Using Attentional RNNs for Local Named Entity Disambiguation

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 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 broadly be divided into local and global approaches, where local algorithms disambiguate mentions independently using local textual context while global approaches assume some coherence among mentions within a single document and therefore employ a coherency model to simultaneously disambiguate many mentions. Much attention was given recently to global algorithms (Ratinov et al., 2011a; Guo and Barbosa, 2014; Pershina et al., 2015) (there's newer stuff) which demonstrate very good performance on standard datasets. However, while most standard datasets are based on text corpora such as news reports and Wikipedia paragraphs that are indeed relatively well formed and coherent, other corpora such as fragments of web pages can be noisier and less coherent.

Deep Neural Networks (DNN) have recently gained traction as state-of-the-art architectures that can model powerful data representations and complex feature interaction. DNNs achieve state-of-the-art results on a wide variety of tasks ranging from image classification (Krizhevsky et al., 2012) to machine translation (Bahdanau et al., 2014). For the task of NED, He at el. (2013a) used stacked auto-encoders to learn entity and context representations. More recently Sun at el. and Francis-Landau at el. (2015; 2016a) have proposed models that use convolutional neural networks for modeling and combining representations of a range of input signals and granularities.

We propose a novel Deep-Learning based disambiguation algorithm where NED is treated as separating correct entity assignments from corrupt ones given only a local context. Our model is based on Recurrent Neural Networks (RNNs) since they are state-of-the-art language modeling architectures. inspired by Globerson at el (Globerson et al., 2016) that showed an attention model is beneficial for NED, we develop a neural attention mechanism to enable the model to decide which contextual signals are most important when considering a specific entity assignment. We as well describe a novel method for initializing word and entity embeddings used in our model and demonstrate its importance.

To test our method we craft a large-scale dataset of short web fragments containing mentions disambiguated into Wikipedia by thousands of website editors. Our dataset is extracted from a subset of the Wikilinks dataset (Singh et al., 2012) which is re-purposed for NED. It represents a large-scale

and difficult task with count unique mentions disambiguated into $100K$ unique entities and where context is limited and noisy. We demonstrate our model greatly outperforms existing state-of-the-art NED algorithms on this dataset and in addition 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

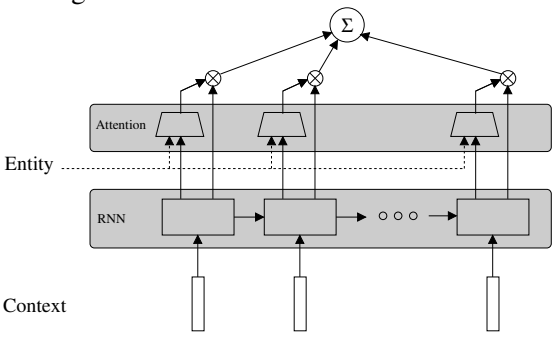
Our DNN model is a discriminative model which takes a pair of local context and candidate entity, and outputs a likelihood for the candidate entity being correct. Both words and entities are represented using embedding dictionaries and we interpret local context as a window-of-words to the left and right of a mention. The left and right contexts are fed into a duo of Attentional RNNs (ARRN) components which process each side and produce a fixed length vector representation. The left context is fed in a forward manner while the right context is fed backwards into the model. Each Attentional RNN uses the candidate entity input to control its attention, allowing it to attend to the most discriminating parts of the context given the candidate at hand.

The output vectors generated by both Attentional RNNs and the embedding of the entity itself are then fed into a classifier network consisting of a hidden layer and an output layer with two output units and a softmax activation. The output units are trained to emit the likelihood of the candidate being a correct or corrupt assignment by optimizing a cross-entropy loss function.

We assume our model is only given examples of correct entity assignments during training and therefore automatically generate examples of corrupt assignments. For each $(context, entity)$ pair where $entity$ is a correct assignment for a given $contex$ we produce k corrupt examples with the same $context$ and a corrupt entity uniformly sampled from all entities in the dataset. Using the combined dataset of correct and corrupt examples our algorithm learns to separate correct assignments from the generated corrupt ones.

In our implementation we have set the hidden layer size to be 300 and used a ReLU non-linearity for this layer. Preliminary evaluations showed the width and depth of the classifier to be of little impact on performance, but using a ReLU non-linearity was found to be important. We have also applied dropout with $p = 0.5$ to the hidden layer.

Figure 1: Attentional RNN Architecture



2.1 Attentional RNN component

Our Attentional RNN component is based on a general RNN unit fitted with an attention mechanism. The mechanics of the Attentional RNN component are depicted in Figure 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 based on the previous hidden state and the next input vector by a function f parametrized by Θ_1. The output at each step is computed from the hidden state using a function g parametrized by Θ_2. This allows the RNN to 'remember' important signals while scanning the context and to recognize signals spanning multiple words.

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

In out implementation we have used a standard GRU unit (Cho et al., 2014), however any RNN can be a drop-in replacement. While an RNN unit can be used as-is in our model by feeding the last output vector o_t directly into the classifier network, we have implemented an attention mechanism that allows the model to be aware of the candidate entity it is evaluating when computing an output. Equation 2 details the equations governing the attention model.

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

The main component in equation 2 is the function r, parametrized by Θ_3, which computes an attention value at each step using $v_{candidate}$, the candidate entity embedding, as a control signal. We use the softmax function to normalize the attention values such that $\sum_{i=1}^{t} a'_i = 1$ and compute the final output o_{attn} as a weighted sum of all the output vectors of the RNN. This allows the attention mechanism to decide on the importance of different context parts when examining a specific candidate. We parametrize our attention function r as a single layer NN as shown in equation 3 where A, B are the layer weights and b is a bias term.

$$r_{\Theta_3}(o_t, v_{candidate}) = Ao_t + Bv_{candidate} + b \quad (3)$$

2.2 Training initial word and entity embeddings

Training our model implicitly trains its dictionaries of both word and entity embedding by error back-propagation. However, as will be shown in section 4, we have found using pre-trained embeddings to significantly improve model performance/greatly reduce training time(??). 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 use the word2vecf library[1] by Levy and Goldberg (2014a) that is adapted from word2vec code and allows to train on a dataset made of $(word, context)$ pairs rather then a textual corpus in string format, as is done in the original word2vec. We exploit this to redefine $context$ as a context entity rather then a contextual word.

We do this by considering each page in Wikipedia to represent a unique entity, enumerated by the $pageid$ identifier in Wikipedia database and having a textual description (the page itself). For each word $\{word_i\}$ in the page we add the pair $(word_i, pageid)$ to our dataset. We however limit our vocabularies by ignoring both rare words that appear less then 20 times and entities that have less then 20 words in their description.

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

For our experiments we trained embeddings of length 300 for 10 iterations over the dataset. We used default values for all other parameters in word2vec.

-needs developing-

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

3 Creating a Web-Fragment based NED Dataset

We introduce a new large-scale NED dataset of web-fragments crawled from the web. Our dataset is derived from the Wikilinks dataset originally collected by Singh at el. (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 was constructed by crawling the web and collecting hyperlinks linking to Wikipedia and the web context they appear in. The anchor texts act as mentions and the link targets in Wikipedia act as ground-truths. Wikilinks contains 40 million mentions covering 3 million entities and collected from over 10 million web pages.

Wikilinks can be seen as a large-scale, naturally-occurring and crowd-sourced dataset where thousands of human annotators provide ground-truths for mentions of interest. Its web sourcing entails 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 in particular represents an interesting test-case that supplements existing standard datasets such as CoNLL (Hoffart et al., 2011), ACE and Wiki (Ratinov et al., 2011a) as these are all sourced from mostly coherent and well formed text such as news articles and Wikipedia pages. Wikilinks emphasizes utiliz-

[1] Available at https://bitbucket.org/yoavgo/word2vecf

ing strong and adaptive local disambiguation techniques, and marginalizes the utility of coherency based global approaches.

The original dataset exists in a number of formats, and we have chosen a version with only short local contexts[2] since it renders the size of the dataset a manageable 5Gb of compressed data (compared to 180Gb for the full texts). Following are the filtering and preprocessing steps used to create a NED evaluation dataset from Wikilinks:

- we resolved 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 $pageid$ column as a unique identifier for Wikipedia pages (entities). To reduce loss of unresolved mentions due to malformed URLs we compared page names using a case-insensitive and normalized[4] title matching. We discarded mentions where the ground-truth could not be resolved, resulting in retention of 97% of the mentions.

- We collected all pairs of mention m and entity e appearing in the dataset and computed the following two statistics: how many times m refers to e: $\#\{e|m\}$ and the conditional probability of e given m: $p(e|m) = \#\{e|m\}/\sum_{e'}\#\{e'|m\}$. Examining these distributions revealed many mentions belong to two extremes: either they had very little ambiguity or had a number of candidate entities each appearing very few times. We have deemed the former to be unambiguous and not-interesting, and the latter to be suspected as noise with high probability. We therefore designed a procedure to filter both this cases: We retained only mentions for whom at least two ground-truth entities have $\#\{e|m\} \geq 10$ and $p(e|m) \geq 0.1$.

- Finally, We randomly 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 basis but rather collected all origin domains and assigned each domain along with all mentions collected from it into a split collectively.

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 (m, e) pairs where $\#\{e|m\} \leq 3$ 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 Evaluation

In this section we describe the setup used when evaluating our model and present evaluation results for two datasets. We evaluate the effect of initializing word and entity embeddings on our model as well.

4.1 Wikilinks

Prior to evaluating our method on the Wikilinks dataset we have collected the following statistics from the Wikilinks training set: $P(e)$ is the prior probability of seeing entity e in the training set and $P(e|m)$ is the probability of seeing entity e as a ground-truth for mention m.

When evaluating a NED system it is required to use some method for generating candidate entities first. We use a simple method where given mention m, we considered all candidates for whom $P(e|m) > 0$ as candidates. This simple method gives 97% ground-truth recall on the test set. We used GRUs as our RNN unit and used fixed size left and right contexts, using a 20 word window to each side of the mention. In cases were the context was shorter than the fixed size, we padded it with a special PAD symbol. Further, we filtered out stop words according to NLTK's stop-word list. The optimization of the model was carried out using standard back propagation and an 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 the word and entity embeddings themselves. A single epoch

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

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

[4] Normalization was done using the unidecode python library

with $2.6M$ mentions and $k = 5$ for corrupt example generation was used for training the model taking half a day using a 20-core CPU machine.

We have used the following methods as base line on the Wikilink dataset:

- **Yamada et al.** (2016b) have created a state-of-the-art NED system for jointly mapping entities and words onto the same vector space via the skip-gram model and using these for disambiguation. As Wikilinks has only one mention per fragment we use only the local features described by Yamada at el.

- **Cheng et al.** (2013) addressed the Disambiguation to Wikipedia, or the "Wikificaiton" task, with a combination of local and global approaches. Mentions are disambiguated locally by generating a ranked list of candidates for each mention using the GLOW Wikification system proposed by Ratinov et al. (2011b). We compare our results to the ranking step of the algorithm, were a linear Ranking SVM is trained over a set of local and global features to return the list of candidates sorted by their likelihood.

- We include Most Probable Sense (MPS) as a baseline. This baseline picks the entity with the highest $P(e|m)$ as the correct mention. This simple baseline is notoriously known to give competitive results in many NED datasets

4.2 CoNLL

CoNLL is an evaluation corpus created by Hoffart et al. (2011) commonly used for benchmarking NED solutions (Globerson et al., 2016; Hachey et al., 2013; Yamada et al., 2016b; Pershina, 2015). CoNLL was composed by manually annotating Reuters newswire articles from 1996. It contains 1393 documents from a period of 12 days split into train, development and test sets. Following previous works we have only evaluated our method on non-NIL mentions. For candidate generation we used the publicly available candidate dataset by Pershina at el. (2015) with over 99% gold sense recall.

CoNLL has a training set with 18505 non-NIL mentions, which preliminary experiments showed is not sufficient to train our model on. We therefore resorted to a more complex training method where we first trained our model on a large corpus of mentions derived from Wikipedia cross references and then fine tuned the resulting model on CoNLL training set. To derive the Wikipedia training corpus we have extracted all cross-reference links from Wikipedia along with their context, resulting in over 80 million training examples. Due to constrained resources we set $k = 1$ for corrupt example generation and trained 1 epoch, which took around 4 days to train. The resulting model was then fine-tuned on CoNLL training set, where corrupt examples were produced by considering all possible candidates for each mention.

We have also found that using traditional statistical and string based features along with our model further improves its performance. We therefor used a setting similar to Yamada et al. (2016a) where a Gradient Boosted Regression Tree was fitted with our models prediction score as a feature along with 7 other statistical and string based features. The statistical features are prior probability $P(e)$ and conditional probability $P(e|m)$ as described above, along with a feature counting the number of candidates generated for the mention and a feature giving the maximum conditional probability of the entity for all mentions in the document. For string similarity features we used the edit distance between the mention and the entity title in Wikipedia, a feature indicating weather the mention is a prefix or postfix of the entity Wikipedia title and a feature indicating weather the Wikipedia entity title is a prefix or postfix of the mention. Following Yamada we used sklearn's GradientBoostingClassifier implementation (Pedregosa et al., 2011) with a deviance loss and set the learning rate, number of estimator and maximum depth of a tree to 0.02, 10000 and 4, respectively.

As a baseline we took the standard Most Probable Sense (MPS) prediction, which corresponds to the $\arg\max_{e \in E} P(e|m)$, where E is the group of all candidate entities. We also compare to the following papers - Lazic et al. (2015), Francis-Landau et al. (2016b), He et al. (2013b), Hoffart et al. (2011) and Chisholm et al. (2015b) ,as they are all strong local approaches and a good source for comparison.

4.3 Results

The micro and macro accuracy scores on CoNLL test-b are displayed in table 2.

Wikilinks test set	
Model	**Micro accuracy**
ARNN	**64.8**
GBRT: Base + ARNN features	**66.8**
Yamada et al.(partial)	59.8 **
Cheng et al.	*
Baseline (MPS)	55.9

Table 1: Evaluation on Web-Fragment data (Wikilinks)

CoNLL test-b		
Model	Micro accuracy	Macro accuracy
ARNN	87.3	88.6
GBRT: Base + ARNN features	86.9	89.1
Yamada et al. (partial)	90.9	92.4
Lazic et al.	86.4	-
Francis-Landau et al.	85.5	-
He et al.	84.82	83.37
Hoffart et al.	79.57	80.71
Chisholm et al.	88.7	-
Baseline (MPS)	77	77

Table 2: Evaluation on CoNLL. Bold font denotes the models offered in this study

add insights regarding the results and comparison

4.4 Model sensitivity

Noam: this should be extended elaborate on:
!! ESA embedding initialization
!! attention vs. no attention

Wikilinks test set	
Model	**Micro accuracy**
ARNN w/o ESA init.	61
ARNN w/ ESA init. w/o Attention	64.1
ARNN w/ ESA & Attention	64.8

Table 3: ARNN Model sensetivity

5 Conclusions

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

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

Xiao Cheng and Dan Roth. 2013. Relational Inference for Wikification. *Empirical Methods in Natural Language Processing*, (October):1787–1796.

Andrew Chisholm and Ben Hachey. 2015a. Entity disambiguation with web links. *Transactions of the Association for Computational Linguistics*, 3:145–156.

Andrew Chisholm and Ben Hachey. 2015b. Entity Disambiguation with Web Links. *Transactions of the Association for Computational Linguistics*, 3(0):145–156.

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.

Matthew Francis-Landau, Greg Durrett, and Dan Klein. 2016a. Capturing semantic similarity for entity linking with convolutional neural networks. *arXiv preprint arXiv:1604.00734*.

Matthew Francis-Landau, Greg Durrett, and Dan Klein. 2016b. Capturing Semantic Similarity for Entity Linking with Convolutional Neural Networks. pages 1256–1261.

Jerome H Friedman. 2001. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232.

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

Amir Globerson, Nevena Lazic, Soumen Chakrabarti, Amarnag Subramanya, Michael Ringgaard, and Fernando Pereira. 2016. Collective Entity Resolution with Multi-Focal Attention. *Acl*, pages 621–631.

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.

Ben Hachey, Will Radford, Joel Nothman, Matthew Honnibal, and James R. Curran. 2013. Evaluating entity linking with wikipedia. *Artificial Intelligence*, 194:130–150.

Zhengyan He, Shujie Liu, Mu Li, Ming Zhou, Longkai Zhang, and Houfeng Wang. 2013a. Learning entity representation for entity disambiguation. In *ACL (2)*, pages 30–34.

Zhengyan He, Shujie Liu, Mu Li, Ming Zhou, Longkai Zhang, and Houfeng Wang. 2013b. Learning Entity Representation for Entity Disambiguation. pages 30–34.

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.

Nevena Lazic, Amarnag Subramanya, Michael Ringgaard, and Fernando Pereira. 2015. Plato: A Selective Context Model for Entity Resolution. *Transactions of the Association for Computational Linguistics*, 3:503–515.

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.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(Oct):2825–2830.

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.

Maria Pershina. 2015. Personalized Page Rank for Named Entity Disambiguation. (Section 4):238–243.

Lev Ratinov, Dan Roth, Doug Downey, and Mike Anderson. 2011a. 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.

Lev Ratinov, Dan Roth, Doug Downey, and Mike Anderson. 2011b. Local and Global Algorithms for Disambiguation to Wikipedia. *Acl 2011*, 1:1375–1384.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. *CONLL '03 Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003*, 4:142–147.

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.

Yaming Sun, Lei Lin, Duyu Tang, Nan Yang, Zhenzhou Ji, and Xiaolong Wang. 2015. Modeling mention, context and entity with neural networks for entity disambiguation. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1333–1339.

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*.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Advances in Neural Information Processing Systems*, pages 2773–2781.

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

Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. 2016b. Joint Learning of the Embedding of Words and Entities for Named Entity Disambiguation. *arXiv preprint arXiv:1601.01343*, page 10.