Natural Language Engineering (2021), **27**, pp. 35–64
doi:10.1017/S1351324919000627

CAMBRIDGE UNIVERSITY PRESS

ARTICLE

Transfer learning for Turkish named entity recognition on noisy text

Emre Kağan Akkaya and Burcu Can*

Department of Computer Engineering, Hacettepe University, Turkey
*Corresponding author. E-mail: burcucan@cs.hacettepe.edu.tr

(Received 26 October 2018; revised 11 November 2019; accepted 11 November 2019; first published online 28 January 2020)

Abstract

In this article, we investigate using deep neural networks with different word representation techniques for named entity recognition (NER) on Turkish noisy text. We argue that valuable latent features for NER can, in fact, be learned without using any hand-crafted features and/or domain-specific resources such as gazetteers and lexicons. In this regard, we utilize character-level, character n-gram-level, morpheme-level, and orthographic character-level word representations. Since noisy data with NER annotation are scarce for Turkish, we introduce a transfer learning model in order to learn infrequent entity types as an extension to the Bi-LSTM-CRF architecture by incorporating an additional conditional random field (CRF) layer that is trained on a larger (but formal) text and a noisy text simultaneously. This allows us to learn from both formal and informal/noisy text, thus improving the performance of our model further for rarely seen entity types. We experimented on Turkish as a morphologically rich language and English as a relatively morphologically poor language. We obtained an entity-level F1 score of 67.39% on Turkish noisy data and 45.30% on English noisy data, which outperforms the current state-of-art models on noisy text. The English scores are lower compared to Turkish scores because of the intense sparsity in the data introduced by the user writing styles. The results prove that using subword information significantly contributes to learning latent features for morphologically rich languages.

Keywords: Named entity recognition; Transfer learning; Recurrent neural networks; Low-resource language; Noisy text

1. Introduction

Named entity recognition (NER) is an information extraction task in natural language processing that aims to identify and categorize each word into predefined categories. For example, for the sentence *"Thomas Bayes was the son of London Presbyterian minister Joshua Bayes,"* as an NER task, we aim to assign PERSON label for *Thomas Bayes* and *Joshua Bayes* and assign LOCATION label for London, and CORPORATION label for *Presbyterian*. As we can see from the recent studies in the literature, the performance of NER on formal (e.g., newspapers, academic papers) data is very high, particularly for languages like English with comparably poor morphology and abundant annotated data. Recent research such as Lample *et al.* (2016) and Ma and Hovy (2016) achieved over 91% F1 score on English formal data, almost comparable to human annotation performance. So one can prematurely conclude that the NER task has nearly reached its peak performance.

However with the ever-changing nature of the Internet, especially after the emergence of social media, we have been introduced to informal/noisy data (user-generated data) such as user comments and tweets. This new type of data is highly valuable for information extraction tasks such as opinion mining due to being widespread and having almost up-to-date nature. However, noisy

and informal text normally includes missing characters in words (either deliberately or by forgetfulness), missing punctuation, various emojis, slang words, and abbreviations. All of these bring new problems for the existing Turkish NER studies, considering that most of them are either statistical or rule-based models that usually depend on manually crafted features (e.g., capitalization, numerical/date/time patterns, or other rule-based features) and/or external domain-specific resources (e.g., gazetteers, lexicons), therefore ill-suited to noisy data. Some of these studies try to solve these new challenges either by extending their existing feature set to better suit on this new domain or by adding new domain-specific resources. Current state-of-the-art Turkish NER model (Şeker and Eryiğit 2017), which is a conditional random field (CRF)-based model utilizing domain-specific heavy feature engineering and external resources, achieves an F1 score of 91.94% on formal data and only 67.96% (and 63.63% without using *Twitter mention* feature to label mentions as *PERSON*) on noisy data. This is mostly due to Turkish being a morphologically rich and agglutinative language and having scarce annotated data. It is evident that Turkish NER performance is far behind on noisy/informal data, despite being a well-studied topic for formal data. Recent successful studies on other languages, especially on English, address these issues by utilizing neural architectures with the help of auto-generated features.

Turkish being a morphologically complex language, NER receives its own share. Although derivation in Turkish named entities is not very common, inflected named entities are seen quite often in Turkish. Especially, case markers are seen with any type of named entity. For example, in the sentence *"Istanbul'a gideceğim"* (means *I will go to Istanbul."*), the location named entity *"İstanbul"* is inflected in the dative case; *"Onu Ahmet'ten alabilirsin."* (means *"You can take it from Ahmet."*), the person name *"Ahmet"* is inflected in the ablative case. Thereby, due to the common usage of suffixes with named entities, the sparsity problem is introduced in NER task. The problem is even more severe in noisy text, since the morphemes could also be spelled differently by different users on social media. For example, the suffix *"ciğim"* that means *"dear"* when it is used with person names, could be written as *"cim,"* or as *"çim," "çum," "çm," "cm," "cim,"* etc., depending on the last vowel in the name accordingly with the vowel harmony.

This motivates us to research and adopt deep recurrent neural network models for Turkish and obtain valuable features without using any external domain-specific resources or any hand-crafted features. To this end, we propose a transfer learning model that is an extension of the widely used *bidirectional long short-term memory (LSTM)-CRF* model by incorporating an additional CRF layer that we train on another, preferably larger dataset to overcome the annotated data scarcity problem. The model is rather similar to the model presented by von Däniken and Cieliebak (2017), since we also use another CRF layer to further improve the performance. We argue that subword information is crucial in word representation for morphologically rich and agglutinative languages; therefore, the model also utilizes different subword embeddings such as morph2vec (Üstün, Kurfal, and Can 2018), fasttext (Bojanowski *et al*. 2017) and orthographic character-level embeddings. Morph2vec (Üstün *et al*. 2018) is a word representation model that estimates the word embeddings through its morphemes where the segmentation of words is not required *a priori*; therefore, the pretrained word embeddings are mimicked by using an attention mechanism over a list of potential segmentations of each word to obtain the final word representation. On the contrary, fasttext (Bojanowski *et al*. 2017) estimates the word embeddings through the n-grams of each word. To the best of our knowledge, this is the first neural network model without using any hand-crafted features and external resources for Turkish NER. Consequently, we obtain an F1 score of 67.39% on Turkish noisy data and 45.30% on English noisy data, which are both the highest scores for both languages.

The paper is organized as follows: Section 2 reviews the recent work on NER on noisy text for Turkish and also for English, Section 3 describes the word representation methods used for representing each word by a dense vector in the NER models proposed in this paper, Section 4 describes and gives the mathematical definition of the baseline Bi-LSTM-CRF model (Huang, Xu, and Yu 2015) adopted in this article, Section 5 describes the proposed transfer learning model,

https://doi.org/10.1017/S1351324919000627 Published online by Cambridge University Press

Section 6 gives the details on datasets and on the implementation of the models in addition to the experimental results obtained from the proposed models on Turkish and English, and finally Section 7 concludes the paper along with the future goals.

2. Related work

Various methods have been adopted for NER, those include statistical methods (Bikel 1997; Wu, Zhao, and Xu 2003; Suzuki and Isozaki 2008), rule-based models (Petasis *et al*. 2001), and recently deep neural network architectures (Huang *et al*. 2015; Ma and Hovy 2016). Since user-generated text on the Internet is typically different compared to formal text, more latent features need to be defined manually or more sophisticated methods need to be used to learn the latent features automatically from a given text. This is because noisy text is more scarce compared to formal text since it may change from one user to another.

One of the commonly used features would be the meaning of the words. Although the spelling of each word may change from one text to another, the meaning would stay the same. Meaning representation has benefited from distributional approaches a lot. In recent years, distributional models such as Latent Semantic Analysis (Landauer, Foltz, and Laham 1998) have changed direction toward neural models. Word representation models such as word2vec (Mikolov *et al*. 2013) or GloVe (Pennington, Socher, and Manning 2014) have shown superior performance. However, those models learn word representations very well when there is enough contextual information for each word, which will not be true for the scarce and noisy text. Therefore, other neural models that make use of subword information such as characters (Cao and Rei 2016), character n-grams (Bojanowski *et al*. 2017), and morphemes (Üstün *et al*. 2018) have been introduced, which learn the representations of scarce data (i.e., noisy text or any text in a morphologically rich language) better than word-level models. Neural word embeddings obtained from such models have been used as features and have shown superior performance when they are sequentially encoded by LSTMs. Moreover, it has been discovered that an additional CRF layer can learn the named entities by using the latent features learned by the LSTMs, which introduces the well-known Bi-LSTM-CRF (bidirectional long short-term memory and CRF) architecture (Huang *et al*. 2015) as a sequence labeling model.

Here, we review mainly the research on NER for noisy text. Although the main scope of this article is Turkish NER, since we are also inspired by other models on English, we also review the research on English NER on noisy text.

2.1 NER on Turkish noisy data

First of all, it is worth mentioning that all studies reported in this section use the same Turkish noisy dataset, so the reported scores are comparable to each other.[a]

Çelikkaya, Torunoğlu, and Eryiğit (2013) introduce the first study focusing on noisy data for Turkish with a CRF-based model that utilizes hand-crafted morphological and lexical features (e.g., stem, PoS tag, noun case, lower/upper case) along with gazetteers. They reported an F1 score of 19.28% on noisy dataset and 91.64% on formal dataset which can be interpreted as another indication that NER on noisy data does not perform as well as NER on formal text. With the aim of adapting the model for noisy data, Küçük and Steinberger (2014) extend a previous multilingual rule-based NER system by expanding existing domain-specific resources based on the fact that most sentences in the noisy data miss the letters with diacritics (ç, ğ, ı, ö, ş, ü) and the authors employ a normalization scheme using this feature. As a result, they achieved an F1 score of 46.93% on the same Turkish noisy dataset.

Eken and Tantuğ (2015) introduce another CRF-based approach that also makes use of gazetteers (with optional distance-based matching) and numerous features (e.g., apostrophe , case

[a]The details of the noisy dataset are given in Section 6.3.

https://doi.org/10.1017/S1351324919000627 Published online by Cambridge University Press

of the word, start of sentence) along with the word suffixes and prefixes. They reported 46.97% F1 score on a new noisy imbalanced dataset, and 28.53% F1 score on the same Turkish noisy dataset. Okur, Demir, and Özgür (2016) present a regularized averaged multiclass perceptron with hand-crafted features (e.g., word type flags, suffix, prefix, capitalization) along with pretrained embeddings obtained from word2vec (Mikolov *et al.* 2013). They also perform tweet normalization using the model introduced by Torunoğlu and Eryiğit (2014). Consequently, they obtain an F1 score of 48.96% on the noisy dataset.

Şeker and Eryiğit (2017) present the state-of-the-art model on Turkish NER which is another CRF-based model, similar to that of Çelikkaya *et al.* (2013). The authors use an extensive set of morphological and lexical features (e.g., stem, part-of-speech (POS) tags, capitalization, word type and shape flags) and gazetteers. Additionally, they use the existence of *Twitter mentions* as a feature. They also provide the reannotated versions of the two commonly used Turkish datasets: news dataset (Tür *et al.* 2003) and Twitter dataset (Çelikkaya *et al.* 2013). Reannotated versions also include TIMEX (Date, Time) and NUMEX (Money, Percent) types along with previously labeled ENAMEX (Organization, Person, Location) types. Finally, they report an F1 score of 67.96% with Twitter mentions and 63.63% without the mentions on the reannotated version of the Turkish noisy dataset.

2.2 NER on English noisy data

Analogously, all studies reported in this section use the same English noisy dataset, which was provided by the 3rd Workshop on Noisy User-Generated Text at Empirical Methods in Natural Language Processing (EMNLP) (WNUT'2017)[b] so that all the reported results are comparable to each other.

Aguilar *et al.* (2017), the winner of the WNUT'17,[c] apply multitask learning approach with a CRF-based model that incorporates pretrained word embeddings obtained from word2vec (Mikolov *et al.* 2013) and orthographic character-level embeddings trained on a Convolutional Neural Network (CNN) with two-stacked convolutional layers. They also make use of gazetteers for the well-known entities. They report an F1 score of 41.86% on entity level and 40.24% on surface forms.

von Däniken and Cieliebak (2017) use a transfer learning model. One of our proposed models is also based on their model. However, unlike our model, their model incorporates sentence-level embeddings (sent2vec) (Pagliardini, Gupta, and Jaggi 2017) and capitalization features in addition to character-level embeddings trained by a CNN and pretrained word embeddings obtained from fasttext (Bojanowski *et al.* 2017). As a result, they obtain 40.78% F1 score on entity level and 39.33% F1 score on surface forms. Lin *et al.* (2017) follow a similar approach for a CRF-based model and use word embeddings that are obtained from pretrained word embeddings and character-level embeddings obtained from another bidirectional LSTM. They also incorporate syntactic information by using POS tags, dependency roles, and word position, and head position. They achieve an F1 score of 40.42% on entity level and 37.62% on surface forms.

Sikdar and Gambäck (2017) propose an ensemble-based approach that uses features learned from CRF, support vector machine, and an LSTM. They also use hand-crafted features such as PoS tags, local context, chunk, suffix and prefix, word frequency, and a collection of flags (e.g., is-word-length-less-than-5, is-all-digit). Consequently, they achieve 38.35% F1 score for entity level and 36.31% F1 score for the surface forms.

Williams and Santia (2017) propose a statistical approach, where each word is associated with its context. Context conditional probabilities are used to estimate the named entity tag probabilities. They obtain an F1 score of 26.30% on entity level and 25.26% F1 score on surface

[b]The details of the noisy dataset are given in Section 6.3.
[c]https://noisy-text.github.io/2017/.

https://doi.org/10.1017/S1351324919000627 Published online by Cambridge University Press

forms. Jansson and Liu (2017), inspired by the work of Limsopatham and Collier (2016), use a bidirectional LSTM-CRF model that is similar to our baseline model but instead of orthographic features, Latent Dirichlet Allocation (Blei, Ng, and Jordan 2003) topic models and PoS tags are used as features. As a result, they achieve a performance of 39.98% F1 score on the entity level and 37.77% F1 score on the surface forms.

3. Neural word embeddings

We use neural word embeddings of words as input to our proposed models. We use different levels of word embeddings such as the word-level word embeddings obtained by word2vec (Mikolov *et al.* 2013), character n-gram-level word embeddings obtained by fasttext (Bojanowski *et al.* 2017), morpheme-level word embeddings obtained by morph2vec (Üstün *et al.* 2018), character-level embeddings, and orthographic character-level embeddings. By using these models, we aim to capture orthographic, morphological, and contextual information of words in noisy data.

For the notation that will be used throughout the article, we denote each sentence (i.e., tweet) by $S = (w_1, w_2, ..., w_N)$ that consists of N tokens (i.e., words or other tokens), where the ith token is denoted by w_i.

3.1 Orthographic character-level embeddings

We use an orthographic character encoder similar to that of Aguilar *et al.* (2017) that encodes alphabetic characters as "c" (or "C" if the character is capitalized), numeric characters as "n," punctuation as "p," and other characters as "x." For example, the word *"Türkiye'ye!"* (means *to Turkey*) becomes "Ccccccpccp." Each orthographic encoding is also padded with 0s accordingly with the longest word in the dataset to have a fixed length of orthographic embedding for all words. This allows us to reduce sparsity and capture the shapes and orthographic patterns within the words. We train the embeddings by a character-level CNN. We apply two-stacked convolutional layers and perform global average pooling on the output. Finally, we use a fully connected feed-forward layer with a rectifier linear unit (ReLU) activation function with the final character-level word representation of each word that is denoted by $E_{w_i}^{(c_{cnn})}$. An overview of the architecture is given in Figure 1. Here, the word *"Ankara!"* is first encoded in terms of its characters such as *"Ccccccp,"* and then the orthographic embeddings are fed into the convolutional layers to obtain the character representation for orthographic encoding.

As an alternative approach, we also train the orthographic character-level embeddings using a Bi-LSTM that is simply a combination of two different LSTMs (i.e., forward and backward LSTMs) where one of them takes the input sequence in the forward and the other one in the reverse order. Output of the forward and backward LSTMs are concatenated for the final orthographic character-level word embedding $E_{w_i}^{(c_{Bi-LSTM})}$. The Bi-LSTM model is given in Figure 2. Here, the sentence *"29 ekimde Ankara'ya"* is first encoded in terms of its orthographic characters such as *"nn ccccc Ccccccpcc,"* and then the embeddings of the orthographic characters are fed into a Bi-LSTM to obtain a character-level orthographic word embedding.

3.2 Character-level word embeddings

We also learn the character-level word embeddings using the actual characters rather than the character types unlike the orthographic word embeddings. For example, the word *"Bravo"* is first encoded in terms of the character embeddings of *"B,"* *"r,"* *"a,"* *"v,"* and *"o."*

We use another Bi-LSTM to learn the character-level word embeddings. To this end, the Bi-LSTM is fed by the character embeddings of the word. We obtain the character-level word embeddings denoted by $E_{w_i}^{(c)}$ by concatenating the vectors that are output by both LSTMs from both directions.

https://doi.org/10.1017/S1351324919000627 Published online by Cambridge University Press

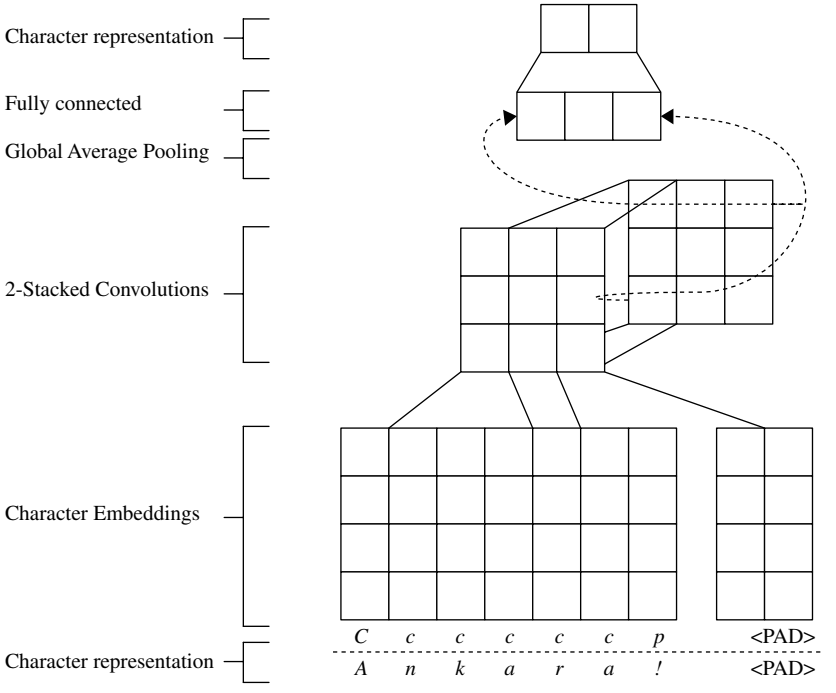

Figure 1. Character-level word embedding using CNN (Aguilar *et al*. 2017).

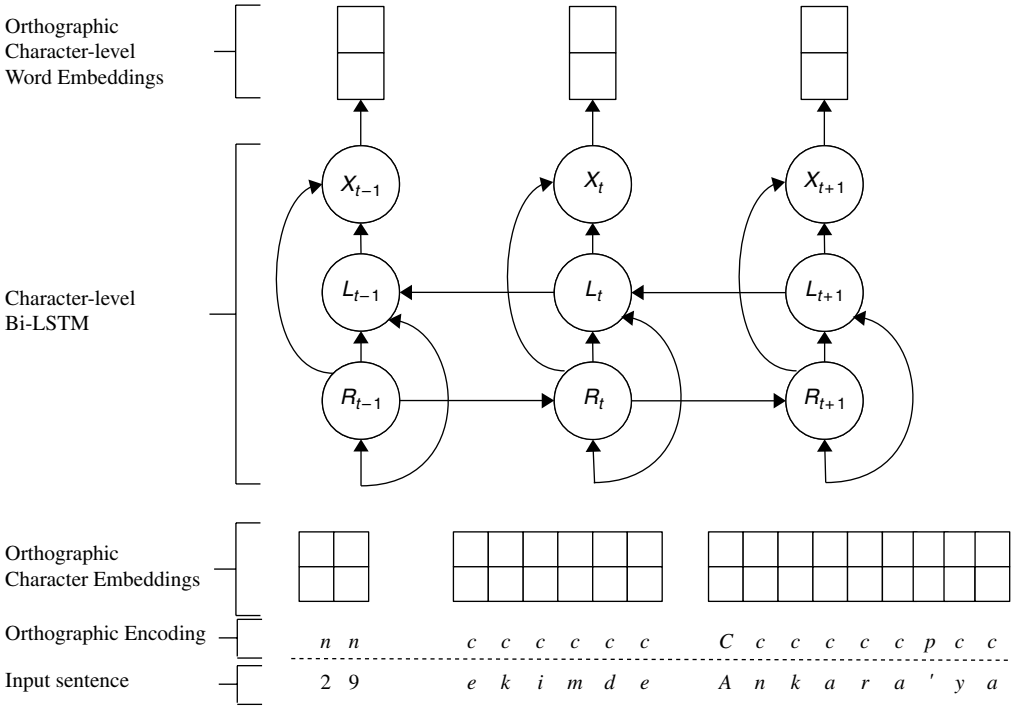

Figure 2. Character-level word embedding using a bidirectional LSTM.

https://doi.org/10.1017/S1351324919000627 Published online by Cambridge University Press

3.3 Character n-gram-level word embeddings

Fasttext (Bojanowski *et al*. 2017) is an extension of word2vec (Mikolov *et al*. 2013), and it is comparably better at capturing word representation for morphologically rich languages such as Turkish. This is due to its ability to form vector representation of words from their vectors of character n-grams. As a result, this allows us to generate word embeddings $E_{w_i}^{(c_{ngram})}$ using n-grams even for out-of-vocabulary words, which is a common case for noisy text and also agglutinative languages.

3.4 Morpheme-level word embeddings

Morph2vec (Üstün *et al*. 2018) is another representation learning model that utilizes subword information to learn the word embeddings. The algorithm takes a list of candidate morphological segmentations of all words in the training data that are suggested by an unsupervised morphological segmentation system (Üstün and Can 2016). Given that each word has multiple sequences of candidate morphological segmentations, the final word representation $E_{w_i}^{(m)}$ is a weighted sum of the morpheme-level word embeddings of all segmentations of that word. An attention mechanism is used on top of the model in order to learn the weights, where the mechanism assigns more weight to the correct segmentation of the word. We incorporate morpheme-level word embeddings that we obtain from pretrained morph2vec embeddings in our proposed models in this article.

It can be argued that words in an informal text may not have proper morphemes. For example, *"gidiyorum"* in Turkish (means *"I am going"*) is usually written as *"gidiyom"* by combining the present participle suffix *-iyor* with the person ending *-um*. However, morph2vec (Üstün *et al*. 2018) builds the word embeddings from several segmentations of the word that are likely to include the portions of the suffixes in the corrupted form.

3.5 Word-level word embeddings

Word2vec (Mikolov *et al*. 2013) has been one of the leading word representation methods that has shown superior performance in capturing syntactic and semantic features of words. The method aims to estimate word embeddings $E_{w_i}^{(w)}$ using their contextual information similar to other aforementioned methods, but it does not make use of any subword information and all words are considered as distinct tokens.

3.6 Final word embeddings

The final word embeddings that we use as input to the proposed models are the concatenation of fasttext (Bojanowski *et al*. 2017), morph2vec (Üstün and Can 2016), word2vec (Mikolov *et al*. 2013), character-level word embeddings, and orthographic character-level embeddings (either CNN-based or LSTM-based):

$$E_i = E_{w_i}^{(w)} \circ E_{w_i}^{(c_{ngram})} \circ E_{w_i}^{(m)} \circ E_{w_i}^{(c)} \circ E_{w_i}^{(c_{cnn|Bi-LSTm})} \tag{1}$$

An overview of the approach is given in Figure 3. Each vertical stacked box represents a different level of word embedding for the given input word.

After concatenating the different-level word embeddings, we apply *dropout* on the final word embedding E_i. This prevents the model from solely depending on one type of word embedding and, therefore, ensures a better generalization. We assign dropout rate $r = 0.5$.

https://doi.org/10.1017/S1351324919000627 Published online by Cambridge University Press

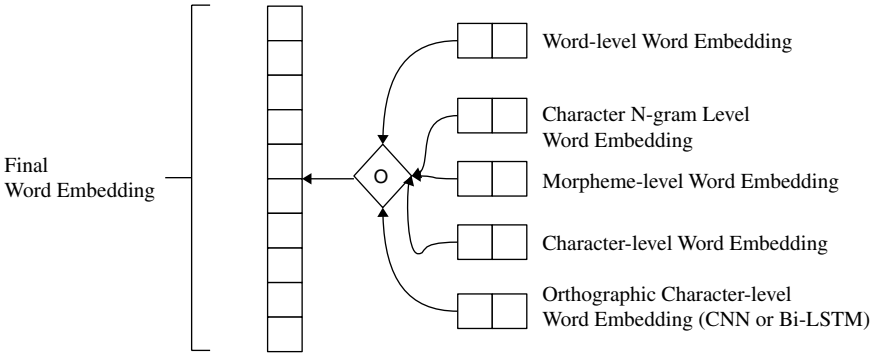

Figure 3. Overview of the final word embeddings. After concatenating embeddings obtained from *fasttext*, *word2vec*, *morph2vec*, and character-level word embeddings, orthographic character-level embeddings, we apply dropout for better generalization.

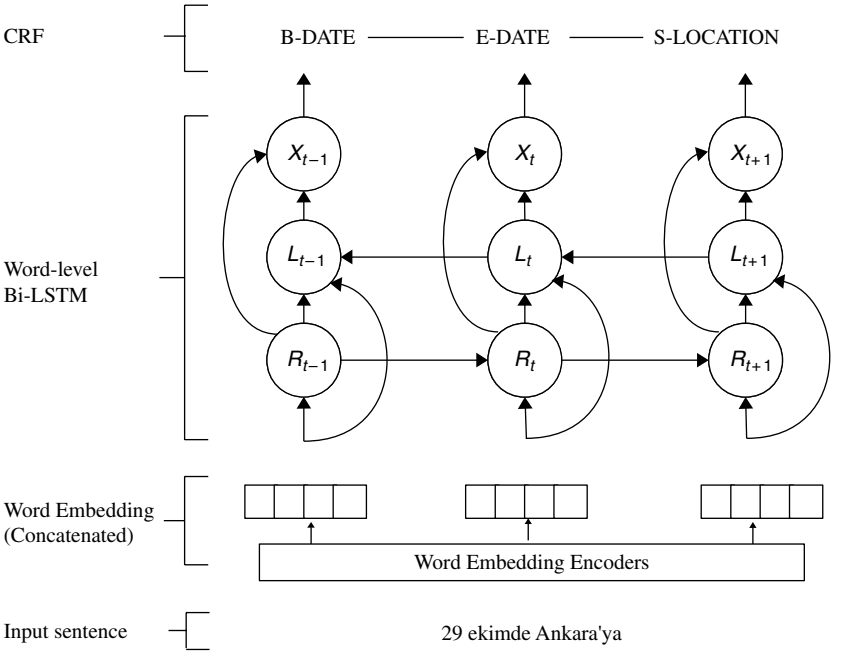

Figure 4. Architecture of our baseline Bi-LSTM-CRF model. We learn latent features by using a Bi-LSTM that is fed by the combined word embeddings and then we feed the output of each Bi-LSTM state to CRF in order to predict the label sequence. Here, Word Embedding Encoders are namely *word2vec* (Mikolov *et al*. 2013), *fasttext* (Bojanowski *et al*. 2017), *morph2vec* (Üstün and Can 2016), character-level word embedding, and orthographic character-level embedding methods. The Turkish input sequence *"29 ekimde Ankara'ya"* means *"To Ankara on 29th October"*.

4. Baseline model

Our baseline model is founded on the well-known bidirectional LSTM-CRF (Bi-LSTM-CRF) model proposed for sequence labeling, which is similar to that of Huang *et al.* (2015), Chiu and Nichols (2015), Lample *et al*. (2016), Ma and Hovy (2016).

A Bi-LSTM is fed by the final word embeddings E_i in order to learn the higher order latent features for the NER task, and another layer with a linear-chain CRF is fed by the LSTM outputs of each word to compute a prediction of the label sequence. Overview of the baseline model is given in Figure 4. Word embeddings are encoded as given in Figure 3.

https://doi.org/10.1017/S1351324919000627 Published online by Cambridge University Press

4.1 Bidirectional LSTM layer

Given an input sentence (i.e., tweet) $S = \{w_1, w_2, \ldots, w_n\}$, bidirectional LSTM is used to process the words sequentially. To this end, the combined word embedding E_i of each word in the sentence is given as input to the bidirectional LSTM layer that is composed of a forward LSTM $LSTM_{forward}$ and a backward LSTM $LSTM_{backward}$ (Hochreiter and Schmidhuber 1997). Latent feature vectors $\overrightarrow{R_t}$ and $\overleftarrow{L_t}$ are learned as an output of the LSTMs at time step t:

$$\overrightarrow{R_t} = LSTM_{forward}(E_{1:N}, t) \tag{2}$$

$$\overleftarrow{L_t} = LSTM_{backward}(E_{1:N}, t) \tag{3}$$

The outputs of the LSTMs are concatenated to build a single vector output from the Bi-LSTM as follows:

$$X_t = \overrightarrow{R_t} \circ \overleftarrow{L_t} \tag{4}$$

where X_t denotes the concatenated output vector for each word. We also apply dropout on X_t for a better generalization. Weights of the Bi-LSTM are initialized using *uniform Glorot initialization* (Glorot and Bengio 2010) that initializes the weights by drawing samples from a Gaussian distribution with mean = 0.0 and variance based on the fan-in (input units in the weight tensor) and fan-out (output units in the weight tensor) of the weight.

4.2 CRF layer

We use a linear-chain CRF to predict the sequence of labels $Y = (y_1, y_2, ..., y_N)$ for the sentence S where y_i denotes the named entity label of the ith word w_i in S. The prediction score of a sequence is defined as follows:

$$C(S, Y) = \sum_{i=0}^{N} A_{y_i, y_{i+1}} + \sum_{i=1}^{N} P_{i, y_i} \tag{5}$$

where the score is estimated over a sequence of size N. Here, P is the matrix of scores that is the output by the Bi-LSTM and A is the matrix that denotes the transitions from the previous label to the next label. P has a size of $N \cdot k$, where k is the number of the distinct entity tags. The concatenated representation of each word X_t is linearly projected onto a layer that has a size of k. Therefore, the matrix defines the scores of labeling each word in the sequence with the possible k tags, which is not a proper probability distribution yet. In other words, P_{i,y_i} is the score of the tag y_i for a given a word w_i. By defining a log-linear model using the scores, the probability of the output sequence of Y becomes:

$$p(Y|S) = \frac{e^{C(S,Y)}}{\sum_{\widetilde{Y} \in Y_S} e^{C(S,Y)}}, \tag{6}$$

where Y_S denotes the set of possible label sequences for S. Finally, the goal becomes to maximize the log probability of the predicted label sequence. Building the log-linear model gives us the form:

$$log(p(Y|S)) = C(S, Y) - log\left(\sum_{\widetilde{Y} \in Y_S} e^{C(S,\widetilde{Y})}\right) \tag{7}$$

The correctly predicted sequence of labels is the one that maximizes Equation 7:

$$\arg\max_{\widetilde{Y} \in Y_s} C(S, \widetilde{Y}). \tag{8}$$

Weights of the CRF layers are initialized using *uniform Glorot* distribution. Both the parameter estimation and decoding are performed by dynamic programming.

https://doi.org/10.1017/S1351324919000627 Published online by Cambridge University Press

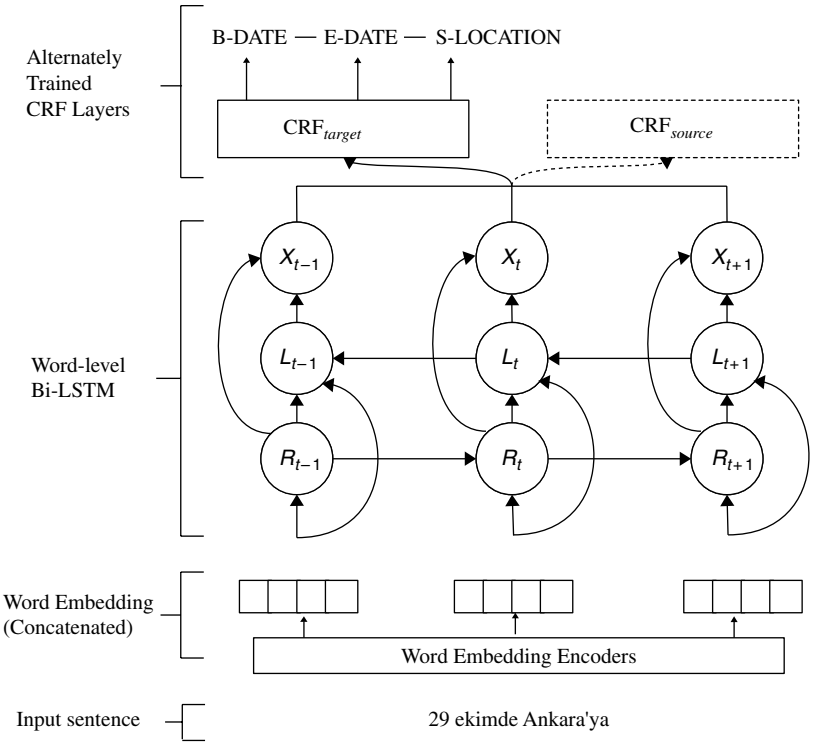

Figure 5. Overview of the transfer learning model that incorporates an additional CRF layer. CRF layers are alternately trained on different datasets so that the shared layers learn from both datasets and therefore learning can be transferred from the source dataset to the target dataset. The Turkish input sequence *"29 ekimde Ankara'ya"* means *"To Ankara on 29th October"*.

5. Transfer learning

The amount of annotated Turkish noisy text is considerably limited. This prevents the basic baseline model from learning especially some of the infrequent types such as DATE, TIME, and PERCENTAGE. To overcome this problem, we incorporate another CRF layer (CRF_{source}) that is trained on a different, but preferably a larger dataset (i.e., source dataset), in addition to the CRF layer (CRF_{target}) that is trained on a small amount of noisy text (i.e., target dataset). Therefore, the model learns from both datasets jointly.

The architecture of the baseline transfer learning model is given in Figure 5. As shown in the figure, lower layer that involves the word-level Bi-LSTM is shared by two CRFs. The embeddings are also shared by both CRF layers. However, the CRF layer involves two independent CRFs, where one of them is trained on the formal text and the other one is trained on the noisy text. Therefore, we transfer the dependencies learnt from the larger and formal text toward the noisy text gradually. The training procedure is performed by doing the gradient updates through each CRF layer alternately. In other words, in every other iteration, the output of one CRF layer is considered to perform the gradient update based on its loss by discarding the output of the other CRF layer. Therefore, the outputs of both CRFs are used alternately, where both CRF outputs are gradually optimized in time. In this way, using the knowledge transferred from the larger text, some dependencies between rare entity types and rare words are also learnt for the noisy text.

This model is an adaptation of the cross-domain transfer learning model proposed by Yang, Salakhutdinov, and Cohen (2017). In their work, the authors introduce various transfer learning architectures for cross-domain, cross-application, and cross-lingual transfer. We adapt the

https://doi.org/10.1017/S1351324919000627 Published online by Cambridge University Press

cross-domain transfer learning architecture by introducing the parameter sharing in the word-level Bi-LSTM, where each domain learns a separate CRF layer. However, the LSTMs are shared across different domains. Cotterell and Duh (2017) also apply a similar transfer learning scheme for low-resource NER with a shared Bi-LSTM across different languages with language-specific CRFs. We particularly used the Bi-LSTM-CRF architecture and not a single Bi-LSTM as suggested by Riedl and Padó (2018) for transfer learning because the best results have already been achieved by this architecture without transfer learning (Reimers *et al.* 2014; Ma and Hovy 2016; Cotterell and Duh 2017).

We further extended the transfer learning architecture by adding extra shared layers on the baseline architecture. Following the various architectures proposed by von Däniken and Cieliebak (2017), we added two ReLUs, a dropout layer, and a linear layer (a feed-forward network) between the Bi-LSTM and the CRF layers. The architecture of the model is given in Figure 6. First, X_t, the output of the Bi-LSTM is passed through an ReLU (Nair and Hinton 2010) layer. Then, a dropout is applied to X_t. The dropout applied output X_t is then passed through a feed-forward network with one hidden layer and ReLU activation, which outputs a score for possible k number of entity tags:

$$score_t = W_2 \cdot ReLU(W_1 h_t + b_1) + b_2 \tag{9}$$

where $W_1 \in R^{d_H \times d_X}$, $b_1 \in R^{d_H}$, $W_2 \in R^{k \times d_H}$, and $b_2 \in R^k$ are the weights of the feed-forward network. Here, d_H is the dimension of the hidden layer and d_X is the dimension of X_t. As seen from the figure, all layers and their parameters are shared by both CRF layers. The motivation behind adding a feed-forward network between the Bi-LSTM layer and the CRF layer is to encode the outputs of the Bi-LSTM by introducing sparsity to lead the negative features to become zero. Otherwise, vanishing gradients problem stands out again due to the many layers that require back-propagation during gradient descent. Therefore, some outputs are forced to be zero by the ReLU unit, and the vanishing gradient problem is naturally solved in this multilayered architecture.

As for the training, analogously, we performed backpropagation using the loss of one of the CRF layers alternately. Therefore, the CRF layer gains generalization through two different datasets from different domains during training.

6. Experiments & Results

We did the experiments for the baseline and the transfer learning models on Turkish, and additionally on English to compare with other related work. First, we describe the datasets, the implementation details of the models, and the evaluation methods that we followed in this work, then we present the experimental results along with a discussion on the results.

6.1 Implementation details

Both the CNN-based ($E_{w_i}^{(c_{cnn})}$) and Bi-LSTM ($E_{w_i}^{(c_{Bi-LSTm})}$)-based orthographic character embeddings have a dimensionality of 30. The CNN-based character embeddings are initialized by uniform Glorot initializer. For the CNN model, 20 is assigned for the maximum word length, where the shorter words are padded with zeros and the longer ones are truncated. The Bi-LSTM-based character-level word representation has a dimensionality of 60.

We trained fasttext (Bojanowski *et al.* 2017) for Turkish with a learning rate of 0.025 for 4 epochs to learn the character n-gram-level word embeddings $E_{w_i}^{(c_{ngram})}$ that have a dimensionality of 200. Character n-gram-level word embeddings have a dimension of 300 for English.

Morpheme-level word embeddings $E_{w_i}^{(m)}$ have a dimensionality of 75 and 50 for English and Turkish, respectively. Word-level word embeddings $E_{w_i}^{(w)}$ have a dimension of 400 for both English and Turkish.

https://doi.org/10.1017/S1351324919000627 Published online by Cambridge University Press

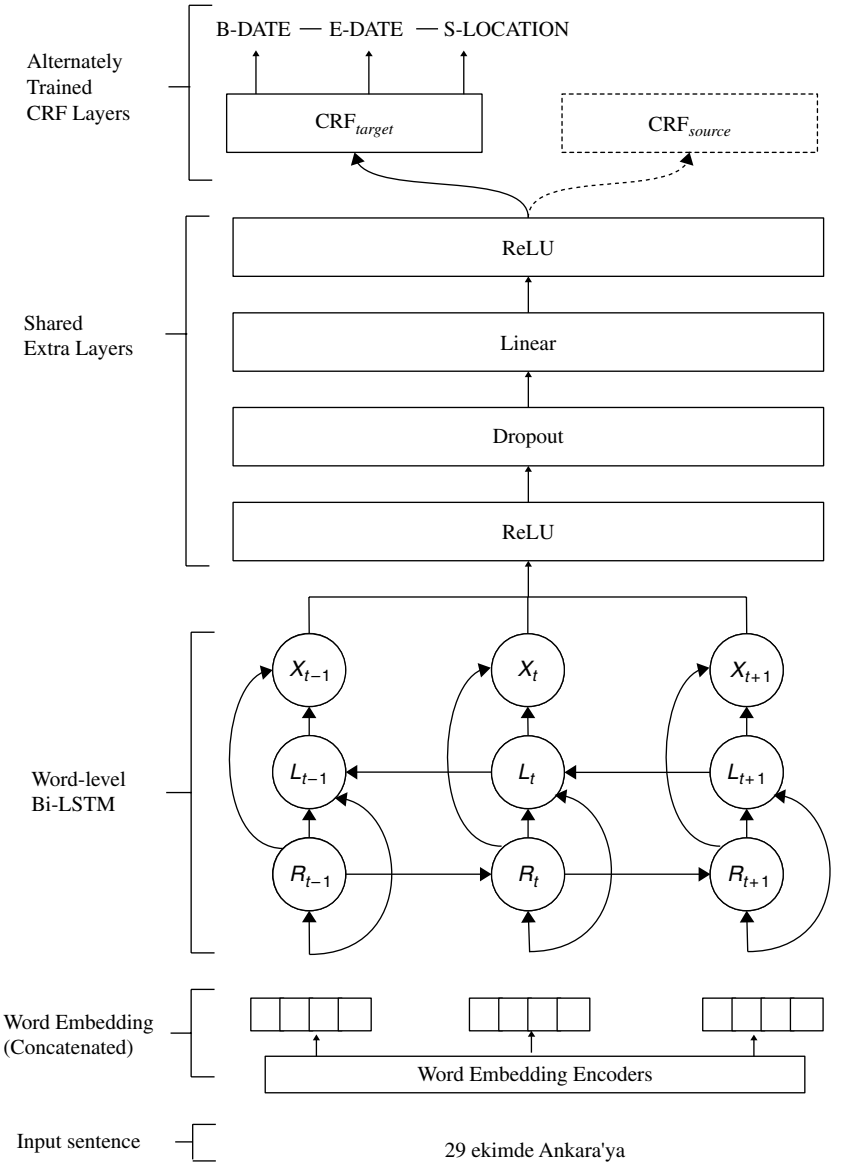

Figure 6. Overview of the extended transfer learning model that incorporates an additional CRF layer. CRF layers are alternately trained on different datasets so that the shared layers learn from both datasets and therefore learning can be transferred from the source dataset to the target dataset. The Turkish input sequence *"29 ekimde Ankara'ya"* means *"To Ankara on 29th October"*.

Weights of the shared ReLU and linear layers in transfer learning models are initialized using uniform Glorot initializer and biases are set to 0.

During all experiments, both the baseline and transfer learning models are trained using back-propagation and the parameters are optimized using Stochastic Gradient Descent algorithm. We trained both models for 100 epochs and set the learning rate to 0.005 in addition to using a gradient clipping of 5.0. Dropout rate of all of the dropout layers is set to 0.5. Hidden dimension of character-level Bi-LSTM and word-level Bi-LSTM layers are set to 30 and 250, respectively.

https://doi.org/10.1017/S1351324919000627 Published online by Cambridge University Press

Table 1. Implementation and training details

Setting/Hyperparameter	Value
Gradient clip	5.0
Learning rate	0.005
lr optimizer	sgd
Batch size	10
Dropout	05
Epochs	100
Hidden size$_{\text{Bi-LSTM(char)}}$	30
Hidden size$_{\text{Bi-LSTM(word)}}$	250
Dimension$_{\text{fasttext(en)}}$	300
Dimension$_{\text{morph2vec(en)}}$	75
Dimension$_{\text{fasttext(tr)}}$	200
Dimension$_{\text{morph2vec(tr)}}$	50
Dimension$_{\text{word2vec}}$	400
Dimension$_{\text{char}}$	30

Tuning the dimensions or any other hyperparameter did not significantly improve the accuracy of the models. An overview of the hyperparameters is given in Table 1.

All models are implemented using Tensorflow 1.8.0,[d] and the implementations and the related material are publicly available.[e]

6.2 Tagging scheme

When it is thought that a named entity can span multiple consecutive words, a tagging scheme that impose some constraints on determining the possible label of a word is highly useful. Inside, outside, beginning (IOB) format is such a tagging scheme that uses *B* for the token that refers to the beginning of a named entity, *I* for the token that refers to the inside of a named entity, and *O* for the token for other words in the sequence. Inside, outside, beginning, ending, single (IOBES) is a variant of IOB format that further restricts the possible label of a word with additional tokens such as *E* token that is used for specifying the ending of a named entity, and *S* token that is used for the named entities with only one word. Here is an example sentence tagged with the IOBES format:

*Mustafa/**B-PERSON** Kemal/**I-PERSON** Atatürk/**E-PERSON** was born in 1881/**S-DATE** in the former Ottoman/**B-ORGANIZATION** Empire/**E-ORGANIZATION**.*

We follow the IOBES tagging scheme for Turkish and IOB tagging scheme for English to be able to compare with other related work using the same annotated noisy text.

6.3 Datasets

In order to obtain Turkish character n-gram-level word embeddings, we trained Skipgram model of fasttext (Bojanowski *et al.* 2017) on a corpus of 20M Turkish tweets.[f] As for English, we used

[d] https://www.tensorflow.org/.
[e] All source code and related material are available on https://github.com/emrekgn/turkish-ner.
[f] http://www.kemik.yildiz.edu.tr/data/File/20milyontweet.rar

https://doi.org/10.1017/S1351324919000627 Published online by Cambridge University Press

Table 2. Datasets

#	Dataset	Named Entity types	# of tokens	# of NEs
DS-1	TR-tweet	ENAMEX, TIMEX, NUMEX corporation, creative-work, group, location,	55K	1.4K
DS-2	WNUT'17	person, product	104K	3.8K

Table 3. Number of entity types in Turkish noisy dataset, *DS-1*

Entity type	Amount
Person	699
Location	230
Organization	363
Date	56
Time	20
Money	12
Percentage	3
Total	1,383

the pretrained English word embeddings that are provided by fasttext[g] (Bojanowski *et al*. 2017). The word embeddings are obtained from the Continuous Bag of Words model of fasttext trained on Common Crawl,[h] a website that provides web crawl data.

Pretrained word embeddings obtained from word2vec (Mikolov *et al*. 2013) are used to learn the morpheme-level word embeddings by imitating them in morph2vec (Üstün *et al*. 2018).

We use pretrained word2vec (Mikolov *et al*. 2013) embeddings that are trained on a corpus that involves 400M English tweets (Godin *et al*. 2015). As for Turkish, we use pretrained word2vec embeddings that are trained on a news corpus (Boğaziçi University web corpus) that involves 423M words (Sak, Güngör, and Saraçlar 2008, 2011) and 20M Turkish tweets (Sezer, Sezer, and Ünivesitesi 2013).

We experimented on two datasets on Turkish and English that are given in Table 2. *DS-1* (Şeker and Eryiğit 2017) is the reannotated version of the initial Turkish noisy dataset (Çelikkaya *et al*. 2013) that consists of ENAMEX, TIMEX, and NUMEX types. As we can see in Table 3, this is a relatively small dataset with a highly imbalanced entity type distribution. Since the dataset does not have training and test splits, during experiments, we applied 10-fold cross-validation and split the dataset into training, test, and validation sets with ratios of 80%, 10%, and 10%, respectively for Turkish, and we did not apply a cross-validation for English to be able to compare our results with other work participated in the 3rd WNUT'17.[i]

DS-2 is an English noisy dataset (Derczynski *et al*. 2017) that is released by the 3rd WNUT'17 that includes *person, location, corporation, product (consumer goods, service), creative work (song, movie, tv series, book)*, and *group (music band, sports team, noncorporate organizations)* types. This dataset has training, test, and development sets with sizes of 65K, 23K, and 16K tokens. Distribution of the entity types in this dataset is also given in Table 4.

[g] https://s3-us-west-1.amazonaws.com/fasttext-vectors/crawl-300d-2M-subword.zip.
[h] https://commoncrawl.org/.
[i] https://noisy-text.github.io/2017/.

https://doi.org/10.1017/S1351324919000627 Published online by Cambridge University Press

Table 4. Number of entity types in English noisy dataset, *DS-2*

Entity type	Train	Development	Test	Total
Person	660	470	429	1,559
Location	548	74	150	772
Corporation	221	34	66	321
Product	142	114	127	383
Creative work	140	104	142	386
Group	264	39	165	468
Total	1,975	835	1,079	3,889

6.4 Preprocessing

Prior to tokenization of the datasets,

- We replaced the URLs (tokens starting with *http*) with a special token. This allows us to reduce sparsity and allows our model to converge relatively faster.
- We replaced the Twitter mentions (Twitter usernames starting with @ sign) with another special token in DS-1. This reduced the number of *PERSON* entities from 4256 to 699, and we believe this prevents memorizing the mentions in the text.

6.5 Evaluation methods

We evaluate the results with accuracy, precision, and recall. Accuracy measures the overall performance of the model by computing the ratio of correctly labeled tokens to the total number of tokens. However, this results in a highly imbalanced value since most of the tokens are not part of a named entity and, therefore, labeled as OTHER. Precision gives the ratio of correctly labeled named entities (chunks) to the total label predictions, and recall measures the ratio of correctly labeled named entities (chunks) to the total number of correct predictions. Finally, F1 score is computed as the harmonic mean of precision and recall:

$$F1 = \frac{2 * precision * recall}{(precision + recall)} \tag{10}$$

In order to measure the overall performance of any given model for the sequence labeling task, F1 score is commonly chosen over accuracy since it intuitively defines a good measure of the model by taking false negatives and false positives into account, whereas accuracy gives imbalanced results due to highly skewed entity type distribution because most of the tokens do not have an entity label.

6.6 Experimental results on Turkish

We experimented with different combinations of embedding methods to analyze the impact of the word and subword embedding methods used in the baseline and the transfer learning models. To this end, we used word-based word embedding method word2vec (Mikolov *et al.* 2013), character n-gram-level word embedding method fasttext (Bojanowski *et al.* 2017), morpheme-level word embedding method morph2vec (Üstün and Can 2016), character embeddings trained with a Bi-LSTM (and CNN), and orthographic character-level embeddings trained on a character-level Bi-LSTM.

https://doi.org/10.1017/S1351324919000627 Published online by Cambridge University Press

Table 5. Overview of the experimental results of the baseline models on the Turkish noisy dataset, *DS-1*. *Baseline-2* uses extra layers in the Bi-LSTM CRF model. Fasttext (Bojanowski *et al*. 2017), morph2vec (Üstün *et al*. 2018), word2vec Mikolov *et al*. (2013), character-level and orthographic embeddings are denoted in the embeddings column by *ft*, *m2v*, *w2v*, *char* and *ortho* respectively. Acc refers to accuracy, P refers to Precision, and R refers to Recall.

Model	Embeddings	Acc. (%)	P (%)	R (%)	F1 (%)
baseline	char	96.81	48.16	18.05	26.12
baseline	m2v	96.66	63.68	11.34	19.14
baseline	m2v, char	96.70	47.78	14.29	21.88
baseline	m2v, ortho	96.72	60.6	14.11	22.81
baseline	ft	97.69	72.42	49.79	58.91
baseline	ft, char	97.7	74.25	48.69	58.69
baseline	ft, ortho	97.73	71.96	51.10	59.7
baseline	ft, m2v	97.67	71.36	50.02	58.73
baseline	ft, m2v, char	97.72	74.48	49.11	59.03
baseline	ft, m2v, ortho	97.68	73.12	48.70	58.32
baseline	ft, m2v, ortho (cnn)	97.69	74.00	49.46	59.07
baseline	w2v	96.73	68.49	13.23	22.02
baseline	w2v, char	96.77	53.28	19.57	28.45
baseline	w2v, ortho	96.69	60.64	16.57	25.8
baseline	w2v, m2v	97.08	65.34	27.63	38.68
baseline	w2v, m2v, char	97.13	63.08	30.31	40.78
baseline	w2v, ft	97.77	71.29	53.26	60.82
baseline	w2v, ft, char	97.75	70.91	53.19	60.58
baseline	w2v, ft, m2v	97.68	73.49	47.45	57.53
baseline	w2v, ft, m2v, char	97.80	70.71	52.31	59.98
baseline	w2v, ft, m2v, ortho	**97.81**	**73.65**	**52.99**	**61.53**
baseline-2	w2v, ft, m2v, ortho	97.51	69.00	53.00	60.15

The results obtained from the baseline model are given in Table 5. In the baseline model, among using only one type of embedding, fasttext performs the best compared to other embedding types with an F1 measure of 58.91%, where CNN-based orthographic char embeddings perform 26.12%, morph2vec performs 19.14%, and word2vec performs 22.02%. This shows that using character n-grams in representation learning can cope with the sparsity issue better compared to other embedding types. We were expecting a similar performance from the morph2vec embeddings; however, they have not performed as well as fasttext. This might be a sign of ill-formed nature of the noisy text, where the morphemes are degenerated. Since morph2vec is trained on a formal text with exact morpheme boundaries, the noisy text could not benefit from the morphological knowledge adequately.

When the contribution of other embeddings used along with fasttext embeddings is observed, we see that orthographic features contribute the most with an improvement of 0.79% and the other embedding types do not contribute to the performance of the model and rather they degrade the results. We believe that since fasttext embeddings also contain character-level and morpheme-level features, those embeddings do not provide a significant improvement on the

https://doi.org/10.1017/S1351324919000627 Published online by Cambridge University Press

Table 6. Experimental results of the baseline model with fasttext (Bojanowski *et al*. 2017), morph2vec (Üstün *et al*. 2018), word2vec (Mikolov *et al*. 2013) and orthographic character-level embeddings on Turkish noisy dataset, *DS-1*

Entity type	Precision (%)	Recall (%)	F1 score (%)
person	70.61	54.10	61.17
organization	83.28	63.81	71.85
location	68.36	46.28	54.84
date	45.71	27.57	32.85
time	0	0	0
money	0	0	0
percentage	0	0	0
overall	**73.65**	**52.99**	**61.53**

fasttext embeddings. The results are also similar when more embeddings are combined with fasttext embeddings, which is due to a similar reason.

Because of the morphological structure of Turkish, using solely word2vec trained word embeddings does not perform very well. Combining the word embeddings with character embeddings or orthographic embeddings improves the scores, although the final scores are still below 30%. Using morph2vec along with word2vec provides a better improvement compared to character-level word embeddings and orthographic embeddings with an F1 measure of 38.68%. The highest improvement is obtained, when word2vec is combined with fasttext and it gives an F1 measure of 60.82%.

The highest performance is obtained when all embedding types (fasttext, word2vec, morph2vec, and orthographic encoding) are used together, which gives an F1 measure of 61.53%. The highest scores obtained for different entity types are given in Table 6. Our baseline model fails to label the infrequent types such as *time*, *money*, and *percentage* since the annotated noisy data are too small to learn the latent features for such infrequent entity types. However, the frequent entity types such as *person* and *organization* are learned well compared to *location* and *date*.

In order to transfer any learned features from another larger dataset, we added an extra CRF layer where the Bi-LSTM layers are shared by both datasets as described in Section 5. We call this model *transfer learning 1*. We trained the model alternately with different datasets in each epoch. Therefore, the shared layers up to the CRF layers can learn from both of the datasets. As a larger dataset (*source dataset*), we used the reannotated version of the Turkish news corpus with 492K tokens, which was originally provided by Tür *et al*. (2003) and reannotated by Şeker and Eryiğit (2017). The results obtained from the transfer learning models are given in Table 7. By using orthographic character-level word embeddings, character n-gram-level word embeddings, morpheme-level word embeddings, and word-level word embeddings, we obtained an F1 score of 66.17% that is better than the baseline model that incorporates all embedding types.

We also incorporated additional ReLU and linear layers between the Bi-LSTM and CRF layers as described in Section 5. We call the extended model as *transfer learning 2*. The results obtained from the transfer learning model are coherent with the results of the baseline model. Fasttext embeddings perform the best with an F1 measure of 62.47%, whereas using the other embedding types on its own perform comparably poorer similar to the baseline model. Morph2vec embeddings and character embeddings perform alike with F1 measures of 34.62% and 36.74%, respectively, which are still significantly better than the results obtained from the baseline model when those embeddings are used alone. This is possibly due to the inclusion of another larger dataset that compensates the sparsity issue in embeddings.

https://doi.org/10.1017/S1351324919000627 Published online by Cambridge University Press

Table 7. Overview of the experimental results of the transfer learning models on the Turkish noisy dataset, *DS-1*. Fasttext (Bojanowski *et al*. 2017), morph2vec (Üstün *et al*. 2018), word2vec Mikolov *et al*. (2013), character-level and orthographic embeddings are denoted in the embeddings column by *ft*, *m2v*, *w2v*, *char* and *ortho* respectively. *Transfer learning - 1* represents the basic transfer learning architecture without the additional (ReLU, linear) layers between the word-level Bi-LSTM and CRF layers and *transfer learning - 2* is the transfer learning model with additional ReLU and linear layers. Acc refers to accuracy, P refers to Precision, and R refers to Recall

Model	Embeddings	Acc. (%)	P (%)	R (%)	F1 (%)
transfer learning-1	w2v, ft, m2v, ortho	98.03	71.00	62.00	66.17
transfer learning-2	char	96.91	50.98	28.87	36.74
transfer learning-2	m2v	96.9	53.62	25.84	34.62
transfer learning-2	m2v, char	97.04	65.96	24.76	35.92
transfer learning-2	ft	97.77	69.8	56.94	62.47
transfer learning-2	ft, char	97.82	69.29	59.73	64.09
transfer learning-2	ft, m2v	97.81	67.78	61.48	64.27
transfer learning-2	ft, ortho	97.88	68.09	63.04	65.37
transfer learning-2	ft, ortho (cnn)	97.89	69.89	60.56	64.73
transfer learning-2	ft, m2v, ortho	97.87	70.78	60.35	65.12
transfer learning-2	ft, m2v, ortho (cnn)	97.95	74.45	58.94	65.72
transfer learning-2	w2v	96.80	65.82	16.20	25.86
transfer learning-2	w2v, char	96.95	59.97	23.77	33.75
transfer learning-2	w2v, m2v	97.38	65.38	40.15	49.60
transfer learning-2	w2v, m2v, char	97.45	66.28	40.73	50.34
transfer learning-2	w2v, ft	97.89	70.09	59.86	64.46
transfer learning-2	w2v, ft, char	97.91	69.47	61.66	65.18
transfer learning-2	w2v, ft, m2v	97.88	68.19	61.53	64.64
transfer learning-2	w2v, ft, m2v, char	97.86	68.58	60.38	64.19
transfer learning-2	w2v, ft, m2v, ortho	**98.00**	**71.79**	**63.9**	**67.39**

Interestingly, using character embeddings in addition to fasttext embeddings improves the F1 score from 62.47% to 64.09%, whereas in the baseline model adding character embeddings on fasttext embeddings did not make an impact. This is possibly due to the transfer of character embeddings between different domains. However, without transferring any character information between the domains, the fasttext emnbeddings seem to cover character embeddings and this hinders the impact of character embeddings against fasttext embeddings. Similar to the baseline results, using word2vec embeddings or character embeddings along with fasttext embeddings improves the scores by around 2%. Using orthographic embeddings along with fasttext embeddings also improves the scores by around 3%.

Using character embeddings in addition to fasttext and word2vec embeddings still improves the scores with an F1 measure of 65.18%, which was not the case in the baseline model. The highest score is obtained with an F1 measure of 67.39% when again the combination of all embedding types (word2vec, fasttext, morph2vec, orthographic embeddings) is used. Therefore, adding extra layers improved the results considerably.

As an alternative to orthographic character-level embeddings, we also incorporated the character-level embeddings that are trained on a character-level Bi-LSTM (by using the actual

https://doi.org/10.1017/S1351324919000627 Published online by Cambridge University Press

Table 8. Experimental results of transfer learning model (transfer learning - 2) with fasttext (Bojanowski *et al*. 2017), morph2vec (Üstün *et al*. 2018), word2vec (Mikolov *et al*. 2013) and ortographic character-level embeddings on Turkish noisy dataset, *DS-1*

Entity type	Precision (%)	Recall (%)	F1 score (%)
person	71.20	65.52	67.95
organization	82.05	70.23	75.16
location	67.05	64.29	65.27
date	40.47	35.69	35.86
time	10.00	4.00	5.71
money	5.00	6.66	5.71
percentage	0	0	0
overall	**71.79**	**63.9**	**67.39**

Table 9. Experimental results obtained from different dimensions of fasttext character n-gram-level word embeddings (Bojanowski *et al*. 2017)

Dimensionality	Accuracy	Precision (%)	Recall (%)	F1 score (%)
dimensions	97.17	74.26	27.56	40.05
dimensions	97.52	74.42	39.86	51.86
dimensions	97.60	73.36	44.94	55.53
200 dimensions	**97.69**	**72.42**	**49.79**	**58.91**
dimensions	97.68	72.93	48.50	58.15

characters this time instead of replacing the characters with various symbols for the shape of the word) following the work of Lample *et al*. (2016). However, the results obtained from the character-level word embeddings performed comparably poorer.

Additionally, in order to analyze the impact of the additional layers, we performed a separate experiment for the baseline model with a single CRF layer without any transfer learning. The model is called *baseline-2* in Table 5. We used word2vec, fasttext, morph2vec, and orthographic embeddings in this setting. The baseline model with the additional layers gives 60.15% F1 score, which is lower than the baseline model without the additional layers using the same embeddings.

Table 8 presents the highest scores obtained for different entity types in the transfer learning model. We can see that the transfer learning model improves upon the results of the baseline model significantly. Although the overall results on rare entity types (such as *date*, *time*, *money*) are higher compared to the baseline model, transfer learning model still fails to label *percentage* but we believe that this is an expected outcome given that it has only three examples belonging to the percentage type in the whole dataset. Note that we are also using cross-validation so that number of times an entity type is seen in one iteration is further decreased.

We trained Skipgram model of fasttext (Bojanowski *et al*. 2017) on the same corpus of 20M Turkish tweets[j] for different dimensions of character n-gram-level word embeddings to analyze the impact of the character n-gram-level word embeddings' dimensionality. The results for different sizes of embeddings are given in Table 9. The results show that the scores improve with higher dimensionalities, where we obtain the highest scores with 200 dimensional fasttext embeddings.

[j] http://www.kemik.yildiz.edu.tr/data/File/20milyontweet.rar.

https://doi.org/10.1017/S1351324919000627 Published online by Cambridge University Press

Table 10. Comparison with related work on Turkish noisy dataset *DS-1*. All results are tested on the same noisy text and are therefore comparable with each other

Related work	F1 score (%)	Dataset
Şeker and Eryiğit (2017)	63.63	DS-1 v4
Çelikkaya *et al*. (2013)	19.28	DS-1 v1
Küçük and Steinberger (2014)	46.93	DS-1 v2
Eken and Tantuğ (2015)	28.53	DS-1 v3
baseline (ft, m2v, w2v, ortho)	61.53	DS-1 v4
baseline-2 (ft, m2v, w2v, ortho)	60.15	DS-1 v4
transfer learning-1 (ft, m2v, w2v, ortho)	66.17	DS-1 v4
transfer learning-2 (ft, m2v, w2v, ortho)	**67.39**	**DS-1 v4**

The results also support the findings of Yin and Shen (2018), where it was reported that the over-parametrization does not hurt the performance and the performance increases with the dimensionality up to a level, where it degrades slightly and converges so long as the dimensionality increases.

In all experiments, we used orthographic and character embeddings that have a dimensionality of 30. We did further experiments to analyze the impact of the dimensionality of the orthographic embeddings. We used 200 dimensional fasttext embeddings along with orthographic embeddings with different dimensions (30, 50, 100). However, the results did not change considerably, and the F1 score was always around 59–60%, which are in line with the previous results reported in Table 5.

Since we used pretrained word2vec embeddings that are already high dimensional (400), we did not perform further experiments to analyze the dimensionality of the word2vec embeddings. As Yin and Shen (2018) suggest, the higher dimensions of word embeddings perform better compared to lower dimensions to a certain extent.

As for the morph2vec (Üstün *et al*. 2018) embeddings, they are optimized by the authors using 50 and 75 for the morph vector dimensions for Turkish and English, respectively.

6.6.1 Comparison with related work on Turkish

We compare our results with the related work on Turkish noisy dataset *DS-1*. The results are given in Table 10. The related work uses different reannotated versions of the same dataset; therefore, named entity distributions within the datasets may slightly differ; however, the difference between the datasets is not very significant. Therefore, all results are comparable with each other. Şeker and Eryiğit (2017) present the latest version called *DS-1 v4*, which is also used in our experiments. Note that, we replaced any Twitter mentions (*number of mentions labelled as person: 3557*) in the dataset prior to training. We compare our model with the results of Şeker and Eryiğit (2017) that are obtained by replacing the mentions in the tweets, so that we can make a fair comparison. Additionally, we compare our model with the models proposed by Çelikkaya *et al*. (2013), Küçük and Steinberger (2014), and Eken and Tantuğ (2015).

It should be noted that none of the related work on Turkish NER is designed particularly for noisy text. Therefore, those models are trained on formal text, and as an additional experimental setting, the authors also present their results on noisy text by using the Turkish noisy text (from *DS-1 v1* to *DS-1 v4*) only for testing purposes. Therefore, our training sets are different.

Our baseline model and the transfer learning model without the additional layers outperform the models proposed by Çelikkaya *et al*. (2013), Küçük and Steinberger (2014), and Eken and

https://doi.org/10.1017/S1351324919000627 Published online by Cambridge University Press

Tantuğ (2015) significantly with F1 measures of 61.53% and 66.17% respectively, whereas the highest score among the other works is 46.93%. The model proposed by Şeker and Eryiğit (2017) is slightly better with an F1 measure of 63.63%.[k] Nevertheless, our transfer learning model with extra layers outperforms all of the models with an F1 score of 67.39%.

6.6.2 Error analysis

We did a qualitative error analysis to examine the common errors in the results. Frequent person names are usually tagged correctly. However, if they are not frequent or if they are spelled with multiple vowels to give a shouting effect (e.g., *Tülaaaaaaaayy*, where the correct name is *Tülay*), then they may not be tagged correctly. In some circumstances, the location names are also tagged as PERSON especially when the person names are followed straight away by location names. This mistagging does not occur when organization names are followed by location names.

Another frequent error type occurs when the organization names span across few words. Those organization names are usually confused with the location names. This mistagging also occurs when the organization name is not frequent enough. Another interesting usage is seen with the organization names that are shortened by the name of the location since the organization belongs to that location. For example, instead of using *Trabzonspor* (the football team that belongs to the city *Trabzon*), it is shortened to *Trabzon* to refer to the team. This requires more information to extract the correct meaning of the named entity and usually such names are mistagged by our models. Those are the typical tagging errors of the organization entries. Apart from these, frequent and single word organization names are tagged correctly by the models. Abbreviated organization names are also tagged correctly whether or not capitalized (e.g., *"FB"* for the football team name *"Fenerbahçe"*, *"gs"* for the football team name *"Galatasaray"*). Even some organization names that include spelling errors are tagged correctly by our model. However, some of the misspelled organization names are not tagged as *organization*, but instead tagged as *other* in the gold data. Therefore, although those organization names are tagged correctly by our model, they are counted wrong. For example, *Fenev* (the name of the football team *Fener* is misspelt) is tagged as *organization* correctly by our model.

Location names that span across few words also usually cannot be identified properly, and only the first word is tagged correctly. Infrequent location names are also tagged incorrectly. Another error occurs because of the non-ASCII characters in the location names. Since we do not perform any preprocessing on the data, those location names also cannot be identified correctly.

The inflection of named entities also have a significant impact on tagging. The inflectional morphemes such as case markers or possessive morphemes are seen frequently with the location names. To our observation, the frequent inflectional morphemes do not affect the tagging. For example, *"samsunsporuma"* (means *"to my team samsunspor"*) is tagged correctly even though it has got two inflectional suffixes (i.e., *"um"* for *"my"* and *"a"* for *"to"*). However, infrequent morphemes lead to mistagging with the location names. Person names are also sometimes inflected with the suffix *ciğim* (means "dear" and usually abbreviated as *cim* in the informal text) as a salutation and they cannot be tagged correctly.

The infrequent named entities are learned better in transfer learning, which is an expected result. Even some of the frequent named entities that are inflected can be correctly tagged in transfer learning model. Otherwise, the errors are common in baseline and transfer learning. Therefore, the main contribution of transfer learning is the compensation of the infrequent named entities using a larger corpus. When we also compare the results obtained from different levels of word embeddings, it shows that using subword information improves the tagging significantly.

[k] Although 67.96% is reported by Şeker and Eryiğit (2017), this score is obtained by including Twitter mentions in both training and test data. Twitter mentions appear in almost any tweet, which are easy to detect and therefore increase the scores naturally. Therefore, we compare our results with their score without using Twitter mentions to have a fair comparison.

https://doi.org/10.1017/S1351324919000627 Published online by Cambridge University Press

Table 11. A list of incorrect tags in Turkish

	Examples
Predicted	Ege\ S-LOCATION Üniversitesi\ O Bölümü\ O
Correct	Ege\ B-LOCATION Üniversitesi\ I-LOCATION Bölümü\ E-LOCATION
Predicted	Doğan\ B-PERSON Diyarbakr\ E-PERSON 5\ O Nolu\ O Cezaevinde\ O
Correct	Doğan\ S-PERSON Diyarbakr\ S-LOCATION 5\ B-LOCATION Nolu\ I-LOCATION Cezaevinde\ E-LOCATION
Predicted	Ziraat\ S-LOCATION Türkiye\ S-LOCATION Kupas \ O
Correct	Ziraat\ B-ORGANIZATION Türkiye\ I-ORGANIZATION Kupas \ E-ORGANIZATION
Predicted	istanbul\ S-LOCATION şehir\ S-LOCATION tiyatrolarnda\ O
Correct	istanbul\ B-ORGANIZATION şehir\ I-ORGANIZATION tiyatrolarnda\ E-ORGANIZATION
Predicted	FENERBAHCEEE\ O
Correct	FENERBAHCEEE\ S-ORGANIZATION
Predicted	bu\ O hafta\ O cuma\ S-DATE
Correct	bu\ B-DATE hafta\ I-DATE cuma\ E-DATE
Predicted	Gizemcim\ O
Correct	Gizemcim\ S-PERSON

However, the subword information in noisy text does not need to be syntactic (morphological units) as suggested and character n-gram-level features help in tagging substantially.

As for the *date* label, the week days can be tagged correctly. However, analogously, if they span over multiple words, they cannot be identified.

A list of examples to errors in our Turkish results is given in Table 11.

6.7 Experimental results on English

We performed a similar set of experiments by combining various word representations to measure the effect of different word and subword representation levels for the English noisy text. Analogously, we employed word-based word embedding method word2vec (Mikolov *et al.* 2013), character n-gram-level word embedding method fasttext (Bojanowski *et al.* 2017), morpheme-level word embedding method morph2vec (Üstün and Can 2016), character embeddings trained with a Bi-LSTM (and CNN), and orthographic character-level embeddings trained on a character-level Bi-LSTM. The overview of the English results is given in Tables 12 and 13 for the baseline and the transfer learning models, respectively.

Among using solely character-level embeddings, morph2vec (Üstün and Can 2016), fasttext (Bojanowski *et al.* 2017), or word2vec (Mikolov *et al.* 2013), the highest results are obtained from word2vec (Mikolov *et al.* 2013) with an F1 measure of 39.04% in the surface level and an F1 measure of 36.55% in the entity level, which gives a completely different picture from the Turkish results where the highest score was obtained from fasttext with an F1 measure of 58.91%. The English results are both lower than that of Turkish, and moreoever word-level embeddings are more beneficial in English compared to Turkish. Due to the morphological divergence between the two languages, obtaining a better performance from word-level word embeddings is an expected result. However, the performance is still not satisfactory compared to the highest result in Turkish when using a single type of word embedding. Using orthographic character-level word embeddings in addition to word2vec contributed the most with an F1 measure of 40.55% in the surface level and 37.82% in the entity level. Although the other embedding types do not

https://doi.org/10.1017/S1351324919000627 Published online by Cambridge University Press

Table 12. The results of the baseline model on the English noisy dataset, *DS-2*. *Baseline-2* uses extra layers in the Bi-LSTM CRF model. Fasttext (Bojanowski *et al*. 2017), morph2vec (Üstün *et al*. 2018), word2vec (Mikolov *et al*. 2013), character-level and orthographic character-level embeddings are denoted in the embeddings column by *ft*, *m2v*, *w2v*, *char* and *ortho* respectively

		Entity level (%)				Surface form (%)			
Model	Embeddings	Accuracy	Precision	Recall	F1 Score	Accuracy	Precision	Recall	F1 Score
baseline	char	92.51	21.46	5.19	8.36	92.51	20.87	5.56	8.77
baseline	m2v	92.55	10.71	0.28	0.54	92.55	12	0.31	0.61
baseline	m2v, char	92.51	26.67	5.57	9.21	92.51	25.7	5.77	9.42
baseline	m2v, ortho	92.56	24.2	4.92	8.17	92.56	23.7	5.24	8.58
baseline	ft	93.1	52.6	7.51	13.15	93.1	51.43	7.55	13.16
baseline	ft, char	93.14	52.87	7.7	13.44	93.14	52.38	8.07	13.99
baseline	ft, ortho	93.31	47.54	15.21	23.05	93.31	45.02	14.68	22.13
baseline	ft, m2v	93.07	54.41	6.86	12.19	93.07	52.85	6.81	12.07
baseline	ft, m2v, char	93.1	46.67	7.79	13.35	93.1	46.11	8.07	13.74
baseline	ft, m2v, ortho	93.39	48.39	15.31	23.26	93.39	45.45	14.68	22.19
baseline	ft, m2v, ortho (cnn), w2v	94.14	66.23	28.39	39.74	94.14	65.87	26.1	37.39
baseline	ft, m2v, ortho, w2v	94.19	66.81	28.39	39.84	94.19	66.76	26.31	37.74
baseline	w2v	93.96	66.91	25.14	36.55	93.96	67.48	27.47	39.04
baseline	w2v, char	94	64.9	26.07	37.19	94	64.57	28.21	39.27
baseline	w2v, ortho	94.11	66.43	26.44	37.82	94.11	66.29	29.21	40.55
baseline	w2v, m2v	93.98	66.25	24.4	35.66	93.98	66.67	22.64	33.8
baseline	w2v, m2v, char	94.08	67.79	26.16	37.75	94.08	68.47	23.9	35.43
baseline	w2v, ft	93.96	65.87	25.42	36.68	93.96	65.59	23.38	34.47
baseline	w2v, ft, char	94.05	66.59	26.07	37.47	94.05	66.57	24.21	35.51
baseline	w2v, ft, m2v	94.08	66.59	26.25	37.66	94.08	65.71	24.11	35.28
baseline	w2v, ft, m2v, char	94.1	66.74	26.44	37.87	94.1	66.76	24.63	35.99
baseline-2	w2v, ortho	**94.3**	**64.51**	**30.52**	**41.44**	**94.3**	**64.81**	**33.33**	**44.02**

contribute on top of the word2vec embeddings in the surface level, character-level word embeddings, orthographic embeddings, and fasttext embeddings slightly contribute to the word-level word embeddings; however, the contribution is not more than 0.6%.

Combining word2vec embeddings with other embeddings obtained from different levels still does not change the results, and the highest results in the surface level still remain the same as the one obtained from using solely word2vec embeddings. However, in the entity level, the highest performance is obtained by using word2vec, fasttext, morph2vec, and orthographic word embeddings, which gives an F1 measure of 39.84%. This is around 3% higher than the results obtained from using solely word2vec. However, in the surface level, the highest results are obtained by using word2vec and orthographic character embeddings with an F1 score of 40.55%.

Without using any word-level word embeddings, the results are far behind the highest obtained score in English, and most of them are below 20%. This concludes that word-level word embeddings of a morphologically poor language such as English bear further information compared to

https://doi.org/10.1017/S1351324919000627 Published online by Cambridge University Press

Table 13. The results of the transfer learning model on the English noisy dataset, *DS-2*. Fasttext (Bojanowski *et al*. 2017), morph2vec (Üstün *et al*. 2018), word2vec (Mikolov *et al*. 2013), character-level and orthographic character-level embeddings are denoted in the embeddings column by *ft*, *m2v*, *w2v*, *char* and *ortho* respectively. *Transfer learning - 1* represents the basic transfer learning architecture without the additional (ReLU, linear) layers between the word-level Bi-LSTM and CRF layers and *transfer learning - 2* is the transfer learning model with additional Relu and linear layers

		Entity level (%)				Surface form (%)			
Model	Embeddings	Accuracy	Precision	Recall	F1 Score	Accuracy	Precision	Recall	F1 Score
transfer learning-1	ft, ortho	91.4	24.47	21.24	22.74	91.4	26.38	20.02	22.77
transfer learning-2	char	92.53	21.27	4.36	7.24	92.53	13.89	2.62	4.41
transfer learning-2	m2v, char	92.69	32.28	3.8	6.8	92.69	32.54	4.3	7.59
transfer learning-2	ft	93.84	70.47	22.36	33.94	93.84	69.04	20.34	31.42
transfer learning-2	ft, char	93.93	62.69	26.35	37.1	93.93	60.51	24.74	35.12
transfer learning-2	ft, m2v, ortho	93.63	48.81	22.91	31.19	93.63	46.98	21.17	29.19
transfer learning-2	ft, ortho, w2v	94.22	55.67	33.67	41.97	94.22	54.45	31.45	39.87
transfer learning-2	ft, m2v, ortho, w2v	94.17	57.01	33.21	41.97	94.17	56.14	31.13	40.05
transfer learning-2	w2v	93.97	60.32	31.17	41.1	93.97	59.87	34.46	43.74
transfer learning-2	w2v, char	94.14	60.51	30.98	40.98	94.14	60.09	34.58	43.9
transfer learning-2	w2v, ortho	94.05	55.98	32.56	41.17	**94.05**	**61.59**	**35.83**	**45.30**
transfer learning-2	w2v, m2v	93.77	53.87	30.98	39.34	93.77	54.56	28.83	37.72
transfer learning-2	w2v, m2v, char	94.12	59.89	30.89	40.76	94.12	59.14	28.83	38.76
transfer learning-2	w2v, ft	93.98	58.3	31.26	40.7	93.98	58.09	29.35	39
transfer learning-2	w2v, ft, char	94.19	61.76	31.91	42.08	94.19	62.12	30.08	40.54
transfer learning-2	w2v, ft, m2v	94.22	62.92	31.63	42.1	94.22	63.51	29.56	40.34
transfer learning-2	w2v, ft, m2v, char	**94.23**	**62.59**	**32.75**	**43.00**	94.23	62.21	30.71	41.12

other embedding types, and the other levels of word embeddings hardly contribute on top of the word-level word embeddings.

Here, orthographic character-level embeddings are trained on CNN instead of Bi-LSTM performed poorer, and thus, we used only Bi-LSTM-trained character-level word embeddings in all experiments on English.

The highest results obtained from different entity types are given in Table 14. We obtain the highest scores again for the most frequent entity types such as *person* and *location*, whereas the other sparse entity types such as *corporation*, *product*, *creative-work*, or *group* cannot be detected as accurate as the frequent types.

In both transfer learning models, we used the English noisy dataset released by the 2nd WNUT'16[1] as a *source dataset*. Therefore, both source and target datasets are noisy, but sizes of the datasets are different. The first transfer learning model, *transfer learning - 1*, has not improved upon the baseline and the results are even worse for this model. We obtained an F1 score of 22.77% for the entity level and 22.74% for the surface level from transfer learning - 1 by using fasttext embeddings and orthographic character-level word embeddings.

As for the transfer learning model with additional layers, the results are significantly improved upon the baseline model accordingly. For example, using solely word2vec embeddings in the

[1]https://noisy-text.github.io/2016/.

https://doi.org/10.1017/S1351324919000627 Published online by Cambridge University Press

Table 14. The results of the baseline model with word2vec (Mikolov *et al*. 2013), and orthographic character-level embeddings on English noisy dataset, *DS-2*

	Entity level (%)			Surface form (%)		
Entity type	Precision	Recall	F1 score	Precision	Recall	F1 score
corporation	29.41	15.15	20.00	28.00	14.89	19.44
creative-work	53.85	4.93	9.07	53.85	5.83	10.53
group	48.72	11.52	18.63	47.06	12.6	19.88
location	69.57	42.67	52.89	69.01	40.83	51.31
person	74.79	41.59	53.45	75.25	53.41	62.47
product	53.85	5.51	10.00	50.00	5.56	10.00
overall	**66.43**	**26.44**	**37.82**	**66.29**	**29.21**	**40.55**

Table 15. Experimental results of transfer learning model (transfer learning - 2) with word2vec (Mikolov *et al*. 2013) and ortographic character-level embeddings on English noisy dataset, *DS-2*

	Entity level (%)			Surface form (%)		
Entity type	Precision	Recall	F1 score	Precision	Recall	F1 score
corporation	37.84	21.21	27.18	40.00	21.28	27.78
creative-work	41.67	7.04	12.05	43.48	8.33	13.99
group	52.73	17.58	26.36	50.00	18.90	27.43
location	42.78	53.33	47.48	57.27	52.5	54.78
person	69.33	48.60	57.14	72.15	61.29	66.28
product	41.67	7.87	13.25	39.13	8.33	13.74
overall	**55.98**	**32.56**	**41.17**	**61.59**	**35.83**	**45.30**

baseline model gives an F1 measure of 39.04%, whereas it improves up to 43.74% in the surface level. The highest score is obtained with an F1 measure of 45.3%, when orthographic embeddings are combined with the word2vec embeddings, which was also the highest in the baseline model. Likewise, fasttext embeddings do not perform well on the transfer learning model for English. Therefore, the results obtained from baseline model transfer learning model for different levels of embeddings are coherent with each other.

We also performed another experiment with the baseline model with additional layers similar to Turkish, which is called *baseline-2* in Table 12. We used only word2vec and orthographic character embeddings in this setting, since it gives the highest score in the surface level for the baseline model without the additional layers. Using the additional layers improves the F1 score up to 44.02%, which is higher than F1 score of 40.55% obtained from the baseline model without the additional layers using the same embeddings. In Turkish, using additional layers in the baseline model does not help, whereas in English the additional layers contribute significantly.

Table 15 presents the highest obtained results for different entity types for the transfer learning model. It is clearly seen that transfer learning helps the model to learn rarely seen entity types better compared to the baseline model, thus the overall results on both entity level and surface forms are significantly improved.

https://doi.org/10.1017/S1351324919000627 Published online by Cambridge University Press

Table 16. Comparison with related work on English noisy dataset *DS-2*. All results are comparable with each other

	F1 score (%)	
Related work	Entity level	Surface form
Jansson and Liu (2017)	39.98	37.77
Williams and Santia (2017)	26.30	25.26
Sikdar and Gambäck (2017)	38.35	36.31
Lin *et al*. (2017)	40.42	37.62
von Däniken and Cieliebak (2017)	40.78	39.33
Aguilar *et al.* (2017)	**41.86**	40.24
baseline (w2v, ortho)	37.82	40.55
baseline-2 (w2v, ortho)	41.44	44.02
transfer learning-1 (w2v, ortho)	22.74	22.77
transfer learning-2 (w2v, ortho)	41.17	**45.30**

6.7.1 Comparison with related work on English

We present a comparison of our proposed models to the related work on English noisy dataset *DS-2*. The results are given in Table 16. Our transfer learning model with additional layers achieves competitive results for the entity level, whereas our baseline model and the transfer learning model with additional layers outperform all other models including the highest scoring models competed in WNUT'17, that are proposed by von Däniken and Cieliebak (2017) and Aguilar *et al.* (2017). The highest score in related work was achieved by Aguilar *et al.* (2017) with 40.24% F1 score. Our transfer learning model gives 45.30% F1 score for the surface forms using the word2vec and orthographic character-level embeddings. However, applying McNemar test[m] (McNemar 1947) between the model proposed by Aguilar *et al.* (2017) and our transfer learning-2 model does not strongly imply this difference ($p = 0.248$) in the surface level. If we compare the transfer learning-2 model with the transfer learning model proposed by von Däniken and Cieliebak (2017) in the surface level, McNemar test confirms the significance of this difference, $p < 0.05$. Therefore, our transfer learning-2 model significantly outperforms the transfer learning model of von Däniken and Cieliebak (2017) in the surface level. Additionally, our baseline model with additional layers (baseline-2) gives 41.44% F1 score for the entity level, which is competitive to that of Aguilar *et al.* (2017), where their highest reported result is 41.86% for the entity level. However, McNemar test between the baseline-2 and the model of Aguilar *et al.* (2017) shows that this difference is not significant in the entity level ($p = 1.0$). The same also applies for the difference between the transfer learning model of von Däniken and Cieliebak (2017) and baseline-2. On the other hand, McNemar test shows that the difference between baseline-2 model and the model proposed by Lin *et al.* (2017) is significant ($p < 0.05$).

It should be noted that both Aguilar *et al.* (2017) and von Däniken and Cieliebak (2017) make use of hand-crafted features such as capitalization or domain-specific knowledge such as gazetteers, whereas our models do not use any external resource.

[m] In particular, we applied McNemar–Bowker test that allows multiple categories in the results, whereas the original version of McNemar test allows only binary categories.

https://doi.org/10.1017/S1351324919000627 Published online by Cambridge University Press

Table 17. A list of incorrect tags in English

	Examples
Predicted	Living\ O Computer\ O Museum\ O
Correct	Living\ B-LOCATION Computer\ I-LOCATION Museum\ I-LOCATION
Predicted	Groep\ O Klein\ O
Correct	Groep\ B-PERSON Klein\ I-PERSON
Predicted	Supreme\ O Court\ O judge\ O
Correct	Supreme\ B-PERSON Court\ I-PERSON judge\ I-PERSON
Predicted	Great\ O Southern\ B-GROUP Television\ I-GROUP
Correct	Great\ B-CORPORATION Southern\ I-CORPORATION Television\ I-CORPORATION
Predicted	Snickers\ B-CORPORATION
Correct	Snickers\ B-PRODUCT
Predicted	Zealandia\ O
Correct	Zealandia\ B-LOCATION
Predicted	Amazon\ B-CORPORATION Echo\ O
Correct	Amazon\ B-PRODUCT Echo\ I-PRODUCT
Predicted	Turkish\ O military\ O
Correct	Turkish\ B-GROUP military\ I-GROUP
Predicted	rival\ O cannibal\ O car\ O gangs\ O
Correct	rival\ B-GROUP cannibal\ I-GROUP car\ I-GROUP gangs\ I-GROUP

6.7.2 Error analysis

Since English is not a morphologically rich language, the errors do not occur because of the inflection of the named entities. Instead, most of the errors are due to the sparsity in the data. The variety of the proper names (i.e., word types) in English data is quite intense compared to Turkish data. Hence, the infrequent named entities cannot be identified analogously to Turkish. For example, although the name *"Thomas Jane"* is correctly tagged as *person*, *"Groep Klein"* cannot be tagged correctly as *person* since it is not as frequent as the former.

Analogously, *location*, *organization*, and *person* names that span across multiple words and that are also infrequent cannot be tagged correctly. However, the frequent named entities with multiple words can be tagged correctly (e.g., *"Fly Community Theater"*). The English text is overcapitalized compared to Turkish text and even the common names could be capitalized. This leads to mistagging especially in *location* names. For example, *"Hotel Housekeepers Needed"* is incorrectly tagged as *location* because of the location word *"hotel"* that is capitalized. On the other hand, *"newzealand"* is tagged as *other*, since the word embeddings do not help in those multiword entities.

Corporation names are usually tagged as *group*. Moreover, product names and creative work are tagged as *corporation* in general. Since the number of those entities are not sufficient to be learned in the noisy text, they cannot be identified properly. For example, *corporation* names are identified correctly, if they are frequent (e.g., *"reddit"*). Most of the time, the group names cannot be identified and tagged as *other*. The same also applies for the product names. For example, *"Chevrolet Corvette"* is tagged correctly as *product*, whereas *"Centrelink"* and *"Sudocrem"* are tagged as *other*.

A list of examples to errors in our English results is given in Table 17.

https://doi.org/10.1017/S1351324919000627 Published online by Cambridge University Press

7. Conclusion & Future work

Various attempts have been made on Turkish NER recently. However, the results are still not satisfactory for noisy text. In this article, we have investigated using deep neural networks along with transfer learning instead of using rule-based or statistical approaches for NER on noisy Turkish text. Noisy text has its own difficulties because of the very sparse orthography of words that highly depend on the user style. Moreover, Turkish brings more challenges due to its morphologically rich structure, which introduces more sparsity in the text. We have investigated the effects of using different word and subword-level word representation methods such as word-level, character n-gram-level, morpheme-level, and orthographic character-level embeddings to mitigate the sparsity in text. We did not use any hand-crafted features and external resources unlike the other existing studies on Turkish NER on noisy text. We investigated transfer learning between a formal text and a noisy (informal) text in order to deal with the sparsity issue in the noisy text.

Therefore, we obtained the highest scores for Turkish NER on noisy text by using a combination of word-level and subword embeddings with transfer learning between a formal text and noisy text. We have also experimented with English as a relatively morphologically poor language and obtained the highest surface-level score and competitive entity-level scores on the English noisy dataset.

The results show that subword information plays a vital role for the NER task, especially on morphologically rich languages. More importantly, we can successfully learn valuable information without using hand-crafted features or domain-specific external resources. Furthermore, it is also proven that transfer learning approach can indeed effectively be used to tackle the problem of data scarcity.

Since our model is not domain and language specific, we believe that it can also be effectively trained and used for other morphologically rich languages, especially those with data sparsity problem. Therefore, experimenting for different languages remains as a future goal.

Supplementary materials

For supplementary material for this article, please visit https://doi.org/10.1017/S1351324919000627

References

Aguilar G., **Maharjan S.**, **Monroy A.P.L. and Solorio**, **T.** (2017). A multi-task approach for named entity recognition in social media data. In *Proceedings of the 3rd Workshop on Noisy User-Generated Text*, Copenhagen, Denmark. Association for Computational Linguistics, pp. 148–153.

Bikel D.M., **Miller S.**, **Schwartz R. and Weischedel R.** (1997). Nymble: A high-performance learning name-finder. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, ANLC'97. Stroudsburg, PA, USA: Association for Computational Linguistics, pp. 194–201.

Blei D.M., **Ng A.Y. and Jordan M.I.** (2003). Latent Dirichlet allocation. *Journal of Machine Learning Research* **3**, 993–1022.

Bojanowski P., **Grave E.**, **Joulin A. and Mikolov T.** (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics* **5**, 135–146.

Cao K. and Rei M. (2016). A joint model for word embedding and word morphology. In *Proceedings of the 1st Workshop on Representation Learning for NLP*. Association for Computational Linguistics, pp. 18–26.

Çelikkaya G., **Torunoğlu D. and Eryiğit G.** (2013). Named entity recognition on real data: A preliminary investigation for Turkish. In *2013 7th International Conference on Application of Information and Communication Technologies (AICT)*. IEEE, pp. 1–5.

Chiu J.P. and Nichols E. (2015). Named entity recognition with bidirectional LSTM-CNNs. arXiv preprint arXiv:1511.08308.

Cotterell R. and Duh K. (2017). Low-resource named entity recognition with cross-lingual, character-level neural conditional random fields. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, Taipei, Taiwan. Asian Federation of Natural Language Processing, pp. 91–96.

Derczynski L., **Nichols E.**, **van Erp M. and Limsopatham N.** (2017). Results of the WNUT2017 shared task on novel and emerging entity recognition. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, Copenhagen, Denmark. Association for Computational Linguistics, pp. 140–147.

Eken B. and Tantuğ C. (2015). Recognizing named entities in Turkish tweets. In *Proceedings of the Fourth International Conference on Software Engineering and Applications, Dubai, UAE*.

Glorot X. and Bengio Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In Teh, Y.W. and Titterington, M. (eds), *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, Chia Laguna Resort, Sardinia, Italy, volume 9 of *Proceedings of Machine Learning Research*. PMLR, pp. 249–256.

Godin F., Vandersmissen B., De Neve W. and Van de Walle R. (2015). Multimedia lab @ ACL WNUT NER shared task: Named entity recognition for Twitter microposts using distributed word representations. In *Proceedings of the Workshop on Noisy User-generated Text*. Association for Computational Linguistics, pp. 146–153.

Hochreiter S. and Schmidhuber J. (1997). Long short-term memory. *Neural Computation* **9**(8), 1735–1780.

Huang Z., Xu W. and Yu K. (2015). Bidirectional LSTM-CRF models for sequence tagging. arXiv preprint arXiv:1508.01991.

Jansson P. and Liu S. (2017). Distributed representation, LDA topic modelling and deep learning for emerging named entity recognition from social media. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pp. 154–159.

Küçük D. and Steinberger R. (2014). Experiments to improve named entity recognition on Turkish tweets. arXiv preprint arXiv:1410.8668.

Lample G., Ballesteros M., Subramanian S., Kawakami K. and Dyer C. (2016). Neural architectures for named entity recognition. arXiv preprint arXiv:1603.01360.

Landauer T.K., Foltz P.W. and Laham D. (1998). An introduction to latent semantic analysis. *Discourse Processes* **25**(2–3), 259–284.

Limsopatham N. and Collier N. H. (2016). Bidirectional LSTM for named entity recognition in Twitter messages. In *Proceedings of the 2nd Workshop on Noisy User-generated Text*, Osaka, Japan, pp. 145–152.

Lin B.Y., Xu F., Luo Z. and Zhu K. (2017). Multi-channel BiLSTM-CRF model for emerging named entity recognition in social media. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*. Association for Computational Linguistics, pp. 160–165.

Ma X. and Hovy E. (2016). End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. arXiv preprint arXiv:1603.01354.

McNemar Q. (1947). Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika* **12**(2), 153–157.

Mikolov T., Chen K., Corrado G. and Dean J. (2013). Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.

Nair V. and Hinton G.E. (2010). Rectified linear units improve restricted Boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10. USA: Omnipress, pp. 807–814.

Okur E., Demir H. and Özgür A. (2016). Named entity recognition on Twitter for Turkish using semi-supervised learning with word embeddings. In *LREC*.

Pagliardini M., Gupta P. and Jaggi M. (2017). Unsupervised learning of sentence embeddings using compositional n-gram features. arXiv preprint arXiv:1703.02507.

Pennington J., Socher R. and Manning C.D. (2014). Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar. Association for Computational Linguistics, pp. 1532–1543.

Petasis G., Vichot F., Wolinski F., Paliouras G., Karkaletsis V. and Spyropoulos C.D. (2001). Using machine learning to maintain rule-based named-entity recognition and classification systems. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, ACL'01, Stroudsburg, PA, USA: Association for Computational Linguistics, pp. 426–433.

Reimers N., Eckle-Kohler J., Schnober C., Kim, J. and Gurevych I. (2014). Germeval-2014: Nested named entity recognition with neural networks. In *Proceedings of the KONVENS GermEval Shared Task on Named Entity Recognition*, Hildesheim, Germany.

Riedl M. and Padó S. (2018). A named entity recognition shootout for German. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, Melbourne, Australia. Association for Computational Linguistics, pp. 120–125.

Sak H., Güngör T. and Saraçlar M. (2008). Turkish language resources: Morphological parser, morphological disambiguator and web corpus. In *Advances in Natural Language Processing*, pp. 417–427. Springer.

Sak H., Güngör T. and Saraçlar M. (2011). Resources for Turkish morphological processing. *Language Resources and Evaluation* **45**(2), 249–261.

Şeker G.A. and Eryiğit G. (2017). Extending a CRF-based named entity recognition model for Turkish well formed text and user generated content 1. *Semantic Web* **8**(5), 625–642.

Sezer B., Sezer T. and Ünivesitesi M. (2013). TS Corpus: Herkes için Türkçe derlem. In *Proceedings 27th National Linguistics Conference, May*, pp. 3–4.

Sikdar U.K. and Gambäck B. (2017). A feature-based ensemble approach to recognition of emerging and rare named entities. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, Copenhagen, Denmark. Association for Computational Linguistics, pp. 177–181.

https://doi.org/10.1017/S1351324919000627 Published online by Cambridge University Press

Suzuki J. and Isozaki H. (2008). Semi-supervised sequential labeling and segmentation using giga-word scale unlabeled data. In *Proceedings of ACL-08: HLT*, pp. 665–673. Association for Computational Linguistics.

Torunoğlu D. and Eryiğit G. (2014). A cascaded approach for social media text normalization of Turkish. In *Proceedings of the 5th Workshop on Language Analysis for Social Media (LASM)*, Gothenburg, Sweden. Association for Computational Linguistics, pp. 62–70.

Tür G., Hakkani-Tür D. and Oflazer K. (2003). A statistical information extraction system for Turkish. *Natural Language Engineering* **9**(2), 181–210.

Üstün A. and Can B. (2016). Unsupervised morphological segmentation using neural word embeddings. In Král P. and Martín-Vide C. (eds), *Statistical Language and Speech Processing*, pp. 43–53. Cham: Springer International Publishing.

Üstün A., Kurfal M. and Can B. (2018). Characters or morphemes: How to represent words? In *Proceedings of The Third Workshop on Representation Learning for NLP*, Melbourne, Australia. Association for Computational Linguistics, pp. 144–153.

von Däniken P. and Cieliebak M. (2017). Transfer learning and sentence level features for named entity recognition on tweets. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, Copenhagen, Denmark. Association for Computational Linguistics, pp. 166–171.

Williams J. and Santia G. (2017). Context-sensitive recognition for emerging and rare entities. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, Copenhagen, Denmark. Association for Computational Linguistics, pp. 172–176.

Wu Y., Zhao J. and Xu B. (2003). Chinese named entity recognition combining statistical model with human knowledge. In *Proceedings of the ACL 2003 Workshop on Multilingual and Mixed-language Named Entity Recognition*, Sapporo, Japan. Association for Computational Linguistics, pp. 65–72.

Yang Z., Salakhutdinov R. and Cohen W.W. (2017). Transfer learning for sequence tagging with hierarchical recurrent networks. arXiv preprint arXiv:1703.06345.

Yin Z. and Shen Y. (2018). On the dimensionality of word embedding. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18. USA: Curran Associates Inc, pp. 895–906.

Cite this article: Akkaya EK and Can B (2021). Transfer learning for Turkish named entity recognition on noisy text. *Natural Language Engineering* **27**, 35–64. https://doi.org/10.1017/S1351324919000627