Joint Learning of Named Entity Recognition and Entity Linking

Pedro Henrique Martins[Ψ] **Zita Marinho**[ϑ♏] and **André F.T. Martins**[Ψ♄]
[Ψ]Instituto de Telecomunicações [ϑ]Priberam Labs [♏]Institute of Systems and Robotics [♄]Unbabel
pedrohenriqueamartins@gmail.com, zita.marinho@priberam.pt,
andre.martins@unbabel.com.

Abstract

Named entity recognition (NER) and entity linking (EL) are two fundamentally related tasks, since in order to perform EL, first the mentions to entities have to be detected. However, most entity linking approaches disregard the mention detection part, assuming that the correct mentions have been previously detected. In this paper, we perform joint learning of NER and EL to leverage their relatedness and obtain a more robust and generalisable system. For that, we introduce a model inspired by the Stack-LSTM approach (Dyer et al., 2015). We observe that, in fact, doing multi-task learning of NER and EL improves the performance in both tasks when comparing with models trained with individual objectives. Furthermore, we achieve results competitive with the state-of-the-art in both NER and EL.

1 Introduction

In order to build high quality systems for complex natural language processing (NLP) tasks, it is useful to leverage the output information of lower level tasks, such as named entity recognition (NER) and entity linking (EL). Therefore NER and EL are two fundamental NLP tasks.

NER corresponds to the process of detecting mentions of named entities in a text and classifying them with predefined types such as person, location and organisation. However, the majority of the detected mentions can refer to different entities as in the example of Table 1, in which the mention "Leeds" can refer to "Leeds", the city, and "Leeds United A.F.C.", the football club. To solve this ambiguity EL is performed. It consists in determining to which entity a particular mention refers to, by assigning a knowledge base entity id.

In this example, the knowledge base id of the entity "Leeds United A.F.C." should be selected.

Leeds' Bowyer fined for part in fast-food fracas.		
	NER	EL
Separate	Leeds-ORG	Leeds
Joint	Leeds-ORG	Leeds_United_A.F.C.

Table 1: Example showing benefits of doing joint learning. Wrong entity in red and correct in green.

In real world applications, EL systems have to perform two tasks: mention detection or NER and entity disambiguation. However, most approaches have only focused on the latter, being the mentions that have to be disambiguated given.

In this work we do joint learning of NER and EL in order to leverage the information of both tasks at every decision. Furthermore, by having a flow of information between the computation of the representations used for NER and EL we are able to improve the model.

One example of the advantage of doing joint learning is showed in Table 1, in which the joint model is able to predict the correct entity, by knowing that the type predicted by NER is Organisation.

This paper introduces two main contributions:

- A system that jointly performs NER and EL, with competitive results in both tasks.
- A empirical qualitative analysis of the advantage of doing joint learning vs using separate models and of the influence of the different components to the result obtained.

2 Related work

The majority of NER systems treat the task has sequence labelling and model it using conditional random fields (CRFs) on top of hand-engineered features (Finkel et al., 2005) or bi-directional Long Short Term Memory Networks (LSTMs) (Lample

Action	Buffer	Stack	Output	Entity
	[Obama, met, Donald, Trump]	[]	[]	
Shift	[met, Donald, Trump]	[Obama]	[]	
Reduce-PER	[met, Donald, Trump]	[]	[(Obama)-PER]	Barack_Obama
Out	[Donald, Trump]	[]	[(Obama)-PER, met]	Barack_Obama
Shift	[Trump]	[Donald]	[(Obama)-PER, met]	Barack_Obama
Shift	[]	[Donald, Trump]	[(Obama)-PER, met]	Barack_Obama
Reduce-PER	[]	[]	[(Obama)-PER, met, (Donald Trump)-PER]	Barack_Obama, Donald_Trump

Table 2: Actions and stack states when processing sentence "Obama met Donald Trump". The predicted types and detected mentions are contained in the Output and the entities the mentions refer to in the Entity.

et al., 2016; Chiu and Nichols, 2016). Recently, NER systems have been achieving state-of-the-art results by using word contextual embeddings, obtained with language models (Peters et al., 2018; Devlin et al., 2018; Akbik et al., 2018).

Most EL systems discard mention detection, performing only entity disambiguation of previously detected mentions. Thus, in these cases the dependency between the two tasks is ignored. EL state-of-the-art methods often correspond to local methods which use as main features a candidate entity representation, a mention representation, and a representation of the mention's context (Sun et al., 2015; Yamada et al., 2016, 2017; Ganea and Hofmann, 2017). Recently, there has also been an increasing interest in attempting to improve EL performance by leveraging knowledge base information (Radhakrishnan et al., 2018) or by allying local and global features, using information about the neighbouring mentions and respective entities (Le and Titov, 2018; Cao et al., 2018; Yang et al., 2018). However, these approaches involve knowing the surrounding mentions which can be impractical in a real case because we might not have information about the following sentences. It also adds extraneous complexity that might implicate a longer time to process.

Some works, as in this paper, perform end-to-end EL trying to leverage the relatedness of mention detection or NER and EL, and obtained promising results. Kolitsas et al. (2018) proposed a model that performs mention detection instead of NER, not identifying the type of the detected mentions, as in our approach. Sil and Yates (2013), Luo et al. (2015), and Nguyen et al. (2016) introduced models that do joint learning of NER and EL using hand-engineered features. (Durrett and Klein, 2014) performed joint learning of entity typing, EL, and coreference using a structured CRF, also with hand-engineered features. In contrast, in our model we perform multi-task learning (Caruana, 1997; Evgeniou and Pontil, 2004), using learned features.

3 Model Description

In this section firstly, we briefly explain the Stack-LSTM (Dyer et al., 2015; Lample et al., 2016), model that inspired our system. Then we will give a detailed explanation of our modifications and of how we extended it to also perform EL, as showed in the diagram of Figure 1. An example of how the model processes a sentence can be viewed in Table 2.

3.1 Stack-LSTM

The Stack-LSTM corresponds to an action-based system which is composed by LSTMs augmented with a stack pointer. In contrast to the most common approaches which detect the entity mentions for a whole sequence, with Stack-LSTMs the entity mentions are detected and classified on the fly. This is a fundamental property to our model, since we perform EL when a mention is detected.

This model is composed by four stacks: the *Stack*, that contains the words that are being processed, the *Output*, that is filled with the completed chunks, the *Action* stack, which contains the previous actions performed during the processing of the current document, and the *Buffer*, that contains the words to be processed.

For NER, in the Stack-LSTM there are three possible types of actions:

- *Shift*, that pops a word off the *Buffer* and pushes it into the *Stack*. It means that the last word of the *Buffer* is part of a named entity.

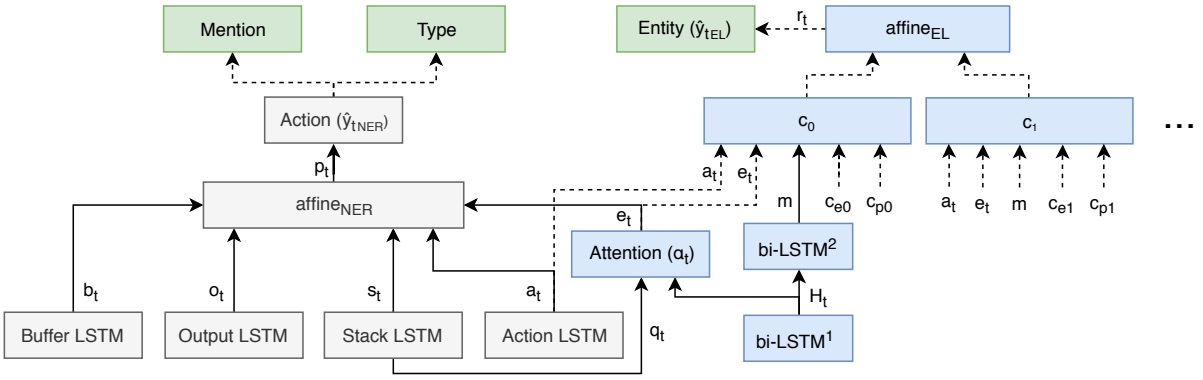

Figure 1: Simplified diagram of our model. The dashed arrows only occur when the action is *Reduce*. The blocks in blue correspond to our extensions to the Stack-LSTM and the green blocks correspond to the model's predictions. The grey blocks correspond to the stack-LSTM, the blue blocks to our extensions, and the green ones to the outputs.

- *Out*, that pops a word off the *buffer* and inserts it into the *Output*. It means that the last word of the *Buffer* is not part of a named entity.
- *Reduce*, that pops all the words in the *Stack* and pushes them into the *Output*. There is one action *Reduce* for each possible type of named entities, e.g. *Reduce-PER* and *Reduce-LOC*.

Moreover, the actions that can be performed at each step are controlled: the action *Out* can only occur if the stack is empty and the actions *Reduce* are only available when the *Stack* is not empty.

3.2 Our model

NER. To better capture the context, we complement the Stack-LSTM with a representation $\boldsymbol{v}_t$ of the sentence being processed, for each action step t. For that the sentence $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{|w|}$ is passed through a bi-directional LSTM, being $\boldsymbol{h}^1_w$ the hidden state of its 1st layer (bi-LSTM^1 in Figure 1), that corresponds to the word with index w:

$$\{\boldsymbol{h}^1_1, \ldots, \boldsymbol{h}^1_{|w|}\} = \text{BiLSTM}^1(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{|w|}).$$

We compute a representation of the words contained in the *Stack*, $\boldsymbol{q}_t$, by doing the mean of the hidden states of the 1st layer of the bi-LSTM that correspond to the words contained in the stack at action step t, set $\mathcal{S}_t$,:

$$\boldsymbol{q}_t = \frac{\sum_{k \in \mathcal{S}_t} \boldsymbol{h}^1_k}{|\mathcal{S}_t|}.$$

This is used to compute the attention scores $\boldsymbol{\alpha}_t$:

$$z_{tw} = \boldsymbol{u}^\top (\boldsymbol{W}_1 \boldsymbol{h}^1_w + \boldsymbol{W}_2\, \boldsymbol{q}_t)$$
$$\boldsymbol{\alpha}_t = \text{softmax}(\boldsymbol{z}_t),$$

where $\boldsymbol{W}_1$, $\boldsymbol{W}_2$, and $\boldsymbol{u}$ are trainable parameters. The representation $\boldsymbol{v}_t$ is then obtained by doing the weighted average of the bi-LSTM 1st layer's hidden states:

$$\boldsymbol{v}_t = \sum_{w=1}^{|w|} \boldsymbol{h}^1_w\, \alpha_{tw}.$$

To predict the action to be performed, we implement an affine transformation (affine$_{\text{NER}}$ in Figure 1) whose input is the concatenation of the last hidden states of the *Buffer* LSTM $\boldsymbol{b}_t$, *Stack* LSTM $\boldsymbol{s}_t$, *Action* LSTM $\boldsymbol{a}_t$, and *Output* LSTM $\boldsymbol{o}_t$, as well as the sentence representation $\boldsymbol{v}_t$.

$$\boldsymbol{d}_t = [\boldsymbol{b}_t;\ \boldsymbol{s}_t;\ \boldsymbol{a}_t;\ \boldsymbol{o}_t;\ \boldsymbol{v}_t]$$

Then, for each step t, we use these representations to compute the probability distribution $\boldsymbol{p}_t$ over the set of possible actions $\mathcal{A}$, and select the action $\widehat{\boldsymbol{y}}_{t_{\text{NER}}}$ with the highest probability:

$$\boldsymbol{p}_t = \text{softmax}(\text{affine}(\boldsymbol{d}_t))$$
$$\widehat{\boldsymbol{y}}_{t_{\text{NER}}} = \underset{a \in \mathcal{A}}{\arg\max}\ (\boldsymbol{p}_t(a)).$$

The NER loss function is the cross entropy, with the gold action for step t being represented by the one-hot vector $\boldsymbol{y}_{t_{\text{NER}}}$:

$$\mathcal{L}_{NER} = -\sum_{t=1}^{T} \boldsymbol{y}_{t_{\text{NER}}}^{\top} \log(\boldsymbol{p}_t).$$

where T is the total number of action steps for the current document.

EL. When the action predicted is *Reduce*, a mention is detected and classified. This mention is then disambiguated by selecting its respective entity knowledge base id. The disambiguation step is performed by ranking the mention's candidate entities.

The candidate entities $c \in \mathcal{C}$ for the present mention are represented by their entity embedding $\boldsymbol{c}_e$ and their prior probability c_p. The prior probabilities were previously computed based on the co-occurrences between mentions and candidates in Wikipedia.

To represent the mention detected the 2nd layer of the sentence bi-LSTM (bi-LSTM^2 in Figure 1), is used, being the representation $\boldsymbol{m}$ obtained by averaging the hidden states $\boldsymbol{h}_w^2$ that correspond to the words contained in the mention, set $\mathcal{M}$:

$$\{\boldsymbol{h}_1^2, \ldots, \boldsymbol{h}_{|w|}^2\} = \text{BiLSTM}^2(\boldsymbol{h}_1^1, \ldots, \boldsymbol{h}_{|w|}^1)$$
$$\boldsymbol{m} = \frac{\sum_{w \in \mathcal{M}} \boldsymbol{h}_w^2}{|\mathcal{M}|}.$$

These features are concatenated with the representation of the sentence $\boldsymbol{v}_t$, and the last hidden state of the *Action* stack-LSTM $\boldsymbol{a}_t$:

$$\boldsymbol{c}_i = [\boldsymbol{c}_{ei};\ c_{pi};\ \boldsymbol{m};\ \boldsymbol{v}_t;\ \boldsymbol{a}_t].$$

We compute a score for each candidate with affine transformations ($\text{affine}_{\text{EL}}$ in Figure 1) that have $\boldsymbol{c}$ as input, and select the candidate entity with the highest score, $\widehat{\boldsymbol{y}}_{t_{\text{EL}}}$:

$$\boldsymbol{l}_t = \text{affine}(\tanh(\text{affine}(\boldsymbol{c}_i, \ldots, \boldsymbol{c}_n)))$$
$$\boldsymbol{r}_t = \text{softmax}(\boldsymbol{l}_t)$$
$$\widehat{\boldsymbol{y}}_{t_{\text{EL}}} = \arg\max_{c \in \mathcal{C}} (\boldsymbol{r}_t(c)).$$

The EL loss function is the cross entropy, with the gold entity for step t being represented by the one-hot vector $\boldsymbol{y}_{t_{\text{EL}}}$:

$$\mathcal{L}_{EL} = -\sum_{t=1}^{T} \boldsymbol{y}_{t_{\text{EL}}}^{\top} \log(\boldsymbol{r}_t)).$$

where T is the total number of mention that correspond to entities in the knowledge base.

Due to the fact that not every mention detected has a corresponding entity in the knowledge base, we first classify whether this mention contains an entry in the knowledge base using an affine transformation followed by a sigmoid. The affine's input is the stack LSTM last hidden state $\boldsymbol{s}_t$:

$$d = \text{sigmoid}(\text{affine}(\boldsymbol{s}_t)).$$

The NIL loss function, binary cross-entropy, is given by:

$$\mathcal{L}_{NIL} = -(y_{NIL} \log(d) + (1 - y_{NIL}) \log(1 - d)),$$

where y_{NIL} corresponds to the gold label, 1 if mention should be linked and 0 otherwise.

During training we perform teacher forcing, i.e. we use the gold labels for NER and the NIL classification, only performing EL when the gold action is *Reduce* and the mention has a corresponding id in the knowledge base. The multi-task learning loss is then obtained by summing the individual losses:

$$\mathcal{L} = \mathcal{L}_{NER} + \mathcal{L}_{EL} + \mathcal{L}_{NIL}.$$

4 Experiments

4.1 Datasets and metrics

We trained and evaluated our model on the biggest NER-EL English dataset, the AIDA/CoNLL dataset (Hoffart et al., 2011). It is a collection of news wire articles from Reuters, composed by a training set of 18,448 linked mentions in 946 documents, a validation set of 4,791 mentions in 216 documents, and a test set of 4,485 mentions in 231

documents. In this dataset, the entity mentions are classified as person, location, organisation and miscellaneous. It also contains the knowledge base id of the respective entities in Wikipedia.

For the NER experiments we report the F1 score while for the EL we report the micro and macro F1 scores. The EL scores were obtained with the Gerbil benchmarking platform, which offers a reliable evaluation and comparison with the state-of-the-art models (Röder et al.). The results were obtained using strong matching settings, which requires exactly predicting the gold mention boundaries and their corresponding entity.

4.2 Training details and settings

In our work, we used 100 dimensional word embeddings pre-trained with structured skip-gram on the Gigaword corpus (Ling et al., 2015). These were concatenated with 50 dimensional character embeddings obtained using a bi-LSTM over the sentences. In addition, we use contextual embeddings obtained using a character bi-LSTM language model by Akbik et al. (2018). The entity embeddings are 300 dimensional and were trained by Yamada et al. (2017) on Wikipedia. To get the set of candidate entities to be ranked for each mention, we use a pre-built dictionary (Pershina et al., 2015).

The LSTM used to extract the sentence and mention representations, $\boldsymbol{v}_t$ and $\boldsymbol{m}$ is composed by 2 hidden layers with a size of 100 and the ones used in the Stack-LSTM have 1 hidden layer of size 100. The feedforward layer used to determine the entity id has a size of 5000. The affine layer used to predict whether the mention is NIL has a size of 100. A dropout ratio of 0.3 was used throughout the model.

The model was trained using the ADAM optimiser (Kingma and Ba, 2014) with a decreasing learning rate of 0.001 and a decay of 0.8 and 0.999 for the first and second momentum, respectively.

4.3 Results

Comparison with state of the art models. We compared the results obtained using our joint learning approach with state-of-the-art NER models, in Table 3, and state-of-the-art end-to-end EL models, in Table 4. In the comparisons, it can be observed that our model scores are competitive in both tasks.

System	Test F1
Flair (Akbik et al., 2018)	**93.09**
BERT Large (Devlin et al., 2018)	92.80
CVT + Multi (Clark et al., 2018)	92.60
BERT Base (Devlin et al., 2018)	92.40
BiLSTM-CRF+ELMo (Peters et al., 2018)	92.22
Our model	92.43

Table 3: NER results in CoNLL 2003 test set.

System	Validation F1		Test F1	
	Macro	Micro	Macro	Micro
Kolitsas et al. (2018)	**86.6**	**89.4**	**82.6**	**82.4**
Cao et al. (2018)	77.0	79.0	80.0	80.0
Nguyen et al. (2016)	-	-	-	78.7
Our model	82.8	85.2	81.2	81.9

Table 4: End-to-end EL results on validation and test sets in AIDA/CoNLL.

Comparison with individual models. To understand whether the multi-task learning approach is advantageous for NER and EL we compare the results obtained when using a multi-task learning objective with the results obtained by the same models when training with separate objectives. In the EL case, in order to perform a fair comparison, the mentions that are linked by the individual system correspond to the ones detected by the multi-task approach NER.

These comparisons results can be found in Tables 5 and 6, for NER and EL, respectively. They show that, as expected, doing joint learning improves both NER and EL results consistently. This indicates that by leveraging the relatedness of the tasks, we can achieve better models.

System	Validation F1	Test F1
Only NER	95.46	92.34
NER + EL	**95.72**	**92.52**

Table 5: Comparison of Named Entity Recognition multi-task results with single model results.

System	Validation F1		Test F1	
	Macro	Micro	Macro	Micro
Only EL	81.3	83.5	79.9	80.2
NER + EL	**82.6**	**85.2**	**81.1**	**81.8**

Table 6: Comparison of Entity Linking results multi-task results with single model results.

Ablation tests. In order to comprehend which components had the greatest contribution to the

obtained scores, we performed an ablation test for each task, which can be seen in Tables 7 and 8, for NER and EL, respectively. These experiments show that the use of contextual embeddings (Flair) is responsible for a big boost in the NER performance and, consequently, in EL due to the better detection of mentions. We can also see that the addition of the sentence representation (sent rep $\boldsymbol{v}_t$) improves the NER performance slightly. Interestingly, the use of a mention representation (ment rep $\boldsymbol{m}$) for EL that is computed by the sentence LSTM, not only yields a big improvement on the EL task but also contributes to the improvement of the NER scores. The results also indicate that having a simple affine transformation selecting whether the mention should be linked, improves the EL results.

System	Validation F1	Test F1
Stack-LSTM	93.54	90.47
+ Flair	95.40	92.16
+ sent rep	95.55	92.22
+ ment rep	**95.72**	**92.52**
+ NIL	95.68	92.43

Table 7: Ablation test for Named Entity Recognition.

System	Validation F1		Test F1	
	Macro	Micro	Macro	Micro
Stack-LSTM	81.95	84.76	80.37	80.12
+ Flair	82.59	85.75	80.86	81.05
+ sent rep	82.31	85.43	80.49	80.62
+ ment rep	82.64	85.17	81.07	81.76
+ NIL	**82.78**	**85.23**	**81.19**	**81.94**

Table 8: Ablation test for Entity Linking.

5 Conclusions and Future Work

We proposed doing joint learning of NER and EL, in order to improve their performance. Results show that our model achieves results competitive with the state-of-the-art. Moreover, we verified that the models trained with the multi-task objective have a better performance than individual ones. There is, however, further work that can be done to improve our system, such as training entity contextual embeddings and extending it to be cross-lingual.

Acknowledgements

This work was supported by the European Research Council (ERC StG DeepSPIN 758969), and by the Fundação para a Ciência e Tecnologia through contracts UID/EEA/50008/2019 and CMUPERI/TIC/0046/2014 (GoLocal). We thank Afonso Mendes, David Nogueira, Pedro Balage, Sebastião Miranda, Gonçalo Correia, Erick Fonseca, Vlad Niculae, Tsvetomila Mihaylova, Marcos Treviso, and the anonymous reviewers for helpful discussion and feedback.

References

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649. Association for Computational Linguistics.

Yixin Cao, Lei Hou, Juanzi Li, and Zhiyuan Liu. 2018. Neural collective entity linking. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 675–686. Association for Computational Linguistics.

Rich Caruana. 1997. Multitask learning. *Machine learning*, 28(1):41–75.

Jason PC Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional lstm-cnns. *Transactions of the Association for Computational Linguistics*, 4:357–370.

Kevin Clark, Minh-Thang Luong, Christopher D. Manning, and Quoc Le. 2018. Semi-supervised sequence modeling with cross-view training. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1914–1925. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Greg Durrett and Dan Klein. 2014. A joint model for entity analysis: Coreference, typing, and linking. *Transactions of the association for computational linguistics*, 2:477–490.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. 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 334–343. Association for Computational Linguistics.

Theodoros Evgeniou and Massimiliano Pontil. 2004. Regularized multi–task learning. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 109–117. ACM.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 363–370. Association for Computational Linguistics.

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.

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.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv:1412.6980 [cs.LG]*, pages 1–13.

Nikolaos Kolitsas, Octavian-Eugen Ganea, and Thomas Hofmann. 2018. End-to-end neural entity linking. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 519–529. Association for Computational Linguistics.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*.

Phong Le and Ivan Titov. 2018. Improving entity linking by modeling latent relations between mentions. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1595–1604.

Wang Ling, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015. Two/too simple adaptations of word2vec for syntax problems. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1299–1304. Association for Computational Linguistics.

Gang Luo, Xiaojiang Huang, Chin-Yew Lin, and Zaiqing Nie. 2015. Joint entity recognition and disambiguation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 879–888.

Dat Ba Nguyen, Martin Theobald, and Gerhard Weikum. 2016. J-nerd: joint named entity recognition and disambiguation with rich linguistic features. *Transactions of the Association for Computational Linguistics*, 4:215–229.

Maria Pershina, Yifan He, and Ralph Grishman. 2015. Personalized page rank for named entity disambiguation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 238–243.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of NAACL-HLT*, pages 2227–2237, New Orleans, Louisiana.

Priya Radhakrishnan, Partha Talukdar, and Vasudeva Varma. 2018. Elden: Improved entity linking using densified knowledge graphs. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1844–1853.

Michael Röder, Ricardo Usbeck, and Axel-Cyrille Ngonga Ngomo. Gerbil–benchmarking named entity recognition and linking consistently. *Semantic Web*, (Preprint):1–21.

Avirup Sil and Alexander Yates. 2013. Re-ranking for joint named-entity recognition and linking. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 2369–2374. ACM.

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 *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

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. *TACL*, 5:397–411.

Yi Yang, Ozan Irsoy, and Kazi Shefaet Rahman. 2018. Collective entity disambiguation with structured gradient tree boosting. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 777–786.