Turkish Journal of Electrical Engineering and Computer Sciences

Volume 30 | Number 5 Article 15

1-1-2022

# Transmorph: a transformer based morphological disambiguator for Turkish

HİLAL ÖZER

EMİN ERKAN KORKMAZ

Turkish Journal of Electrical Engineering & Computer Sciences
http://journals.tubitak.gov.tr/elektrik/
Research Article
Turk J Elec Eng & Comp Sci
(2022) 30: 1897 – 1913

doi:10.3906/elk-1300-0632.3912

# Transmorph: a transformer based morphological disambiguator for Turkish

**Hilal ÖZER***, **Emin Erkan KORKMAZ**
Department of Computer Engineering, Faculty of Engineering, Yeditepe University, İstanbul, Turkey

**Received:** 24.01.2022 • **Accepted/Published Online:** 28.04.2022 • **Final Version:** 22.07.2022

**Abstract:** The agglutinative nature of the Turkish language has a complex morphological structure, and there are generally more than one parse for a given word. Before further processing, morphological disambiguation is required to determine the correct morphological analysis of a word. Morphological disambiguation is one of the first and crucial steps in natural language processing since its success determines later analyses. In our proposed morphological disambiguation method, we used a transformer-based sequence-to-sequence neural network architecture. Transformers are commonly used in various NLP tasks, and they produce state-of-the-art results in machine translation. However, to the best of our knowledge, transformer-based encoder-decoders have not been studied in morphological disambiguation. In this study, in addition to character level tokenization, three input subword representations are evaluated, which are unigram, bytepair, and wordpiece tokenization methods. We have achieved the best accuracy with character input representation which is 96.25%. Although the proposed model is developed for Turkish language, it is not language-dependent, so it can be applied to a larger set of languages.

**Key words:** Natural language analysis, agglutinative languages, machine learning methods, morphological disambiguation, morphological analysis, transformer network

## 1. Introduction

Morphologically rich languages have posed significant challenges in natural language processing (NLP) research, the most significant of which is data sparseness. The morphological analysis enhances NLP systems by decreasing the lexicon size and the impacts of data sparsity. Moreover, many applications such as syntactic parsing, text-to-speech synthesis, word sense disambiguation, and spelling correction rely on correct word analyses.

The morphological disambiguation is an essential first step for higher level analysis of agglutinative languages such as Turkish, Hungarian, Finnish and Czech. On the other side, languages with less morphology than Turkish, like German and French, can also benefit from it. If the morphology of the words is considered, more accuracy in NLP tasks such as part-of-speech (POS) tagging, dependency parsing [1] and named entity recognition [2, 3] can be achieved. However, due to the enormous number of tags, morphosyntactic tagging is more challenging in agglutinative languages.

Turkish is an agglutinative language, meaning that morphemes are appended to the end of root words. A word's morphological analysis can be defined as a series of tags that corresponds to the morphemes. Because of the complex morphology and ambiguity of Turkish, there are usually several potential analyses for a particular word. Each of them is regarded as a unique interpretation of a specific word with a distinct morpheme sequence. For example, the Turkish word "dolar" has five different analyses as presented in Table 1. In morphological

*Correspondence: hilal.ozer@std.yeditepe.edu.tr

analysis column of the table, the morphological features are listed. The first element at each row is the root of the word. The root is followed by the root part of speech which are "Noun" and "Verb" in these analysis. "A3sg" and "A3Pl" indicate number/person agreement and "Pnon" indicates nonpossessive agreement. "Pos" is a polarity feature and Nom is a case feature. "Aor" is a marker that is used with verbs and indicates its tense. DB is the derivational boundary which means a new word is derived with the following part of speech feature "Adj" at the last example. "AorPart" is used with adjectives and Aorist is a participle feature.

**Table 1**. Five parses of the Turkish word "dolar" are presented.

| **Morphologic analysis** | **Meaning** |
|---|---|
| dolar + Noun + A3sg + Pnon + Nom | dollar |
| do + Noun + A3pl + Pnon + Nom | C, third note of the musical alphabet |
| dol + Verb + Pos + Aor + A3sg | it fills |
| dola + Verb + Pos + Aor + A3sg | she wraps |
| dola + Verb + Pos ^DB + Adj + AorPart | wrapper |

Computational linguists usually handle the problem by morphological disambiguation (MD). MD can be defined as the process of determining the contextually accurate morphological parse of a word in a text in which all potential parses are given. MD systems use the context to identify the correct parse from a set of potential parses for a given word provided by an analyzer. MD generates semantic and syntactic information about a word, such as polarity, tense, its POS tag, and whether it is possessive, accusative, or genitive. While this information is crucial for certain NLP tasks, such as semantic role labeling and dependency parsing, other tasks such as named entity recognition [4], topic modeling, and machine translation [5] can also take advantage of this information.

Turkish words can accept a number of derivational and inflectional suffixes when used in a sentence. It is very common to generate words that nearly equate to a phrase in English like *"git+ ebil + ecek + se + k → if we will be able to go"*. Moreover, many derivations can be found in a single word. In running text, however, a word's average number of morphemes is approximately three. Furthermore, each word has around two unique morphological analyses on average.

Turkish's agglutinative morphology allows for an extremely wide vocabulary, because of highly generative derivational processes. The morphological ambiguity results because of the following reasons without considering the context.

- The root of the word may be ambiguous in terms of its part of speech.
- Homograph morphemes that have numerous interpretations.
- Because of the variety of root forms and morphographemic processes that result in similar surface structures, it is possible to segment word forms in a number of different ways.

In [6], a large corpus is created composed of Turkish text of 500 million words derived from news articles and some statistics presented based on this dataset. Almost 4.1M distinct words are identified, with the most common 50K/300K word forms comprising 89%/97% of them, respectively. The number of distinct morpheme combinations in 360M words is reported as 46K, which surpasses the number of distinct stems, and this results in nearly infinite vocabulary size and various challenges in most of the tasks. As a result, the morphological

disambiguation in Turkish language is a difficult task because of the high number of potential word forms and various morphological parses required to catch them all.

In NLP tasks, a large vocabulary causes many problems. One of them is sparseness. In language modeling, using sublexical units rather than words is one technique to reduce the vocabulary size and overcome data sparsity. Morphological analyzers provide sublexical units that enhance the statistical language model coverage and performance.

Traditionally, morphologically complex languages have been analyzed in two stages: (i) A finite-state transducer-based morphological analyzer creates all possible morphological analyses for a word; (ii) A statistical disambiguation model determines the proper analysis for each word depending on its context. Developing state machines for morphological analysis that are designed to capture the grammatical rules of a language is not an easy task that requires significant effort and linguistic knowledge. The proposed MD model in this study does not require a morphological analyzer and accomplishes morphological analysis and disambiguation in one step.

The recent morphological disambiguation studies are mostly based on the recurrent neural network (RNN)[7] architectures. Although they have shown good performances on the morphological disambiguation task [3, 8–12], RNN architecture is composed of memory units that allow storing data to model short-term dependencies. Usually, RNNs are not sufficient to capture long term dependencies. The character positions of the input and output sequences are used to factor computation in recurrence models. The longer the distance, the more difficult it is to learn the symbol dependencies. Moreover, parallel training is not possible with RNN architectures since the output of the previous state is required to compute the next state. This is again a problem for longer sequences, especially when there are memory restrictions. Large memory requirements with longer sequences end up with limiting the batch size, which can affect the performance of the model.

The transformer model is introduced in [13] which achieved state-of-the-art results in many sequence-to-sequence tasks. The attention mechanism, which is the fundamental building block of the transformer, handles long range dependencies with ease. Attention mechanism highly improves the sequence-to-sequence models allowing the model to focus on the relevant part of the input sequence and this prevents performance drops caused by the long dependencies. The transformer is solely an attention-based network exempt from recurrence. This enables the model to capture the context for any position of the input sequence. The sentence is processed as a whole and the context that gives meaning to each word in the phrase can be detected. Moreover, the transformer model is more parallelizable compared to RNNs. Context is an important factor in selecting the correct morphological parse of the word. Especially the sentential context can be helpful in disambiguation of the word. A better context representation can improve the disambiguation performance. The capability of transformers on morphological disambiguation has not been studied yet. We used transformers with a character based input representation which also helps the model to handle rare or unseen words better.

We aim to make the following contributions to the literature in this study:

- We proposed TransMorph, a transformer-based neural network for the morphological analysis and disambiguation tasks, and showed that TransMorph is comparable to the current state-of-the-art models by verifying its effectiveness with comprehensive experiments.
- We automatically created an unambiguous dataset which is composed of sentences with only unambiguous words. When it is used together with an ambiguous dataset, the model accuracy is enhanced.
- We investigated the performance of the character representation and various other subword representations of sentences as the input to transformer-based encoder-decoder architecture.

In the following section, the previous studies on morphological analysis and disambiguation are summarized. Section 3 presents a transformer-based morphological disambiguator. Lastly, in Section 4, the experiments and their results are discussed.

## 2. Related work

The morphological structure of natural language sentences is decomposed by using morphological analyzers [14, 15]. The finite-state automata (FSA) and transducers (FST) are used to encode morphological grammars and analyzers. As most computational morphological approaches, these methods are rule-based and hand crafted. A list of handwritten rules needs to be implemented in a certain order for each word of the sentence. The process is costly and requires a savant for each particular language in order to define the handcrafted rules. Statistical approaches are also utilized for morphological analysis [16], and they are of importance to avoid domain-specific rules. These methods use a training corpus, and an algorithm is run to compute the probability of cooccurrence of all ordered tag pairs. Both of the approaches share the same objective; to achieve morphological decomposition.

### 2.1. Morphological analyzers

Although it is not the first one, Koskenniemi's two-level morphology model [14] was the most well-known and successful rule-based model ever. It used standard machinery that consists of a finite-state transducer in which language-specific morphotactics are embedded. A large set of morphological rules, a lexicon and morphemes are required to implement rule-based models. PC-KIMMO is introduced as a two-level morphological analyzer in which users can build their own rule sets, and lexicons [17].

The first two-level analyzer in Turkish is developed by Oflazer on PC-KIMMO based on a root lexicon consisting of 23,000 root words [15]. The phonetic rules of Turkish are encoded with 22 two-level rules and morphotactics as a finite state machine in this study. In [18] an affix driven approach is used instead of root driven approach. The affixes of the word are determined at first, and then the stem is determined without using any lexicon. In [19] a freely available Turkish morphological analyzer was developed: TrMorph. Another two-level freely available morphological analyzer is [20], and it consists of morphotactic and morphophonological rules. Lastly, there are two open source Turkish morphological analyzers [21, 22].

Since it was time-consuming to identify the morphotactic and morphophonological rules, statistical methods emerged. The rule-based morphological implementations can be improved with weighted and statistical Finite State Transducers (FSTs). Sak developed a stochastic Finite-State morphological parser[16].

### 2.2. Morphological disambiguator

The number of possible parses for a selected word is usually more than one in Turkish due to the complex morphology and ambiguity. Each parse is regarded as a distinct interpretation of a single word that has a different sequence of morphemes. There are two different approaches to the solution of this problem. While some computational linguists handle the problem as POS tagging [23] others consider this as a morphological disambiguation problem. The morphological disambiguation is much harder than POS tagging because it requires all roots, suffixes and the corresponding tags to be identified in addition to the POS tag.

The rule-based morphological disambiguation algorithms are composed of a list of handwritten rules that needs to be applied in a particular order to each word in the text. The statistical ones (also referred as Stochastic model) employ a training corpus to compute the probability of all ordered tag pairs that are

co-occurring. There are also hybrid models that use both rule-based and probabilistic methods. Recently deep learning methods have emerged which have competitive results. The traditional morphological disambiguators require a morphological analyzer to generate candidate morphological parses, and then the disambiguator makes a selection among these parses. The recent methods, especially the deep learning models, do not need these prerequisite steps and can accomplish morphological analysis and disambiguation in one stage.

#### 2.2.1. Rule based models

The first morphological disambiguation model for Turkish is a rule-based disambiguator developed by Oflazer et al. [24] and enables one to write rules in the form $C_1 : A_1; C_2 : A_2; ...C_n : A_n$. where each $C_i$ is a constraint in lexical form that should be satisfied for the corresponding action $A_i$ to be executed. These rules are used after all words in the sentence are morphologically analyzed. The constraints can be morphological features (such as case, part of speech of the word or agreement) or positional features of the word. The actions are deleting the matching parse, assigning the matching parse, composing a new parse or null action. After rule based approach they combined hand-crafted rules with statistical and learned rules that are extracted in an unsupervised manner from a training corpus [25]. Also, in [26], a pure rule-based disambiguation model is developed. Although these early models claimed to have high accuracy (97%–98%), they all utilize very small corpora and they are based on handmade rules. Therefore, it is difficult to claim that the reported performance can be achieved for the general case.

#### 2.2.2. Statistical models

The first Turkish statistical morphological disambiguator is proposed in [27]. In this study, n-gram language models are utilized using inflection groups for morphological disambiguation. Using smaller units for data sparseness is found helpful, and the best performing model obtained is the trigram model with 93.95% accuracy. In [19], three models are compared for disambiguation and the model that ignores the root word and considers only the frequency of the sequence of suffixes has the best result with 94% accuracy.

#### 2.2.3. Machine learning and deep learning models

Machine learning and deep learning approaches are commonly applied for Turkish MD problem, as well. In [28], a rule-based approach utilizing decision lists is used, named as the Greedy prepend algorithm. The features of surface form of the ambiguous words are used in the method together with the preceding and following words, and this approach achieved 91.23% accuracy without the help of a morphological analyzer and 95.82% accuracy when the morphological analysis is carried out by an external morphological analyzer. In [29] the MD problem is approached as a classification problem. Ten different classification algorithms are experimented, and the highest result is 95.61% accuracy with J48 Tree algorithm. Sak et al.[30], used the previous work in [27] which is the baseline statistical trigram model that obtains the n-best list candidates of morphological parses. A perceptron algorithm using 23 features is used to rerank candidate parse lists. The perceptron algorithm increased the baseline model's accuracy from 93.95% to 96.80% in MD.

Shen et al. [8] used bidirectional long short-term memory (LSTM)[31] networks separately to obtain stem, morphological tag and context representations. Using these embeddings, each morphological analysis is scored using CRF and the Viterbi algorithm. In [9], the Turkish morphological analysis and disambiguation problems are solved jointly in one step with a character based encoder-decoder model based on LSTMs. The key contribution is the sequence decoder that takes the word related features and context information as input and

creates root characters and morphological tags as a sequence one at a time with 97.67% accuracy. [10] shows a joint contextual lemmatizer and morphological tagger. They represented word representations using LSTM over character vectors and used bidirectional LSTMs to learn context representations from word embeddings with a lemma decoder which is designed to estimate the fewest edit operations between lemmata and surface words. They measured the F1 score as 93.71% which is calculated over the predicted and actual individual morphological tags. In [11], a deep neural architecture is applied with the Viterbi algorithm to the MD problem. In order to get better representations, the root of the words are trained with the Skip-gram algorithm [32] which improved the results presented by 85% accuracy on the test set. In [12], Turkish morphological segmentation is considered as a character sequence-to-sequence learning problem, and an attention-based neural network model is proposed to solve it. A bidirectional LSTM is used as encoder to encode the input sentence and an attention mechanism is used in the decoder to guide the morphological segmentation model focus on certain contexts of the current character to be tagged. A total of 92.6% accuracy on Turkish morphological tagging is achieved. In [33] a vector-space model is proposed for MD that accomplishes morphological ambiguity by locating the correct candidates of ambiguous words near to the unambiguous neighbors. The model named learning word-vector quantization achieved 98.4% accuracy in their own dataset. [34] a special type of sequence-to-sequence networks, pointer networks[35] are used for MD task and 96.6% F1 score is achieved in Universal Dependencies v2.2 dataset[36].

## 3. Materials and methods

Developing a model that converts an input symbol sequence to a target symbol sequence is the challenge of the sequence-to-sequence problem. Each input symbol is assumed to be encoded into a vector representation and the input sequence is expressed as a series of input vectors:

$$X_{1:n} = \{x_1, \ldots, x_n\}. \tag{1}$$

The solution is to discover a transformation function that converts an input sequence of $n$ vectors $X_{1:n}$ to a series of $m$ target vectors $Y_{1:m}$, where the number of target vectors $m$ is not known at the beginning and relies on the input.

$$f = X_{1:n} \rightarrow Y_{1:m} \tag{2}$$

We defined the MD problem as a sequence-to-sequence problem and defined a mapping of input sentence as sequence of characters $X_{1:n}$ to output morphological analysis of sentence $Y_{1:m}$ of variable length $m$. Given an input sequence $X_{1:n}$, a conditional probability distribution of target vectors $Y_{1:m}$ is established by using the transformer-based encoder-decoder model.

$$p_{\theta_{enc},\theta_{dec}}(\, Y_{1:m} \,|\, X_{1:n} \,) \tag{3}$$

The input sentence $X_{1:n}$ is encoded by the encoders to the sequence of context vectors $\widehat{X}_{1:n}$ creating a mapping:

$$f_{\theta_{enc}} = X_{1:n} \rightarrow \widehat{X}_{1:n}. \tag{4}$$

Then, the decoder component computes the conditional probability distribution of sequence of target vectors $Y_{1:n}$, given the sequence of encoded context vectors $\widehat{X}_{1:n}$.

$$p_{\theta_{dec}}(\, Y_{1:n} \,|\, \widehat{X}_{1:n} \,) \tag{5}$$

The distribution can be factored down to a product of conditional distributions of the target vector $y_i$, given all prior target vectors $Y_{0:i-1}$ and the encoded context vectors $\widehat{X}_{1:n}$, applying Bayes' rule [37].

$$p_{\theta_{dec}}(Y_{1:n} \mid \widehat{X}_{1:n}) = \prod_{i=1}^{n} p_{\theta_{dec}}(y_i \mid Y_{0:i-1}, \widehat{X}_{1:n}) \tag{6}$$

As a result, the transformer-based decoder converts all prior target vectors $Y_{0:i-1}$ and the encoded hidden states $\widehat{X}_{1:n}$ to the logit vector $l_i$. To construct the conditional distribution $p_{\theta_{dec}}(y_i|Y_{0:i-1}, \widehat{X}_{1:n})$, logit vector $l_i$ is subsequently processed using the softmax operation. Different from the decoders that are based on RNN, the distribution of the target vector $y$ is directly dependent on all preceding target vectors.

The previous encoder-decoder MD models that employed RNN [3, 8–12] are indirectly dependent on preceding target vectors. In decoders that are based on RNN, current target vector $y_i$'s distribution depends directly on the prior hidden state $c_{i-1}$ and the preceding target vector $y_{i-1}$. $c_{i-1}$ has dependency on all of the prior target vectors $y_0, y_1, \ldots, y_{i-2}$. As a result, the RNN-based encoder-decoder models the conditional distribution indirectly.

Our proposed model is founded on the sequence-to-sequence encoder-decoder network approach for machine translation [38]. While language models condition on the previous steps to predict the next word, machine translation models condition on both the context information and previous words to predict the next word. Hence they can be named as the conditional language model. The sequence-to-sequence learning is about training models in order to transform sequences from one domain to sequences in another domain. The sequence-to-sequence architectures composed of an encoder and a decoder are well suited for such problems. The Encoder maps the input sequence into a n-dimensional vector and sends it to the Decoder that turns it into an output sequence.

Since we modeled MD as a sequence-to-sequence problem, for instance given an input sentence $X =$ "Geldim." is represented as characters $x_1 = G$, $x_2 = e$, $x_3 = l$, $x_4 = d$, $x_5 = i$, $x_6 = m$, $x_7 = .$, $x_8 = EOS$. The goal is to produce the morphological analysis Y ="g e l Eor Verb Pos Past A1sg Eow . Punct" as a sequence $y_0 = BOS$, $y_1 = g$, $y_2 = e$, $y_3 = l$, $y_4 = Eor$, $y_5 = Verb$, $y_6 = Pos$, $y_7 = Past$, $y_8 = A1sg$, $y_9 = Eow$, $y_{10} = .$, $y_{11} = Punct$. The start and the end of the sentences are represented with "BOS" (Beginning of sentence) and "EOS" (End of sentence) tokens in sequence-to-sequence models. In the output representation, "Eor" (End of root) token is added at the end of root characters in order to separate root from morphological tags. Moreover, "Eow" (End of word) token is added at the end of each word to distinguish the analysis of separate words. This process is described in Figure 1.

### 3.1. Transformer-based model

In order to solve the defined sequence-to-sequence problem, we propose the model named as TransMorph which is based on a transformer-based encoder-decoder architecture [13]. In this section, while we briefly cover the transformer network, we will also describe the modeling options for utilizing the transformer architecture as a morphological disambiguator. At first, the input character sequence is sent into the Character Embedding and Position Embedding modules, which generate an encoded representation for each character capturing the meaning and position. Positional Embeddings are used to preserve the information related to order of the sequence. On the target side, the output sequence of the analysis is also sent to Output Embeddings and Positional Embeddings modules. TransMorph is comprised of a stack of three encoder layers and three decoder

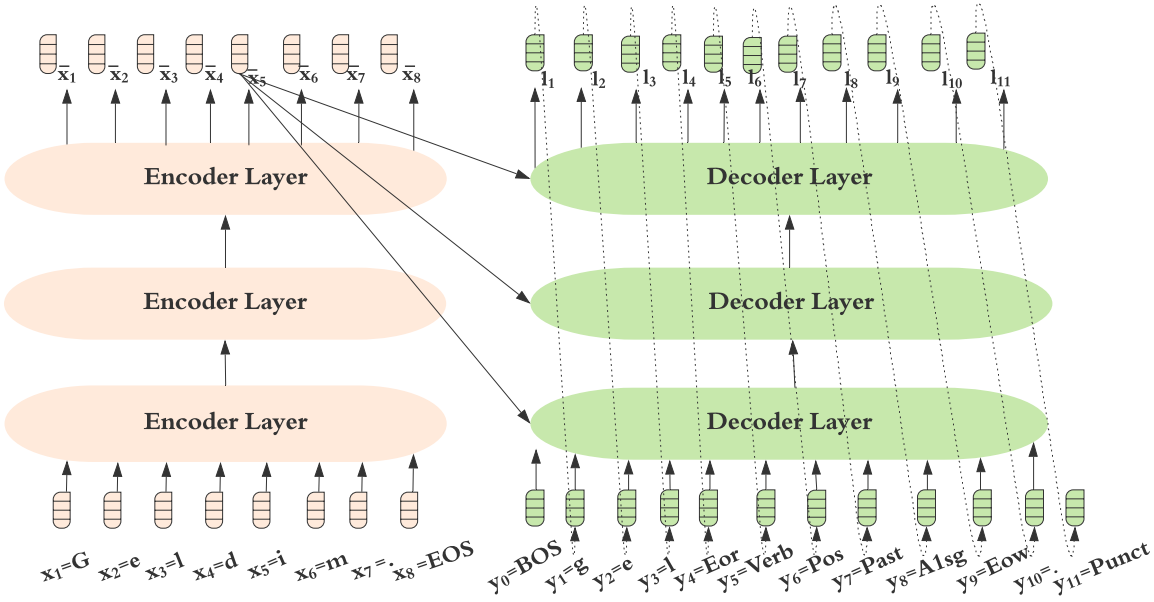


**Figure 1**. The input/output representation in transformer-based MD.

layers. The encoder encodes input embedding into the context vector, and then the decoder decodes this context vector to lemmas one character at a time and also to the morphological analysis of the sentence. Each encoder and decoder layer is organized into a multihead self-attention layer with relative position bias [39] and position-wise fully connected feed-forward layers. Relative position bias is applied to establish relative position relations among all positions, which can improve local relationships without overlooking them. These two sublayers employed residual connection with dropout followed by normalization. The decoder adds a third sublayer, which executes masked multihead attention with relative position bias on the encoder's output. The other important parameter is the number of layers in the feed-forward network. MD problem converged well with 4 feed-forward layers instead of 2048, which is used in the original implementation. The model's hyperparameter setting is given in Table 2, and the architecture of the TransMorph is shown in Figure 2.

**Table 2**. Hyperparameter setting.

| **Transformer parameter setup** | |
|---|---|
| Input embedding size | 512 |
| Number encoder/decoder layer | 3 |
| Number attention heads | 8 |
| Head dimension | 512 |
| Feed forward dimension | 4 |
| Dropout | 0.1 |

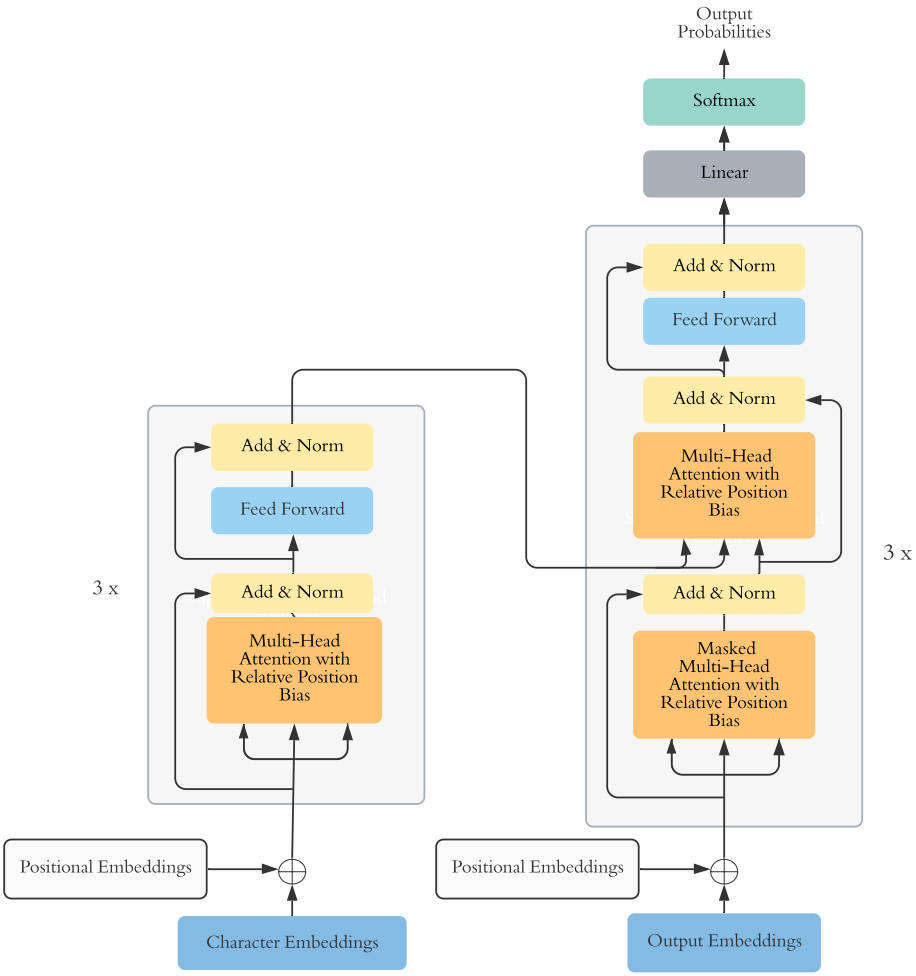


**Figure 2**. Architecture of transformer-based morphological disambiguation model: TransMorph.

### 3.2. Self-attention and multihead attention

Self-attention computes a weighted average of feature representations, where the weights determine which parts of the feature vectors should be paid more attention. The attention layer inputs 3 parameters $Q$, $V$ and $K$ which are referred as the query, the value, and the key. These three parameters are obtained with the input sequence of n tokens of $d$ dimensions $X \in \mathbb{R}^{n \times d}$ and its projection to the matrices respectively $W_Q \in \mathbb{R}^{d \times d_q}$, $W_V \in \mathbb{R}^{d \times d_v}$ and $W_K \in \mathbb{R}^{d \times d_k}$. The computation of Q, V, K are carried out as follows:

$$Q = XW_Q, \quad V = XW_V, \quad K = XW_K. \tag{7}$$

On the other side, the self-attention can be formulated as:

$$S = D(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_q}})V \tag{8}$$

The input embedding and position embedding are supplied to all these three parameters, in the self-attention of the first encoder, which then creates an encoded representation for each character in the input

sequence. The representation includes the attention scores for each character as well. Each self-attention module utilized inserts its own attention scores to the representation of each character, as this representation travels through all of the encoders in the stack.

On the target side, the output embedding and position embedding are again supplied to all three parameters, key, query, and value in the self-attention module of the first decoder. Now an encoded representation for each character is created in the target sequence, including the attention scores of the character as well. This is given to the query parameter of encoder-decoder attention of the first decoder after going through a normalization layer (layer norm). Meanwhile, the output of the stack's last encoder is sent to the value and key parameters. Therefore, the encoder-decoder attention receives a representation of both the target sequence (from the decoder self-attention) and the input sequence (from the encoder stack). As a result, it generates a representation of the attention scores for each target sequence element. The representation also includes the effect of the attention scores obtained from the input sequence. Then the representation obtained is passed through the stack of Decoders. In this process, the attention scores generated by all self-attentions and encoder-decoder attentions are added to the representation of each character.

In multihead attention, the attention module performs its calculations for a number of times in parallel and each output is named as an attention head. The knowledge explored by multiple heads are combined to obtain a final score. This makes the transformer more powerful by encoding many associations and nuances for each input element. The embedding vectors are split logically among several heads, which implies that different parts of the embedding can capture different aspects related to the meaning of each word. For example, while one part learns about the tense (present, past, future) of the word, the other part might be concentrating on the word's person agreement (I am, he is).

### 3.3. Features

To effectively reveal the relationship between words and morphemes, different experiments are carried out by utilizing characters and subword tokens as the basic unit for the input representation. We train the proposed model with character, unigram [40], BPE [41] and wordpiece [42] encodings separately. On the target side, for the output of decoders, the root is represented as characters, and the morphological features POS tag and morphemes are set as tokens as they are.

## 4. Experiments and results

### 4.1. Data sets

In the experiments, three different morphologically tagged data sets are employed. These datasets are composed of both derivational and inflectional morphologically tagged data. The first one is TrMor2018 [9] which is recently used in MD studies. The second one is the previous version of the first data set; trMor2006 [28]. The last dataset is an unambiguous dataset. They are all composed of both derivational and inflectional morphologically tagged data.

#### 4.1.1. TrMor2018 and TrMor2006

Both TrMor2006[28] and TrMor2018[9] are semiautomatically generated morphology data sets. Since TrMor2006 training data set has limited accuracy, a new data set is required. The sentences in TrMor2018 data set are first disambiguated automatically. Then, subset of the original data set is selected randomly and this subset is disambiguated manually. These manually disambiguated sentences are compared with the results of

the automatically disambiguated sentences and the errors in the automatically disambiguated sentences are corrected. This process is repeated several times, until a data set with a noise level of %3 is obtained. The ambiguous and unambiguous word counts are shown in Table 3. It is the most recent data set that has been used in MD studies. It is used both for training and testing the MD model in this research. Since previous studies were done with the previous version of TrMor2018 which is trMor2006 [28], our model is also tested on trMor2006 in order to compare the results with prior research.

**Table 3**. Data set word count statistics.

| Data set | Ambiguous | Unambiguous | Total |
|---|---|---|---|
| TrMor2018 | 215024 | 243866 | 460669 |
| TrMor2006 | 399216 | 436406 | 837524 |
| Unambigous | - | 1336231 | 1336231 |

#### 4.1.2. Unambigous data set

The unambiguous sentences at the Hürriyet News data set and the BOUN Web corpus [6] are combined to obtain an unambiguous data set. These two data sets are used to filter sentences that are composed of only unambiguous words.

We used Hürriyet Api[1] to query news data and collected 50M tokens (6M sentences). The sentences are analyzed with Zemberek Library [43] in order to determine unambiguous sentences which are composed of only unambiguous words. We also filtered the sentences that had unknown and foreign words. Then the sentences are reanalyzed by Oflazer's finite-state transducers for Turkish morphological analysis [15]. After these operations, $11,842$ unambiguous sentences are obtained.

The BOUN Web Corpus was created in 2008 [6]. It has 500M tokens. Again, we have morphologically analyzed its sentences by Kemal Oflazer's finite-state transducers [15]. $298,703$ unambiguous sentences are detected in this corpus. These sentences are combined with the unambiguous sentences obtained from Hürriyet News data and finally an unambiguous data set is obtained that consists of $310,109$ sentences after dropping duplicate sentences.

### 4.2. Evaluation metrics

The most important indicator for the performance of the MD model is accuracy. This metric is also generally used in the previous studies. The model's prediction is compared with the morphological analysis tag of the word in the utilized data set in order to determine if the analysis is correct. The accuracy is defined as the percentage of the correctly analyzed words among all analyzed words.

$$Accuracy = \frac{\#\text{ correctly analyzed words}}{\#\text{ all analyzed words}} \tag{9}$$

### 4.3. Training

Parameters of the proposed model are optimized with Momentumized, Adaptive, Dual Averaged Gradient Method (MADGRAD) [44] which belongs to the AdaGrad adaptive gradient methods. The initial learning rate

[1]GitHub (2022). Hürriyet API [online]. http://github.com/hurriyet/developers.hurriyet.com.tr[accessed 19 January 2022].

is set as 0.0003 and then it is reduced by a factor of 0.1 when there is no improvement in the loss value for 10 epochs. We used a batch size of 8. Moreover, while preparing the training and test sets, the sentences selected have a maximum word count of 50 and a maximum morphological analysis length of 300 (counting root as one). We set that limit to prevent a negligible amount of examples to affect encoder and decoder input/output length excessively. The selected limits are far above the average sentence length. Depending on input and target data, the maximum Encoder length is set to 385 and the maximum decoder length to 525. The other model parameters are given in Table 2. Lastly, the model is trained for 350 epochs on the data sets.

The code of the model and the unambiguous data set created in this study is released upon publication.[2] The experiments were performed on the i7-10750H PC, with memory of 32 GB, RTX2070 GPU (8 GB memory), Python 3.7. and Google Colab Pro personal accounts.

### 4.4. Results and discussion

Three experiments are conducted to examine the efficiency of the proposed model. In the first one, the model is evaluated on data sets and compared to other competitive models. In the second experiment, the model is tested with three subword input representations. Lastly, in the third experiment the all-in-one morphological disambiguator and analyzer TransMorph is compared with a model that has a separate analyzer and disambiguator.

#### 4.4.1. First experiment

In the first experiment, we used character input representation and tokenized input sentence with character tokenizer in SentencePiece library [40]. Then 10-fold cross-validation tests are applied on both TrMor2006 and TrMor2018 data sets and the MD task accuracy results are presented in Table 4.

**Table 4**. Ten-fold cross-validation results on data sets.

| Data set | Accuracy(%) |
|---|---|
| TrMor2006 | 94.74 |
| TrMor2018 | 96.19 |

Additionally, in order to explore the effect of an unambiguous data set, the training data of trMor2018 is enlarged with the unambiguous data defined in 4.1.2. Then, the tests are rerun with the same test data, and 96.25% accuracy is obtained with this new training dataset.

The lemmatization and morphological tagging accuracy is also evaluated. The lemmatization accuracy is the ratio of correctly identified lemmata to the total number of lemmata. The morphological tagging accuracy is the ratio of the number of words whose all morphological tags are correctly identified over the total number of words. The results of the experiment shown in Table 5.

**Table 5**. Ten-fold cross-validation results TrMor2018.

| Data set | Lemmatization acc.(%) | Morph. tagging acc.(%) | POS tagging acc.(%) |
|---|---|---|---|
| TrMor2018 | 98.42 | 97.44 | 97.57 |

[2]GitHub (2022). TransMorph Model [online]. http://github.com/hozerk/TransMorph

One important tag identified in MD is the POS tag. It is identified by considering the last POS tag in the analysis. TransMorph identified POS tags with 97.57% accuracy on the TrMor2018 data set.

In order to verify the suggested method, a comparison is also carried out with some existing methods in the literature:

1. Yüret and Türe [28] used a rule-based approach based on decision lists.
2. Sak et al. [30] used a perceptron algorithm to rank morphological analysis.
3. Shen et al. [8] used Bidirectional LSTMs and CRF-based model.
4. Akyürek et al.[9] developed a sequence-to-sequence LSTM based method.

Among these methods, the first three methods depend on a morphological analyzer and use the candidate parses that are received from an analyzer in order to select the best parse among them. Only Akyürek's method and our proposed model accomplish morphological analysis and disambiguation in one step.

The TransMorph model is also evaluated on a small test set of TrMor2006 that has been manually disambiguated and composed of 958 tokens. The aim was to compare the performance of TransMorph to earlier models presented in Table 6. We used the same training set, namely TrMor2006 as previous researchers used, and tested on the small test set utilized in these studies. Although it seems that the proposed model has a lower performance, it should be noted that our model makes morphological analysis and disambiguation in parallel. The first three methods in the table utilize the analysis of words obtained from a morphological analyzer and make disambiguation for only ambiguous words. The correct analysis of unambiguous words is directly provided by the analyzer in these studies. Moreover, in the experiments, we did not exclude any of the tags. [9] states that "Prop" (proper noun) and digit tags are excluded in the experiments. When we excluded these tags, the accuracy increased to 96.17% which is higher than [9] on this test set. As a result, it can be claimed that TransMorph shows a satisfactory performance compared to similar methods in the literature.

We also compared the proposed model's accuracy with that of Akyürek et al. in [9] on the data set TrMor2018 and presented the results in Table 7. In their article it is stated that they masked "Prop" and digit tags. We had not made any exclusion of tags in the evaluation of results in our initial evaluations. In order to make a comparison under the same circumstances, we masked these tags and the accuracy metric for the tests of TrMor2018 increased to 97%. As a result, the proposed method has comparable results with [9].

**Table 6**. Comparative results on manually tagged test set of TrMor2006.

| **Method** | **TrMor2006** |
|---|---|
| Yuret and Ture, 2006 [28] | 95.82 |
| Sak et al., 2007 [30] | 96.28 |
| Shen et al., 2016 [8] | 96.41 |
| Akyürek et al., 2019 [9] | 95.94 |
| **TransMorph** | **95.08** |

**Table 7**. Comparative results on TrMor2018.

| **Method** | **TrMor2018** |
|---|---|
| Akyürek et al., 2019 [9] | 97.67 |
| **TransMorph** | **97** |

When we have analyzed the errors in the experiments, we saw that 41% of the errors occurred in predicting the lemma of the word. Moreover, about 58% of the errors consisted of cases where the lemma was predicted correctly but the morphological tagging was done incorrectly. In 30% of the morphological tagging errors, the root POS tag was selected erroneously. In 29% of them the POS tag of the root was erroneously

predicted as adjective instead of noun. In 22.9% of them, noun should have been adjective. In 50.6% of morphological tagging errors, the POS tag of the word was erroneously selected. In 27.88% of them the POS tag was erroneously predicted as adjective instead of noun. In 17.51% of them, predicted noun should have been adjective; in 10% of them, verb should have been noun. Lastly, 36.5% of all the morphological tagging errors are cases where only the "Prop" tag was incorrectly decided and if the "Prop" tag was decided correctly, then the morphological tagging would have been correct.

We noticed that using relative position bias in the multihead attention distinctly helped the model to converge faster compared to using the absolute positional embeddings. It is because the relative representations depend on how far the elements are from one another. The morphemes have mostly local relationships among each other, and the relative position bias is better at capturing local relations. Moreover, in parameter tuning, we see that the MD model converged with the lower layer dimensions of the feed-forward layer. This observation corroborates findings of [45] that lower layers tend to capture the shallow patterns like n-grams and suffixes, while upper layers learn the semantic patterns better.

We have carried out an extra experiment with RNN, using a 2 layer LSTM encoder-decoder architecture with similar input and output representations in order to make a comparison with our model. However, the LSTM model did not exhibit a convergence on the morphological disambiguation task since we used a single character-level encoder.

#### 4.4.2. Second experiment

A second experiment is conducted to see if the subword input representations are as effective as the character input representation. We obtained the character, unigram, and BPE tokenizers by training the Sentencepiece Library [40] with the BOUN Web corpus [6] where the vocabulary size is 16000. For the wordpiece encodings, we used the pretrained tokenizer[3]. Ten-fold cross-validation tests were conducted on TrMor2018 dataset with the unigram, BPE, and wordpiece subword input representations. The experiment shows that the character-level representation is better than the subword representations in learning morphological data in the transformer-based encoder-decoder architecture as in Table 8.

**Table 8**. Evaluation of accuracies of MD models with subword input representations.

| **Method** | **TrMor2018** |
|---|---|
| Character | 96.19 |
| Unigram | 95.11 |
| BPE | 94.74 |
| Wordpiece | 93.48 |

#### 4.4.3. Third experiment

In order to see if it is computationally advantageous to solve morphological analysis and disambiguation in parallel, a comparison is made with a model that has a separate analyzer and disambiguator. For this new model, initially a transformer-based morphological analyzer and then a transformer-based disambiguator are trained separately on TrMor2018 dataset. The transformer-based analyzer is trained with the same settings as

[3]Zenodo (2020). BERTurk model [online]. Website http://zenodo.org/record/3770924#.YeiAz71BxPZ [accessed 19 January 2022].

the transformer encoder-decoder architecture of TransMorph. All possible analyses of the words are given as the target output for the new analyzer during the training phase. After training with 10-fold cross-validation, the created analyses are used to train the disambiguator.

The disambiguator is constructed with two encoders, one encoder is used to encode candidate morphological analyses and the other is used to encode the input sentence. The encoders are combined serially to the decoder as in [46]. The output of the encoders are fed to the decoder with cross-attention.

When we compare the separated model with TransMorph, TransMorph is more advantageous in terms of training time. The training time of the analyzer is almost similar to Transmorph for one epoch. The training time of the disambiguator is 50% longer than TransMorph. Separating the analyzer and disambiguator into two models results about 2.5 times longer training time. Moreover, the separated model's accuracy is 95.9% on the MD task on TrMor2018 dataset which is below TransMorph's accuracy 96.19%.

## 5. Conclusion

In this study, we presented a method using the transformer-based encoder-decoder to solve the problem of MD in Turkish. To the best of our knowledge, this is the first study that uses a transformer-based encoder-decoder model for the Turkish MD problem. Transformers can represent dependencies in long sequences better than RNN's, and they can be parallelized as well. We have shown that using only character features as input, it is possible to achieve 96.25% accuracy, which is a comparable accuracy with the previous competitive models. Since our model is not language-dependent, it can be applied to other languages as well. In the experiments, it was observed that the character-level representation is better than the subword representations for the morphological analysis and disambiguation task when the transformer-based encoder-decoder is utilized. Moreover, we experimented that an all-in-one transformer-based morphological disambiguator performs better in terms of training time and accuracy compared to a separated morphological analyzer and disambiguator that is also based on the transformer model. In this study, also a public unambiguous data set is created that is composed of sentences with only unambiguous words.

## References

[1] Sennrich R, Schneider G, Volk M. Warin M. A New Hybrid Dependency Parser for German. In: Proceedings of Biennial GSCL Conference; Potsdam, Germany; 2009. pp. 115-124.

[2] Kalender M, Korkmaz EE. Turkish Entity Discovery with Word Embeddings. Turkish Journal of Electrical Engineering & Computer Sciences 2017; 25: 2388-2398. doi:10.3906/elk-1512-102

[3] Güngör O, Güngör T, Üsküdarli S. The Effect of Morphology in Named Entity Recognition with Sequence Tagging. Natural Language Engineering 2018; 25 (1): 147–169. doi:10.1017/S1351324918000281

[4] Kalender M, Korkmaz EE. Thinker - entity linking system for Turkish language. IEEE Transactions on Knowledge and Data Engineering 2018; 30 (2): 367–380 doi:10.1109/TKDE.2017.2761743

[5] Pan Y, Li X, Yang Y, Dong R. Multi-Source Neural Model for Machine Translation of Agglutinative Language. Future Internet 2020; 12 (6): 96. doi:10.3390/fi12060096

[6] Sak H, RT, Saraçlar M. Turkish Language Resources: Morphological Parser, Morphological Disambiguator and Web Corpus. In: Advances in Natural Language Processing: GoTAL; Gothenburg, Sweden; 2008. pp. 417–427.

[7] Rumelhart DE, Hinton GE, Williams RJ. Learning internal representations by error propagation. Parallel Distributed Processing 1987; 1: 318-362, MIT Press.

[8] Shen Q, Clothiaux D, Tagtow E, Littell P, Dyer C. The role of context in neural morphological disambiguation. In Proceedings of the Conference on Computational Linguistics (COLING-2016); Osaka, Japan, 2016. pp. 181–191.

[9] Akyürek E, Dayanık E, Yuret D. Morphological analysis using a sequence decoder. Transactions Of The Association For Computational Linguistics 2019; 7: 567–579. doi:10.1162/tacl_a_00286

[10] Yıldız E, Tantuğ C. Morpheus: A Neural Network for Jointly Learning Contextual Lemmatization and Morphological Tagging. In: Proceedings of the 16th Workshop on Computational Research in Phonetics, Phonology, and Morphology; Florence, Italy; 2019. pp. 25-34.

[11] Yildiz E, Tirkaz C, Sahin HB, Eren MT, Sonmez O.A morphology-aware network for morphological disambiguation. In: Proc. of the Thirtieth AAAI Conference on Artificial Intelligence; Phoenix, ARI, USA; 2016. pp. 2863–2869.

[12] Zhu S. A Neural Attention Based Model for Morphological Segmentation. Wireless Personal Communications 2018; 102 (4): 2527–2534. doi: 10.1007/s11277-018-5274-8

[13] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L et al. Attention is all you need. In: Advances in Neural Information Processing Systems; Long Beach, CA, USA; 2017. pp. 6000–6010.

[14] Koskenniemi K. Two-Level Morphology: A General Computational Model For Word-Form Recognition and Production. PhD, University of Helsinki, Helsinki, 1983.

[15] Oflazer K. Two-level description of Turkish morphology. Literary and Linguistic Computing 1994; 9 (2): 137–148. doi:10.1093/llc/9.2.137

[16] Sak H, r T, Saraçlar M. A Stochastic Finite-State Morphological Parser for Turkish. In: Proceedings of the ACL-IJCNLP 2009 Conference short papers; Suntec, Singapore; 2009. pp. 273–276.

[17] Antworth EL. PC-KIMMO: A Two-Level Processor for Morphological Analysis; Dallas, TX, USA: Summer Institute of Linguistics, 1990.

[18] Eryiğit G, Adali. E. An Affix Stripping Morphological Analyzer For Turkish. In: Proceedings of the IASTED International Conference on Artificial Intelligence and Applications; Innsbruck, Austria; 2004. pp. 299-304.

[19] Çöltekin Ç. A set of open source tools for Turkish natural language processing. In: Proceedings of the 9th International Conference on Language Resources and Evaluation; Reykjavik, Iceland; 2014. pp. 1079–86.

[20] Kayabaş A, Schmid H, Topcu AE, Kılıç Ö. TRMOR: a finite-state-based morphological analyzer for Turkish. Turkish Journal of Electrical Engineering & Computer Sciences 2019; 27: 3837–3851. doi:10.3906/elk-1902-125

[21] Öztürel A, Kayadelen T, Demirşahin I. A syntactically expressive morphological analyzer for Turkish. In Proceedings of the 14th International Conference on Finite-State Methods and Natural Language Processing; Dresden, Germany; 2019. pp. 65–75.

[22] Yıldız OT, Avar B, Ercan G. An Open, Extendible, and Fast Turkish Morphological Analyzer. In Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019).; Varna, Bulgaria; 2019. pp. 1364–1372.

[23] Eşref Y, Can B. Using Morpheme-Level Attention Mechanism for Turkish Sequence Labelling. In: 2019 27th Signal Processing and Communications Applications Conference; Sivas, Turkey; 2019. pp. 1–4.

[24] Oflazer K, Kuruöz I. Tagging and Morphological Disambiguation of Turkish Text. In: Proceedings of the fourth conference on Applied natural language processing; Stuttgart, Germany; 1994. pp. 144-149. doi:10.3115/974358.974391

[25] Oflazer K, Tür G. Combining Hand-Crafted Rules and Unsupervised Learning in Constraint based Morphological Disambiguation. In: Proceedings of Conference on Empirical Methods in Natural Language Processing; Philadelphia, PA, USA; 1996.

[26] Daybelge T, Çiçekli I. A Rule-Based Morphological Disambiguator for Turkish. In: Proceedings of Recent Advances in Natural Language Processing; Borovets, Bulgaria; 2007. pp:145-149.

[27] Hakkani-Tur D, Oflazer K, Tür G. Statistical morphological disambiguation for agglutinative languages. Journal of Computers and Humanities 2002; 36 (4): 381–410. doi:0.1023/A:1020271707826

[28] Yüret D, Türe F. Learning Morphological Disambiguation Rules for Turkish. In: Human Language Technology Conference/North American chapter of the Association for Computational Linguistics Annual Meeting; New York City, NY, USA; 2006. pp. 328–334.

[29] Görgün O, Yıldız OT. A Novel Approach to Morphological Disambiguation for Turkish. In: Computer and Information Sciences II; London, UK; 2011. pp. 77-83.

[30] Sak H, r T, Saraçlar M. Morphological Disambiguation of Turkish Text with Perceptron Algorithm. In: International Conference on Intelligent Text Processing and Computational Linguistics(CICLing); Mexico City, Mexico; 2007. pp. 107–118.

[31] Hochreiter S, Schmidhuber J. Long short-term memory. Neural Computation 1997; 9 (8): 1735 - 1780. doi:10.1162/neco.1997.9.8.1735

[32] Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J. Distributed representations of words and phrases and their compositionality. In: Proceedings of the 26th International Conference on Neural Information Processing Systems (NIPS'13); Lake Tahoe, NV, USA; 2013. pp. 3111–3119.

[33] Orhan U, Arslan E. Learning Word-vector Quantization: A Case Study in Morphological Disambiguation. ACM Transactions on Asian and Low-Resource Language Information Processing 2020; 19 (5): 1–18.

[34] Seker A, Tsarfaty R. A Pointer Network Architecture for Joint Morphological Segmentation and Tagging. In Findings of the Association for Computational Linguistics: EMNLP; Online; 2020. pp. 4368–4378.

[35] Vinyals O, Fortunato M, Jaitly N. Pointer networks. In Advances in Neural Information Processing Systems 28 (NIPS 2015); Montreal, Canada; 2015. pp. 2692–2700.

[36] Nivre J, Marneffe MC, Ginter F, Goldberg Y, Hajic J et al. Universal dependencies v1: A multilingual treebank collection. In Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16), Portoroz,ˇ Slovenia; 2016. pp.1659–1666.

[37] Bayes T. An Essay Toward Solving a Problem in the Doctrine of Chances. Philosophical Transactions 1763; 53: 370-418.

[38] Sutskever I, Vinyals O, Le Q. Sequence to sequence learning with neural networks. In: International Conference on Learning Representations (ICLR); Montreal, Canada; 2014. pp. 3104–3112.

[39] Shaw P, Uszkoreit J, Vaswani A. Self-Attention with Relative Position Representations. In: North American Chapter of the Association for Computational Linguistics (NAACL); New Orleans, LA; 2018. pp. 464–468.

[40] Kudo T, Richardson J. SentencePiece: A Simple and Language Independent Subword Tokenizer and Detokenizer for Neural Text Processing. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations; Brussels, Belgium; 2018. pp. 66–71.

[41] Sennrich R, Haddow B, Birch A. Neural Machine Translation of Rare Words with Subword Units. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics; Berlin, Germany; 2016. pp. 1715–1725.

[42] Wu Y, Schuster M, Chen Z, Le QV, Norouzi M et al. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. ArXiv abs/1609.08144, 2016

[43] Akın AA, Akın MD. Zemberek, an open source nlp framework for turkic languages. Structure 2007; 10: 1-5.

[44] Defazio A,Jelassi S. Adaptivity without Compromise: A Momentumized, Adaptive, Dual Averaged Gradient Method for Stochastic Optimization. ArXiv abs/abs/2101.11075, 2021

[45] Geva M,Schuster R, Berant J, Levy O. Transformer Feed-Forward Layers Are Key-Value Memories. In: Proceedings of Empirical Methods in Natural Language Processing (EMNLP); Punta Cana, Dominican Republic; 2021. pp. 5484–5495.

[46] Libovicky J, Helcl J, Marecek D. Input combination strategies for multi-source transformer decoder. In Proceedings of the Third Conference on Machine Translation; Brussels, Belgium; 2018, pp. 253–260.