cross-document co-reference task. cross-document co-reference entails clustering mentions referring to the same entity across a set of documents without consulting a predefined knowledge base of entities, and is many-a-time regarded as a downstream task for knowledge base population (KBP). Wikilinks is constructed from crawling websites and collecting hyperlinks into Wikipedia and the web context they appear

[1] Available at https://code.google.com/archive/p/word2vec/

in. These hyperlinks are assumed to be ground-truths for linking mentions (hyperlink anchor-text) to distinct entities (Wikipedia pages). Wikilinks contains 40 million mentions covering 3 million entities collected from over 10 million web pages.

Despite the difference between a NED and a co-reference task, we believe the nature of Wikilinks makes it directly applicable to NED as well. It can be seen as a large-scale, naturally-occurring and crowd-sourced dataset where thousands of human annotators provide ground truth for mentions of interest. Its web sourcing entails an added interest since it contains every kind of noise expected from automatically gathered web content, including many faulty, misleading and peculiar ground truth labels on the one hand, and on the other hand noisy, malformed and incoherent textual context for mentions.

While noise in crowd-sourced data is arguably a necessary trade-off for quantity, we believe the contextual noise represents an interesting test case supplementing existing standard datasets for NED such as CoNLL (Hoffart et al., 2011) and ACE and Wiki (Ratinov et al., 2011) as these are sourced from mostly coherent and well formed text such as news articles and Wikipedia pages. Such a test case emphasize utilizing strong and adaptive local disambiguation techniques, and marginalizes the utility of coherency based global approaches.

elaborate on intralink dataset

3.1 Preprocessing the Wikilinks dataset

In order to utilize the Wikilinks dataset for NED we have first performed a number of filtering and preprocessing steps. Since we have opted for focusing on local disambiguation we choose a version of Wikilinks with only local contexts for mentions and URLs to the containing page rather then the much larger version with full pages [2]. This sums to around 5Gb of compressed data (compared to 180Gb for the full texts).

The first preprocessing step we took was resolving ground-truth links using a $7/4/2016$ dump of the Wikipedia database [3]. The same dump was consistently used throughout this research. We used the $page$ and $redirect$ tables for resolution and kept the database id as a unique identifier for Wikipedia pages (entities). We discarded mentions where the ground-truth could not be resolved. This resulted in retaining over 97% of the mentions. We then permuted the order of mentions within the data and split it into train, evaluation and test set. We split the data $90\%/10\%/10\%$ respectively. Since websites might include duplicate or closely related content we did not assign mentions into splits on an individual base but rather collected all origin domains and assigned each domain along with all mentions collected from it into a split collectively.

The last and most important task was filtering the data. We first counted how many times each mention refers to each entity: $\#\{e|m\}$ and calculated the prior conditional-probability of an entity given a mention: $p(e|m) = \#\{e|m\}/\sum_{e'} \#\{e'|m\}$. Examining these distributions revealed most mentions either had very little ambiguity or appeared very few times and had a number of ground truths each appearing just a handful of times. We have deemed unambiguous mentions to be not-interesting and scarce mentions to be suspected as noise with a high probability and designed a procedure to filter both this cases. We therefore decided to retain only mentions for whom at least two ground-truth entities have $\#\{e|m\} \geq 10$ and $p(e|m) \geq 0.1$.

This procedure aggressively filtered the dataset and we were left with $2.6M$ training, $300K$ test and $300K$ evaluation samples. We believe that doing so filters uninteresting cases while emitting a dataset that is large scale yet manageable in size for research purposes. We note that we have considered filtering $(mention, entity)$ pairs where $\#\{entity|mention\} \leq 3$ since these are suspected as additional noise however decided against this procedure as it might filter out long tail entities, a case which was deemed interesting by the community.

4 Experiments

4.1 Training details

We now describe our setup for evaluating the proposed model for the task of NED. Following the motivation for disambiguating the Wikilinks data set, which was explained in section ReferencesWeb-Fragment based NED Dataset, we trained our model on $2.6M$ mentions. This procedure was carried out with a 20-core CPU machine during XX days add description of the machine and timing.

[2] Both available at http://www.iesl.cs.umass.edu/data/wiki-links

[3] Recent Wikipedia dumps are found at https://dumps.wikimedia.org/

Prior to the training we generated a basic counts/statistics file of mentions and entities from our corpus in a statistics file, for the creation of additional features and filtering mentions. [What else?]. Specifically, it held a mention-count, an entity-count, a mention-links with the distribution over the possible candidates and a count for all words appearing in the context of a mention.

Due to framework limitations we have trained our RNNs with a fixed size left and right contexts. These where set a 20 word window to each side of the mention. When the context is shorter on either side, we pad it with a special PAD symbol. We use a stop-word list to filter less informative stop-words.

The input in the model was streamed as pairs of entity-candidate. Following Mikolov at el. (Mikolov et al., 2013) we used a Negative Sampling approach where for each positive example the network is trained on, it is also trained on k negative samples. In this framework, the context remains the same , while incorrect candidates are randomly sampled from all possible entities. We deviate from Mikolov , which picks negative examples according to their distribution in the training corpus, by uniform sampling from all possible entities . double check - In uniform sampling the ratio of positive to negative samples for each entity is proportional to the prior probability of the entity within the corpus, which bias the network toward ranking common entities higher.

The optimization of the model was carried out using standard back propagation and AdaGrad optimizer(Duchi et al., 2011). We allowed the error to propagate through all parts of the network and fine tune all trainable parameters, including fine-tuning the word and entity embeddings themselves. The scale of the data enabled us to run only 1 epoch over the data.

CoNLL 2003 NER data is a an evaluation that is commonly used in research for benchmarking NED solutions (**?**; **?**; **?**; **?**). This has driven us to hold additional models, trained on 18505 non-NIL mentions from the CoNLLs training set, and evaluated on test-b set with 4485 cases.

CoNLL The CoNLL corpus is a public, free and annotated dataset, which is compromised out of a relatively large amount of news articles written in 1996 by the Reuters newswire. It shares 1393 documents from two time periods of overall 12 days, which are split into train, development and test set according to $68 - 15 - 17\%$. The performance on the corpus is usually measured using micro accuracy (p@1), which is a the average over the label of the mentions, having the golden-sense in their candidate set. While disregarding the performance of the candidate generation process, this measure can be delusive when the recall of the candidate generation is low. Therefore, it is common to use candidate generation resources, such as YAGO (Hoffart et al., 2011) or PPRforNED (**?**), with over 99% candidate recall for better comparability of the results. Another broadly used measure is the macro accuracy, which averages the precision of all documents.Typical and recent p@1 results of local disambiguation approaches on the test-b dataset have ranged between 80% to 89% (see table **??**), when very recently the 90% barrier was broken by Yamada et al.(**?**). Results on CoNLL appear sometimes in literature under the AIDA system, which is an online disambiguation system offered by Hoffart that contains the CoNLL 2003 NER task.

4.2 Learning to rank

We had two kinds of models trained on the CoNLL corpus. Initially, we used the pure Attentional RNN architecture, trained over 20% is this right? of the intralinks corpus with fine-tuning. Inspired by Yamada's approach, we extended our model to be part of a parent Gradient Boosting Regression Tree (GBRT) classifier that is known for demonstrating neat performance in ranking applications (**?**). For every data fragment, we took the preceding model's prediction, which is the probability that a candidate suits the input mention given the context, and fed it into a feature vector. In this manner we generated an input vector for every pair of mention-candidate from the CoNLL training set and let the model iterate over the entire data. Practically, we used sklearn's GradientBoostingClassifier implementation (**?**) with parameters similar to those reported by Yamada. We trained with the logistic-regression deviance and set the learning rate, number of estimator and maximum depth of a tree to 0.02, 10000 and 4, respectively. For the CoNLL experiments we also added Yamada's base and string similarity features into the input vector. We find it important to note that in contrast to many other studies, we didn't focus on surpassing current state of the art results on the CoNLL task, as it is a precisely edited and specific newspaper

corpus with stories covering merely two weeks of reports (**?**).

4.3 Count and string based features Should appear as a distinct section?

Besides of embeddings of the context and entities' the model's input consisted several more features. We implemented the base features that were introduced by Yamada using statistics from Wikipedia that were computed prior to the training. In this manner we obtained the conditional and marginal entity prior, $P(e|m)$ and $P(e)$, which are normalized measures giving the number of times entity e was linked to a specific or to any mention m in the dataset. Information about the source document was captured by the max prior feature that is takes the maximum prior probability of

4.4 Comparison with other methods

- Yamada et al. (**?**)
- Cheng et al. (**?**)

4.5 Results

** metrics

4.6 Feature study

5 Conclusions

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Franois Chollet. 2015. keras. `https://github.com/fchollet/keras`.

Jeffrey Dalton, Laura Dietz, and James Allan. 2014. Entity query feature expansion using knowledge base links. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pages 365–374. ACM.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159.

Evgeniy Gabrilovich and Shaul Markovitch. 2007. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *IJcAI*, volume 7, pages 1606–1611.

Zhaochen Guo and Denilson Barbosa. 2014. Entity linking with a unified semantic representation. In *Proceedings of the 23rd International Conference on World Wide Web*, pages 1305–1310. ACM.

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 *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 782–792. Association for Computational Linguistics.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Omer Levy and Yoav Goldberg. 2014a. Dependency-based word embeddings. In *ACL (2)*, pages 302–308.

Omer Levy and Yoav Goldberg. 2014b. Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems*, pages 2177–2185.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Maria Pershina, Yifan He, and Ralph Grishman. 2015. Personalized page rank for named entity disambiguation. In *Proc. 2015 Annual Conference of the North American Chapter of the ACL, NAACL HLT*, volume 14, pages 238–243.

Lev Ratinov, Dan Roth, Doug Downey, and Mike Anderson. 2011. Local and global algorithms for disambiguation to wikipedia. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1375–1384. Association for Computational Linguistics.

Sameer Singh, Amarnag Subramanya, Fernando Pereira, and Andrew McCallum. 2012. Wikilinks: A large-scale cross-document coreference corpus labeled via links to Wikipedia. Technical Report UM-CS-2012-015.

The Theano Development Team, Rami Al-Rfou, Guillaume Alain, Amjad Almahairi, Christof Angermueller, Dzmitry Bahdanau, Nicolas Ballas, Frédéric Bastien, Justin Bayer, Anatoly Belikov, et al. 2016. Theano: A python framework for fast computation of mathematical expressions. *arXiv preprint arXiv:1605.02688*.

Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. 2016. Joint learning of the embedding of words and entities for named entity disambiguation. *arXiv preprint arXiv:1601.01343*.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Franois Chollet. 2015. keras. `https://github.com/fchollet/keras`.

Jeffrey Dalton, Laura Dietz, and James Allan. 2014. Entity query feature expansion using knowledge base links. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pages 365–374. ACM.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159.

Evgeniy Gabrilovich and Shaul Markovitch. 2007. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *IJcAI*, volume 7, pages 1606–1611.

Zhaochen Guo and Denilson Barbosa. 2014. Entity linking with a unified semantic representation. In *Proceedings of the 23rd International Conference on World Wide Web*, pages 1305–1310. ACM.

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 *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 782–792. Association for Computational Linguistics.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Omer Levy and Yoav Goldberg. 2014a. Dependency-based word embeddings. In *ACL (2)*, pages 302–308.

Omer Levy and Yoav Goldberg. 2014b. Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems*, pages 2177–2185.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Maria Pershina, Yifan He, and Ralph Grishman. 2015. Personalized page rank for named entity disambiguation. In *Proc. 2015 Annual Conference of the North American Chapter of the ACL, NAACL HLT*, volume 14, pages 238–243.

Lev Ratinov, Dan Roth, Doug Downey, and Mike Anderson. 2011. Local and global algorithms for disambiguation to wikipedia. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1375–1384. Association for Computational Linguistics.

Sameer Singh, Amarnag Subramanya, Fernando Pereira, and Andrew McCallum. 2012. Wikilinks: A large-scale cross-document coreference corpus labeled via links to Wikipedia. Technical Report UM-CS-2012-015.

The Theano Development Team, Rami Al-Rfou, Guillaume Alain, Amjad Almahairi, Christof Angermueller, Dzmitry Bahdanau, Nicolas Ballas, Frédéric Bastien, Justin Bayer, Anatoly Belikov, et al. 2016. Theano: A python framework for fast computation of mathematical expressions. *arXiv preprint arXiv:1605.02688*.

Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. 2016. Joint learning of the embedding of words and entities for named entity disambiguation. *arXiv preprint arXiv:1601.01343*.