Natural Language Engineering (2022), **28**, pp. 763–795
doi:10.1017/S1351324921000371

CAMBRIDGE UNIVERSITY PRESS

ARTICLE

Joint learning of morphology and syntax with cross-level contextual information flow

Burcu Can[1,2,*], Hüseyin Aleçakır[3], Suresh Manandhar[4] and Cem Bozşahin[3]

[1]Department of Computer Engineering, Hacettepe University, Ankara, Turkey, [2]Research Institute of Information and Language Processing, University of Wolverhampton, Wolverhampton WV1 1LY, UK, [3]Cognitive Science Department, Informatics Institute, Middle East Technical University, Ankara, Turkey and [4]Madan Bhandari University of Science and Technology Development Board, Karyabinayak, Nepal
*Corresponding author. E-mail: b.can@wlv.ac.uk

(Received 29 April 2020; revised 8 November 2021; accepted 8 November 2021; first published online 20 January 2022)

Abstract

We propose an integrated deep learning model for morphological segmentation, morpheme tagging, part-of-speech (POS) tagging, and syntactic parsing onto dependencies, using cross-level contextual information flow for every word, from segments to dependencies, with an attention mechanism at horizontal flow. Our model extends the work of Nguyen and Verspoor (2018, *Proceedings of the CoNLL Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. The Association for Computational Linguistics, pp. 81–91) on joint POS tagging and dependency parsing to also include morphological segmentation and morphological tagging. We report our results on several languages. Primary focus is agglutination in morphology, in particular Turkish morphology, for which we demonstrate improved performance compared to models trained for individual tasks. Being one of the earlier efforts in joint modeling of syntax and morphology along with dependencies, we discuss prospective guidelines for future comparison.

Keywords: Morphology; Syntax; Dependency parsing; Morphological tagging; Morphological segmentation; Recurrent neural networks; Attention

1. Introduction

NLP tasks such as morphological segmentation, morpheme tagging, part-of-speech (POS) tagging, and syntactic parsing onto semantic dependencies have been widely investigated for extracting the meaning of a clause. These tasks can be considered to be different stages in reaching semantics: morphological segmentation and morpheme tagging deal with the morphemes inside a word;[a] POS tagging captures the words in a particular context, usually in a clause; dependency parsing views each clause as a sequence of words and the relations between them, such as subject, modifier, complement.

In these tasks, morpheme tagging labels each morpheme in a word with the syntactic information of the morpheme in that particular context, for example, person or tense. POS tagging assigns a syntactic category to each word in a context (e.g., noun and adjective). Dependency parsing finds relations such as argument, complement, and adjunct between words in terms of word–word dependencies. Dependency labeling assigns a tag (e.g., subject, object, and modifier) to every dependency relation.

[a]We take it to be a different problem than word delineation and identification, for example, in Chinese, Korean, and Japanese, which is a problem of orthography rather than morphology. We assume that the word is already identified.

Different aspects of language such as phonology, morphology, parts of speech, and dependency relations of words must be linked to each other to understand the interaction of these tasks.

For example, in Turkish, a plural marker on a noun pluralizes the noun to which it is attached, whereas a plural on the verb—which has the same phonological shape as nominal plural—indicates (as agreement) a plural syntactic subject.

Morphology and phonology interact to bring about the dependencies, and the interaction has implications for syntax. For example, in Tagalog, a verb is not available to surface syntax unless it is inflected for voice or marked for recent past. The phonological shape of the voice morpheme depends on the verb class, to be realized as prefix, infix or suffix. It has been argued that Tagalog's unvoiced and bare verbs are precategorial, that is, without an identifiable POS (Foley 1998). Their POS is determined by an interaction of morphology, phonology, and syntax, which call for joint consideration of these aspects.

Furthermore, the question of context in a clause, that is, left and right neighboring of linguistic elements in an expression, is relevant to morphology in ways that require its involvement from the ground up, rather than as an add-on mechanism serving only morphological disambiguation, which would in most cases disambiguate the morphemes of one word depending on its neighbors. A striking example of an extended role for morphology is from the language Kwakw'ala (Anderson 1992). In this language, nominal morphological markers of case, deictic status, and possession are not marked on the word itself but on the preceding word, as "suffixes" of that word. However, these suffixes bear no morphological relation to their phonological host.[b] Therefore, these morphemes need right context beyond their phonological host word to identify the stem of these inflections.

Providing a cross-level contextual information flow as we propose intends to inform higher levels so that a combination of the morpheme, the POS tag and the morpheme tag can use attention to right context. By "cross-level contextual information," we mean phonological, morphological, syntactic, and semantic (dependency) information considered altogether, for each word, rather than exclusively horizontal (or "cascaded") information flow, for example, first segmenting each word, then tagging and morphological analysis of each word, then syntax, then semantics. It also means that, for example, syntactic properties of a word not only depends on its own morphological properties but also on morphological properties of words in its context. This is true of all information projected upward, to dependencies.

In the current work, the only aspect we project exclusively horizontally is attention to phonemes of words in morphological tagging.

In the other direction of surface structure, that is, in requiring a left context, we do not have to go far to find striking examples which need left context. It is a fact of English morphosyntax in which "group genitives" show their true syntactic potential by semantically scoping over phrases but phonologically appearing in an unrelated word, for example, *Every man I know's taste in wallpaper is appalling* (Anderson *et al.* 2006). The genitive marker in the example bears no morphological or semantic relation to the word on which it appears. The left context that is needed to determine the semantic argument of the genitive can therefore span many words.

It is with these facts in mind that Anderson *et al.* (2006) propose relating word-level inflection to leftmost or rightmost daughters of a *syntactic* phrase. Clearly, it will require joint consideration of phonology, syntax, morphology, and semantics with the context in mind from the ground up.

In this article, we propose to start doing that by looking at a moving window of words at every level, by changing information flow from horizontally cascaded processes to a cross-level one, from morphological segments up to dependencies for every word in a moving window. This choice addresses agglutinating languages quite naturally, albeit within a limited window.[c]

[b]The language is strictly verb–subject–object, so there is always a preceding word for every nominal.

[c]We look forward to having access to data in languages such as Kwakw'ala for extending it to the right context, and for more elaborate English annotation for examples such as the one above to better understand the left context to support model training.

https://doi.org/10.1017/S1351324921000371 Published online by Cambridge University Press

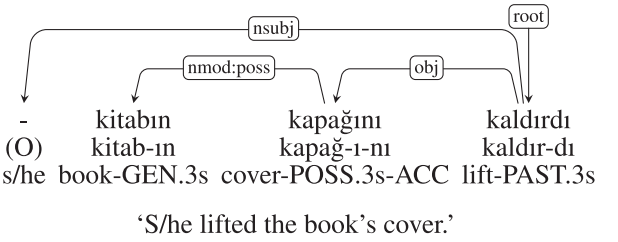

Figure 1. A Turkish clause with labeled dependency relations between morphologically complex words. First line: The orthographic form ("-" is for "null"). Second line: morphological segments. Third line: morphological tags (-GEN=genitive, .3s=3rd person singular, -ACC=accusative, -PAST=past tense). Dependencies in the article are arrowed (head to dependent) and labeled UD dependencies (de Marneffe *et al*. 2021).

We introduce a joint learning framework for morphological segmentation, morpheme tagging, POS tagging, and dependency parsing, with particular modeling emphasis on agglutinating languages such as Turkish. We have tested our system with other languages as well.

The proposed joint learning is a stronger form of multitask learning. Instead of cascaded horizontal processes which are common in multitasks, we propose a different design. We can motivate the design choices as follows.

In Turkish, morphology is agglutinating. Syntactic information is encoded in inflectional morphemes, which are suffixed to the word. An example is given in Figure 1. In the example, *-ı* in *kapağ-ı-nı* indicates possession, which is an inflection. It also reveals that its host word is a nominal. It is in a possessive dependency relation with "book."

If the stem were a verb, for example, *yap-tığ-ı-nı* (do-COMP-POSS.3s-ACC) "his/her doing," the same phonological shape *ı* would mark agreement with subordinate subject of this verb, rather than possession. This is relevant to choice of dependency. There is no possessive meaning in *yaptığ-ı-nı,* as can be observed in *Ben [[sigara içme-nin] kanser yap-tığ-ı-nı] bilmiyordum* (I [cigarette smoking] -GEN.3s cancer do-COMP-POSS.3s-ACC I-not-know) "I did not know that smoking caused cancer." There is possessive meaning in *kapağ-ı-nı*. One implication is that their morphological tags must be different, because of differences in semantics.[d]

Morphology's role is crucial and it can be nonlocal in setting up the correct relation of meaning. As we shall see later, in Figures 9(c) and 10, in many cases looking at the previous word is not enough to disambiguate the possessor even in examples where we can determine that the shape has possession semantics; therefore, decoding on this basis alone would not be sufficient (cf. Akyürek, Dayanık, and Yuret 2019). It needs an attention mechanism or its functional equivalent. Therefore, a standard method that involves learning morphology, syntax, and semantics in a horizontal pipeline process would eventually suffer in such circumstances.

Joint learning is a preparation for these concerns. It is defined as learning some related tasks in parallel by using shared knowledge contained in various tasks; therefore, each task can help other tasks to be learned better (Caruana 1997). Representational autonomy of the tasks is the expedient to joint modeling.

We represent distributional meaning in terms of neural vector representations that help to learn some underlying form of a sentence. We use neural character and morpheme representations to learn the word representations for learning POS tags and dependency relations between

[d]This has been a source of variation in Turkish linguistics, which has been carried over to computational linguistics. For example, Göksel and Kerslake (2005) (p.135) mark the dependent as GEN and head as POSS, for both the genitive-possessive construction and for agreement in finite subordination. However, Kornfilt (2001) (p.187), who is the main resource for calling the elements on subordinate verbs "agreement markers" rather than possession markers, since 1980s, marks the first one as GEN and the second as 3SG, for third person singular. To limit interpretation, when the actual morph is shown, we use both notations, for example, GEN.3s for third person singular genitive and POSS.2s for second person singular possessive.

https://doi.org/10.1017/S1351324921000371 Published online by Cambridge University Press

words. In learning morphological segmentation of words, we aim to obtain a word representation based on morphemes. Recent work shows that representations that are learned from words as separate tokens are not good enough to capture the syntactic and semantic information of words in agglutinating languages because of severe sparsity. Character-level or morpheme-level representations lead to a better prediction of embeddings in such languages (e.g., Üstün, Kurfalı, and Can 2018). Vania and Lopez (2017) provide a comparison of word representations. Their results confirm that although character-level embeddings are effective for many languages, they do not perform as well as the models with explicit morphological information. We use both character and morpheme representations in learning POS tags and dependencies.

Our framework is built upon the joint POS tagging and dependency parsing model of Nguyen and Verspoor (2018), to which we introduce two more layers for morphological segmentation and morphological tagging. The results show that a unified framework benefits from joint learning of various levels, from morphology to dependencies. Our results with joint learning of morphological tagging and dependency parsing are encouraging compared to the model of Nguyen and Verspoor (2018), who model them separately.

The rest of the paper is organized as follows. Section 2 reviews related work in five tasks that we model. Section 3 describes the proposed joint learning framework and the components involved in the architecture. Section 4 presents experimental results and detailed error analysis. A brief section concludes (Section 5).

2. Related work

Adequate treatment of morphology has been one of the long-standing problems in computational linguistics. Some work in the literature deals with it as a segmentation task that only aims to segment each word into its morphemes without identifying the syntactic or semantic role of the morpheme (Goldsmith 2001; Creutz and Lagus 2007; Can and Manandhar 2010; Goldwater, Griffiths, and Johnson 2011). Some work deals with morphology as both segmentation and sequence labeling, which is called morphological tagging (Müller *et al.* 2015; Cotterell and Heigold 2017; Dayanık, Akyürek, and Yuret 2018). POS tagging (Schütze 1993; Clark 2000; Van Gael, Vlachos, and Ghahramani 2009) and dependency parsing (Kudo and Matsumoto 2002; Oflazer *et al.* 2003; Nivre and Nilsson 2005) are well-known tasks in computational linguistics.

In recent years, deep neural networks have been extensively used for all of these tasks. Regarding morphological analysis, Heigold, Neumann, and van Genabith (2017) present an empirical comparison between convolutional neural network (CNN) (Lecun *et al.* 1998) architectures and long-short-term memory network architectures (LSTMs) (Hochreiter and Schmidhuber 1997). Character-based architectures are meant to deal with sparsity. LSTMs give slightly better results compared to CNNs. Cotterell and Heigold (2017) apply transfer learning by jointly training a model for different languages in order to learn cross-lingual morphological features. Shared character embeddings are learned for each language as in joint learning, under a joint loss function. Dayanık *et al.* (2018) propose a sequence-to-sequence model for morphological analysis and disambiguation that combines word embeddings, POS tags, and morphological context.

Deep neural networks have also been used for POS tagging. Dos Santos and Zadrozny (2014) employ both word embeddings and character embeddings obtained from a CNN. Ling *et al.* (2015) introduce a model to learn word representations by composing characters. The open vocabulary problem is addressed by employing the character-level model, and final representations are used for POS tagging. The results are better compared to that of Dos Santos and Zadrozny (2014).

Joint learning has re-attracted attention in recent years (Sanh, Wolf, and Ruder 2019; Liu *et al.* 2019; Li *et al.* 2020) after the resurgence of neural networks. To our knowledge, there has not been any study that combines morphological segmentation, morpheme tagging, POS tagging, and dependency parsing in a single framework using deep neural networks, especially for morphologically rich languages. There have been various studies that combine some of these

https://doi.org/10.1017/S1351324921000371 Published online by Cambridge University Press

taks. Nguyen and Verspoor (2018) introduce a model that extends the graph-based dependency parsing model of Kiperwasser and Goldberg (2016) by adding an extra layer to learn POS tags through another layer of bidirectional LSTM (BiLSTM), which is used as one of the inputs to dependency LSTMs. Yang *et al*. (2018) propose a joint model to perform POS tagging and dependency parsing based on transition-based neural networks. Zhang *et al*. (2015) introduce a joint model that combines morphological segmentation, POS tagging, and dependency parsing. The model does not perform morpheme tagging. It is based on randomized greedy algorithm that jointly predicts morphological segmentations, POS tags, and dependency trees. Straka (2018) proposes a pipeline model called UDPipe 2.0 that performs sentence segmentation, tokenization, POS tagging, lemmatization, and dependency parsing. The pipeline framework follows a bottom-up approach without jointly training the tasks. Kondratyuk and Straka (2019) extends UDPipe 2.0 with a BERT replacement of embedding and projection layers, which is trained across 75 languages simultaneously.

Our proposed model differs from these models in trying to combine five different tasks in a single learning framework, with the expectation that all five tasks improve by joint modeling, and with cross-level information. The baseline of our framework, Nguyen and Verspoor (2018), addresses only POS tagging and dependency parsing. We extend their model with extra layers for morphological segmentation and identification, enabling the model to perform both morphological and syntactic analysis for morphologically rich languages.

3. Joint learning of morphology and syntax

In the base model of Nguyen and Verspoor (2018), one component is for POS tagging. It is based on a two-layer BiLSTM (Hochreiter and Schmidhuber 1997) which encodes sequential information that comes from each word within a sentence. Encoded information is passed through a multilayer perceptron with a single layer that outputs the POS tags. Their second component is for graph-based dependency parsing. It also involves BiLSTM, which learns the features of dependency arcs and their labels by using the predicted POS tags obtained from the POS tagging component, word embeddings, and character-level word embeddings.

The overview of our architecture is shown in Figure 2. We have added one component to learn the morphological segmentation and another to do morphological identification to obtain the morpheme labels. The vectorial representations used in the model are illustrated in Figure 3. We now describe the architecture.

3.1 Morphological segmentation

The lowest-level component in our joint learning framework performs segmentation, which uses characters as proxies for phonological shape. For agglutinating languages, in particular, the task is to make the segmentation morphologically salient, that is, corresponding to a morph. The overall sub-architecture of the segmentation component is given in Figure 4. A BiLSTM ($\text{BiLSTM}_{\text{seg}}$) which encodes characters in a word is used to learn the sequential character features for segment boundaries. We feed this BiLSTM with one hot character encoding of each character in the word. A latent feature vector v_i^{char} for a character c_i (ith character in the word) is learned as an output of the BiLSTM at a time step:

$$v_i^{\text{char}} = \text{BiLSTM}_{\text{seg}}(e_{1:n}, i) \tag{1}$$

where $e_{1:n}$ denotes a sequence of one hot vectors for each character in the word.

Each output vector is reduced to a single dimension by a multilayer perceptron (MLP) with one hidden layer to predict a segment boundary at a given step:

$$\hat{y}_i = \text{sigmoid}\left(\text{MLP}_{\text{seg}}\left(v_i^{\text{char}}\right)\right) \tag{2}$$

https://doi.org/10.1017/S1351324921000371 Published online by Cambridge University Press

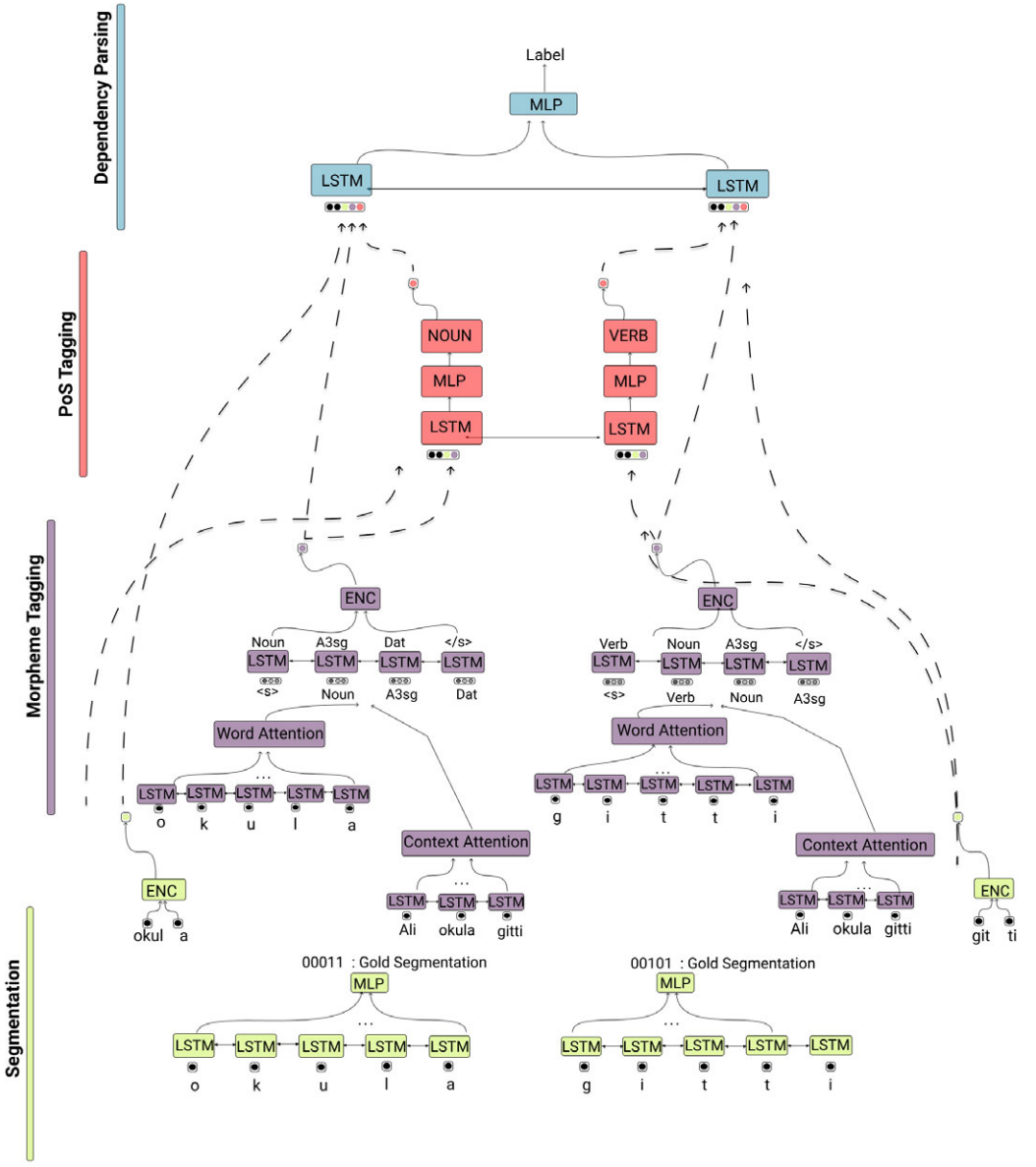

Figure 2. The layers of the proposed joint learning framework. The sentence "Ali okula gitti." ("Ali went to school") is processed from morphology up to dependencies.

where $\hat{y}_i$ denotes prediction probability of being a segment boundary after the ith character in the word. MLP has one sigmoid output that predicts whether there is a segment boundary at a time step. 1 indicates that there is a segment boundary after the character and 0 indicates that the word will not be split at that point.

The sigmoid is a continuous function. We use it for binary tagging, where a value above 0.6 refers to a morpheme boundary and a value below 0.6 refers to a non-boundary during testing. Therefore, we amplify the values above the threshold to predict a segmentation boundary during testing. However, we sample a segmentation boundary during training based on a probability value sampled randomly.

https://doi.org/10.1017/S1351324921000371 Published online by Cambridge University Press

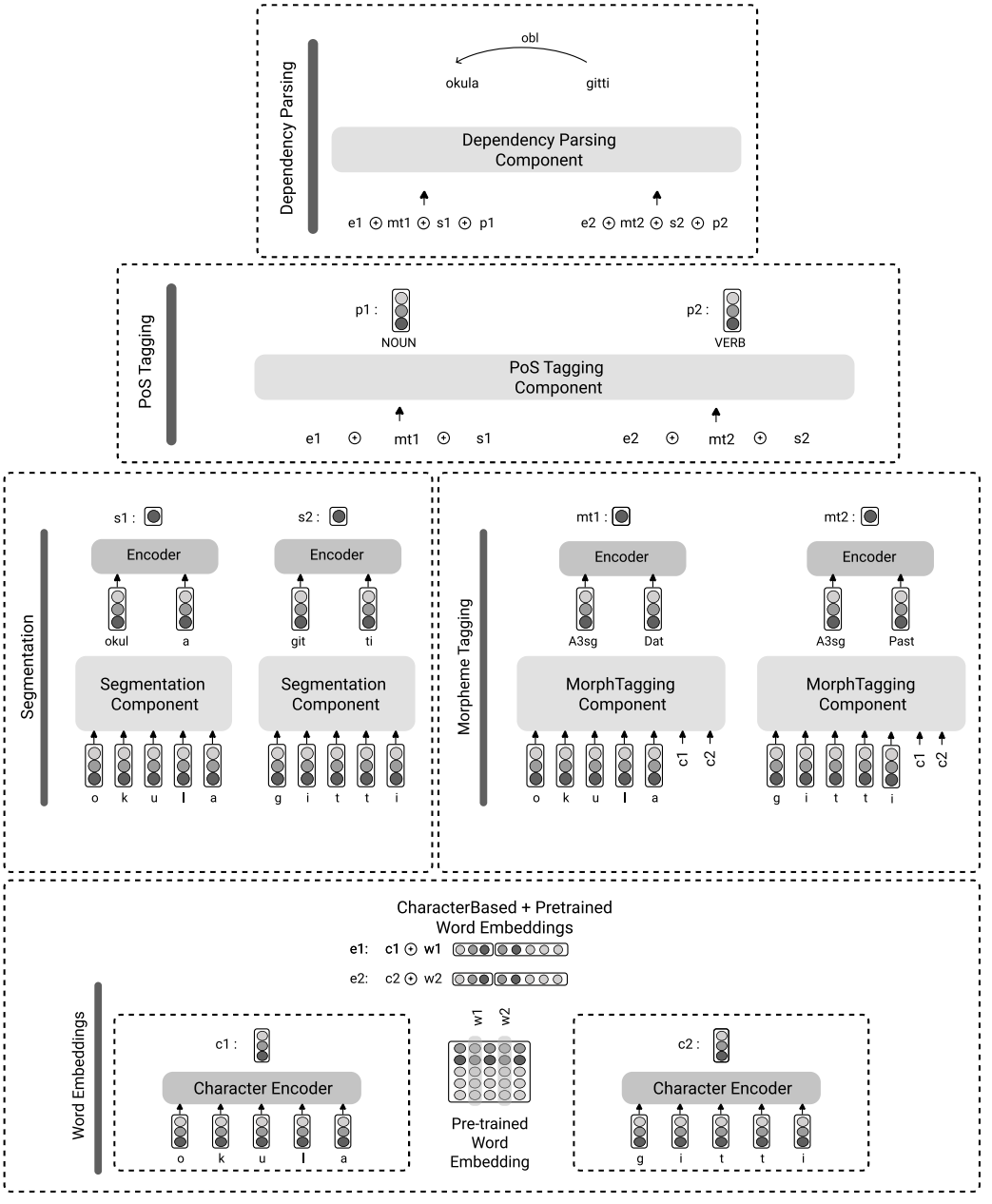

Figure 3. The layers of the proposed joint learning framework working on the sentence "okula gitti" ("(he/she) went to school"). The vectors *c*, *w*, *e*, *mt*, *s*, and *p* denote character, word, concatenated character and word, morphological tag, segment, and POS tag, respectively.

Based on the predicted outputs for each position inside the word, we compute the loss defined by binary cross-entropy as follows for Y number of characters inside a word:

$$\mathscr{L}_{\text{seg}} = -\sum_{i=1}^{Y} \left(y \log(\hat{y}_i) - (1-y) \log\left(1-\hat{y}_i\right) \right) \tag{3}$$

where $\hat{y}$ denotes the predicted segmentation and y denotes the gold segmentation.

https://doi.org/10.1017/S1351324921000371 Published online by Cambridge University Press

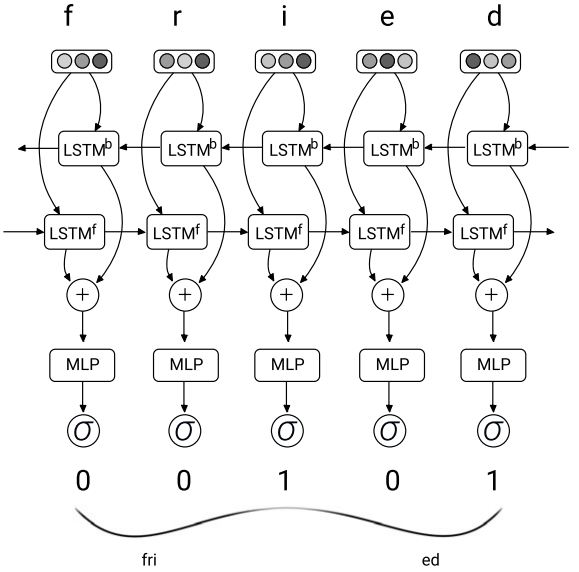

Figure 4. The sub-architecture of the morphological segmentation component. The one hot vector of each character of a word is fed into a BiLSTM. The resulting vector obtained from each state of the BiLSTM is fed into a multilayer perceptron with one hidden layer and with a sigmoid activation function in the output layer. The output of the sigmoid function is a value between 0 and 1, where 1 corresponds to a segment boundary and 0 indicates a non-boundary. In the example, there is a boundary after the character *i*, that is, 1 is the output from the MLP at that time step.

3.2 Morphological tagging

The second additional component in the model is for morphological tagging. We use an encoder-decoder model which takes each word as input, encodes the characters of the word along with the context of the word, and performs decoding by generating the morpheme tags of the word.

Decoding is performed in a sequential manner by predicting morpheme tags of a word one by one using LSTM. However, this sequential generation may not refer to the actual order of morphemes in a word. It reveals the correspondent morpeme tags within a word regardless of their position. Therefore, decoding is not tied to agglutination, and portmanteau, circumfixation and lexically marked differences in a word are representable (e.g., run/ran).

The overall sub-architecture of the morphological tagging component is depicted in Figure 5. Here, we define a two-layer BiLSTM for the character encoder, where each character is encoded as a vector which is the output of that time step in BiLSTM that involves all the sequential information to the right and left of the current character, denoted as m_i^{char}:

$$m_i^{char} = \text{BiLSTM}_{\text{c\_encode}}\left[\overrightarrow{h_i^c} : \overleftarrow{h_i^c}\right] \tag{4}$$

where $\overrightarrow{h_i^c}$ is the forward representation and $\overleftarrow{h_i^c}$ is the backward representation obtained for the *i*th character in the current word. In order to decode a token into a morphological tag through the decoder, we apply an attention mechanism to learn which characters in the actual word have more impact on the current tag that will be generated by the decoder.

We estimate the weight of each character c_i on each decoded token z_j as follows, which is normalized over all characters in the original word:

$$\alpha_{ji} = \frac{\exp\left(\text{score}\left(c_i, z_j\right)\right)}{\Sigma_{y'=1}^{Y} \exp\left(\text{score}\left(c_i', z_j\right)\right)} \tag{5}$$

https://doi.org/10.1017/S1351324921000371 Published online by Cambridge University Press

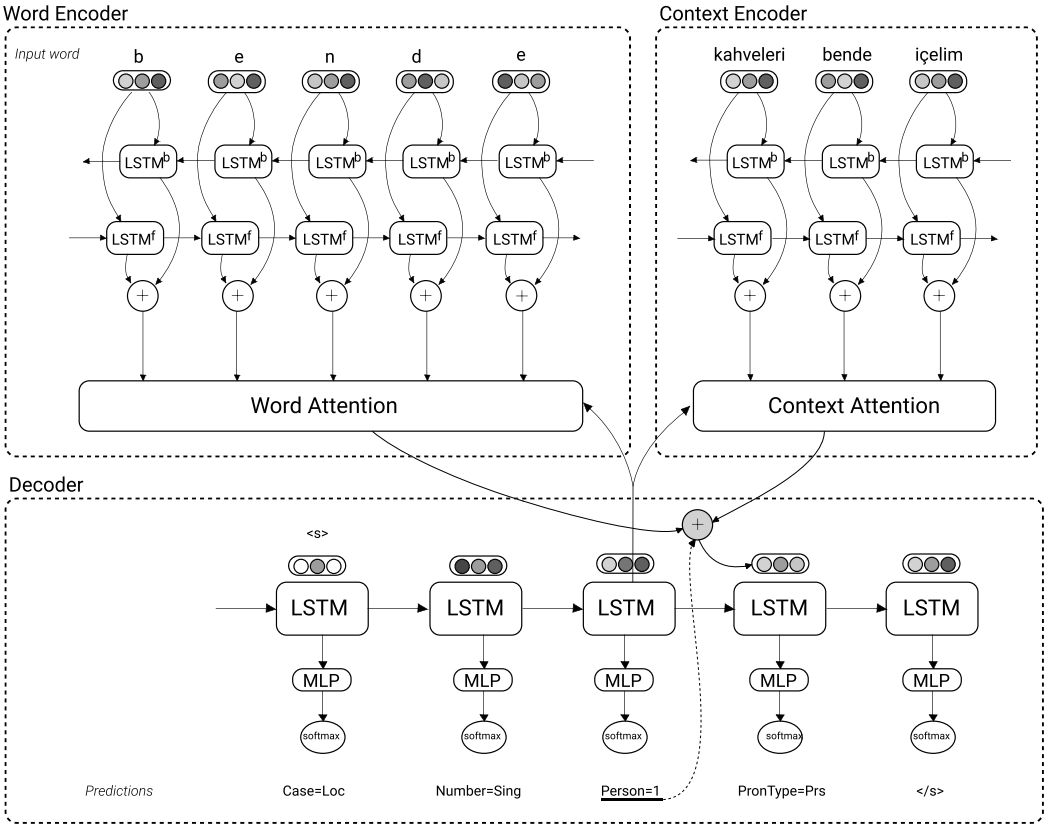

Figure 5. The encoder–decoder sub-architecture of the morphological tagging component. The top part is the encoder and the bottom part is the decoder. The example is "kahveleri bende içelim" ("let's drink coffee at my place").

Y is the length of the encoded word. The unnormalized score(c_i, z_j) that measures how much contribution each character has on the output tokens is computed with the relation between the encoded character c_i and the decoded state z_j as follows:

$$\text{score}\left(c_i, z_j\right) = v_1 \cdot \tanh\left(\left[W_1 m_i^{\text{char}}\right]; \left[W_2 z_j\right]\right) \tag{6}$$

where v_1, W_1, and W_2 are the parameters to be learned during training. This is a one-layer neural network that applies *tanh* function to its output. It corresponds to the attention mechanism for the character encoder.

Once the weights are estimated, the total contribution of all characters to the current word is computed as the weighted sum of each character at time step t:

$$c_t = \Sigma_s \alpha_{ts} m_s^{\text{char}} \tag{7}$$

where c_t is the attended word characters of the current word that will be one of the inputs to the decoder.

For the word encoder, we define a second one-layer BiLSTM, where each context word is encoded as a vector, m_i^{word}:

$$m_k^{\text{word}} = \text{BiLSTM}_{\text{w\_encode}}\left[\overrightarrow{h_k^w}; \overleftarrow{h_k^w}\right] \tag{8}$$

where $\overrightarrow{h_k^w}$ is the forward representation and $\overleftarrow{h_k^w}$ is the backward representation obtained from the jth context word in the current sentence. The input to $\text{BiLSTM}_{\text{w\_encode}}$ is generated by another

https://doi.org/10.1017/S1351324921000371 Published online by Cambridge University Press

BiLSTM that processes the characters of each word and produces a character-level word embedding of a given word using character embeddings. Another attention mechanism is used for encoding the context to predict the correct morphological tags based on the context, thereby enabling morphological disambiguation. To this end, we estimate the weight of each context word w_k (including the current word) on each decoded token z_j as follows, which is normalized over all words in the sentence:

$$\alpha_{jk} = \frac{\exp\left(\text{score}\left(w_k, z_j\right)\right)}{\Sigma_{i'=1}^{N} \exp\left(\text{score}\left(w'_k, z_j\right)\right)} \tag{9}$$

where N denotes the number of words in the sentence. The unnormalized score $score(w_k, z_j)$ that measures how much contribution each context word has on the output tokens is computed based on the encoded word w_k. The resulting decoded state t_x is defined as follows:

$$\text{score}\left(w_k, z_j\right) = v_2 \cdot \tanh\left(\left[W_3 m_k^{\text{word}}\right]; \left[W_4 z_j\right]\right) \tag{10}$$

where v_2, W_3, and W_4 are the parameters to be learned during training. This is a one-layer neural network that applies *tanh* function to its output. It is the attention mechanism for the context encoder.

Once the weights are estimated, the total contribution of all words in the sentence is computed as the weighted sum of each word at time step t:

$$wv_t = \Sigma_s \alpha_{ts} m_s^{\text{word}} \tag{11}$$

where wv_t is the attended context words that will be another input to be fed into the decoder.

The input of the decoder for each time step t is therefore the concatenation of the embedding of the morpheme tag produced in the last time step $t-1$, the weighted sum of the BiLSTM outputs of both word encoder and context encoders:

$$f_t = c_t \circ wv_t \circ z_{t-1} \tag{12}$$

The decoder is a unidirectional LSTM with the output at each time step t:

$$\begin{aligned} d_t^{\text{token}} &= \text{LSTM}_{\text{decode}}(f_{1:Y}, t) \\ \hat{y}_t &= \text{softmax}\left(\text{MLP}_{\text{mtag}}\left(d_t^{\text{token}}\right)\right) \end{aligned} \tag{13}$$

where $\hat{y}_t$ is the predicted morpheme tag at time step t; the tth morpheme tag generated for the current word. The output of each decoder state is fed into a one-layer MLP with a *softmax* function to predict the next morpheme tag.

Finally, categorical cross-entropy loss is computed for each position in the decoder. The morphological tagging loss ($\mathscr{L}_{mtag}$) is

$$\mathscr{L}_{mtag} = -\sum_{t}^{\mathbb{Y}} y_t \log\left(\hat{y}_t\right) \tag{14}$$

3.3 Word vector representation

We use word-level, character-level, and morpheme-level word embeddings for both POS tagging and dependency parsing. Word-level word embeddings are pretrained from word2vec (Mikolov *et al.* 2013). We learn the character embeddings through a BiLSTM. The output of the BiLSTM is used for character-level word embeddings.

For the morpheme-level embeddings, we have two types. One is generated from actual morphemes. The other one is generated from morpheme tags. For the embeddings of the actual morphemes, we use morph2vec (Üstün *et al.* 2018) to pretrain the morpheme embeddings. Once

https://doi.org/10.1017/S1351324921000371 Published online by Cambridge University Press

the morpheme embeddings are learned, we use a BiLSTM that encodes each morpheme embedding to predict the morpheme-level word embeddings. We use another character BiLSTM for the unseen morphemes that are fed with character embeddings. For the embeddings of the morpheme tags, we randomly initialize the morpheme tag embeddings, where the morpheme tags are predicted by the morpheme tagging component described above.

The final vector e_i for the ith word w_i in a given sentence $s = w_1, w_2, \cdots w_N$ is

$$e_i = e^{(w)}_{w_i} \circ e^{(c)}_{w_i} \circ e^{(m)}_{w_i} \circ e^{(mt)}_{w_i} \tag{15}$$

where $e^{w}_{w_i}$ is the word-level word embedding, $e^{c}_{w_i}$ is the character-level word embedding, $e^{m}_{w_i}$ is the morpheme-level word embedding, and $e^{mt}_{w_i}$ is the encoding of the sequence of morpheme tags for the given word.

3.4 POS tagging

We use a BiLSTM to learn the latent feature vectors for POS tags in a sentence. We feed the sequence of vectors $e_{1:N}$ for the sentence to $\text{BiLSTM}_{\text{POS}}$. The output of each state is a latent feature vector $v^{(POS)}_i$ that refers to the feature vector of the ith word w_i in the sentence:

$$v^{\text{POS}}_i = \text{BiLSTM}_{\text{POS}}(e_{1:n}, i) \tag{16}$$

In order to predict the POS tag of each word, we use an MLP with *softmax* output function similar to the joint model of Nguyen and Verspoor (2018):

$$\hat{\text{POS}}_i = \underset{T}{\arg\max}\, \text{softmax}\left(\text{MLP}_{\text{POS}}\left(v^{POS}_i\right)\right) \tag{17}$$

where POS_i denotes the predicted POS tag of the ith word in the sentence and T is the set of all POS tags. We use the categorical cross-entropy loss $\mathscr{L}_{POS}$.

$$\mathscr{L}_{POS} = -\sum_{t}^{\mathbb{Y}} POS_t \log\left(\hat{POS}_t\right) \tag{18}$$

where POS_t is the gold POS tag.

3.5 Dependency parsing

We employ a BiLSTM ($\text{BiLSTM}_{\text{dep}}$) to learn the dependency latent feature vectors for dependency relations between words. We feed $\text{BiLSTM}_{\text{dep}}$ with the POS tag embeddings and the word embeddings e_i. Once the POS tags are predicted, we represent each POS tag with a vector representation $e^{(p)}_{p_i}$, which is randomly initialized. The final input vector representations are defined as follows for the dependency parsing component:

$$x_i = e^{(p)}_{p_i} \circ e_i \tag{19}$$

$$= e^{(p)}_{p_i} \circ e^{(w)}_{w_i} \circ e^{(c)}_{w_i} \circ e^{(m)}_{w_i} \circ e^{(mt)}_{w_i} \tag{20}$$

We feed the sequence of vectors $x_{1:N}$ into $\text{BiLSTM}_{\text{dep}}$. Latent feature vectors $v_{1:N}$ are obtained from $\text{BiLSTM}_{\text{dep}}$:

$$v_k = \text{BiLSTM}_{\text{dep}}\left[\overrightarrow{h^d_k}:\overleftarrow{h^d_k}\right] \tag{21}$$

where $\overrightarrow{h^d_k}$ is the forward representation and $\overleftarrow{h^d_k}$ is the backward representation obtained from the output of the kth state in $\text{BiLSTM}_{\text{dep}}$.

https://doi.org/10.1017/S1351324921000371 Published online by Cambridge University Press

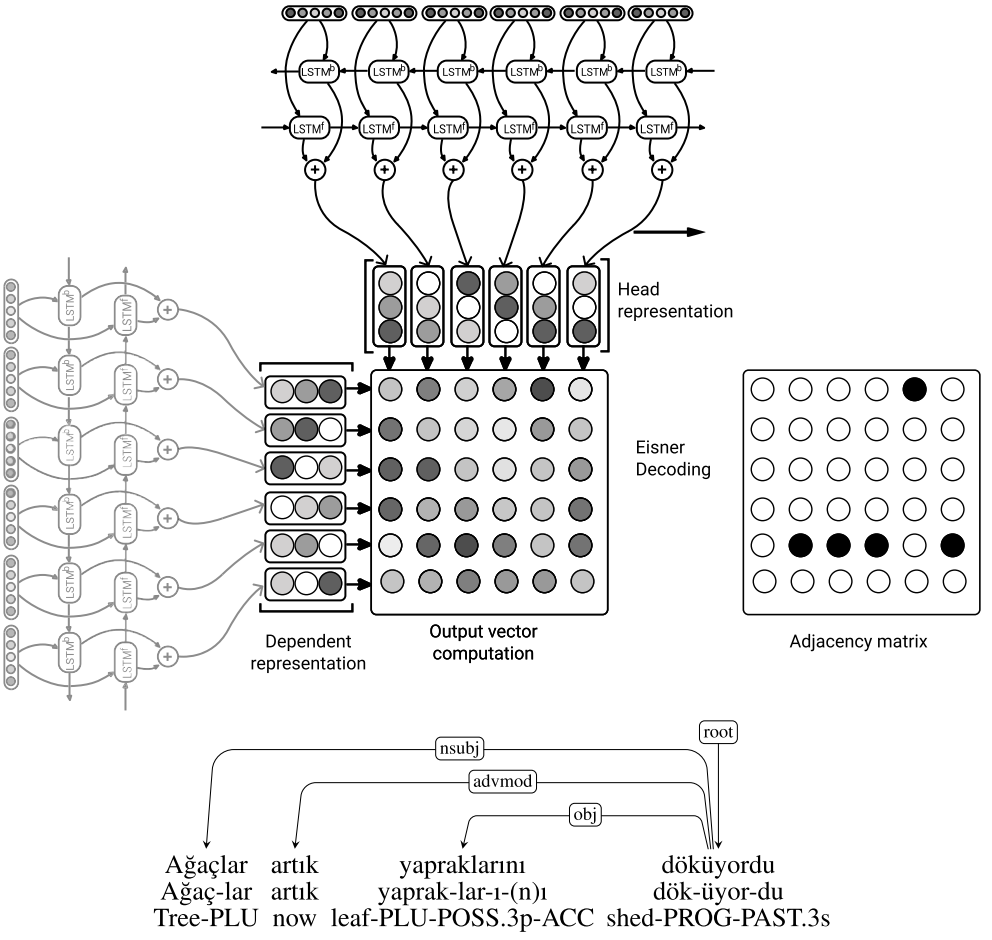

Figure 6. The pointer network which is used to predict a dependency score between words in a sentence. Some Turkish dependencies are shown for exemplifying head-dependent relations.

To predict the dependency arcs, we follow Nguyen and Verspoor (2018) and McDonald, Crammer, and Pereira (2005). Features are enriched with morpheme embeddings obtained from morph2vec (Üstün *et al.* 2018). The arcs are scored by an MLP that produces a score denoted by $\text{score}_{\text{arc}}^{(i,j)}$ with a single output node:

$$\text{score}_{\text{arc}}(i,j) = \text{MLP}_{\text{arc}}\left(v_i \circ v_j \circ \left(v_i * v_j\right) \circ |v_i - v_j|\right) \tag{22}$$

The MLP consists of a pointer network (Figure 6) which predicts whether two words in a sentence have a dependency relation or not.

Once the scores are predicted, we use the decoding algorithm of Eisner (1996) to find the projective parse tree with the maximum score:[e]

$$\text{score(s)} = \arg\max_{\hat{t}\in T(s)} \sum_{(h,m)\in\hat{t}} \text{score}_{\text{arc}}(h,m) \tag{23}$$

[e]The version we used is at https://github.com/LxMLS/lxmls-toolkit/.

https://doi.org/10.1017/S1351324921000371 Published online by Cambridge University Press

where h denotes the head word of the dependency, m denotes the modifier word, T_s is the set of all possible parse trees for the sentence s, and $\hat{t}$ is the parse tree with the maximum score.

We also predict the labels for each arc with another MLP, which is MLP_{rel}, with *softmax* output. An output vector $v_{(h,m)}$) is computed for an arc (h,m):

$$v_{(h,m)} = \text{MLP}_{\text{rel}}\left(v_h \circ v_m \circ (v_h * v_m) \circ |v_h - v_m|\right) \tag{24}$$

3.6 Cross-level contextual information flow

In the architecture which we have described so far, every time step in each layer is fed with only that time step's output obtained from one layer below.

We further investigate data sharing between the layers in various time steps. For this, we incorporate contextual information from the previous word into the input of the layers for the current word, by adding contextual information obtained from morpheme tagging encoding and morpheme encoding of the previous word to POS and dependency layers of the current word. Similarly, we incorporate POS tagging encoding of the previous word into the dependency layer of the current word. An illustration of the cross-level contextual information flow is in Figure 7.

We do this by various methods. One of the methods is weighted sum:

$$e'_{w_i} = \alpha_1 e_{w_{i-1}} + \alpha_2 e_{w_i} \tag{25}$$

where α_1 and α_2 are weights in combining the encoding of the previous word $e_{w_{i-1}}$ and encoding of the current word e_{w_i}. The updated encoding of the current word is denoted by e'_{w_i}.[f] The encodings may correspond to either morpheme-based word embeddings, morpheme tag encodings, or POS tag encodings of words, depending on which layer the cross-level contextual information flow applies.

The second method is elementwise multiplication between the encoding of the previous word and the encoding of the following word:

$$e'_{w_i} = e_{w_{i-1}} \odot e_{w_i} \tag{26}$$

In the third method, we experimented with contextual information by using MLP to let the full joint model learn how to combine the encodings of successive words:

$$e'_{w_i} = \text{MLP}_{\text{ver}}\left(e_{w_{i-1}} \circ e_{w_i}\right) \tag{27}$$

First, the encodings of the previous and the current word are concatenated, then fed into a one-layer MLP. The parameters of the MLP are learned jointly with the model.

In the fourth method, we employed a concatenative (or additive) attention network to learn the weights of the contextual information automatically. We used a fixed size window for the history to estimate the weights.[g] The weight of a previous word is estimated as:

$$\alpha_i = \frac{exp(\text{score}(w_{i-1}, x_i))}{\Sigma^Y exp(\text{score}(w', x_i))} \tag{28}$$

where w_{i-1} is the previous word, x_i is the encoding of the ith token (particularly, the POS tag or the dependency relation), and Y is the set of context words (including the current word itself). Therefore, α_i gives the relevance of w_{i-1} for x_i in that layer. The unnormalized score (w_{i-1}, x_i) that estimates the weight of each context word is defined as follows:

$$\text{score}_1\ (w_{i-1}, x_i) = v_3 \cdot \tanh\left(\left[W_4 e_{w_{i-1}}\right]; [W_5 x_i]\right) \tag{29}$$

[f]We use manually set weights, $\alpha_1 = 0.3$ and $\alpha_2 = 0.7$. The weights are assigned empirically as a result of several experiments performed on the development set.

[g]We set the window size to 2 or 5 to the left, and 1 or 2 to the right to incorporate narrower and larger contextual information. The results are discussed in Section 4.

https://doi.org/10.1017/S1351324921000371 Published online by Cambridge University Press

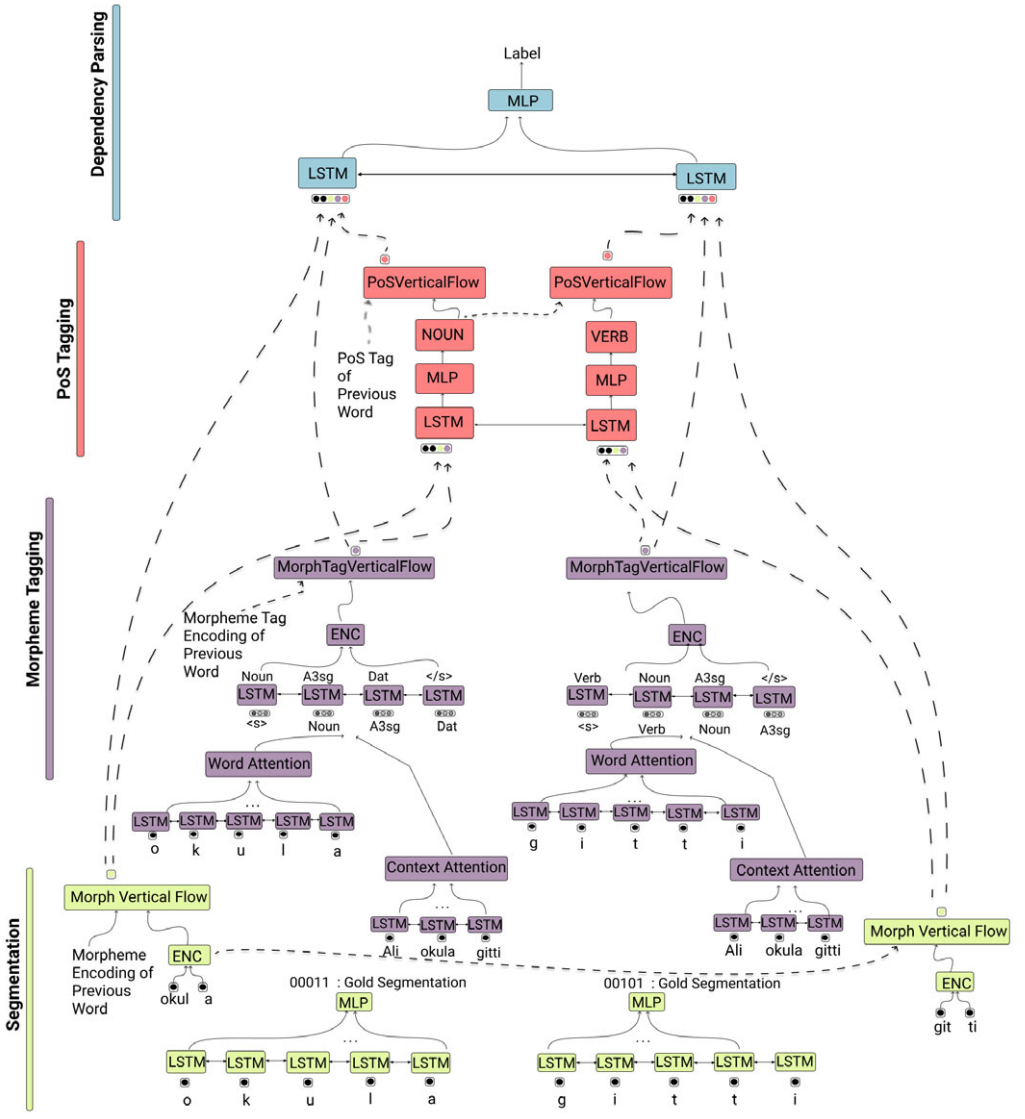

Figure 7. An illustration of the overall architecture with the cross-level contextual information flows between layers, using the example "Ali okula gitti." ("Ali went to school").

where $e_{w_{i-1}}$ is the encoded information of w_{i-1} (either with morphemic information, morpheme tags, or POS tags) that is obtained from a lower layer to be fed into an upper layer, and v_3, W_4, and W_5 are the parameters to be learned during training. Once the weights are learned, max pooling is applied for the final encoding that will be fed into the upper layer.

In the fifth method, we used a dot-product attention model. The unnormalized score $\text{score}_2(w_{i-1}, x_i)$, which estimates the weight of each context word is defined as follows:

$$\text{score}_2(w_{i-1}, x_i) = x_i^T \cdot e_{w_{i-1}} \tag{30}$$

We also employed CNNs to blend the contextual information using convolution and pooling layers along with a rectifier unit.

An illustration of the attention network that allows cross-level contextual information flow for two previous words is given in Figure 8. The figure shows the attention network between the POS

https://doi.org/10.1017/S1351324921000371 Published online by Cambridge University Press

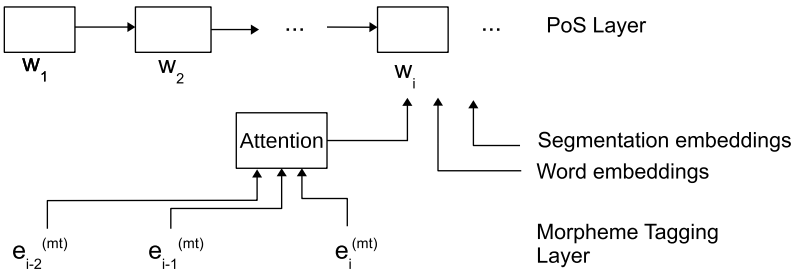

Figure 8. An illustration of morpheme tag cross-level contextual information flow between POS layer and morpheme tagging layer. The context contains only two previous words in the example. The POS layer also takes the morpheme segmentation embeddings and word embeddings which are also based on their own attention networks; they are excluded from the figure for the sake of simplicity.

layer and morpheme tagging layer, where e^{mt}_{i-2} and e^{mt}_{i-1} correspond to the previous two words but particularly to their morpheme tags and w_i is the current word with a POS tag p_i. Therefore, the attention network learns how the morpheme tags of two previous words affect the POS tag of the current word. We generalize it further to a larger context to the left and right, and between various layers in the architecture. A similar cross-level contextual information flow is also built between dependency layer and morpheme tagging layer, or dependency layer and segmentation layer; or POS tagging layer and segmentation layer. Attention is replaced with all the methods mentioned (weighted sum, elementwise multiplication, MLP, concatenative attention, dot-product attention, and CNN), to combine the contextual information in each layer to feed into upper layer.

3.7 Model training

In the proposed joint learning framework, each component has its own error that contributes to the total loss of the model $\mathscr{L}$:

$$\mathscr{L} = \mathscr{L}_{\text{seg}} + \mathscr{L}_{\text{mtag}} + \mathscr{L}_{\text{POS}} + \mathscr{L}_{\text{arc}} + \mathscr{L}_{\text{rel}} \tag{31}$$

We use training and validation corpora to train the model. During training, at each epoch, the model is first trained by Adaptive Moment Estimation (Adam) optimization algorithm (Kingma and Ba 2015). Then, per-token accuracy is calculated using the validation set. At every tenth epoch, all momentum values are cleared in the Adam Trainer. We applied step-based learning rate decay and early stopping.

3.8 Realization

We implemented our model in DyNet v2.0 (Neubig *et al*. 2017), a dynamic neural network toolkit. The implementation of the model is publicly available at https://github.com/halecakir/JointParser. For the morphological segmentation component, we split each token into characters. We represent each character with a 50-dimensional character embedding. Using characters as input, we build a BiLSTM with 50-dimensional unit size to encode the character context. We feed an MLP with a sigmoid output that has an output size of 1 with the encoded characters to capture the segment boundaries.

Based on the segmentation probabilities obtained from the MLP, we created a morpheme list that makes up the input word. We encode morphemes with 50-dimensional embeddings. We feed another BiLSTM with 50-dimensional units to encode each input morpheme. We concatenate the 200-dimensional word-level word embeddings, 50-dimensional character-level word embeddings, and 100-dimensional morpheme-level word embeddings for the final word representation.

https://doi.org/10.1017/S1351324921000371 Published online by Cambridge University Press

For the morpheme tagging component, we use a two-layered BiLSTM with 128-dimensional unit size to encode the character context. The decoder is also based on a two-layered BiLSTM with 128-dimensional unit size that is fed with the attended characters that are encoded by the encoder.

For the POS tagging component, we use a two-layered BiLSTM with 128-dimensional unit size to encode the word context. We feed the output of the word context into an MLP with a *softmax* activation function and 100-dimensional hidden unit size. The output size of the MLP is equal to the number of distinct POS tags in a language. *Softmax* gives the probability vector for the POS categories. We use the negative log-likelihood function as a loss function for the POS tagging component.

For the dependency parsing component, we use a two-layered BiLSTM with 128-dimensional unit size and a pointer network with 100-dimensional hidden unit size.[h] We use negative log-likelihood function for training.

We adopt dropout to penalize some random weights to zero and observed significant improvement when we applied dropout with a rate of 0.3 after each LSTM.

4. Experiments and results

We have tested the system in various languages. First, we report Turkish results.

4.1 Turkish data

We used the UD Turkish Treebank for both training and evaluation. The dataset is a semi-automatic conversion of the IMST Treebank (Sulubacak *et al.* 2016), which is a re-annotated version of the METU-Sabancı Turkish Treebank (Oflazer *et al.* 2003). All of the three treebanks share the same raw data, with 5635 sentences from daily news reports and novels.

For the pretrained word embeddings, we use pretrained 200-dimensional word embeddings trained on Boun Web Corpus (Sak, Güngör, and Saraçlar 2008) provided by CoNLL 2018 Shared Task. We also pre-train morph2vec (Üstün *et al.* 2018) to learn the morpheme embeddings on METU-Sabancı Turkish Treebank. We obtain the gold segments from rule-based Zemberek (Akın and Akın 2007), to train the segmentation component by disambiguating segmentation of a word in the given context in METU-Sabancı Treebank. Each word token then has only one interpretation.

4.2 Turkish results

We performed several experiments with the alternating components of the proposed model. All models are trained jointly with a single loss function. Depending on the examined joint tasks, we excluded either morpheme tagging or segmentation or both tasks from the full model. We observed how the lower levels that correspond to morpheme tagging and segmentation tasks affect the upper levels that refer to POS tagging and dependency parsing.

We followed the standard evaluation procedure of CoNLL 2018 shared task defined for dependency parsing, morphological tagging, and POS tagging. For the POS tagging, we evaluated our results with an accuracy score that is the percentage of the correctly tagged words. For the dependency parsing task, we present two different evaluation scores: labeled attachment score (LAS) and unlabeled attachment score (UAS). The attachment score is the percentage of the words that have the correct head and the dependency arc. LAS measures the percentage of the words that have the correct head with the correct dependency label, whereas UAS only measures the percentage of the words that have the correct head without considering the dependency labels.

[h] All dimensions in the network are determined empirically.

https://doi.org/10.1017/S1351324921000371 Published online by Cambridge University Press

Table 1. Experimental results for different joint models

	POS	UAS	LAS	FEATS	SEG
POS-DEP (Baseline)	92.16	70.42	62.71	–	–
MorphTag-POS-DEP	93.72	70.99	63.75	87.34	–
SEG-POS-DEP	94.51	70.99	62.62	–	98.90
SEG-MorphTag-POS-DEP	**94.78**	**71.00**	**63.92**	**87.59**	**98.97**

We evaluate segmentation results also for accuracy, which is the percentage of the correctly segmented words. We evaluate morpheme tagging with an accuracy measure, which is in this case the percentage of correctly tagged words. We evaluate morpheme tagging based on all of the morpheme tags that each word bears. We assume that all the morpheme tags of the word must be correct in order to count the word as correctly tagged. It is labeled FEATS in the tables (universal morphological tags).

Because gold segments are not provided in the CoNLL datasets, we segmented the test set with Zemberek morphological segmentation tool of Akın and Akın (2007) and evaluated training based on Zemberek results. Therefore, our segmentation results are computed relative to Zemberek.

All the results obtained for the proposed models of Turkish are given in Table 1. We compare them with the joint POS tagging and dependency parsing model of Nguyen and Verspoor (2018), which is the base model for us. Recall that it has two layers, given as POS-DEP (Baseline) in the table. Our baseline with these two layers only approximately replicate the results of Nguyen and Verspoor (2018). We obtained an UAS score of 70.42% and a LAS score of 62.71% from the baseline model. The results of the baseline in Table 1 are obtained using Nguyen and Verspoor (2018)'s own implementation. However, we have been unable to replicate their reported scores identically, possibly due to some minor parametric differences because of hyperparameters. POS tagging accuracy is the highest with 94.78% when all the components (i.e., addition of morpheme tagging and morphological segmentation) are included in a full joint model. The same also applies to UAS, LAS, morpheme tagging accuracy, and segmentation accuracy, which are the highest when all the components are adopted in the model. This shows that all the layers in the model contribute to each other during learning.

Ours being the first attempt to combine five tasks in a single joint model, we compare the performance of each component separately with different models. Results of dependency parsing are in Table 2. The table shows the UAS and LAS scores of different dependency parsing models. We compare our results with the joint POS tagging and dependency parsing model by Nguyen and Verspoor (2018), with the winning contribution to CoNLL 2018 shared task that incorporates deep contextualized word embeddings (Che *et al*. 2018), with the horizontal pipeline system that performs tokenization, word segmentation, POS tagging, and dependency parsing by Qi *et al*. (2018), with the tree-stack LSTM model by Kırnap, Dayanık, and Yuret (2018), and with the dependency parser that incorporates morphology-based word embeddings proposed by Özateş *et al*. (2018).

The results show that our model has nontrivial improvement on the baseline model of Nguyen and Verspoor (2018). Moreover, our model outperforms most of the models that participated in CoNLL 2017 and CoNLL 2018. The only models that perform better than our model are the ones proposed by Che *et al*. (2018) and Qi *et al*. (2018). Their models give a UAS score of 72.25% and 71.07%, LAS score of 66.44% and 64.42%, respectively, whereas our model gives a UAS score of 71.00% and LAS score of 63.92%. Our UAS score is competitive with the deep contextualized model by Che *et al*. (2018); however, there is a 2.5% difference in the LAS scores of the two models. As the authors of that work also state, deep contextualized word embeddings (BERT, Devlin *et al*. 2019) affect parsing accuracy significantly.

https://doi.org/10.1017/S1351324921000371 Published online by Cambridge University Press

Table 2. Comparison of Turkish dependency parsing results with other models

Model	UAS	LAS
Che *et al*. (2018)	**72.25**	**66.44**
Qi *et al*. (2018)	71.07	64.42
Nguyen and Verspoor (2018)	70.53	62.55
Straka (2018)	69.34	63.07
Kırnap *et al*. (2018)	65.93	58.75
Özateş *et al*. (2018)	57.53	50.33
SEG-MORPH-POS-DEP	71.00	63.92
MORPH-POS-DEP	70.99	63.75
POS-DEP	70.42	62.71
SEG-POS-DEP	70.99	62.62

Table 3. The comparison of the Turkish POS tagging results with other models

Model	Accuracy (UPOS)
Che *et al*. (2018)	**94.78**
Straka (2018)	93.58
Qi *et al*. (2018)	93.20
Özateş *et al*. (2018)	91.64
Kırnap *et al*. (2018)	91.64
Nguyen and Verspoor (2018)	92.93
SEG-MORPH-POS-DEP	**94.78**
SEG-POS-DEP	94.51
MORPH-POS-DEP	93.72

Apart from the usage of contextualized word embeddings, our base model is similar to their model. Our full model performs morpheme tagging and morphological segmentation additionally. Qi *et al*. (2018) presents another prominent model which competed in CoNLL shared task 2018. They were placed second in the shared task with their UAS and LAS scores. Similar to our model, they also introduce a neural pipeline system that performs tokenization, segmentation, POS tagging, and dependency parsing. Their architecture is similar to ours, differently they use biaffine classifier for POS tagging similar to that of Dozat and Manning (2017).

The comparison of our POS tagging results with other POS tagging models is given in Table 3. We compare our model with Che *et al*. (2018), Qi *et al*. (2018), Özateş *et al*. (2018), and Kırnap *et al*. (2018). Our model outperforms other models with an accuracy of 94.78% performing similar to Che *et al*. (2018), which uses deep contextualized word embeddings.

https://doi.org/10.1017/S1351324921000371 Published online by Cambridge University Press

Table 4. The comparison of the Turkish morphological tagging results (FEATS) with other models

Model	FEATS
Straka (2018)	**91.25**
Dayanık *et al*. (2018)	89.54
Qi *et al*. (2018)	89.43
Che *et al*. (2018)	86.23
Özateş *et al*. (2018)	86.15
Kırnap *et al*. (2018)	86.15
SEG-MORPH-POS-DEP	87.59
MORPH-POS-DEP	87.34

The comparison of our morphological tagging results (FEATS) with other models is given in Table 4. We compare our model with Straka (2018), Dayanık *et al*. (2018), Che *et al*. (2018), Qi *et al*. (2018), Özateş *et al*. (2018), and Kırnap *et al*. (2018). The best performing model is by Straka (2018) with an accuracy of 91.25%. It is worth mentioning that none of these models performs a joint learning for morphological tagging. Our model is the only joint model that performs morphological tagging along with other syntactic tasks. The results are very competitive with other models although they fall slightly behind them.

4.3 The effect of cross-level contextual information flow

The cross-level contextual information flow has been investigated with the following methods: weighted sum, elementwise multiplication, MLP, and attention network. We include only the previous word as context in weighted sum, elementwise multiplication, and MLP, and we incorporate various lengths of contextual information both from left and right in the attention mechanism.

The results of the proposed methods for cross-level contextual information flow on Turkish are given in Table 5. Highest results are obtained with basic dot-product attention.

Attention networks resolve the issue of manual weight assignment by learning the weight of each context word automatically during training. However, attention with concatenation requires a separate development set for hyperparameter tuning to find the optimum weights for the contextual words. Therefore, it has to be performed for every language and dataset separately using a development set, which is preferably different than the training dataset. We incorporated various lengths of contextual knowledge both for left and right context. For a smaller context, we incorporated two previous words; and for a larger context, we incorporated five previous words. The results obtained from two previous words are comparably lower than that of five previous words. We also incorporated right context with one and two following words including previous words as well. The results with smaller context again performs better than a larger context. This could be due to various lengths of sentences in dataset where there is not enough history for especially larger contexts.

We split the dataset for training and test sets to include various lengths of contextual knowledge for shorter (having less than 10 words) and longer sentences (having 10 or more words). We used five words in the left and one word on the right for longer sentences and used two words in the left and one word on the right for shorter sentences. The overall results are comparably better than that of using a fixed length of context for all sentences, especially in dependency parsing. However,

https://doi.org/10.1017/S1351324921000371 Published online by Cambridge University Press

Table 5. The results of cross-level contextual information flow on Turkish

Method	UAS	LAS	UPOS	FEATS	SEG
Weighted sum	**71.59**	64.42	94.85	86.32	99.02
Elementwise mult.	70.61	63.49	94.86	86.74	99.03
MLP	70.74	63.77	93.75	86.92	98.92
Attention (2 left)	70.89	64.21	94.44	87.13	99.01
Concat attention (5 left)	70.55	63.33	94.29	87.20	98.85
Concat attention (2 left + 1 right)	70.29	63.04	93.31	87.42	99.06
Concat attention (2 left + 2 right)	69.43	62.01	93.30	86.87	99.02
Concat attention (long(5l, 1r)+short(2l, 1r))	71.58	63.07	92.82	83.73	98.69
Dot-product attention (2 left + 1 right)	71.51	**64.81**	**94.90**	**87.18**	**99.02**
CNN (2 left + 1 right)	70.22	63.22	94.22	86.47	98.96

Table 6. The results of joint model on other languages (English, Czech, Hungarian, and Finnish) with (w) and without (w/o) cross-level contextual information flow. The bold ones are the highest scores for the given language

	Language	UAS	LAS	UPOS	FEATS
w/o cross-level flow	English	84.97	80.76	94.36	88.70
	Finnish	86.35	83.17	95.64	88.95
	Hungarian	**76.74**	70.03	92.47	64.09
	Czech	**89.71**	86.27	98.69	81.16
w cross-level flow	English	**88.09**	**85.09**	**95.40**	**91.34**
	Finnish	**86.62**	**83.56**	**95.72**	**90.16**
	Hungarian	77.64	**71.64**	**92.51**	**78.61**
	Czech	89.59	**87.08**	**98.77**	**81.99**

POS tagging and morphological tagging results fall behind that of using a fixed amount of context. This could be due to the size of the training sets when split. However, it still shows that using a larger context indeed helps at the syntactic level.

4.4 Results on English, Czech, Hungarian, and Finnish

There is nothing in our system which is specific to Turkish. We have applied our joint system to other languages, where we use Universal Dependencies v2.3 corpora. The results for English, Czech, Hungarian, and Finnish are given in Table 6. We excluded the morphological segmentation layer from the joint model for these experiments, since gold morphological segmentations are not available for these languages; only morphological tags are available in UD treebanks.

We performed experiments both without using cross-level contextual information flow and with using attention mechanism for the cross-level information flow. We used only previous two words for the contextual information, since the highest scores are obtained from this setting on

https://doi.org/10.1017/S1351324921000371 Published online by Cambridge University Press

Turkish. We applied concatenative attention network for English, Hungarian, and Czech, dot-product attention for Finnish due to the highest evaluation scores.

As can be seen from the results, the cross-level contextual information flow considerably improves the results in all languages. The results demonstrated a substantial improvement, especially in English LAS and Hungarian FEATS.

We compared the highest scores obtained in these languages with other state-of-the-art participated in CoNLL Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies (Zeman and Hajič 2018). The results are given in Table 7. There are 26 participants who shared their results on the given languages. We only compared our scores with the top five systems participated in the shared task. We also provide the scores of Nguyen and Verspoor (2018) to show the improvement obtained by the joint model and cross-level contextual information flow.

Our joint model outperforms all other participants in English for both UAS and LAS. Although we have a significant improvement over the baseline model by Nguyen and Verspoor (2018) in Finnish, Hungarian, and Czech, our model falls behind the top scores in other languages. However, the joint model still performs well on other languages when all participants in the shared task are considered. For example, our model is ranked 11th in Finnish, 13th in Hungarian, and 14th in Czech according to LAS scores; 10th in Finnish, 13th in Hungarian, and 15th in Czech according to UAS scores; 5th in English, 10th in Finnish, and Hungarian, 6th in Czech according to universal POS (UPOS) scores.

4.5 Error analysis for Turkish

We explored the errors made by the proposed model variations in greater detail in one language, Turkish. The following experiments aim to classify parsing errors with respect to structural properties of the dependency graphs. In all experiments, four different joint models have been trained and tested with the standard split of Turkish IMST universal dependencies.[i] Error analysis has been carried out using four properties: sentence length, projectivity, arc type, and POS.

4.5.1 The effect of sentence length

We first try to see the effect of sentence length on the robustness of the four model variations. For that purpose, the test data have been divided into equally sized sentence length intervals. The interval length is 10. The UAS and LAS scores according to different sentence lengths are given in Tables 8 and 9, respectively. It can be seen that the longer the sentences are, the lower both scores are for all model variations. However, MorphTag-POS-DEP and SEG-MorphTag-POS-DEP models perform rather well in shorter sentences (with length 0–20) with regard to both UAS and LAS.

What is interesting about the scores is that the models that have higher UAS and LAS scores in shorter sentences are worse in longer sentences (length 30–50). Parsing errors gradually increase as the sentence length increases. In literature, a similar outcome has been reported for other languages, for example, for English and Vietnamese (McDonald and Nivre 2007; Van Nguyen and Nguyen 2015).

4.5.2 Effect of projectivity

This effect concerns dependencies.

An arc in a dependency tree is projective if there is a directed path from the head word to all the words between the two endpoints of the arc, and therefore there are no crossing edges in the dependency graph. Projectivity is less of a problem in languages with more strict order such as English. However, non-projective trees can be seen more frequently in languages such as Czech,

[i]Turkish IMST universal dependencies: https://universaldependencies.org/treebanks/tr_imst/.

https://doi.org/10.1017/S1351324921000371 Published online by Cambridge University Press

Table 7. Comparison to other models on English, Czech, Hungarian, and Finnish

	Model/system	UAS	LAS	UPOS	FEATS
English (en_ewt)	Joint model	**88.09**	**85.09**	95.40	91.34
	Nguyen and Verspoor (2018)	87.55	84.71	95.48	–
	HIT-SCIR (Che *et al*. 2018)	86.79	84.57	95.64	94.60
	LATTICE (Lim *et al*. 2018)	86.90	84.02	**95.94**	94.60
	Stanford (Qi *et al*. 2018)	86.40	83.87	95.46	95.99
	CEA LIST (Duthoo and Mesnard 2018)	85.40	82.88	95.01	95.55
	NLP-Cube (Boros, Dumitrescu, and Burtica 2018)	85.49	82.79	95.25	**96.03**
	UDPipe 2.0 (Straka 2018)	85.01	82.51	95.43	95.99
Finnish (fi_tdt)	Joint model	86.57	83.37	95.80	90.19
	Nguyen and Verspoor (2018)	86.07	82.92	96.12	–
	HIT-SCIR (Che *et al*. 2018)	**90.95**	**88.73**	**97.30**	92.17
	LATTICE (Lim *et al*. 2018)	88.39	85.42	97.12	92.06
	Stanford (Qi *et al*. 2018)	88.91	87.64	96.83	95.26
	CEA LIST (Duthoo and Mesnard 2018)	88.44	85.99	96.53	94.78
	NLP-Cube (Boros *et al*. 2018)	87.06	83.74	95.52	92.41
	UDPipe 2.0 (Straka 2018)	88.10	85.72	96.75	**95.43**
Hungarian (hu_szeged)	Joint model	77.64	71.64	92.51	78.61
	Nguyen and Verspoor (2018)	76.18	69.75	92.07	–
	HIT-SCIR (Che *et al*. 2018)	**87.21**	**82.66**	**96.43**	88.09
	LATTICE (Lim *et al*. 2018)	80.49	74.21	91.58	88.09
	Stanford (Qi *et al*. 2018)	80.49	77.64	95.35	91.43
	CEA LIST (Duthoo and Mesnard 2018)	81.54	76.96	95.40	**92.75**
	NLP-Cube (Boros *et al*. 2018)	81.52	75.85	94.97	89.37
	UDPipe 2.0 Straka (2018)	83.08	78.51	95.48	92.41
Czech (cs_pdt)	Joint model	89.59	87.08	98.77	81.99
	Nguyen and Verspoor (2018)	89.64	87.04	98.74	–
	HIT-SCIR (Che *et al*. 2018)	**93.44**	**91.67**	**99.05**	92.40
	LATTICE (Lim *et al*. 2018)	92.73	90.13	98.99	92.40
	Stanford (Qi *et al*. 2018)	92.44	90.38	98.52	94.25
	CEA LIST (Duthoo and Mesnard 2018)	91.38	89.06	98.44	92.75
	NLP-Cube (Boros *et al*. 2018)	91.63	89.45	98.76	**95.22**
	UDPipe 2.0 Straka (2018)	92.32	90.32	99.01	96.85

https://doi.org/10.1017/S1351324921000371 Published online by Cambridge University Press

Table 8. LAS by length

	<=10	<=20	<=30	<=40	<=50
POS-DEP	70.27	59.95	55.33	54.77	**55.20**
SEG-POS-DEP	71.46	60.18	57.63	53.18	54.75
MorphTag-POS-DEP	**71.92**	60.95	**55.94**	54.32	52.94
SEG-MorphTag-POS-DEP	71.48	**61.43**	55.79	**59.09**	52.49
Number of tokens	3,875	3,516	1,952	440	221

Table 9. UAS by length

	<=10	<=20	<=30	<=40	<=50
POS-DEP	77.73	67.52	63.88	63.18	60.63
SEG-POS-DEP	78.86	67.95	**64.04**	61.59	**61.54**
MorphTag-POS-DEP	**79.07**	68.29	63.17	62.05	59.28
SEG-MorphTag-POS-DEP	78.71	**68.57**	62.50	**65.45**	60.63
Number of tokens	3,875	3,516	1,952	440	221

Table 10. Statistics about test partition of Turkish IMST universal dependencies

	Tokens	Sentences
Projective	7928	855
Non-projective	2076	120
Total	10,004	975

German and Turkish. Table 10 shows some statistics about the test set. Turkish being a free order language, the number of non-projective dependency graphs is quite significant.

However, it is not only relatively free word order in languages such as Turkish that leads to various number of non-projective dependencies. Turkish root relativization may be nested Figure 9(a)–(b), or crossing Figure 9(c), without manipulating free word order effects.

In the case of long-range dependencies crossing clause boundaries, Turkish is *always* crossing, as can be seen from Figure 10; therefore, a dataset that contains abundance of such examples would afford very little chance of adequate coverage when non-projective dependencies are eliminated.

There is no limit to crossings in Turkish, because there is no limit to long-range relativization. N-level embedded complementation in an expression requires n crossings to capture the dependency relation of the extracted element with its verb, as shown in Figure 10. Therefore, there is no such thing as mildly non-projective Turkish long-range relative clause dependency (cf. Kuhlmann and Nivre 2006).

We point out that filtering out crossing dependencies for evaluation as, for example, done by Sulubacak and Eryiğit (2018) would take away a lot of non-crossing dependencies as well, so all kinds of parsers, projective and non-projective, would be affected by this elimination in their training and performance in simpler cases.

https://doi.org/10.1017/S1351324921000371 Published online by Cambridge University Press

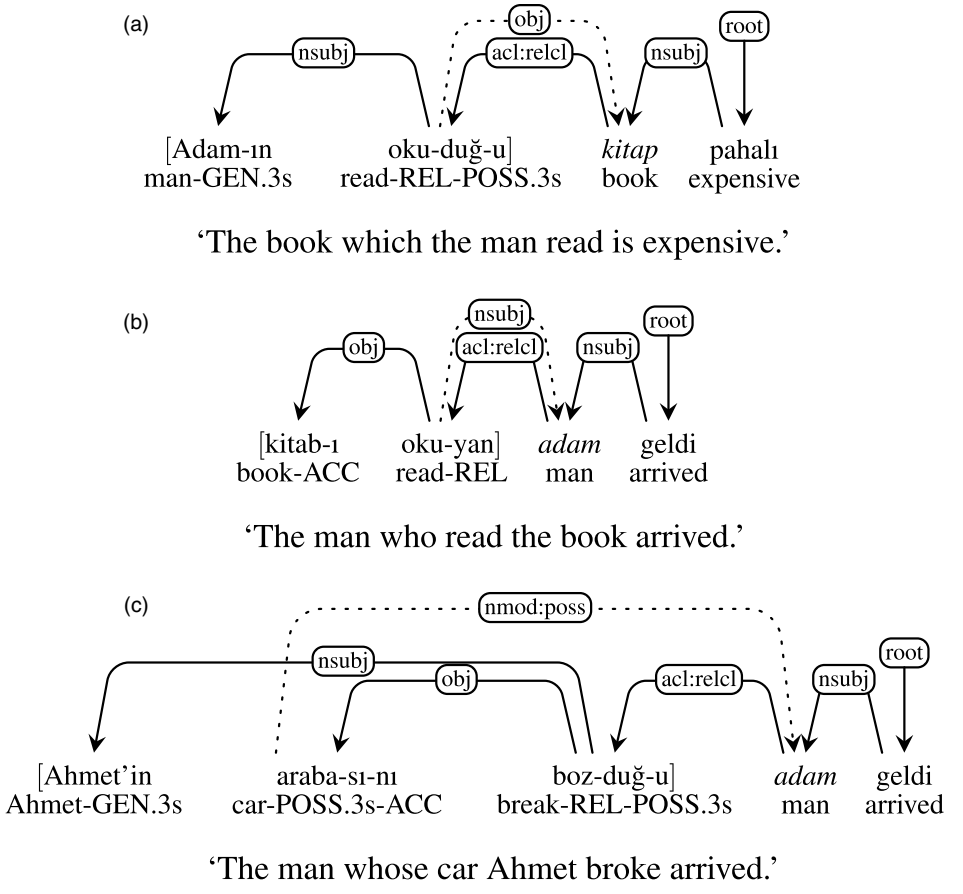

Figure 9. Dependencies in Turkish relative clauses (in brackets), and with the relativized noun (in italic). They are (a) root-clause object relativization; (b) root-clause subject relativization; (c) relativization out of possessive NPs, here from "man's car": "*adam-GEN.3s araba-POSS.3s*." Dotted edges indicate which dependency relation "acl:rel" dependency is expected to capture.

We designed the joint-task cross-level contextual information flow with these considerations in mind, so that all kinds of dependencies when available in standard datasets for evaluation can be modeled. Adding attention along with cross-level contextual information handling must be regarded as the first-step toward handling such data with dependencies when they are available, which would allow a meaningful comparison of dependency parsing with, for example, CCG parsing (Hockenmaier and Steedman 2007; Çakıcı, Steedman, and Bozşahin2018), where so-called non-projectivity is handled by bounded function composition on complement clauses.

We hope that our sticking to dependency parsing for the moment may help assess current state of the art and provide a prospect for the future with such kind of data. In particular, we suggest reporting constructions by genera (control, suboordination, relativization, coordination, etc.), cross-listed by bounded and long-range dependencies, as also pointed out by Hockenmaier and Steedman (2007). Without such standardizations, significance tests for improvement or degradation over datasets would not reveal much about what is learned about a language in a model.

Figure 11 gives the percentages of projective and non-projective trees according to sentence length. The results show that as the complexity of sentence increases; therefore, the length, the probability of generating non-projective trees also increases. The number of projective trees is

https://doi.org/10.1017/S1351324921000371 Published online by Cambridge University Press

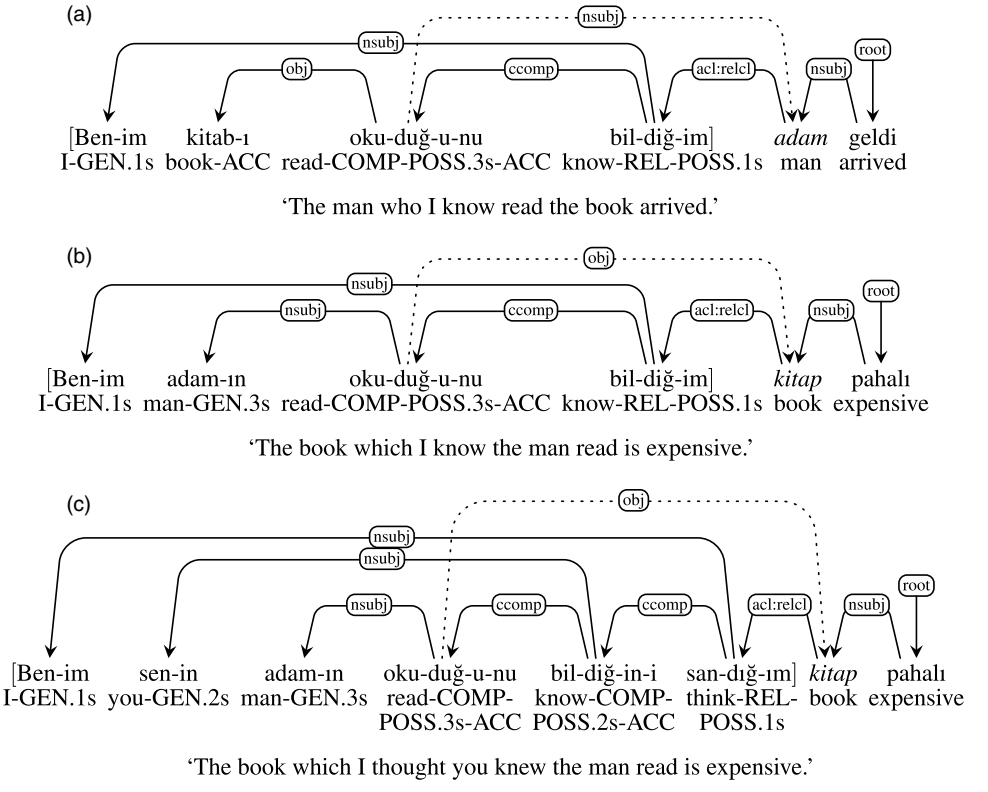

Figure 10. Unbounded dependencies in Turkish relative clauses (in brackets), and with the relativized noun (in italic). They are (a) long-range subject relativization and (b–c) long-range object relativization. Dotted edges indicate which dependency relation "acl:rel" dependency is expected to capture.

higher for the sentence lengths smaller than 30. The number of non-projective trees is higher for the sentence lenghts greater than 40. For example, for all the sentences that are longer than 50, the dependency tree is non-projective.

We analyze the labeled and unlabeled accuracy scores obtained from the projective and non-projective trees in testing. As can be seen in Table 11, the MorphTag-POS-DEP model outperforms other models both for labeled and unlabeled dependency arcs in projective trees. However, for the non-projective trees, the SEG-MorphTag-POS-DEP model outperforms other models. We aim in the long run for wide-coverage systems with reasonable approximation of semantics; therefore, we take it as a good sign that the full model is responsive to non-projective, therefore more complex sentences.

4.5.3 The effect of constructions

Finally, we analyze the syntactic properties of dependency relations based on their POS labels obtained from different models. We compare the results based on dependent word's POS tag to see whether it plays a role in the accuracy of dependencies. Following McDonald and Nivre (2007), we make use of coarse POS categories instead of UPOS categories in order to observe the performance differences between the models in loss of useful syntactic information. In coarse POS categories, nouns include nouns and proper nouns, pronouns consist of pronouns and determiner, and verbs include verbs and auxiliary verbs.

UAS and LAS accuracies of the four models are given in Tables 12 and 13, respectively for the coarse POS categories. As can be seen from Table 12, there is a slight improvement of UAS

https://doi.org/10.1017/S1351324921000371 Published online by Cambridge University Press

Table 11. Results for projective and non-projective sentences

	Model	UAS	LAS
Projective	POS-DEP	72.97	65.05
	SEG-POS-DEP	73.65	65.98
	MorphTag-POS-DEP	**73.75**	**66.30**
	SEG-MorphTag-POS-DEP	73.25	66.16
Non-Projective	POS-DEP	60.69	53.81
	SEG-POS-DEP	60.84	54.62
	MorphTag-POS-DEP	60.45	54.05
	SEG-MorphTag-POS-DEP	**62.43**	**55.39**

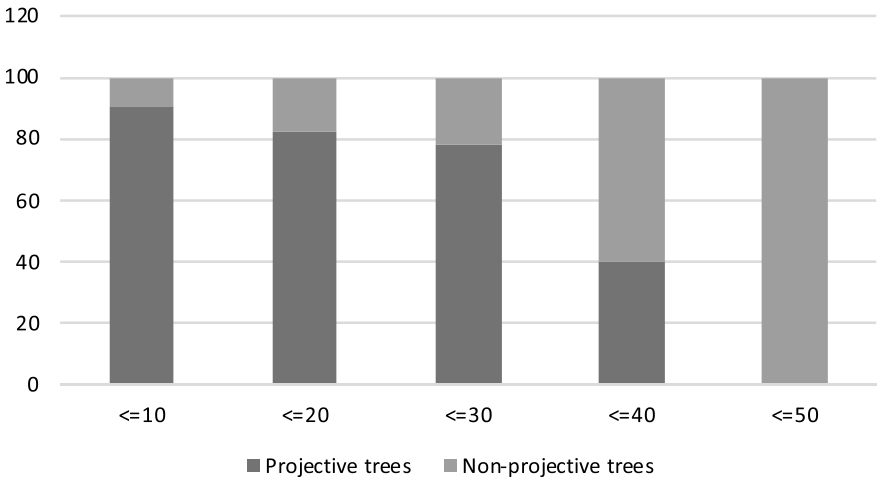

Figure 11. Projective and non-projective tree percentages grouped by the sentence length.

in conjunctions (CONJ), and substantial improvement in verbs (VERB) and adpositions (ADP) in the SEG-MorphTag-POS-DEP model. The MorphTag-POS-DEP model captures the unlabeled arcs for the adjective (ADJ) and adverb (ADV) dependents better compared to other models. The scores also show that there is a slight increase in the unlabeled accuracy for the noun (NOUN) and pronoun (PRON) dependents in the SEG-POS-DEP model. The POS-DEP model does not outperform other models for any of the POS categories. Only the UAS score for the adverb dependents is the same as the one obtained from the MorphTag-POS-DEP. LAS and UAS scores are given for all dependency relations in Tables A1 and A2, respectively, in Appendix A.

The overall results show that each component in the model has an impact on the UAS scores for different POS categories of the dependents, and the full joint model that includes all the components has a higher overall UAS accuracy for different dependent word's POS categories.

As can be seen from Table 13, the SEG-MorphTag-POS-DEP model tends to be more accurate in nouns (NOUN) and conjunctions (CONJ). With the addition of morpheme tags, the MorphTag-POS-DEP model has higher LAS accuracies for adjectives (ADJ), adverbs (ADV), and verbs (VERB). There is a considerable improvement of LAS in adpositions (ADP) after

https://doi.org/10.1017/S1351324921000371 Published online by Cambridge University Press

Table 12. UAS by coarse POS categories

Dependent POS	Total	MT POS-DEP	MT SEG-POS -DEP	MT MorphTag -POS-DEP	MT SEG-MorphTag -POS-DEP
ADJ	939	72.52	72.63	**74.01**	72.31
ADP	360	76.11	81.39	78.89	**80.56**
ADV	345	**68.70**	65.22	**68.70**	66.09
CONJ	390	75.38	74.10	74.62	**75.64**
NOUN	3145	65.47	**66.83**	66.20	65.98
PRON	530	76.60	**77.17**	75.85	76.61
VERB	2121	72.09	71.28	72.99	**73.51**

Table 13. LAS by coarse POS categories

Dependent POS	Total	MT POS-DEP	Mt SEG-POS -DEP	Mt MorphTag -POS-DEP	MT SEG-MorphTag -POS-DEP
ADJ	939	57.51	59.74	**60.38**	60.06
ADP	360	73.61	**79.44**	76.39	78.06
ADV	345	60.58	58.26	**61.74**	60.58
CONJ	390	69.49	66.92	66.41	**69.74**
NOUN	3145	53.90	55.86	55.74	**56.00**
PRON	530	**68.68**	67.73	67.36	67.36
VERB	2121	66.95	66.38	**68.18**	67.89

incorporating segmentation information in the joint SEG-POS-DEP model; however, the SEG-POS-DEP model does not outperform other models for other POS categories. The only POS category where the POS-DEP model outperforms other models is pronouns (PRON), which can be a sign that pronouns are usually not inflected and the morph tags and segmentation do not contribute much for the pronoun category.

4.6 Ablation

In order to measure the contribution of components in the model, we remove one component tentatively, and test. It is an ablation analysis, and, in the previous sections, we have done a coarse version of it.

There is also another version of ablation analysis that measures the influence of the individual components in the model by incrementally replacing the component with their gold annotations. This version of ablation analysis is reported by Qi *et al.* (2018). Using this approach, researchers have been able to decide whether an individual component is useful for higher-level tasks and estimate the limitations of their studies under perfect conditions. This way, the errors that flow through the network from different components are filtered out to measure the actual effect of each component on the final scores.

https://doi.org/10.1017/S1351324921000371 Published online by Cambridge University Press

Table 14. Pipeline ablation results

	POS	UAS	LAS	UFeats
POS-MorphTag-SEG-DEP	94.78	71.00	63.92	87.59
POS-MorphTag-GoldSEG-DEP	94.90	70.93	64.05	87.81
POS-GoldMorphTag-SEG-DEP	97.30	**72.34**	**66.68**	100
POS-GoldMorphTag-GoldSEG-DEP	**97.36**	71.97	65.94	100

The results obtained from the pipeline ablation analysis of the joint model are given in Table 14. The first row gives the results of the full joint model without gold annotations. In the second row (POS-MorphTag-GoldSEG-DEP), the morpheme segmentation component is replaced with the gold morphological segments rather than predicting them, and the rest of the components are left as they are. In the third row (POS-GoldMorphTag-SEG-DEP), the morphological tagging component is replaced with the gold morpheme tags.

Finally, both morpheme segmentation and morphological tagging components are replaced with their gold annotations in the fourth row (POS-GoldMorphTag-GoldSEG-DEP). The results show that using the gold morpheme tags contributes to the dependency scores the most. However, the highest improvement is obtained when both gold morphological segmentations and gold morpheme tags are incorporated into the model for POS tagging. We are encouraged by this result because of our motivation to add morpheme information to joint learning from the beginning.

When individual components are replaced with the gold annotations incrementally, from bottom tasks to the upper layers, there is still a huge gap between the ablation scores and the gold scores. This is where linguistic theories can fill in the gap, for morphology, phonology, and syntax, by narrowing the solution space of every task of a joint model.

5. Conclusion and future work

We presented a joint learning framework for learning POS tagging, morphological tagging, morphological segmentation and dependency parsing to the extent of obtaining dependency relations of who does what to whom. We use a deep neural architecture where each layer learns a specific level, and this knowledge is shared through other layers.

The results show that using morphological knowledge such as morpheme tags and morphs themselves improve both dependency parsing and POS tagging. Morphemic knowledge plays an important role in morphologically rich languages.

Applying deep contextualized word embeddings such as BERT (Devlin *et al*. 2019) or Elmo (Peters *et al*. 2018) remain as a future goal in this work. We plan to replace the LSTMs with self-attention mechanisms of variable reach at the levels of the current architecture, as a controlled means of extending the context. We believe that using self-attention mechanism instead of LSTMs will handle the long-term dependencies better, especially in longer sentences, and in larger left and right contexts.

Acknowledgments. This research has been supported by Scientific and Technological Research Council of Turkey (TUBITAK), project number EEEAG-115E464.

References

Akn A.A. and Akn M.D. (2007). Zemberek, an open source NLP framework for Turkic languages. *Structure* **10**, 1–5.

Akyürek E., Dayanık E. and Yuret D. (2019). Morphological analysis using a sequence decoder. *Transactions of the Association for Computational Linguistics* **7**, 567–579.

https://doi.org/10.1017/S1351324921000371 Published online by Cambridge University Press

Anderson S.R. (1992). *A-Morphous Morphology*. Cambridge, UK: Cambridge University Press.

Anderson S.R., Brown L., Gaby A. and Lecarme J. (2006). Life on the edge: there's morphology there after all! *Lingue e Linguaggio* 1(1), 33–47.

Boros T., Dumitrescu S.D. and Burtica R. (2018). NLP-cube: end-to-end raw text processing with neural networks. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, Brussels, Belgium. Association for Computational Linguistics, pp. 171–179.

Can B. and Manandhar S. (2010). Clustering morphological paradigms using syntactic categories. In *Proceedings of the Multilingual Information Access Evaluation I. Text Retrieval Experiments: 10th Workshop of the Cross-Language Evaluation Forum, Revised Selected Papers*. Berlin, Heidelberg: Springer, pp. 641–648.

Caruana R. (1997). Multitask learning. *Machine Learning* **28**, 41–75.

Çakıcı R., Steedman M. and Bozşahin C. *(2018). Wide-coverage parsing, semantics, and morphology. In Oflazer K. and Saraçlar M. (eds), Turkish Natural Language Processing. Springer, pp. 153–174.*

Che W., Liu Y., Wang Y., Zheng B. and Liu T. (2018). Towards better UD parsing: deep contextualized word embeddings, ensemble, and treebank concatenation. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. Association for Computational Linguistics, pp. 55–64.

Clark A. (2000). Inducing syntactic categories by context distribution clustering. In *Proceedings of the 2nd Workshop on Learning Language in Logic and the 4th Conference on Computational Natural Language Learning-Volume 7*. Association for Computational Linguistics, pp. 91–94.

Cotterell R. and Heigold G. (2017). Cross-lingual character-level neural morphological tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pp. 748–759.

Creutz M. and Lagus K. (2007). Unsupervised models for morpheme segmentation and morphology learning. *ACM Transactions Speech Language Processing* **4**, 3:1–3:34.

Dayanık E., Akyürek E. and Yuret D. (2018). MorphNet: a sequence-to-sequence model that combines morphological analysis and disambiguation. CoRR, abs/1805.07946.

de Marneffe M.-C., Manning C.D., Nivre J. and Zeman D. (2021). Universal dependencies. *Computational Linguistics* **47**(2), 255–308.

Devlin J., Chang M.-W., Lee K. and Toutanova K. (2019). Bert: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT 2019*, Minneapolis, Minnesota. Association for Computational Linguistics, pp. 4171–4186.

Dos Santos C.N. and Zadrozny B. (2014). Learning character-level representations for part-of-speech tagging. In *Proceedings of the 31st International Conference on International Conference on Machine Learning (ICML 2014) - Volume 32*. JMLR.org, pp. II–1818–II–1826.

Dozat T. and Manning C.D. (2017). Deep biaffine attention for neural dependency parsing. In *5th International Conference on Learning Representations, ICLR 2017, Conference Track Proceedings*. OpenReview.net.

Duthoo E. and Mesnard O. (2018). CEA LIST: processing low-resource languages for CoNLL 2018. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, Brussels, Belgium.

Eisner J.M. (1996). Three new probabilistic models for dependency parsing: an exploration. In *Proceedings of COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics*, pp. 340–345.

Foley W.A. (1998). Symmetrical voice systems and precategoriality in Philippine languages. In *LFG Conference, Workshop on Voice and Grammatical Functions in Austronesian Languages*, The University of Queensland, Australia.

Göksel A. and Kerslake C. (2005). *Turkish: A Comprehensive Grammar*. London: Routledge.

Goldsmith J. (2001). Unsupervised learning of the morphology of a natural language. *Computational Linguistics* **27**(2), 153–198.

Goldwater S., Griffiths T.L. and Johnson M. (2011). Producing power-law distributions and damping word frequencies with two-stage language models. *Journal of Machine Learning Research* **12**, 2335–2382.

Heigold G., Neumann G. and van Genabith J. (2017). An extensive empirical evaluation of character-based morphological tagging for 14 languages. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Association for Computational Linguistics, pp. 505–513.

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

Hockenmaier J. and Steedman M. (2007). CCGbank: a corpus of CCG derivations and dependency structures extracted from the Penn Treebank. *Computational Linguistics* **33**(3), 356–396.

Kingma D.P. and Ba J. (2015). Adam: a method for stochastic optimization. In Bengio, Y. and LeCun Y. (eds), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7–9, 2015, Conference Track Proceedings*.

Kiperwasser E. and Goldberg Y. (2016). Simple and accurate dependency parsing using bidirectional LSTM feature representations. *Transactions of the Association for Computational Linguistics* **4**, 313–327.

Kırnap Ö., Dayanık E. and Yuret D. (2018). Tree-stack LSTM in transition based dependency parsing. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. Association for Computational Linguistics, pp. 124–132.

https://doi.org/10.1017/S1351324921000371 Published online by Cambridge University Press

Kondratyuk D. and Straka M. (2019). 75 languages, 1 model: parsing universal dependencies universally. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, Hong Kong, China. Association for Computational Linguistics, pp. 2779–2795.

Kornfilt J. (2001). Functional projections and their subjects in Turkish clauses. In *The Verb in Turkish*. John Benjamins.

Kudo T. and Matsumoto Y. (2002). Japanese dependency analysis using cascaded chunking. In *Proceedings of the 6th Conference on Natural Language Learning - Volume 20*, COLING-02, Stroudsburg, PA, USA. Association for Computational Linguistics, pp. 1–7.

Kuhlmann M. and Nivre J. (2006). Mildly non-projective dependency structures. In *Proceedings of COLING-ACL*, Sydney, pp. 507–514.

Lecun Y., Bottou L., Bengio Y. and Haffner P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE* **86**(11), 2278–2324.

Li J., Liu X., Yin W., Yang M., Ma L. and Jin Y. (2020). Empirical evaluation of multi-task learning in deep neural networks for natural language processing. Neural Computing and Applications.

Lim K., Park C., Lee C. and Poibeau T. (2018). SEx BiST: a multi-source trainable parser with deep contextualized lexical representations. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, Brussels, Belgium. Association for Computational Linguistics, pp. 143–152.

Ling W., Dyer C., Black A.W., Trancoso I., Fermandez R., Amir S., Marujo L. and LuÍs T. (2015). Finding function in form: compositional character models for open vocabulary word representation. In *EMNLP*. The Association for Computational Linguistics, pp. 1520–1530.

Liu X., He P., Chen W. and Gao J. (2019). Multi-task deep neural networks for natural language understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy. Association for Computational Linguistics, pp. 4487–4496.

McDonald R., Crammer K. and Pereira F. (2005). Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL'05, Stroudsburg, PA, USA. Association for Computational Linguistics, pp. 91–98.

McDonald R. and Nivre J. (2007). Characterizing the errors of data-driven dependency parsing models. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, Prague, Czech Republic. Association for Computational Linguistics, pp. 122–131.

Mikolov T., Chen K., Corrado G. and Dean J. (2013). Efficient estimation of word representations in vector space. In Bengio Y. and LeCun, Y. (eds), *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2–4, 2013, Workshop Track Proceedings*.

Müller T., Cotterell R., Fraser A. and Schütze H. (2015). Joint lemmatization and morphological tagging with lemming. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, Lisbon, Portugal. Association for Computational Linguistics, pp. 2268–2274.

Neubig G., Dyer C., Goldberg Y., Matthews A., Ammar, W., Anastasopoulos A., Ballesteros M., Chiang D., Clothiaux D., Cohn, T., Duh K., Faruqui M., Gan C., Garrette D., Ji Y., Kong L., Kuncoro A., Kumar G., Malaviya C., Michel P., Oda, Y., Richardson M., Saphra N., Swayamdipta S. and Yin P. (2017). Dynet: the dynamic neural network toolkit. arXiv preprint arXiv:1701.03980.

Nguyen D.Q. and Verspoor K. (2018). An improved neural network model for joint POS tagging and dependency parsing. In *Proceedings of the CoNLL Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. The Association for Computational Linguistics, pp. 81–91.

Nivre J. and Nilsson J. (2005). Pseudo-projective dependency parsing. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL'05, Stroudsburg, PA, USA. Association for Computational Linguistics, pp. 99–106.

Oflazer K., Say B., Hakkani-Tür D.Z. and Tür G. (2003). *Building a Turkish Treebank*. Dordrecht, Netherlands: Springer, pp. 261–277.

Özateş B., Özgür A., Gungor T. and Öztürk B. (2018). A morphology-based representation model for LSTM-based dependency parsing of agglutinative languages. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. Association for Computational Linguistics, pp. 238–247.

Peters M., Neumann M., Iyyer M., Gardner M., Clark C., Lee K. and Zettlemoyer L. (2018). Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, New Orleans, Louisiana. Association for Computational Linguistics, pp. 2227–2237.

Qi P., Dozat T., Zhang Y. and Manning C.D. (2018). Universal dependency parsing from scratch. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. Association for Computational Linguistics, pp. 160–170.

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

https://doi.org/10.1017/S1351324921000371 Published online by Cambridge University Press

Sanh V., Wolf T. and Ruder S. (2019). A hierarchical multi-task approach for learning embeddings from semantic tasks. In *Proceedings of the AAAI Conference on Artificial Intelligence* vol. 33, Hawaii, USA, pp. 6949–6956.

Schütze H. (1993). Part-of-speech induction from scratch. In *Proceedings of the 31st Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, pp. 251–258.

Straka M. (2018). UDPipe 2.0 prototype at CoNLL 2018 UD shared task. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, Brussels, Belgium. Association for Computational Linguistics, pp. 197–207.

Sulubacak U. and Eryiğit G. (2018). Implementing universal dependency, morphology and multiword expression annotation standards for Turkish language processing. Turkish Journal of Electrical Engineering & Computer Sciences.

Sulubacak U., Gokirmak M., Tyers F., Çöltekin Ç., Nivre J. and Eryiğit G. (2016). Universal dependencies for Turkish. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, Osaka, Japan. The COLING 2016 Organizing Committee, pp. 3444–3454.

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*. Association for Computational Linguistics, pp. 144–153.

Van Gael J., Vlachos A. and Ghahramani Z. (2009). The infinite HMM for unsupervised POS tagging. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2*. Association for Computational Linguistics, pp. 678–687.

Van Nguyen K. and Nguyen N.L.-T. (2015). Error analysis for Vietnamese dependency parsing. In *2015 Seventh International Conference on Knowledge and Systems Engineering (KSE)*. IEEE, pp. 79–84.

Vania C. and Lopez A. (2017). From characters to words to in between: do we capture morphology? In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vancouver, Canada. Association for Computational Linguistics, pp. 2016–2027.

Yang L., Zhang M., Liu Y., Sun M., Yu N. and Fu G. (2018). Joint PoS tagging and dependence parsing with transition-based neural networks. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* **26**(8), 1352–1358.

Zeman D. and Hajič J. (eds) (2018). *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, Brussels, Belgium. Association for Computational Linguistics.

Zhang Y., Li C., Barzilay R. and Darwish K. (2015). Randomized greedy inference for joint segmentation, POS tagging and dependency parsing. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pp. 42–52.

A. Appendix

Table A1. LAS by dependency relation

DEPREL	Total	MT POS-DEP	Mt SEG-POS -DEP	MT MorphTag -POS-DEP	MT SEG-MorphTag -POS-DEP
acl	249	**65.46**	63.45	**65.46**	63.45
advmod	465	68.39	65.81	66.67	**68.60**
amod	567	62.96	**66.14**	65.96	65.96
appos	4	0.00	**25.00**	0.00	0.00
aux	48	**85.42**	81.25	**85.42**	83.33
case	348	75.57	**82.18**	78.16	80.17
cc	154	57.14	51.95	**57.79**	**57.79**
ccomp	4	**75.00**	75.00	50.00	**75.00**
compound	527	41.18	40.80	42.50	**46.11**
conj	643	43.86	40.90	**47.12**	44.17

https://doi.org/10.1017/S1351324921000371 Published online by Cambridge University Press

Table A1. Continued

DEPREL	Total	MT POS-DEP	Mt SEG-POS -DEP	MT MorphTag -POS-DEP	MT SEG-MorphTag -POS-DEP
cop	132	88.64	**89.39**	87.88	87.88
det	339	76.40	77.29	77.88	**78.76**
discourse	26	**50.00**	34.62	42.31	46.15
fixed	6	0.00	50.00	**66.67**	33.33
flat	167	55.09	**60.48**	50.30	55.69
mark	17	35.29	47.06	29.41	**52.94**
nmod	1276	55.02	58.78	59.17	**59.40**
nsubj	655	52.21	**53.74**	50.38	49.01
nummod	106	42.45	**48.11**	43.40	41.51
obj	774	60.85	56.33	**63.57**	62.02
obl	756	59.39	**64.55**	60.05	60.05
parataxis	1	0.00	0.00	0.00	0.00
punct	1765	72.12	**73.37**	71.56	72.24
root	975	79.28	78.56	**79.59**	79.49

Table A2. UAS by dependency relation

DEPREL	Total	MT POS-DEP	MT SEG-POS -DEP	MT MorphTag -POS-DEP	MT SEG-MorphTag -POS-DEP
acl	249	72.69	72.29	71.49	**74.30**
advmod	465	**77.20**	74.41	75.70	75.91
amod	567	**79.72**	78.66	79.54	78.66
appos	4	**50.00**	**50.00**	**50.00**	25.00
aux	48	87.50	85.42	**89.58**	**89.58**
case	348	77.87	**83.91**	80.17	81.90
cc	154	61.04	55.84	61.69	**62.34**
ccomp	4	**100.00**	**100.00**	**100.00**	75.00
compound	527	48.77	47.63	50.47	**52.18**
conj	643	44.95	43.86	**49.46**	46.03

https://doi.org/10.1017/S1351324921000371 Published online by Cambridge University Press

Table A2. Continued

DEPREL	Total	MT POS-DEP	MT SEG-POS -DEP	MT MorphTag -POS-DEP	MT SEG-MorphTag -POS-DEP
cop	132	**90.91**	90.15	88.64	**90.91**
det	339	83.48	85.25	**85.84**	85.25
discourse	26	**65.38**	50.00	**65.38**	61.54
fixed	6	66.67	66.67	83.33	**100.00**
flat	167	65.27	**67.66**	61.08	60.48
mark	17	76.47	82.35	82.35	**88.24**
nmod	1276	66.22	67.55	66.77	**67.79**
nsubj	655	69.47	**71.91**	69.62	69.77
nummod	106	64.15	**66.98**	66.04	66.04
obj	774	77.52	77.00	**78.81**	76.74
obl	756	70.50	**73.68**	71.16	70.90
parataxis	1	**100.00**	**100.00**	**100.00**	0.00
punct	1765	72.12	**73.43**	71.67	72.29
root	975	79.28	78.56	**79.59**	79.49

Cite this article: Can B, Aleçakır H, Manandhar S and Bozşahin C (2022). Joint learning of morphology and syntax with cross-level contextual information flow. *Natural Language Engineering* **28**, 763–795. https://doi.org/10.1017/S1351324921000371