Joint Learning of Named Entity Recognition and Dependency Parsing using Separate Datasets

Arda Akdemir and Tunga Güngör

Bogazici University, Istanbul, Turkey
{arda.akdemir,gungort}@boun.edu.tr

Abstract. Joint learning of different NLP-related tasks is an emerging research field in Machine Learning. Yet, most of the recent models proposed on joint learning require a dataset that is annotated jointly for all the tasks involved. Such datasets are available only for frequently used languages. In this paper, we propose a novel BiLSTM CRF based joint learning model for dependency parsing and named entity recognition tasks, which has not been employed before for any language to the best of our knowledge. We also propose a novel joint learning method which uses different datasets for each task to solve the joint learning problem. This enables joint learning of various tasks for languages that have limited amount of annotated datasets. Our model, tested on a frequently used NER dataset for Turkish, has comparable results with the state-of-the-art systems. We also show that our proposed model outperforms the joint learning model which uses a single dataset.

Keywords: Joint Learning · Named Entity Recognition · Dependency Parsing · Turkish.

1 Introduction

Named Entity Recognition (NER) is the task of detecting and categorizing the entities in a given text. Entities contain valuable information related to various Natural Language Processing (NLP) tasks which makes NER an important and a popular research area.

Nadeau et al. [12] give a detailed survey of the work done until 2007 in this area. Many of the systems relied on manually crafted feature sets and used statistical machine learning methods to make NER predictions. Most feature based models require third party tools, like morphological analyzers, to annotate a given dataset for the features [14]. Recent state-of-the-art works focus on representing each word with character and word embeddings [9], and learn the entity tags by using Bidirectional Long Short Term Memory (BiLSTM) layers. One of the main motivations of these approaches is to relieve NLP models from relying on manually crafted features. Even though these systems do not rely on manually crafted feature sets, for joint learning of multiple tasks these systems require a dataset to be annotated jointly for all the tasks involved. However, such

jointly annotated datasets usually do not exist, especially for less frequently used languages such as Turkish.

In this paper, we propose a novel joint learning model that attempts to solve this joint annotation problem. Our model jointly learns named entity recognition and dependency parsing using separate datasets for each task similar to the approach used for named entity recognition and morphological disambiguation [5]. Dependency parsing information is shown to improve the performance of feature based NER systems. Based on this observation, we incorporate dependency parser output into the named entity recognition component to learn both tasks jointly. Our results show that joint learning on separate gold-labeled datasets for each task outperforms joint learning on a single dataset annotated automatically using a third party tool [13] for the dependency parsing tags. Moreover, our model is the first model in the literature that attempts joint learning of dependency parsing and named entity recognition. Our main contributions can be summarized as follows:

- We implement a novel joint learning model for dependency parsing and named entity recognition.
- We propose a novel way of learning these tasks where we make use of different datasets for each task.

The paper is organized as follows: Section 2 describes the related work on NER and joint learning. Section 3 describes the datasets used for each task. Section 4 explains the details of the proposed joint learning model. Section 5 gives the results obtained. Section 6 concludes the paper. All the work we have done can be accessed and reproduced from the relevant GitHub repository which includes the source codes, the results obtained and explanation about how the model should be run [1].

2 Related Work

Most of the recent work on NER make use of neural network models and especially BiLSTM based systems. Chiu et al. [2] use a BiLSTM based neural network model and learn the character embeddings using Convolutional Neural Networks (CNNs). Lample et al. [9] propose a system based on BiLSTMs and Conditional Random Fields (CRFs) where CRFs are used to find the optimal tag sequence using the Viterbi decoding algorithm. Ma et al. [11] propose a model that combines the previous two works. They use CNNs to learn character embeddings and use BiLSTMs together with a CRF layer. Their model currently holds the state-of-the-art result for the NER task on the frequently used CoNLL 2003 English dataset with 91.21% F1 score.

Related work for Turkish, which make use of manually crafted feature sets, show that NER performance increases when syntactical and morphological features are employed [14]. Their findings show the importance of using features

[1] https://github.com/ardakdemir/Named-Entity-Recognition-in-Turkish-Using-Deep-Learning-Models-and-Joint-Learning

for morphologically rich languages like Turkish. Following the state-of-the-art work for English, Gungor et al. [5] use a BiLSTM and CRF based model, and show that morphological features improve the NER performance. Using a similar BiLSTM based deep learning architecture, Gunes et al. [4] obtained the state-of-the-art score for the commonly used NER dataset for Turkish [16].

Joint learning is an emerging research area in NLP. Joint learning of POS tagging and dependency parsing is shown to improve the dependency parser performance [13, 10] for the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies. Syntactic parsers are used to increase the performance of various NLP tasks. However, these models are in pipeline format and are not trained jointly. Hashimoto et al. [6] attempt joint learning of various NLP tasks of different levels of complexity. They follow the work of Sogard et al. [15] which shows using outputs of different levels of the neural network for different tasks outperforms using the outputs at the same level for each task. Similarly, Gungor et al. [5] showed that joint learning of morphological disambiguation and named entity recognition improves the named entity recognition performance when different layer outputs are used for each task.

3 Datasets

This section describes the datasets used in this paper. For the named entity recognition task for Turkish we used a frequently used NER dataset [16]. The dataset was created as part of a work which tackles four information extraction tasks including NER. It is made up of articles from a national newspaper. Table 1 gives the number of entities in each set. For the configuration of our model which uses a single dataset annotated for both tasks, we used a dependency parser to annotate this dataset for dependency parsing.

The IOB tagging scheme is used for the NER task. The dataset is annotated for three entity types: Location, Organization and Person. Including the label of the non-entity words we have the following seven entity tags: B-PER, I-PER, B-LOC, I-LOC, B-ORG, I-ORG, and O. In the IOB scheme, each entity is labeled with the 'I' prefix unless the entity token is an immediate successor of a separate entity token with same entity type. In that case, 'B' prefix is used to overcome the entity boundary ambiguity problem.

Table 1. Number of annotated entities in the Turkish NER dataset.

Subset	**LOC**	**ORG**	**PER**
Training	6,720	9,260	6,249
Development	769	1,412	824
Test	907	1,174	670

For dependency parsing we used the IMST-UD dataset provided by the Universal Dependency framework for the CoNLL 2018 Shared Task [17]. The dataset

is in CoNLL-U format which was designed to form a universal format for dependency datasets of multiple languages. The dataset is a semi-automatic conversion of the IMST Treebank, which is itself a reannotated version of the METU-Sabanci Turkish Treebank. The dataset is made up of 5,635 sentences from daily news reports and novels. An example annotated sentence from this dataset is given in Fig. 1. The example sentence in Turkish is: 'Karşısında, pantolonu dizlerine dek ıslak, önlük torbası ham eriklerle dolu İbrahim dikiliyordu', which corresponds to the following English sentence: 'Ibrahim was standing against him with his pants wet up to the knees and with his bag filled with plums.' The joint learner only makes use of the surface form, dependency relation type, and dependency head index fields which correspond to 2^{nd}, 7^{th}, and 8^{th} fields, respectively.

```
# sent_id = mst-0006
# text = Karşısında, pantolonu dizlerine dek ıslak, önlük torbası ham eriklerle dolu İbrahim dikiliyordu.
1   Karşısında  karşı   ADJ NAdj    Case=Loc|Number=Sing|Number[psor]=Sing|Person=3|Person[psor]=3  14  amod    _   SpaceAfter=No
2   ,   ,   PUNCT   Punc    _   14  punct   _   _
3   pantolonu   pantolon    NOUN    Noun    Case=Nom|Number=Sing|Number[psor]=Sing|Person=3|Person[psor]=3  4   nmod:poss   _   _
4   dizlerine   diz NOUN    Noun    Case=Dat|Number=Plur|Number[psor]=Sing|Person=3|Person[psor]=3  6   obl _   _
5   dek dek ADP PCDat   _   4   case    _   _
6   ıslak   ıslak   ADJ Adj _   13  amod    _   SpaceAfter=No
7   ,   ,   PUNCT   Punc    _   14  punct   _   _
8   önlük   önlük   NOUN    Noun    Case=Nom|Number=Sing|Person=3   9   nmod:poss   _   _
9   torbası torba   NOUN    Noun    Case=Nom|Number=Sing|Number[psor]=Sing|Person=3|Person[psor]=3  14  obl _   _
10  ham ham NOUN    Noun    Case=Nom|Number=Sing|Person=3   11  nmod    _   _
11  eriklerle   erik    NOUN    Noun    Case=Ins|Number=Plur|Person=3   12  obl _   _
12  dolu    dolu    ADJ Adj _   13  amod    _   _
13  İbrahim İbrahim PROPN   Prop    Case=Nom|Number=Sing|Person=3   14  nsubj   _   _
14  dikiliyordu dikil   VERB    Verb    Aspect=Prog|Mood=Ind|Number=Sing|Person=3|Polarity=Pos|Polite=Infm|Tense=Past   0   root    _
SpaceAfter=No
15  .   .   PUNCT   Punc    _   14  punct   _   _
```

Fig. 1. An example sentence from the IMST-UD Treebank dataset used by the dependency parser of the joint learner.

4 Methodology

This section explains in detail the joint learning model we propose. The joint learning model is a BiLSTM CRF based neural network model that has two main components corresponding to the two tasks being tackled. We begin by explaining the overall model which will be followed by detailed descriptions of each component in separate sections.

The overall architecture of the model is shown in Fig. 2. The first component in the network produces the output for the dependency parsing task. Given a token sequence, BiLSTM is used to calculate the vector representation of each word. These vectors are given to two separate Multilayer Perceptrons (MLP) which output the scores for the directed dependency arcs between each pair of tokens and the relation type prediction for the predicted dependency arcs, respectively. The second part of the network is responsible for the named entity recognition task. It takes as input the output produced by the dependency parsing component in vector format, in addition to the same inputs given to the dependency parser. BiLSTM is used to find the vector representation of each

token, by taking into account the prediction of the dependency parser. These vectors are given to a MLP which outputs scores for all possible NER tags for each token. A CRF layer is used to find the highest scoring tag sequence for the calculated tag scores.

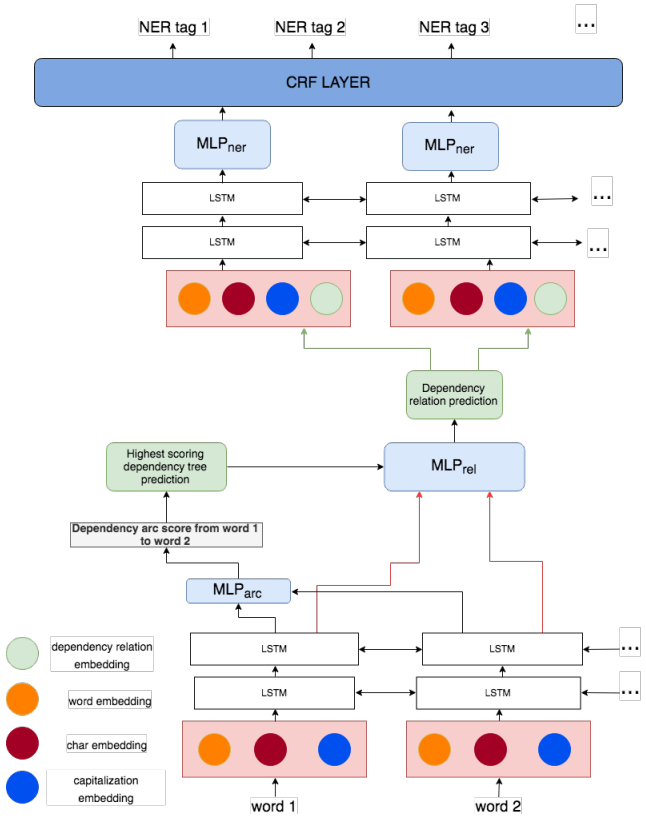

Fig. 2. Architecture of the joint learning model for dependency parsing and named entity recognition.

4.1 Dependency Parsing Component

The dependency parsing component of our proposed model is similar to the dependency parsing component of the joint POS Tagging Dependency Parsing (jPTDP) system [13]. Word, character and capitalization embeddings of the tokens of a given sentence are given as input to the dependency parser. Following the related works [2], in addition to the word and character representations, we feed the model with the vector representation of the capitalization feature

of each token. Capitalization information is important for the NER task. BiLSTMs are used to produce vector representations of each token by analyzing the given sentence in both directions. All embeddings are initialized randomly for all words, capitalization types, and characters in the training set. The capitalization embedding maps four capitalization types into vector representations: all lower case, First Letter Only, ALL UPPER CASE, and miXEd Case. Let $\mathbf{e}_w$ denote the vector representation of a given token w. Let $\mathbf{w}_{emb}$, $\mathbf{c}_{emb}$, and $\mathbf{cap}_{emb}$ represent the corresponding word, character, and capitalization embeddings of the word, respectively. The vector representation is calculated by concatenating the above mentioned three vectors:

$$\mathbf{e}_w = \mathbf{w}_{emb} \circ \mathbf{c}_{emb} \circ \mathbf{cap}_{emb}$$

$\mathbf{c}_{emb}$ is calculated by using a BiLSTM layer. For a character x, we randomly initialize a character embedding $\mathbf{c}_x$. To calculate the character embedding for a given word each character embedding $\mathbf{c}_i$ is fed into a BiLSTM which produces forward and backward character representations, $\mathbf{c}^f_{emb}$ and $\mathbf{c}^b_{emb}$. These are concatenated to produce the character embedding:

$$\mathbf{c}_{emb} = \mathbf{c}^f_{emb} \circ \mathbf{c}^b_{emb}$$

So the overall vector representation of a given word is:

$$\mathbf{e}_w = \mathbf{w}_{emb} \circ \mathbf{c}^f_{emb} \circ \mathbf{c}^b_{emb} \circ \mathbf{cap}_{emb}$$

These vector representations for the words are given as input to the first BiLSTM layer. The output of the first BiLSTM layer is the concatenations of the vectors created by going over a given sentence in forward and backward directions. The output of the LSTM layer has length $ldims$ for each direction, thus the output is of length $2 \times ldims$. The second BiLSTM layer thus takes as input a vector of size $2 \times ldims$ for each token in a sentence. The system again goes over these vector representations to create forward and backward vector outputs of size $ldims$.

The outputs of the second BiLSTM layer are used as the input for the multi layer perceptrons (MLP) responsible for calculating the scores for the dependency parsing task. The dependency parsing task contains two sub-tasks: creating a parse tree for a given sentence and labeling the arcs of the parse tree.

Let $\mathbf{e}^{lstm}_{w_i}$ represent the final lstm output for word w_i. Following the related work [13], four vectors are concatenated and given as input to the MLP called $\mathbf{MLP}_{arc}$, which outputs the score for a directed arc from w_i and w_j:

$$\text{score}_{\text{arc}}(i,j) = \mathbf{MLP}_{\text{arc}}(\mathbf{e}^{lstm}_{w_i} \circ \mathbf{e}^{lstm}_{w_j} \circ (\mathbf{e}^{lstm}_{w_i} * \mathbf{e}^{lstm}_{w_j}) \circ |\mathbf{e}^{lstm}_{w_i} - \mathbf{e}^{lstm}_{w_j}|)$$

where $(\mathbf{e}^{lstm}_{w_i} * \mathbf{e}^{lstm}_{w_j})$ and $|\mathbf{e}^{lstm}_{w_i} - \mathbf{e}^{lstm}_{w_j}|$ are element-wise multiplication and absolute element-wise difference, respectively. Each vector is of length $2 \times ldims$ which results in a vector of size $8 \times ldims$. Given these scores Eisner's decoding algorithm [3] is used to find the highest scoring dependency tree. Loss of this subtask, $Loss_{arc}$, is calculated by maximizing the difference between the gold parse tree and the highest scoring incorrect parse tree following the related work [7].

To find the dependency relation type, i.e. the label, for each predicted arc, another MLP called $\mathbf{MLP}_{rel}$ is used. $\mathbf{MLP}_{rel}$ takes the same input with the previous MLP and outputs a vector containing a score for each relation type:

$$\mathbf{scores}_{rel}(i,j) = \mathbf{MLP}_{rel}(\mathbf{e}^{lstm}_{w_i} \circ \mathbf{e}^{lstm}_{w_j} \circ (\mathbf{e}^{lstm}_{w_i} * \mathbf{e}^{lstm}_{w_j}) \circ |\mathbf{e}^{lstm}_{w_i} - \mathbf{e}^{lstm}_{w_j}|)$$

Cross entropy loss, $Loss_{rel}$, is computed over this score vector using the gold label for each token.

4.2 Named Entity Recognition Component

Let rel_j represent the dependency relation type for the dependency arc from w_i to w_j. Each dependency relation type rel_j is represented with an embedding $\mathbf{e}_{rel_j}$. Given the relation type prediction $\mathbf{e}_{rel_j}$ of the dependency parsing component for given words w_i and w_j, the word w_j is represented by concatenating the embedding of this relation to the vector representation of the word:

$$\mathbf{e}^{ner}_{w_j} = \mathbf{e}_{w_j} \circ \mathbf{e}_{rel_j}$$

For a given input sentence with n words, we represent each sentence with the sequence of vector representations $\mathbf{e}^{ner}_{w_i}$ for $1 \leq i \leq n$ and feed this sequence of vectors into LSTMs in forward and backward directions. The LSTM outputs vectors $\mathbf{v}_i$ for each word w_i in a given sentence by taking into account the context in both directions:

$$\mathbf{v}_i = \text{LSTM}_f(\mathbf{e}^{ner}_{w_{1:i}}) \circ \text{LSTM}_b(\mathbf{e}^{ner}_{w_{n:i}})$$

These vectors of size $2 \times ldims$ are fed into a second LSTM layer which outputs the final vector representation of each token:

$$\mathbf{v}^{fin}_k = \text{LSTM}_f(v_{1:k}) \circ \text{LSTM}_b(v_{n:k})$$

Each vector $\mathbf{v}^{fin}_k$ is given as input to an MLP called $\mathbf{MLP}_{ner}$ which produces scores for each possible entity type for each word. The score matrix of size (n, t) is created where $score_{i,j}$ refers to the score for the i^{th} token having the j^{th} tag:

$$\mathbf{Scores}_{ner}(i) = \mathbf{MLP}_{ner}(\mathbf{v}^{fin}_i)$$

During the prediction mode, these scores are normalized into probabilities to be used for finding the optimal tag sequence. For each tag for a given word probability is calculated by normalizing the scores produced for each entity tag type tag_jusing the softmax function:

$$P(i,j) = \frac{exp(score(i,j))}{\sum_{(j' \in tags)} exp(score(i,j'))}$$

where $score(i,j)$ represents the score produced for tag_j for a given word w_i in a sentence.

We model the tag sequence jointly rather than predicting each label independently. For this a CRF [8] is used to find the highest scoring named entity tag sequence. Transitions between named entity tags are important because of the sequential nature of the task and CRFs are used frequently for the NER task [5, 1]. Using the score vectors produced for each word and randomly initialized tag translation probabilities, we find the optimal tag sequence using the Viterbi decoding algorithm.

Negative log likelihood loss $Loss_{ner}$ is used to calculate the loss of the gold label NER tag for each word in the sentence. The transition probabilities between entity tag types are included implicitly in $Loss_{ner}$ because the final prediction of the model is calculated using the Viterbi algorithm which takes into account the transition probabilities.

5 Experiments and Results

We performed various experiments with different versions of the proposed joint learning model. This section will explain the experiments conducted and the results obtained. In this paper we mainly focus on experiments with the following configurations:

- Model 1: The named entity recognition component only. This configuration does not take as input the dependency parsing prediction. It is a typical BiLSTM CRF based named entity recognition model.
- Model 2: The joint learning model using a single dataset annotated jointly for both tasks. The named entity recognition dataset is automatically annotated using a third party dependency parsing tool [13].
- Model 3: Our proposed novel joint learning model which uses different datasets for each task. Extensive results are obtained for this final proposed model with various hyperparameter configurations. The results obtained with this model are compared with the results obtained in the previous two configurations.

For the evaluation of the performance of both of the tasks, we used the frequently used evaluation metrics. For the dependency task, 'Labeled Attachment Score' (LAS) and 'Unlabeled Attachment Score' (UAS) are used. For the named entity recognition task, we use precision, recall and F1-Measure.

We first show the results on the development sets used to fine tune the models. These results are calculated taking into account partial matches of entities so every match between a gold label and a prediction is counted without checking whether a named entity is completely predicted by the system or not. Fig. 3 shows the results obtained for running Model 1 for 40 epochs. The best F1-Measure of **0.915** is obtained in the 34^{th} epoch. As stated, the architecture of this model is the same as shown in Fig. 2, but the part of the neural network responsible for the dependency parsing task is not used. One can think of this model as the dependency part of the network 'silenced'.

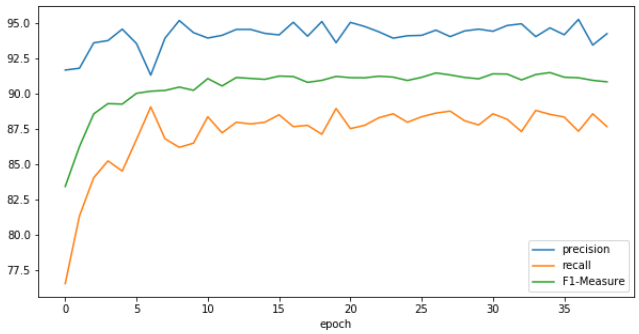

Fig. 3. Results for the NER only model in each epoch tested on the development set.

The next experiment is conducted for Model 2 on the version of the dataset that is annotated jointly for both tasks using a third party dependency parsing tool. The best results obtained for each task and the epoch numbers are given in Table 2 where DEP refers to the dependency parser score and NER refers to the named entity score. The dependency parser results for Model 2 are significantly higher than Model 3 (See Table 3). This is probably due to the fact that the architecture of the tool used for annotating the dataset is quite similar to the dependency parser implemented in our system. This similarity can make it easier for the system to quickly learn and mimic the parser used. The results also show that we could not obtain a performance improvement over Model 1.

Table 2. Best results for the joint learning model on a single dataset.

Task Name	Metric	**Best Results**	**Best Epoch**
DEP	**Average LAS - UAS Accuracy**	0.760	13
NER	**F1-Measure**	0.878	12

Next, we performed extensive experiments for Model 3. First, we give the results for the default configuration of the model on the development sets of each dataset. The parameters of the default configuration are set using the previous works on joint learning [5, 13]. The results with the default configuration are given in Table 3. The results show a relative improvement for the NER performance over using a single dataset annotated automatically.

There are various parameters in our joint learning model. The parameters of the model are given in Table 4. The results obtained for various combinations of word embedding dimensions and LSTM dimensions for Model 3 are given in Table 5.

Table 3. Best results for the joint learning model on development sets for each task.

Task Name	Metric	**Best Results**	**Best Epoch**
DEP	**Average LAS - UAS Accuracy**	0.600	16
NER	**F1-Measure**	0.904	17

Table 4. Parameters of the joint learning model together with the default values.

Parameter Name	**Default Value**
Word embedding size	100
Character embedding size	50
Capitalization feature embedding size	50
Relation embedding size	100
Hidden units	100
Activation function	tanh
Lstm layers	2
Lstm dimensions	128
Enable dependency parsing	True
Enable viterbi decoding	True

We could not observe a significant difference in performance during the grid search of other parameter configurations so we do not include them here. The best F1-Measure scores are obtained for the NER task with the following two configurations: (word embedding dim: 100 , lstm dim: 64) and (word embedding dim: 150 , lstm dim: 64).

Table 5. Results for joint learning model with different parameter combinations.

wemb dim	**lstm dim**	**NER F1-Measure**	**Average LAS & UAS**
50	64	0.905	0.587
	128	0.908	0.580
	256	0.904	0.588
100	64	**0.909**	0.595
	128	0.904	**0.600**
	256	0.907	0.593
150	64	**0.909**	0.594
	128	0.908	0.595
	256	0.908	0.591

Finally, we give the results obtained on the NER test set. We use the CoNLL evaluation metric for consistency with other works on named entity recognition. Table 6 shows the results for all models.

The results given here show that the novel joint learning model (Model 3) brings a relative improvement over using a single dataset annotated jointly for

Table 6. Results for all three models.

	Model 1			Model 2			Model 3		
	Precision	**Recall**	**F1**	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**
PER	89.74	89.89	89.81	92.43	77.82	84.50	86.29	86.66	86.48
LOC	89.95	90.04	89.99	77.07	87.53	81.97	86.84	85.89	86.36
ORG	87.56	86.65	87.10	81.21	75.67	78.34	80.97	76.41	78.63
Overall	89.28	89.15	89.21	83.78	80.51	82.11	85.23	83.91	84.56

both tasks (Model 2) for all entity types. Yet, we could not observe an improvement over the NER only model. The overall F1-Measure for Model 1, Model 2, and Model 3 are 89.21, 82.11 and 84.56, respectively. On the other hand, the joint learning model has several advantages over the NER only model. It enables learning both tasks jointly and, with sufficient amount of training time, Model 3 has the potential to match the Model 1 performance on the NER task while learning an additional task.

6 Conclusion And Future Work

In this paper, we proposed a novel neural network model for joint learning, which attempts to solve the joint annotation problem. We combined two BiLSTM based neural components to jointly learn dependency parsing and named entity recognition on different datasets. Our results show that we can obtain improvements over using a single dataset for joint learning. Yet, the NER component trained without using the dependency parsing prediction outperforms both models.

As future work, we plan to improve the performance of the proposed system in several ways. Using embedding representations for outputs instead of the outputs directly is shown to increase the performance of NLP systems for various tasks. Representation learning is applied to the coarse-grained labeled NER task in various studies. Using output representations of NER labels increases the performance for both coarse-grained and fine-grained NER tasks. Future work includes learning the representations of output labels to calculate losses in a more robust way.

We will also modify the proposed architecture to take into account the dependency prediction of the dependency parsing component in different ways. We will implement a joint learning model that makes use of the head of the dependency arc for a given word as well as the dependency relation.

7 Acknowledgements

We would like to thank Onur Gungor for his kind help and advices about this work. This work was partially supported by JST CREST Grant Number JPMJCR1402, JSPS KAKENHI Grant Numbers 17H01693, and 17K20023JST.

References

1. Chen, T., Xu, R., He, Y., Wang, X.: Improving sentiment analysis via sentence type classification using bilstm-crf and cnn. Expert Systems with Applications **72**, 221–230 (2017)
2. Chiu, J.P., Nichols, E.: Named entity recognition with bidirectional lstm-cnns. arXiv preprint arXiv:1511.08308 (2015)
3. Eisner, J.M.: Three new probabilistic models for dependency parsing: An exploration. In: Proceedings of the 16th conference on Computational linguistics-Volume 1. pp. 340–345. Association for Computational Linguistics (1996)
4. Güneş, A., TantuĞ, A.C.: Turkish named entity recognition with deep learning. In: 2018 26th Signal Processing and Communications Applications Conference (SIU). pp. 1–4. IEEE (2018)
5. Güngör, O., Üsküdarlı, S., Güngör, T.: Improving named entity recognition by jointly learning to disambiguate morphological tags. arXiv preprint arXiv:1807.06683 (2018)
6. Hashimoto, K., Xiong, C., Tsuruoka, Y., Socher, R.: A joint many-task model: Growing a neural network for multiple nlp tasks. In: EMNLP (2017)
7. Kiperwasser, E., Goldberg, Y.: Simple and accurate dependency parsing using bidirectional lstm feature representations. arXiv preprint arXiv:1603.04351 (2016)
8. Lafferty, J., McCallum, A., Pereira, F.C.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data (2001)
9. Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., Dyer, C.: Neural architectures for named entity recognition. arXiv preprint arXiv:1603.01360 (2016)
10. Li, Z., He, S., Zhang, Z., Zhao, H.: Joint learning of pos and dependencies for multilingual universal dependency parsing. Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies pp. 65–73 (2018)
11. Ma, X., Hovy, E.: End-to-end sequence labeling via bi-directional lstm-cnns-crf. arXiv preprint arXiv:1603.01354 (2016)
12. Nadeau, D., Sekine, S.: A survey of named entity recognition and classification. Lingvisticae Investigationes **30**(1), 3–26 (2007)
13. Nguyen, D.Q., Verspoor, K.: An improved neural network model for joint pos tagging and dependency parsing. arXiv preprint arXiv:1807.03955 (2018)
14. Şeker, G.A., Eryiğit, G.: Initial explorations on using crfs for turkish named entity recognition. Proceedings of COLING 2012 pp. 2459–2474 (2012)
15. Søgaard, A., Goldberg, Y.: Deep multi-task learning with low level tasks supervised at lower layers. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). vol. 2, pp. 231–235 (2016)
16. Tür, G., Hakkani-Tür, D., Oflazer, K.: A statistical information extraction system for turkish. Natural Language Engineering **9**(2), 181–210 (2003)
17. Zeman, D., Hajič, J., Popel, M., Potthast, M., Straka, M., Ginter, F., Nivre, J., Petrov, S.: CoNLL 2018 shared task: Multilingual parsing from raw text to universal dependencies. In: Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies. pp. 1–21. Association for Computational Linguistics, Brussels, Belgium (October 2018)