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 et al., 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 are not required a priori, therefore the pre-trained 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 named entity recognition. 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 named entity recognition 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 named entity recognition models proposed in this paper, section 4 describes and gives the mathematical definition of the baseline Bi-LSTM-CRF model (Huang et al., 2015) adopted in this article, section 5 describes the proposed transfer learning model, 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 named entity recognition, those include statistical methods (Bikel et al., 1997; Wu et al., 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 have benefited from distributional approaches a lot. In recent years, distributional models such as Latent Semantic Analysis (LSA) (Landauer et al., 1998) have changed direction towards neural models. Word representation models such as word2vec (Mikolov et al., 2013) or GloVe (Pennington et al., 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 Conditional Random Field) architecture (Huang et al., 2015) as a sequence labeling model.

Here, we review mainly the research on named entity recognition 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 named entity recognition on noisy text.

2.1 Named Entity Recognition 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[1].

Çelikkaya et al. (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 misses 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 character, case of the word, start of sentence) along with the word su xes 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 et al. (2016) present a regularized averaged multiclass perceptron with hand-crafted features (*e.g.* word type flags, su x, prefix, capitalization) along with pre-trained 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.

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

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 tags, capitalization, word type and shape flags) and gazetteers. Additionally, they use the existence of *Twitter mentions* as a feature. They also provide the re-annotated versions of the two commonly used Turkish datasets: news dataset (Tür et al., 2003) and Twitter dataset (Çelikkaya et al., 2013). Re-annotated versions also include TIMEX and NUMEX types along with previously-labeled ENAMEX types. Finally, they report an F1 score of 67.96% with Twitter mentions and 63.63% without the mentions on the re-annotated version of the Turkish noisy dataset.

2.2 Named Entity Recognition 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 EMNLP (WNUT'2017)[2] so that all the reported results are comparable to each other.

Aguilar et al. (2017), the winner of the WNUT'17,[3] apply multi-task learning approach with a CRF-based model that incorporates pre-trained word embeddings obtained from word2vec (Mikolov et al., 2013) and orthographic character-level embeddings trained on a CNN with 2-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 et al., 2017) and capitalization features in addition to character-level embeddings trained by a CNN and pre-trained 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 pre-trained word embeddings and character-level embeddings obtained from another bidirectional LSTM. They also incorporate syntactic information by using part-of-speech (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 (SVM) and an LSTM. They also use hand-crafted features such as PoS tags, local context, chunk, su x and prefix, word frequency and a collection of flags (*e.g.* is-word-length-less-than-5, is-all-digit etc.). 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

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

the named entity tag probabilities. They obtain an F1 score of 26.30% on entity-level and 25.26% F1 score on surface 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 (LDA) (Blei et al., 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

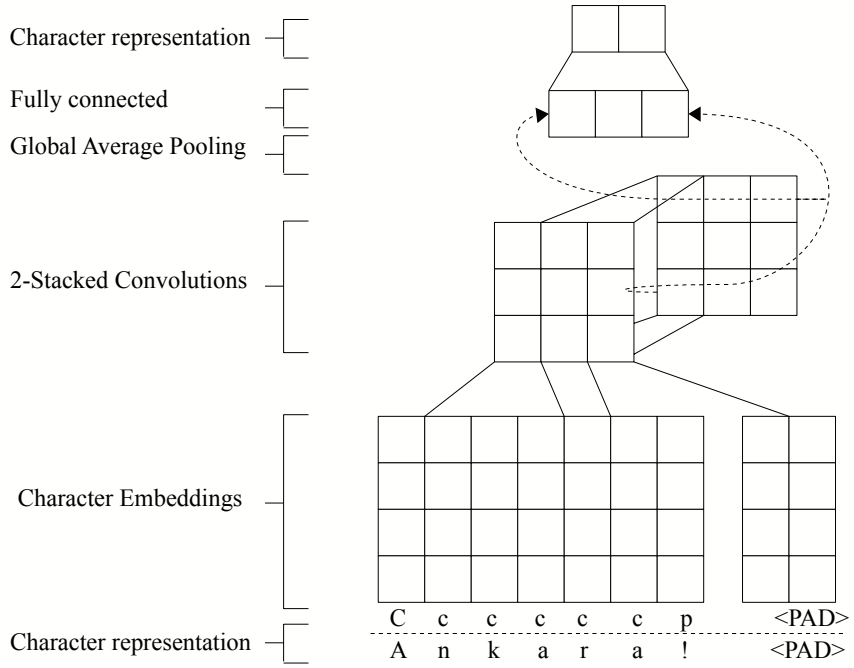

Fig. 1. Character-level word embedding using CNN. (Aguilar et al., 2017)

"nn cccccc 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.

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.

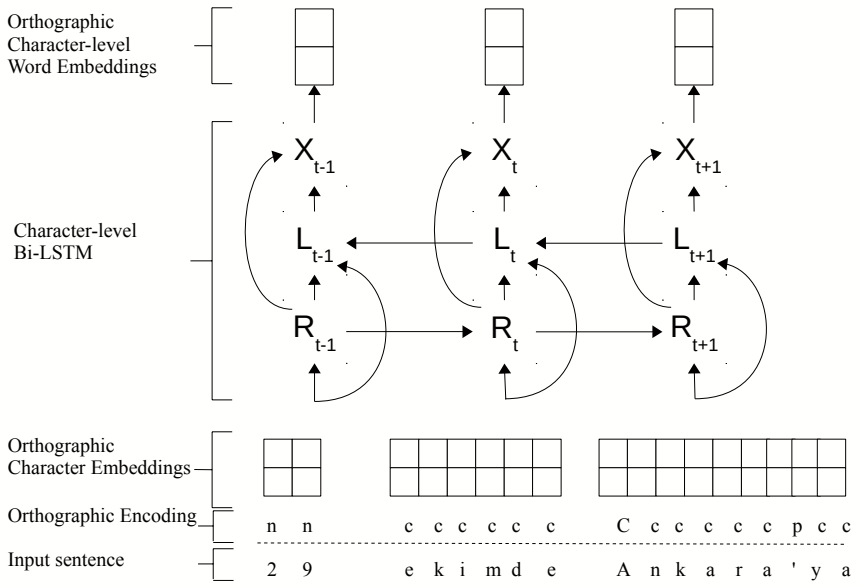

Fig. 2. Character-level word embedding using a bidirectional LSTM

3.4 Morpheme-level Word Embeddings

Morph2vec (Üstün et al., 2018) is another representation learning model that utilizes sub-word 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 pre-trained 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 su x *-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 su xes 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

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**

nition 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 Tantuğ (2015) significantly with a F1 measure 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 a F1 measure of 63.63%[11]. Nevertheless, our transfer learning model with extra layers outperforms all of the models with a 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

[11] 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.

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 Diyarbakır\E-PERSON 5\O Nolu\O Cezaevinde\O
Correct	Doğan\S-PERSON Diyarbakır\S-LOCATION 5\B-LOCATION Nolu\I-LOCATION Cezaevinde\E-LOCATION
Predicted	Ziraat\S-LOCATION Türkiye\S-LOCATION Kupası \O
Correct	Ziraat\B-ORGANIZATION Türkiye\I-ORGANIZATION Kupası \E-ORGANIZATION
Predicted	istanbul\S-LOCATION şehir\S-LOCATION tiyatrolarında\O
Correct	istanbul\B-ORGANIZATION şehir\I-ORGANIZATION tiyatrolarında\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

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 su xes (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 su x *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. 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 are given in Table 12 and Table 13 for the baseline and the transfer learning models respectively.

Amongst using solely character-level embeddings, morph2vec (Üstün and Can, 2016), fasttext (Bojanowski et al., 2017), or word2vec (Mikolov et al., 2013), the

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.*

Model	Embeddings	Entity Level (%)				Surface Form (%)			
		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**

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.*

Model	Embeddings	Entity Level (%)				Surface Form (%)			
		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

highest results are obtained from word2vec (Mikolov et al., 2013) with a F1 measure of 39.04% in the surface level and 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 a 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 a F1 measure of 40.55% in the surface level and 37.82% in the entity level. Although the other embedding types do not 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 do not change the results and the highest results in the surface level still remains 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 a 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 bears further information compared to 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 that are trained on CNN instead of Bi-LSTM performed poorer, 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 Workshop on Noisy User-generated Text workshop, WNUT'16[12] 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.

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

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 Type	Entity Level (%)			Surface Form (%)		
	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**

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 baseline model gives a F1 measure of 39.04%, whereas it improves up to 43.74% in the surface level. The highest score is obtained with a 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.

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 Type	Entity Level (%)			Surface Form (%)		
	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**

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[13] (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

[13] 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.