Arabic Named Entity Recognition: What Works and What's Next

Liyuan Liu
University of Illinois
Urbana-Champaign
ll2@illinois.edu

Jingbo Shang
University of Illinois
Urbana-Champaign
shang7@illinois.edu

Jiawei Han
University of Illinois
Urbana-Champaign
hanj@illinois.edu

Abstract

This paper presents the winning solution to the Arabic Named Entity Recognition challenge run by Topcoder.com. The proposed model integrates various tailored techniques together, including representation learning, feature engineering, sequence labeling, and ensemble learning. The final model achieves a test F_1 score of 75.82% on the AQMAR dataset and outperforms baselines by a large margin. Detailed analyses are conducted to reveal both its strengths and limitations. Specifically, we observe that (1) representation learning modules can significantly boost the performance but requires a proper pre-processing and (2) the resulting embedding can be further enhanced with feature engineering due to the limited size of the training data. All implementations and pre-trained models are made public[1].

1 Introduction

Aiming to identify entities in natural language, named entity recognition (NER) serves as one of the fundamental steps in various applications. In many languages, the performance of NER has been significantly improved because of recent advances in representation learning (Peters et al., 2018; Akbik et al., 2018). To promote the development of Arabic NER, a challenge was hosted on Topcoder.com[2] based on the public Arabic NER benchmark dataset (i.e., the AQMAR dataset) (Mohit et al., 2012). Challenge submissions were required to only use annotations from the training set, and manual reviews on the submitted solutions were further conducted to prevent cheating.

Among 137 registrants competing in the challenge[3], we placed the first by tailoring various techniques and incorporating them all together. Intuitively, it is hard to only rely on feature engineering to capture textual signals, especially for morphologically rich languages like Arabic (Habash, 2010). At the same time, neural networks have demonstrated their great potentials to automate high-quality representation construction in an end-to-end manner. Therefore, we leverage embedding modules to represent words with pre-trained vectors for a better quality. Besides, we observe that handcrafted features can bring a considerable improvement. Consuming all these features, we train multiple LSTM-CRF models to construct the mapping from representations to predictions, and further aggregate their outputs with ensemble learning. Moreover, we incorporate a dictionary-based string matching model and observe that it can improve the recall at some cost of precision, which results in a marginal F_1-score improvement.

Our final ensemble model achieves a test F_1 score of 75.82%, outperforming all other participants as well as the previous state-of-the-arts by significant margins. We further conduct analyses on our solution to get deeper insights on the task: (1) the effectiveness of representation learning and (2) the role of feature engineering.

The rest of paper is organized as follow. The next section discusses related work. Section 3 introduces the problem setting and presents the data analysis. The proposed framework is presented in Section 4, including model ensemble and dictionary-based model. Tailored representations modules are introduced in Section 5. Finally, we discuss the experimental results in Section 6.

[1] https://github.com/LiyuanLucasLiu/ArabicNER

[2] https://www.topcoder.com/challenges/30087004

[3] 220 submissions from 30 participates are made in total.

Proceedings of the Fourth Arabic Natural Language Processing Workshop, pages 60–67
Florence, Italy, August 1, 2019.

2 Related Work

Typically, named entity recognition is conducted as a sequence labeling task. Before deep learning demonstrated its effectiveness, traditional methods rely on handcrafted features (e.g., features based on POS tags) and language-specific resources (e.g., gazetteers) to capture textual signals. Machine learning models like conditional random field (CRF) and hidden Markov model (HMM) are employed to capture the label dependency (Lafferty et al., 2001; Florian et al., 2003; Chieu and Ng, 2002). Many attempts have been made to reduce the reliance on feature engineering or other human endeavors, which makes the NER task be solved in an end-to-end manner (Lample et al., 2016; Ma and Hovy, 2016; Shang et al., 2018). Recent studies have revealed that language model is an effective representation module for NER (Peters et al., 2017, 2018; Liu et al., 2018b; Akbik et al., 2018; Liu et al., 2018a).

At the same time, many approaches have been proposed specifically to solve the NER task in Arabic. Traditional Arabic NER models are mostly rule-basedmodels (Shaalan, 2014). Recently, people have started to attach this task with machine learning methods (Helwe and Elbassuoni, 2017; Gridach, 2016). To further improve the performance, attempts have been made to combine both rule-based and learning-based approaches into a unified framework (Pasha et al., 2014; Abdelali et al., 2016). Besides, incorporating additional supervision from other domains or languages has been explored as well (Darwish, 2013).

3 Problem Setting

In this section, we first introduce the problem setting of sequence labeling. Then, we discuss the aforementioned Arabic NER challenge.

3.1 Sequence Labeling

In the sequence labeling framework, NER problems are usually annotated following the labeling schemes like `BIO` and `IOBES`. These labeling schemes help us encode the information about entities (Ratinov and Roth, 2009). For example, in the `BIO` scheme, when a token sequence is identified as a named entity, its starting token and middle/end tokens are labeled as `B-` and `I-` followed by the type; and all other words are labeled as `O`. The `IOBES` scheme is similar to `BIO` but further use `S-` for singleton entity and `E-` for end-of-entity, respectively.

Using such labels, we define the input sequence as $X = \{x_1, x_2, \ldots, x_T\}$, where x_i is i-th token and its label is y_i. Moreover, we define the character-level input for X as $C = \{c_{1,1}, c_{1,2}, \cdots, c_{1,\_}, c_{2,1}, \cdots, c_{T,\_}\}$, where $\{c_{i,1}, \cdots, c_{i,\_}\}$ are the characters contained in the word x_i and $c_{i,\_}$ is the space character right after x_i. Then, the goal of NER becomes to predict the label y_i for each token x_i in the input sequence X.

3.2 Arabic NER Challenge

The Arabic NER challenge uses the public Arabic NER benchmark dataset (i.e., the AQMAR dataset) (Mohit et al., 2012). Its annotated entities are classified into four types (i.e., "Person", "Location", "Organization" and "Miscellaneous"). This dataset contains 28 hand-annotated Arabic Wikipedia articles, 14 articles are used as the training set, 7 articles are used as the development set, and 7 articles are used as the test set.

Data cleaning is further conducted on this dataset. Specifically, we observed that the label sequence is encoded in a noisy manner. For example, some entities are labelled as {`B-`, **`O`**, `I-`}, while the legit label sequence should be {`B-`, **`I-`**, `I-`}; Some entities are labelled as {`B-`$\mathtt{T}_0$, `I-`$\mathtt{T}_1$} (here, $\mathtt{T}_0$ and $\mathtt{T}_1$ are two different entity types), while the legit label sequence should be {`B-`$\mathtt{T}_0$, `B-`$\mathtt{T}_1$}. In the pursuit of more powerful models and more meaningful comparisons, we conduct a label cleaning to regularize the label sequence. The resulting dataset is released for future study[4], and its statistics are summarized in Table **??**. In the following sections, all comparisons are conducted on this cleaned dataset.

4 Model Framework

As visualized in Figure 1, we design a heterogeneous framework, which incorporates various techniques: (1) It employs representation learning and sequence labeling as the basic sequence labeling model; (2) It leverages ensemble learning to combine outputs from different NER models; and (3) It further incorporates a dictionary-based string matching model.

[4] https://github.com/LiyuanLucasLiu/ArabicNER

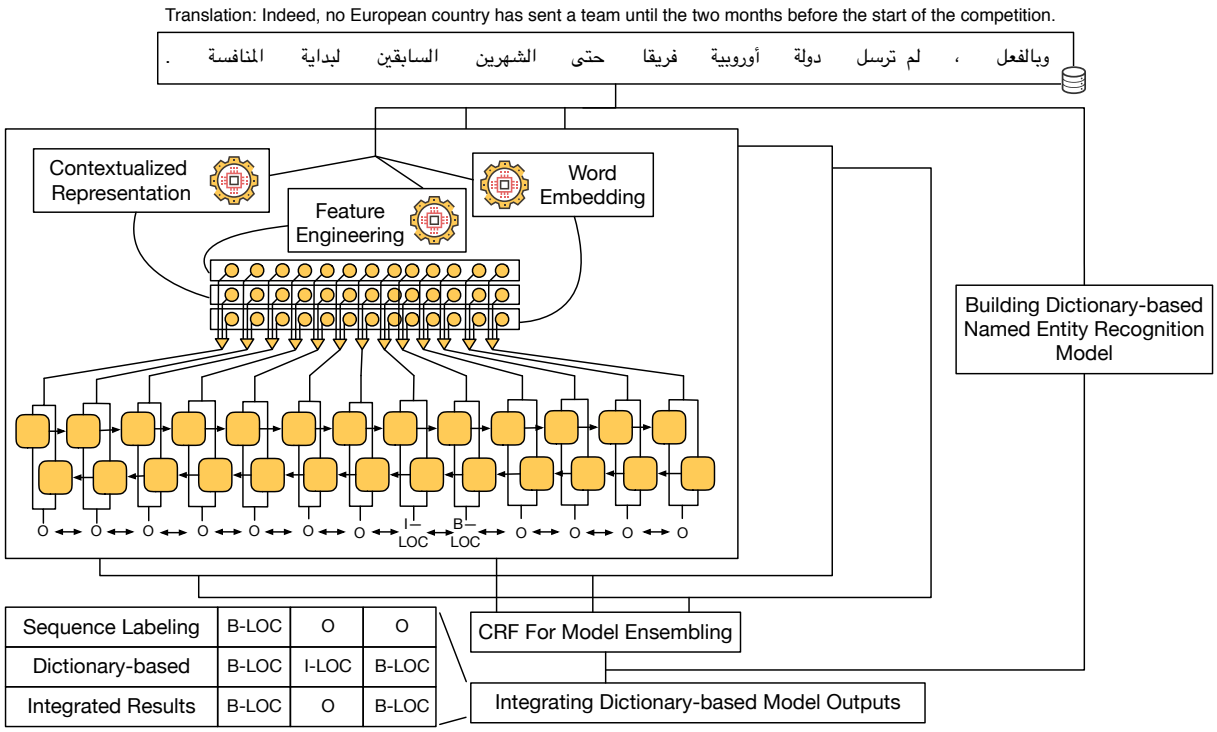

Figure 1: Our proposed framework for Arabic NER.

	Train	Dev	Test
# Sent	1,329	711	606
# Token	36,050	19,519	18,284
# Char	156,941	83,267	80,565
# PER	752	292	424
# LOC	971	146	326
# ORG	234	114	102
# MISC	1,092	660	722

Table 1: Dataset Statistics of the AQMAR dataset.

4.1 Sequence Labeling Model

As to the basic sequence labeling model, we assume there are n different representation modules, namely M_i $(1 \leq i \leq n)$. Given the j-th token in the input sequence, the representation vector produced by module M_i is denoted as $\mathbf{f}_{i,j}$. In this paper, we concatenate the output from different modules as the representation (input of LSTM-CRF), i.e., $\mathbf{f}_j = [\mathbf{f}_{1,j}; \mathbf{f}_{2,j}; \cdots; \mathbf{f}_{n,j}]$. Given the input sequence X, we define its token representations as $\mathbf{F} = \{\mathbf{f}_1, \mathbf{f}_2, \cdots, \mathbf{f}_T\}$. Building upon representation modules, we use LSTM-CRF (Huang et al., 2015) to conduct entity extraction: we first feed $\mathbf{F}$ into Bi-LSTMs, whose outputs are marked as $\mathbf{Z} = \{\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_T\}$. A linear-chain CRF is further leveraged to model the whole label sequence simultaneously. Specifically, for the input sequence $\mathbf{Z}$, CRF defines the conditional probability of $\mathbf{Y} = \{y_1, \cdots, y_T\}$ as

$$p(\mathbf{Y}|\mathbf{Z}) = \frac{\prod_{t=1}^{T} \phi(y_{t-1}, y_t, \mathbf{z}_t)}{\sum_{\hat{\mathbf{Y}} \in \mathbf{Y}(\mathbf{Z})} \prod_{t=1}^{T} \phi(\hat{y}_{t-1}, \hat{y}_t, \mathbf{z}_t)} \quad (1)$$

where $\hat{\mathbf{Y}} = \{\hat{y}_1, \cdots, \hat{y}_T\}$ is a possible label sequence, $\mathbf{Y}(\mathbf{Z})$ refers to the set of all possible label sequences for $\mathbf{Z}$, and $\phi(y_{t-1}, y_t, \mathbf{z}_t)$ is the potential function of the CRF. In this paper, we define the potential function as:

$$\phi(y_{t-1}, y_t, \mathbf{z}_t) = \exp(W_{y_t} \mathbf{z}_t + b_{y_{t-1}, y_t})$$

where W_{y_t} and b_{y_{t-1}, y_t} are the weight and bias.

During the model training, we use the negative log-likelihood of Equation 1 as the loss function. In the inference stage, the predicted label sequence for input X is the one maximizing the probability in Equation 1. Although the denominator in Equation 1 contains an exponential number of terms [5], due to the definition of the potential function, both training and inference can be efficiently conducted using dynamic programming.

[5]The number of terms is exponential to the sequence length T.

The dictionary-based NER model and representation learning modules would be introduced in the following sections.

4.2 Sequence Labeling Model Ensemble

To get better performance, we applied the ensemble learning on sequence labeling results. Specifically, as in Figure 1, multiple NER models are separately trained with the shared representation modules, and their results are combined as the final output.

Specifically, we refer the output of N models as $\{\hat{\mathbf{Y}}_1, \hat{\mathbf{Y}}_2, \cdots, \hat{\mathbf{Y}}_N\}$, where $\hat{\mathbf{Y}}_i = \{\hat{y}_{i,1}, \cdots, \hat{y}_{i,T}\}$. Following the previous work (Nguyen and Guo, 2007), we first construct a list of transition matrices $\{R_1, \cdots, R_{T-1}\}$, where $R_i(j,k) = |\{n|\hat{y}_{n,i} = j, \hat{y}_{n,i+1} = k\}|$ is the number of times that i-th and $i+1$-th tokens are labelled as j and k in $\{\hat{\mathbf{Y}}_n\}$. Also, we calculate $B_i(j) = |\{n|\hat{y}_{n,i} = j\}|$, which is the times of i-th token being labelled as j. Then the integrated label sequence is calculated with dynamic programming:

$$\hat{\mathbf{Y}} = \arg\max \sum_{t=1}^{T-1} R_t(\hat{y}_t, \hat{y}_{t+1}) + \sum_{t=1}^{T} B_t(\hat{y}_t)$$

where $\hat{\mathbf{Y}} = \{\hat{y}_1, \cdots, \hat{y}_T\}$ is the integrated label sequence.

4.3 Dictionary-based NER Model

Besides the sequence labeling ensemble model, we also incorporate a dictionary-based NER model. Specifically, we first build a dictionary to map surface names to their types from the training set, then apply this dictionary via string matching. We will add the dictionary-extracted entities into the final prediction, if and only if they do not conflict with the sequence labeling results. For example, in Figure 1, since the two-word entity (i.e., `B-LOC I-LOC`) detected by the dictionary-based model overlaps with the sequence labeling results, this entity is dropped; At the same time, because the one-word entity (i.e., the second `B-LOC`) detected by the dictionary-based model is not overlapped with any entities detected by the sequence labeling model, it is therefore integrated to the final results. In our experiments, we found this enrichment by the dictionary-based model improves the recall at a relatively smaller cost of the precision, thus improving the F_1 score.

5 Representation Learning Modules

In this section, we introduce the three representation learning modules: (1) word embedding, (2) contextualized representation, and (3) handcrafted features.

5.1 Word Embedding

Based on the distributional hypothesis (*i.e.*, "a word is characterized by the company it keeps" (Harris, 1954)), word embedding methods aim to learn the distributed representations by analyzing their contexts (Mikolov et al., 2013). Recent work shows that word embedding could uncover textual information of various levels (Artetxe et al., 2018). Hence, we leverage word embedding as a part of the word representation. Due to the limited size of the training set, we fix the pre-trained word embedding during the training of NER models. When the pre-trained embedding has a high dimension, we will add a linear projection to further project them to a relatively low dimension.

5.2 Contextualized Representation

Contextualized representations have been widely adopted in the state-of-the-art sequence labeling models. Typically, they rely on bidirectional neural language models to capture the local contextual information before and after a certain word. Such representations provide rich, supplementary information to the context-agnostic information contained in a word embedding. Specifically, character-level language models are first used to provide additional supervision (Liu et al., 2018b), and further exploration observes its effectiveness as the pre-training task to construct contextualized word representation (Akbik et al., 2018).

We present the details of character-level language modeling and integration as below.

Character-Level Language Modeling. A bidirectional character-level language model contains two character-level language models to capture information from two directions. Character-level language modeling aims to model the probability distribution of the character sequence. Typically, the probability of the sequence $\{c_1, \cdots, c_T\}$ is defined in a "forward" manner: $p(c_1, \cdots, c_T) = \prod_{t=1}^{T} p(c_t|c_1, \cdots, c_{t-1})$.

To calculate this conditional probability, we first map the input sequence C to a list of character embedding vectors and pass them into a re-

current neural network, whose output is referred to $\mathbf{h}_t$. Then, the probability $p(c_t|c_1, \cdots, c_{t-1})$ is calculated using the softmax function. The backward language model is the same as the forward language model, except that it decomposes the probability of the sequence $\{c_1, \cdots, c_T\}$ from the end to the front as $p(c_1, \cdots, c_T) = \prod_{t=1}^{T} p(c_t|c_{t+1}, \cdots, c_T)$. Its output for character c_t is denoted as $\mathbf{h}_t^r$. Both language models use negative log-likelihood as the training objective.

Language Model Integration. Using the bidirectional character-level language models, we construct contextualized representations for each word. Specifically, we feed the input character sequence C to language models, and then concatenate the hidden state of the forward language model at $c_{i,-}$ and the hidden state of the backward language model at $c_{i-1,-}$ as the representations for x_i. We refer these two hidden states as $\mathbf{h}_i$ and $\mathbf{h}_i^r$. Due to the complex nature of natural language, large dimensions of $\mathbf{h}_i$ and $\mathbf{h}_i^r$ are usually required in language models, which might lead to overfitting in the NER task. To avoid such cases, we add a linear transformation layer to project $\mathbf{h}_i$ and $\mathbf{h}_i^r$ to a lower dimension. In details, we use $\mathbf{r}_i = W_{cr} \cdot [\mathbf{h}_i, \mathbf{h}_i^r] + \mathbf{b}_{cr}$, where W_{cr} and $\mathbf{b}_{cr}$ are parameters to learn during the training of NER models. The output $\mathbf{r}_i$ is the contextualized representation for x_i.

5.3 Handcrafted Features

Due to the limited amount of available annotations, we further handcraft word shape features to help the model better capture the textual features. Specifically, all words are classified into three classes: (1) We mark all numbers as "num"; (2) For remaining words, if it contains English characters, it would be marked as "en"; (3) Otherwise, it would be marked as "ar". These three categories would be further mapped to three different vectors as the token representation.

Although these handcrafted features are quite simple, similar to existing work (Dozat, 2016), it results in a remarkable performance improvement in our experiments. More discussion on this feature engineering design is included in Section 6.

6 Experiments

In this section, we present the experimental results on the AQMAR dataset.

6.1 Implementation Detail

As to pre-trained language models, we conduct training on the Arabic Wikipedia texts with a vocabulary of 256 characters (out-of-vocabulary characters are mapped to a special <UNK> character). Since the resulting language model would be used to construct contextualized representations for the downstream task, whose input would be space separated, we conduct further preprocessing. Specifically, we first tokenize the text, then concatenate the token sequence by space. To demonstrate the importance of pre-processing, we trained two kinds of language models, one with pre-processing, and the other without.

For pre-trained word embedding, we adopt two sets of pre-trained embedding. One is trained with the word2vec model (Mikolov et al., 2013). It has 100 dimensions and is public available [6]. The other is trained with the Fasttext model (Bojanowski et al., 2017), which is released together with 156 other languages [7]. It has 300 dimensions and would be projected to 100 dimensions before concatenating with other vectors.

6.2 Hyper-parameter

For language model training, we use Nadam (Dozat, 2016) as the optimizer, set the learning rate as 0.002, clip the gradient at 1, set the batch size as 128 and limit the back propagation length to 256. As to the RNN, we use one-layer LSTMs with 2048 hidden states. We set its character embedding to be 128 dimensional and project its outputs to 50 dimension before concatenating with other vectors.

As to the sequence labeling task, we use LSTMs with 250 hidden states in the LSTM-CRF layer, and apply dropout with a ratio of 0.5, and use additional word dropout to each representation module with a ratio of 0.1. Following the previous work (Reimers and Gurevych, 2017), we use Nadam (Dozat, 2016) as the optimizer, set the learning rate as 0.002, clip the learning rate at 1 and set the batch size as 32.

6.3 Performance Comparison

As summarized in Table 2, our final model achieves a F_1 score of 75.82%. Further ablation study is conducted to analyze the effectiveness of each module.

[6] https://github.com/bakrianoo/aravec

[7] https://fasttext.cc/docs/en/crawl-vectors.html

Methods	Pre	Rec	F_1
Final Model	81.06	71.22	75.82
– Dict-based	81.27	70.84	75.70
– Ensemble	79.33	68.99	73.80
– Word shape	76.43	67.13	71.47
– Pre-process	71.60	61.33	66.07
– Language model	66.92	45.96	54.50

Table 2: Model Performance and Ablation Study for the AQMAR dataset.

Ablation Study Setting. In the ablation study, we first detach the dictionary-based NER from the resulting system and refer ensemble sequence labeling model as "– Dict-based". Then, we refer the basic sequence labeling model as "– Ensemble". After that, we detach hand-crafted features and mark the resulting model as "– Word shape". Pre-processing is further removed from language model training, which is marked as "– Pre-process". In the end, we remove language model which leads to a typical LSTM-CRF model (Huang et al., 2015) with pre-trained word embedding, we refer this model as "– Language model". Their results are summarized in Table 2.

Discussion. We find that the dictionary-based NER model[8] improves the recall at the cost of the precision and improves the F_1 score by a small margin. Also, we observe that the results demonstrate the effectiveness of ensemble learning. At the same time, we find the major F_1 improvements come from a better capturing of task-related signals. For example, by properly adding language models or designing handcrafted features, the F_1 boosts significantly. It verifies the effectiveness of contextualized representation. Also, it reveals the weakness of these techniques. Specifically, although the constructed character-level language model has the potential to capture the word shape signals, adding handcrafted features (i.e., word shape) can improve the F_1 from 71.47% to 73.80%. We conjugate this is caused by the limited size of training data with English entities, which limits the model from properly constructing task-related representations. Further comparison between these two models finds their major differences are the predictions for entities containing both Arabic and English and validates our intuition. Besides, we find the pre-processing used in language model training is crucial for the performance, which has a big impact on the model performance (from 66.07% to 71.47%). The main reason is that although pre-trained language models are powerful, they are agnostic to the target task corpus and suffer from their differences.

[8]The dictionary-based NER model achieves Pre: 64.35%, Rec: 8.83%, F_1: 15.53%.

7 Conclusion

In this paper, we introduce the winning solution to the Arabic Named Entity Recognition challenge. First of all, we give a detailed introduction on system design and the integrated technologies. We further conduct ablation study to reveal the effectiveness of each module and figure out all modules bring performance improvements. We observe that properly capturing the task-related features is crucial to the performance. We also noticed the current contextualized representation learning techniques, although effective, could be further enhanced by incorporating handcrafted features to better handle some corner cases.

Acknowledge

Research was sponsored in part by U.S. Army Research Lab. under Cooperative Agreement No. W911NF-09-2-0053 (NSCTA), DARPA under Agreement No. W911NF-17-C-0099, National Science Foundation IIS 16-18481, IIS 17-04532, and IIS-17-41317, DTRA HDTRA11810026, and grant 1U54GM114838 awarded by NIGMS through funds provided by the trans-NIH Big Data to Knowledge (BD2K) initiative (www.bd2k.nih.gov). Any opinions, findings, and conclusions or recommendations expressed in this document are those of the author(s) and should not be interpreted as the views of any U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon. This research was supported by grant 1U54GM114838 awarded by NIGMS through funds provided by the trans-NIH Big Data to Knowledge (BD2K) initiative (www.bd2k.nih.gov). The views and conclusions contained in this paper are those of the authors and should not be interpreted as representing any funding agencies.

References

Ahmed Abdelali, Kareem Darwish, Nadir Durrani, and Hamdy Mubarak. 2016. Farasa: A fast and furious segmenter for arabic. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 11–16.

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.

Mikel Artetxe, Gorka Labaka, Inigo Lopez-Gazpio, and Eneko Agirre. 2018. Uncovering divergent linguistic information in word embeddings with lessons for intrinsic and extrinsic evaluation. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 282–291.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Hai Leong Chieu and Hwee Tou Ng. 2002. Named entity recognition: a maximum entropy approach using global information. In *COLING*.

Kareem Darwish. 2013. Named entity recognition using cross-lingual resources: Arabic as an example. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1558–1567.

Timothy Dozat. 2016. Incorporating nesterov momentum into adam.

Radu Florian, Abe Ittycheriah, Hongyan Jing, and Tong Zhang. 2003. Named entity recognition through classifier combination. In *NAACL*.

Mourad Gridach. 2016. Character-aware neural networks for arabic named entity recognition for social media. In *Proceedings of the 6th workshop on South and Southeast Asian natural language processing (WSSANLP2016)*, pages 23–32.

Nizar Y Habash. 2010. Introduction to arabic natural language processing. *Synthesis Lectures on Human Language Technologies*.

Zellig S Harris. 1954. Distributional structure. *Word*, 10(2-3):146–162.

Chadi Helwe and Shady Elbassuoni. 2017. Arabic named entity recognition via deep co-learning. *Artificial Intelligence Review*, pages 1–19.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *ICML*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of NAACL-HLT*, pages 260–270.

Liyuan Liu, Xiang Ren, Jingbo Shang, Xiaotao Gu, Jian Peng, and Jiawei Han. 2018a. Efficient contextualized representation: Language model pruning for sequence labeling. In *EMNLP*.

Liyuan Liu, Jingbo Shang, Xiang Ren, Frank Fangzheng Xu, Huan Gui, Jian Peng, and Jiawei Han. 2018b. Empower sequence labeling with task-aware neural language model. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. *arXiv preprint arXiv:1603.01354*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Behrang Mohit, Nathan Schneider, Rishav Bhowmick, Kemal Oflazer, and Noah A Smith. 2012. Recall-oriented learning of named entities in arabic wikipedia. In *EACL*.

Nam Nguyen and Yunsong Guo. 2007. Comparisons of sequence labeling algorithms and extensions. In *ICML*, pages 681–688. ACM.

Arfath Pasha, Mohamed Al-Badrashiny, Mona T Diab, Ahmed El Kholy, Ramy Eskander, Nizar Habash, Manoj Pooleery, Owen Rambow, and Ryan Roth. 2014. Madamira: A fast, comprehensive tool for morphological analysis and disambiguation of arabic. In *LREC*, volume 14, pages 1094–1101.

Matthew Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In *Proceedings of the ACL*, volume 1, pages 1756–1765.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of NAACL-HLT*, volume 1, pages 2227–2237.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the ACL*.

Nils Reimers and Iryna Gurevych. 2017. Optimal hyperparameters for deep lstm-networks for sequence labeling tasks. *arXiv preprint arXiv:1707.06799*.

Khaled Shaalan. 2014. A survey of arabic named entity recognition and classification. *Computational Linguistics*.

Jingbo Shang, Liyuan Liu, Xiaotao Gu, Xiang Ren, Teng Ren, and Jiawei Han. 2018. Learning named entity tagger using domain-specific dictionary. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2054–2064.