Turkish Language Morphological Analysis and Disambiguation Using Transformers and Large Language Models

Mete Erdoğan
Koç University
merdogan18@ku.edu.tr

Abstract

The process of morphological analysis involves breaking down words into their smallest units of meaning. This is important for improving accuracy and efficiency in various natural language processing tasks, such as part-of-speech tagging, grammatical error correction, information retrieval, search engine optimization, and text classification. Consequently, in this work, we will be conducting morphological analysis and disambiguation for the Turkish language using the TrMor2018 dataset. The work consists of two separate parts: (i) first part involves the development of supervised models with an emphasis on Transformer models, (ii) whereas the second part involves utilizing Large Language Models (LLMs) with different prompt engineering and fine-tuning methods, applied on morphological analysis and disambiguation as a downstream task.

1 Introduction

	English	Turkish
β	0.466	0.593

Table 1: Heap's law β values calculated using online newspaper corpora showing the relative vocabulary growth of English and Turkish (Hakkani-Tür et al., 2002).

Morphological analysis and disambiguation allows us to identify the affixes that are combined with words to form different tenses or comparative forms. Therefore it enables understanding how languages are structured and what each word means in a certain context and beneficial for many NLP tasks. In Linguistics, Heaps' is an empirical principle that explains the relationship between the length of a document and the number of unique words it contains. It quantifies the type-token relationship as $V_R(n) = Kn^\beta$ where where V_R is the number of distinct words in a document with n words; and K and β are empirically determined parameters where a higher β means having larger vocabulary for a given corpus size. We can see in table 1 that Turkish is much richer than English in terms of its word variety and it has a much larger vocabulary given a corpus with the same amount of words. Analyzing such morphologically complex and agglutinative languages like Turkish is particularly crucial because each word can contain multiple affixes that carry different grammatical and semantic meanings, and it is possible to generate thousands of different forms given a root word (Oflazer and Tür, 1997).

Analysis of the Turkish word "izin"
Yerdeki **izin** temizlenmesi gerek. (that trace/print) **Analysis:** iz+Noun+A3sg+Pnon+Gen
Üzerinde parmak **izin** kalmış. (your trace/print) **Analysis:** iz+Noun+A3sg+P2sg+Nom
Içeri girmek için **izin** alman gerekiyor. (permission) **Analysis:** izin+Noun+A3sg+Pnon+Nom

Table 2: Morphological analyses for the Turkish word *"izin"*. An example context and its contextual meaning is given in paranthesis before each analysis. (Hakkani-Tür et al., 2002)

In table 2, we observe a practical demonstration of morphological analysis for a particular Turkish word that has different meanings and semantic information in different sentences, similar to our TrMor2018 dataset. Consequently, developing a simple system for morphological disambiguation in Turkish is not practical. Instead, a comprehensive statistical analysis of sentences is required to grasp the contextual meaning of each word (Hakkani-Tür et al., 2002).

This work consists of two separate parts:

1. The development of supervised sequence to sequence generation models with an emphasis on Transformers.
2. Utilizing Large Language Models (LLMs) applied on morphological analysis as a downstream task.

using the TrMor2018 dataset.

For the first part of this work, the objective was to create a cutting-edge transformer model that can perform lemmatization and tagging simultaneously. Although there are successful LSTM-based models like Morse (Akyürek et al., 2019), there is currently no well-developed transformer model for sentence-level morphology. Here, the first model we implemented involves a basic character-level Transformer that takes the entire sentence as input and provides the corresponding lemmas and tags for each word in sequence. Next, a more generic model is implemented, that we call as the Morse++ model, with two separate Transformer encoders. The word encoder generates word vectors based on the characters, while the context encoder is intended to capture the meaning of the word in the given sentence. For the second part of the work, ChatGPT's one-shot learning performance was evaluated using different prompting methods for conducting morphological analysis and disambiguation. Then, the Prefix Tuning method was implemented using the multilingual mGPT model with suitable prompting.

2 What you proposed vs. what you accomplished

Almost all the tasks that were proposed were accomplished:

1. Implementing the baselines. basic RNN (without context), Vanilla Tranformer (with and without context).
2. Implementing and experimenting with the generic Transformer Morse++ model.
3. Performing one-shot and few-shot learning experiments with ChatGPT.
4. Performing tuning experiments with LLMs. Prefix Tuning with mGPT.
5. Creating the evaluation metrics tables.

Here are the extra tasks that were performed, but was not mentioned in the proposal:

1. Performing inference with the state-of-the-art Morse model on the same train-test splits for comparison with our Morse++ model.
2. In-context example selection using the Sentence BERT embeddings to enhance ChatGPT's few-shot learning performance.

3 Related work

People started morphological analysis and disambiguation first by attacking the lemmatization and tagging tasks separately, using different models. Then the literature shifted towards performing these tasks jointly. The first attempts used statistical and information theoretical models, whereas now the Deep Learning based models yield the best performance.

First, a method that automatically generates lemma classes by comparing the reversed word forms with the corresponding lemmas, using the shortest edit script was proposed Chrupala, 2006. They then use a standard Naive Bayes classifier to assign lemma classes to the words. This approach treats lemmatization procedure in a similar way to POS tagging.

Then, the Morfette model was introduced as an extension of the previous method, which allows joint lemmatization and tagging (Chrupala et al., 2008). This model consist of two different learning modules, both trained to predict morphological tags and lemmas using a maximum entropy classifier. In addition, there is a third module that dynamically combines the predictions of the previous models and generates a probability distribution over sequences of lemma-tag pairs.

Subsequent research has demonstrated the effectiveness of Deep Learning models in morphological analysis. A recently developed model using a character-based RNN coder in combination with a whole-tag classifier outperformed previous approaches (Heigold et al., 2017).

In addition, the LSTM-based model called Morse was presented for Turkish sentence-level morphological analysis and disambiguation, along with the TrMor2018 dataset (Akyürek et al., 2019). Morse includes three different LSTM encoder architectures. First, there is the word en-

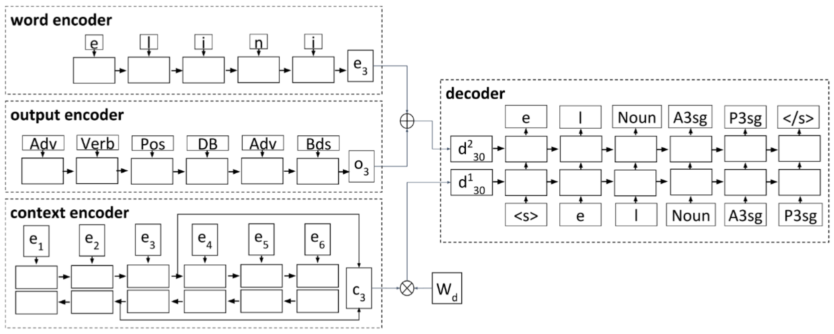


Figure 1: Morse Model illustration for a given Turkish word "elini" (Akyürek et al., 2019).

coder, which uses a unidirectional LSTM to process character embeddings and generate hidden state vectors. The final hidden state vector serves as the word embedding. Second, Morse includes a context encoder, a bidirectional LSTM that processes the word embeddings and generates a context vector for each word. Finally, the model contains an output encoder, a unidirectional LSTM, which uses the morphological tags of the previous word as input. The outputs of all encoders are combined before being fed into the decoder network as shown in figure 1, which generates the output lemma and tags. Morse achieves an impressive lemma+tag accuracy of 97.67% on the TrMor2018 dataset. For comparison, we also evaluated Morse on the same train-val-test $(80\%, 10\%, 10\%)$ split that we tested our Transformer models.

Transformer-based models have proven to be particularly powerful in numerous NLP tasks, especially in sequence to sequence generation Vaswani et al., 2017. For example, Acikgoz et al., 2022 achieved promising results on clause-level morphology tasks using the vanilla transformer architecture and employing various data augmentation techniques. Our Version-1 vanilla transformer model is similar to this approach at its core. However, their task's focus was limited to clause-level analysis, while our task is more complex, involving both lemmatization and sentence-level tagging, which requires processing longer contextual information. The goal of this work is to establish a transformer baseline for Turkish sentence-level morphological analysis and potentially outperform other methods in the literature.

Large Language Models such as BERT (Devlin et al., 2019), XLNet (Yang et al., 2019) and GPT-3 (Brown et al., 2020) that are pre-trained on a large text corpus are also becoming increasingly popular due to its ability of being fine-tuned on specific downstream tasks such as translation, question answering and performing arithmetic operations, and even surpasses many prior fine-tuned state of art approaches.

LLMs can be successfully adapted into different tasks. First, we can directly use the pre-trained models with guidance in k-shot learning scenarios. (Liu et al., 2022) proposes the KATE strategy to find the most suitable examples for in-context learning. The process involves using a certain sentence encoder to convert sources in both the training set and test set to vector representations. Then the most similar examples from the training set is found using cosine similarity on the vector embeddings and knn algorithm to pick the closest k samples for k-shot learning.

LLMs can also be fine-tuned using gradient based optimization procedures. Houlsby et al., 2019 utilizes the adapter-tuning method that adds additional layers on the BERT model and only optimizes the parameters of the additional layers to reduce the computational cost of training the whole model from scratch. Furthermore, the Prefix Tuning method was proposed by (Li and Liang, 2021), that first freezes the LLM model parameter and tries to find the optimal input prompts by concatenating additional virtual tokens that are trained using gradient descent and freezed in the test time. This is similar to finding the best one-shot learning sample but in a continuous optimization setting. Also, Acikgoz et al., 2022 utilized Prefix Tuning method using the mGPT model and applied it to a multilingual morphological analysis task. Prefix tuning method was also used in our work with an mGPT model.

4 Dataset

This work uses the TrMor2018 dataset that was presented in Akyürek et al., 2019. This dataset includes 34673 sentences and 460669 words in total from different genres of Turkish text, and was annotated semi-automatically. This is the state of the art of the Turkish morphological datasets with a disagreement level of 4.4% with the hand-tagged subset. The dataset is available in the GitHub repository mentioned in (Akyürek et al., 2019). The dataset consists of sentences where each word is labeled with is lemma and its morphological tags in the same manner as the Turkish word ”izin” was analysed in table 2. A sample input (source) and output (target) can be seen in table 3 below, and also in figures 5, 6.

Source	Target
herkesin	herkes+Noun+A3sg+Pnon+Gen
iyi	iyi+Adj
durumda	durum+Noun+A3sg+Pnon+Loc
olduğunu	ol+Verb+Pos^DB+Noun+PastPart+A3sg+P3sg+Acc
gördüm	gör+Verb+Pos+Past+A1sg
.	.+Punct

Table 3: An example source and target pair from the TrMor2018 dataset. The source is a sentence and the target is the root (lemma) of each word and the corresponding morphological tag sequence.

	Before	After
Sentence	34673	30722
Word	460669	354139
Input Tokens	2588305	1971640
Target Tokens	3459345	2633028
Unkowns	1501	0

Table 4: TrMor2018 dataset statistics before and after our preprocessing (elimination). Here the Unknowns represent the count of sentences that include a word which’s lemma+tag analysis is not available.

4.1 Data preprocessing

The TrMor2018 dataset contains many unknown tags, meaning that the corresponding lemma+tag analysis is not available. As we are trying to grasp the contextual information of each word, we discard not only these unknown words, but the whole sentences that include such words. Such ambiguities may damage our models’ evaluations. Furthermore, we discard the sentences with word length $>$ 64, sentences that include words with token length $>$ 64, sentences with total input or target token count $>$ 256. Lastly, we eliminate the sentences with total target character length $>$ 512. While calculating the total token length, each output tag such as ”Noun” is counted as length 1; but when calculating the total character length, ”Noun” is counted as length 4 considering each character as one token.

These eliminations are necessary as the sentences with very large context length resulted in GPU memory problems especially with our Transformer Version-1 and the contextual prefix tuning method where we concatenate each word and push it to the GPU in a mini-batch in the training process. We perform all our experiments on this preprocessed data where we can see the before and after preprocessing statistiscs in table 4.

5 Baselines

Our baselines consist of 3 distinct encoder-decoder models, LSTM, LSTM with attention and a Vanilla Transformer, all without concerning the context information and only parsing one word at a time. For instance, the word ”izin” as in table 2 may occur mostly with the analysis ”iz+Noun+A3sg+Pnon+Gen”. Then these models will learn to output the most frequently occuring analysis and will not learn any contextual dependancy.

All baselines were trained with 250 epochs with negative log-likelihood loss objective, and batch size 128. The AdamW optimizer with its default parameters were used. The inverse square root learning rate scheduling with linear warm up (4000 steps) is used as it performed the best. In every 10 epochs the models’ performance in the validation set was evaluated and the model parameters that yielded the best validation set performance was used in the testing. Lastly, %80-%10-%10 train-test-validation split is used and we reported the average results for 3 random splits in the results section.

While tuning the hyperparameters, we could not perform a grid-search like procedure due to resource constrains, but run a couple of experiments with exhaustively searching (embed dims E = 128,256,512, batch sizes = 32,64,128, hidden sizes H = 128,256,512 number of layers = 1,2,4,8, dropout probabilities = 0.1,0.2,0,3, number of attention heads = 2,4,8,16 etc.) and using the parameters that gave higher validation set accuracy. Then the optimal hyperparameters we found for both of the LSTM models were learning rate 1e-3, embed dimension and hidden sizes of 256, number of 2 layers and dropout probability of 0.3. Then for the Transformer, we used learning rate 5e-4, number of 4 layers, 16 attention heads and other parameters were same as the LSTM models. The beam search was also tested but as it did not give any considerable improvement, and the greedy decoding is preferred for faster evaluation.

6 Your approach

In all models, our task will be to perform Morphological analysis where the input is a sentence $S = [w_1, \ldots, w_N]$, where w_i is the i'th word in the sentence. Each word consists of a sequence of characters $w_i = [w_{i1}, \ldots, w_{iL_i}], w_{ij} \in \mathcal{A}$ where $\mathcal{A}$ is the set of alphanumeric characters that constitutes the input space of the models. Then, a sample output for a sentence is $O = [o_1, \ldots, o_N]$ where an output corresponding to a word looks like $o_i = [s_{i1}, \ldots, s_{iR_i}, f_{i1}, \ldots, f_{iM_i}]$ where $s_{ij} \in \mathcal{A}$ is an alphanumeric character in the lemma (root), and $f_{ij} \in \mathcal{T}$ is a morphological feature from a feature set such as $\mathcal{T} = \{\text{Noun,Adj,Nom,A3sg}, \ldots\}$. Then the output space of the model is $A \cup \mathcal{T}$.

6.1 Transformer Models

Two Transformer models were developed using the PyTorch library (Paszke et al., 2019) and trained from scratch using the KUACC Cluster GPU's. We broadly utilized the Neural-Transducer GitHub repository [1] which had Vanilla Transformer implementations on Pytorch (torch.nn.MultiheadAttention). Then we implemented our generic Transformer architectures based on these modules.

6.1.1 Contextual Transformer

The first model that we implemented is a character based vanilla Transformer model like the one proposed in Vaswani et al., 2017. In this model, the whole input sentence S is fed into the transformer encoder where all the characters are concatenated such that the words are separated by space characters as we can see an example tokenization in figure 2. Then the model output will be the serial lemma and token combination corresponding to each word. But the model output O may not have the same word length as the input sentence,

[1]`https://github.com/shijie-wu/neural-transducer`

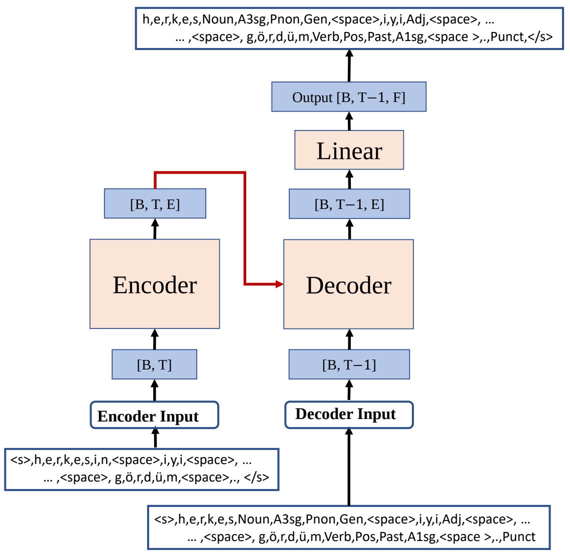

h,e,r,k,e,s,Noun,A3sg,Pnon,Gen,<space>,i,y,i,Adj,<space>, …
… ,<space>, g,ö,r,d,ü,m,Verb,Pos,Past,A1sg,<space >,.,Punct,</s>
Output [B, T−1, F]
Linear
[B, T, E]
[B, T−1, E]
Encoder
Decoder
[B, T]
[B, T−1]
Encoder Input
Decoder Input
<s>,h,e,r,k,e,s,i,n,<space>,i,y,i,<space>, …
… ,<space>, g,ö,r,d,ü,m,<space>,., </s>
<s>,h,e,r,k,e,s,Noun,A3sg,Pnon,Gen,<space>,i,y,i,Adj,<space>, …
… ,<space>, g,ö,r,d,ü,m,Verb,Pos,Past,A1sg,<space >,.,Punct

Figure 2: The Demonstration of input-output structure of the Contextual Transformer model that is equivalent to Vanilla Transformer architecture. The input and output consist of all the words in a given sentence concatenated using a space token in between.

since the transformer decoder will output new tokens until the end of sentence token is generated. Therefore, there may be some alignment issues while calculating the lemma+tag accuracy if an unnecessary word is generated in the middle of the sentence. We trained this model for 250 epochs with the same learning rate scheduling and optimizer as the baselines. Batch size of 64 and learning rate 5e-4 was used where the transformer hidden sizes H=512, embedding dimension E=512, encoder and decoders with 4 layers, dropout probability = 0.3 and 16 attention heads.

6.1.2 Morse++

The second model that will be utilized was inspired by the word and context encoder structure of the Morse and was intended to create a Transformer model as can be seen in figure 4. With this model, we aim to fix the alignment issue of the previous model by outputting the same amount of lemma+tag pairs for the input words, also by taking the contextual information into account.

The data loader of the model will first take a mini batch of B sentences. At first, we aimed to pad all the words using a padding token to have P_2 characters, and all the sentences in the mini batch will be padded to have P_1 words. Then we would have a tensor of size B xP_1xP_2. But this procedure is not realizable as the input of the Word encoder would include zero vectors that consist of

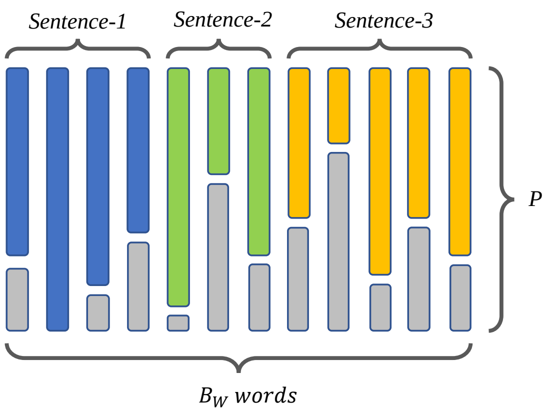


Figure 3: Mini-batch collating procedure of Morse++ model. Each colored rectance represent a word and their length represent token count. The grey rectangles represent the padding where all words are padded to the longest word length P.

only padding in the batch dimension, leading to undefined outputs due to Softmax applied to a zero vector in the attention layer. Therefore, a different approach is used where the input mini-batch is collated as in figure 3. Here, all the words of all sentences that may have different word lengths are concatenated and padded only to fix the length of the words. The sentence lengths are also saved to separate each sentence before feeding it to the context encoder.

After structuring the data as in figure 3, the Word Encoder will be applied. Then, the output of the word encoder will be average pooled in the character dimension that has length P_1. This will result in vectors that represent the word embeddings. Then these vectors will be fed into the context encoder. Lastly the outputs of the Word and the Context encoders will be combined as can be seen with the red and white addition operator in figure 4. Different strategies such as concatenation or addition are used but the concatenation yielded the best performance.

Lastly, the decoder gets both the result of the encoder and the target; and outputs the logits corresponding to each character in the mini-batch. Then the argmax operation will output the intended characters. In this model, there is no alignment problem as the first model as we will know the start and end points of each word. This model was trained for 300 epochs again with inverse square root learning rate scheduling with linear warm up of 4000 steps. Batch size of 8 and learning rate 5e-4 was used where the encoder transformer hidden sizes H=256, embedding dimension E=256 for encoders and E=512 for the decoder as the output of two encoders are concatenated. The encoder and decoders had 6 layers, dropout probability = 0.3 and number of attention heads was 16.

6.2 LLM Models

6.2.1 One-shot Learning with ChatGPT

OpenAI's ChatGPT API is utilized to make one-shot learning experiments on a 200 sample subset of the same Test set we used to evaluate our Transformer models. A sample from the training dataset is given to the ChatGPT and asked to perform a similar analysis to a test set instance using the prompt structure presented in figure 5. We used ChatGPT3.5 Turbo with the user role in our experiments.

Three different prompting methods are utilized, where the first one we call as the "Vanilla" prompting, that is the one as given in figure 5. Secondly, the "With dictionary" prompting where we changed the question by asking the model to generate only the tokens among the given set of characters and morphological tags that are extracted from TrMor2018 dataset (figure 8). Lastly, we used in-context sample selection using the Sentence BERT ("nli-bert base") from the transformers library of Huggingface to calculate the 768-length embedding vectors for each sentence in the training set. Then in the inference time, we also calculate the BERT embedding of the given test sample and find the training set sample that has the lowest cosine distance to the test sample embedding. Then we use this sample in the one-shot learning. Some chosen samples can be seen in table 5 where the model mostly finds the pairs that have the token similarity rather than contextual similarity.

In-context sample selection with Sentence BERT
Test sample: dorian gray gülerek başını salladı . **Train sample:** dorian gray başını salladı .
Test sample: prens huşu ile elini aldı , öpmeye başladı . **Train sample:** prens yeniden onun ellerini öpmeye başladı .
Test sample: arkadaşı ona şaşkınlıkla baktı . **Train sample:** arkadaşı şaşkınlıkla etrafına bakındı .

Table 5: Closest training-set samples selected for one-shot learning using the cosine distance on the sentence BERT embeddings.

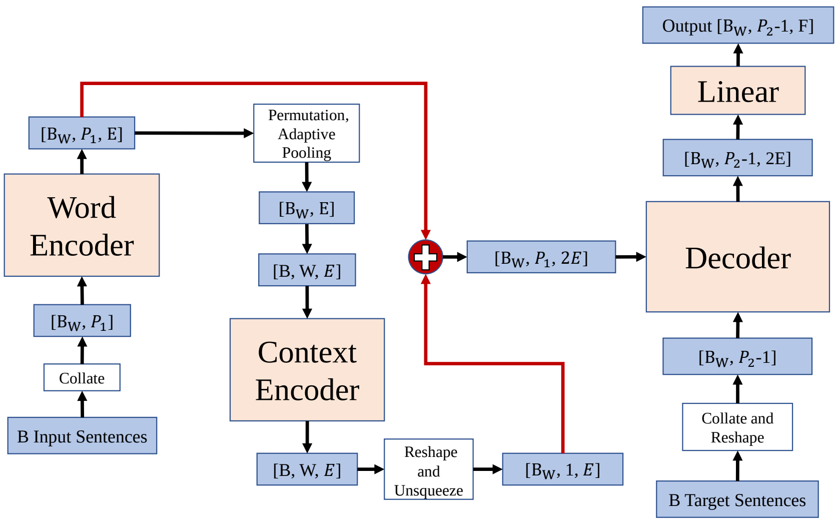
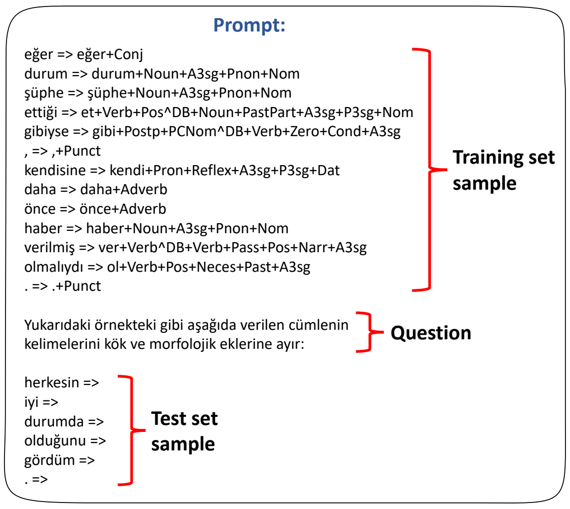


Figure 4: Our Generic Transformer Morse++ with Word and Context Encoders. Here B is the batch size, W is the maximum word count of a sentence in the mini-batch of B sentences, B_W is the collated word count (total word coun in B sentences), E is the embedding dimension, F is the size of the output space, P_1 is the maximum character length of a word in the input mini-batch, P_2 is the maximum character length of a word in the target mini-batch. The input of the decoder is $P_2 - 1$ due to teacher forcing.



Training set sample
Question
Test set sample

Prompt:

```
eğer => eğer+Conj
durum => durum+Noun+A3sg+Pnon+Nom
şüphe => şüphe+Noun+A3sg+Pnon+Nom
ettiği => et+Verb+Pos^DB+Noun+PastPart+A3sg+P3sg+Nom
gibiyse => gibi+Postp+PCNom^DB+Verb+Zero+Cond+A3sg
, => ,+Punct
kendisine => kendi+Pron+Reflex+A3sg+P3sg+Dat
daha => daha+Adverb
önce => önce+Adverb
haber => haber+Noun+A3sg+Pnon+Nom
verilmiş => ver+Verb^DB+Verb+Pass+Pos+Narr+A3sg
olmalıydı => ol+Verb+Pos+Neces+Past+A3sg
. => .+Punct

Yukarıdaki örnekteki gibi aşağıda verilen cümlenin
kelimelerini kök ve morfolojik eklerine ayır:

herkesin =>
iyi =>
durumda =>
olduğunu =>
gördüm =>
. =>
```

Figure 5: In context examples chosen from the training-set for ChatGPT one shot learning using sentence BERT embeddings.

6.2.2 Prefix Tuning with mGPT

The prefix tuning method presented in (Li and Liang, 2021) was implemented using the multilingual mGPT model (Shliazhko et al., 2022). Here, we mainly utilized the OpenPrompt GitHub repository [2]. Again a contextual and non-contextual approaches were implemented by adjusting the input and output pairs.

In the non-contextual version, words are fed into the model one by one; whereas in the contextual model all words in a sentence are concatenated. Here, we used the tilde ("$\sim$") symbol as the separator character in the concatenation of the subsequent words, as it is not frequent in Turkish text. When we use the space character, since it has its own significance that is grasped by the mGPT model in the pretraining, the model has a tendency to remove the spaces that are unnecessary according to the context, but are important for the evaluation to separate consequent words or punctuations.

Then we added two prefix tokens and trained them while freezing the rest of the parameters of the mGPT model. We trained the non-contextual model for 6 epochs, and the contextual model for 10 epochs, both with an initial 5e-5 learning rate and decrease it linearly to zero in training. In the generation, p-sampling (0.9) and beam-search with size-5 are used.

7 Results

The task of generating the lemma+tag pairs for a given word sequentially is different than regular classification, but more aligned with the generation tasks, as the output length may be different for a given input in different context. Therefore, the accuracy of correctly generating the whole lemma+tag pairs is the important metric rather than checking the tokenwise accuracy. Therefore, precision, recall and F1 score metrics that works in more regular classification tasks are not reported,

[2] https://github.com/thunlp/OpenPrompt

Model	Token Acc.	Word Acc.	Sentence Acc.	Edit Dist.
LSTM	98.38 ± 0.06	96.24 ± 0.07	66.87 ± 0.13	0.1168 ± 0.0038
LSTM (Attention)	98.44 ± 0.08	96.25 ± 0.19	67.13 ± 1.28	0.1139 ± 0.0045
Transformer	**98.58 ± 0.08**	97.04 ± 0.11	72.17 ± 0.63	0.1121 ± 0.0053
Transformer (C)	97.82 ± 0.07	96.11 ± 0.11	72.63 ± 0.49	0.1419 ± 0.0044
Morse++ (C)	**98.58 ± 0.03**	**97.14 ± 0.05**	**73.58 ± 0.87**	**0.0973 ± 0.0027**
Morse (C)	**98.75 ± 0.05**	**97.66 ± 0.11**	**77.98 ± 1.09**	**0.0975 ± 0.0048**
mGPT Prefix Tun.	97.32 ± 0.18	94.80 ± 0.20	55.01 ± 0.68	0.1816 ± 0.0266
mGPT Prefix Tun. (C)	95.29 ± 0.21	94.85 ± 0.10	57.36 ± 0.13	0.1616 ± 0.0063

Table 6: Test set results of several methods on the TrMor2018 dataset (Akyürek et al., 2019). The results were obtained using 80%-10%-10% train-test-validation splits and the average results and their standard deviation are presented for 3 random splits. The method names that has the "C" expression means that the model is contextual. The Levenstein distance is used for the edit distance metric. Bolds indicate the best 2 result for each metric.

	Methods	Token Acc.	Word Acc.	Sentence Acc.	Edit Dist.
ChatGPT one-shot learning	**Vanilla**	68.96	32.13	0.00	1.8662
	With dictionary	67.63	32.81	0.50	1.7165
	Example selection	68.99	37.89	2.00	1.6010
mGPT Prefix Tun.	**no-context**	**96.51**	93.72	56.0	0.1971
	with-context	94.62	**94.22**	**60.0**	**0.1870**

Table 7: Test set results evaluated on a 200-sample subset of the same test set used in table 6, of ChatGPT one-shot learning with different prompting methods, and the with and without context prefix tuning methods. "Vanilla" is the regular prompting as given in figure 5, "With dictionary" is the prompting is the one given in figure 8, "Example selection" is the prompting where the one-shot example selected using the closest sentence BERT embedding. Bolds indicate the best result for each metric.

also aligning with (Akyürek et al., 2019). In the result tables 6 and 7, the word (lemma+tag) accuracy, sentence accuracy and edit (Levenstein) distance are reported.

7.1 Transformer Models

The Morse++ model passes all our baseline models in all metrics. However, the Contextual Transformer model described in 6.1.1 performed worse in terms of word accuracy and edit distance and stayed behind the baselines probably because of the previously mentioned alignment problems, although it performed well in terms of sentence accuracy.

In our work, we expected to pass the Morse model, that is the current state-of-the-art model both in TrMor2018 dataset, and other multilingual datasets such as UDv2 (Nivre et al., 2020). However, Morse++ could not pass Morse in Word accuracy and Sentence accuracy, although the Edit (Levenstein) distance was very close. Our non-contextual Transformer baseline even performs very similar to Morse++, probably due to the Context encoder of Morse++ not grasping the contextual information well enough.

7.2 LLM Models

We only worked on a 200 sample subset while experimenting with the ChatGPT due to resource constraints where the API requires payment per generated and used token. Then we present both the ChatGPT one-shot learning and prefix tuning results on this 200 sample subset in table 7.

The results of one-shot learning with ChatGPT is very surprising where the language model is able to correctly label more than 30% of the words and more than 65% of the tokens without any training, but with pure prompting. Our aim with the "With dictionary" prompting was to increase the likelihood of the model to generate tags that are in the vocabulary of TrMor2018 dataset, but it did not yield much improvement over "Vanilla" prompting. Furthermore, in-context example selection using sentence BERT encoding significantly improved the performance with around 5% word accuracy.

The prefix tuning method also performed well and yielded a much better performance than one-shot learning, although could not pass the supervised Transformer models.

Model	Token Acc.	Word Acc.	Sentence Acc.	Edit Dist.
Transformer	98.70±0.15	97.14±0.29	52.31±4.32	0.1036±0.0103
Morse++ (C)	98.72±0.11	97.16±0.35	53.55±4.47	0.1030±0.0075
Morse (C)	98.84±0.13	97.60±0.07	58.87±1.97	0.0823±0.0039

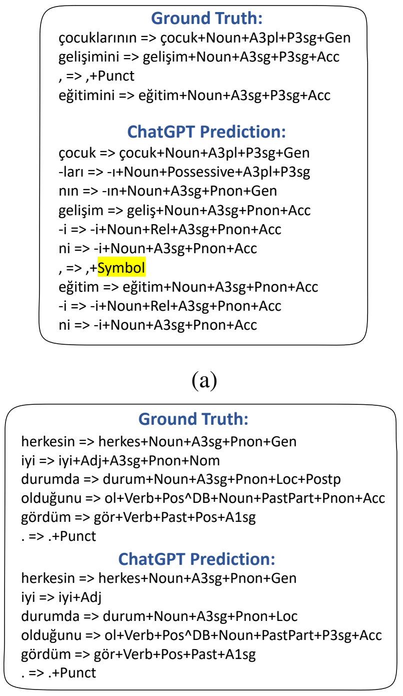
Table 8: Evaluation results of the models on the test set sentences with longer than 20 words. The results were obtained using the same data splits as in table 6, and the average results and their standard deviation are presented for 3 random splits. The method names that has the ”C” expression means that the model is contextual.

8 Error analysis

8.1 Transformer Models

The non-contextual vanilla Transformer yielded 97.04% word accuracy, and shows that more than 97% of the wordforms can be correctly tagged ignoring the context. The result of Morse++ being very close to this non-contextual model suggests the possibility that the Context Encoder of Morse++ (figure 4) does not work. However, when the model outputs are analysed, for example, the Turkish word ”bu” can have two different lemma+tag analysis as ”bu+Pron+Demons+A3sg+Pnon+Nom” or ”bu+Det”. As the second version is more frequent in the dataset, the non-contextual model always outputs that version. However, when we look at Morse++'s outputs for the test set, the model correctly labeled some of the samples that belong to both versions. Therefore, the model is able to grasp the contextual information. However, Morse still performs almost half a percent better than Morse++ probably as the model is not optimized enough.

Further analysis on the models' accuracy on long sentences that consist of more than 20 words is presented in table 8. Again Morse++ and non-contextual Transformer performed similar, and Morse yielded better word and sentence accuracy.


Ground Truth:
çocuklarının => çocuk+Noun+A3pl+P3sg+Gen
gelişimini => gelişim+Noun+A3sg+P3sg+Acc
, => ,+Punct
eğitimini => eğitim+Noun+A3sg+P3sg+Acc

ChatGPT Prediction:
çocuk => çocuk+Noun+A3pl+P3sg+Gen
-ları => -ı+Noun+Possessive+A3pl+P3sg
nın => -ın+Noun+A3sg+Pnon+Gen
gelişim => geliş+Noun+A3sg+Pnon+Acc
-i => -i+Noun+Rel+A3sg+Pnon+Acc
ni => -i+Noun+A3sg+Pnon+Acc
, => ,+Symbol
eğitim => eğitim+Noun+A3sg+Pnon+Acc
-i => -i+Noun+Rel+A3sg+Pnon+Acc
ni => -i+Noun+A3sg+Pnon+Acc

(a)

Ground Truth:
herkesin => herkes+Noun+A3sg+Pnon+Gen
iyi => iyi+Adj+A3sg+Pnon+Nom
durumda => durum+Noun+A3sg+Pnon+Loc+Postp
olduğunu => ol+Verb+Pos^DB+Noun+PastPart+Pnon+Acc
gördüm => gör+Verb+Past+Pos+A1sg
. => .+Punct

ChatGPT Prediction:
herkesin => herkes+Noun+A3sg+Pnon+Gen
iyi => iyi+Adj
durumda => durum+Noun+A3sg+Pnon+Loc
olduğunu => ol+Verb+Pos^DB+Noun+PastPart+P3sg+Acc
gördüm => gör+Verb+Pos+Past+A1sg
. => .+Punct

(b)

Figure 6: ChatGPT Vanilla prompting results (a) Successful prediction (b) Erroneous prediction where the model incorrectly tried to divide the words into its suffixes. Also the model produced some out-of-vocabulary tags such as the ”Symbol” tag (highlighted in the figure), although the TrMor2018 dataset labels punctuations with ”Punct” tag.

8.2 LLM Models

In many one-shot learning cases, we get a result that is close to the ground truth as in figure 6a. However, some predictions incorrectly try to divide the words into its suffixes or may produce out-of-vocabulary tags as in figure 6b. To solve this issue with output length, we perform a couple of passes on the samples with incorrect length until the desired output length is reached, owing to the randomness in the LLM generation. Figure 7 demonstrates how the number of outputs with correct length increases for different prompting methods. We observe that the sentence BERT embedding based example selection yields more aligned results with the ground truth sentence length. Also, we tried to solve the out-of-vocabulary tags (generating ”Symbol” instead of ”Punct”) generated by the model using the ”With dictionary” prompting, no significant improvement was observed.

In table 9 we see an example where all prompting methods were not successful. Here the ”Vanilla” method separated the suffix of the first word and wrote its meaning rather than giving morphological tokens. Here, ”With dictionary” prompt's output format was also different than the intended version, due to the provided long tag-set

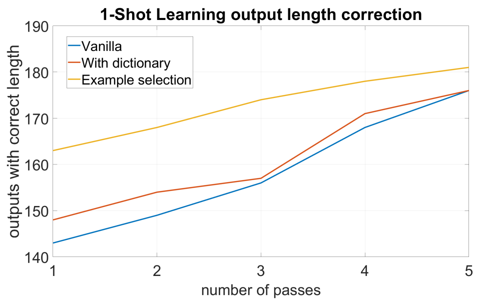


Figure 7: The change of the number of ChatGPT one-shot learning outputs with the correct length. The false-length outputs are repetitively fed into the model until desired output size is reached. ”Vanilla” method is the basic prompt as in figure 5, ”With dictionary” is the prompt that asks the model to use only the tokes in a given list, ”Example selection” is the prompts that chose the 1-shot example using the sample with the lowest cosine distance in the sentence BERT embedding space.

in the prompt question, causing the model to forget the one-shot example. Lastly, the ”Example selection” method’s output length is also false and could not be corrected even after 5 passes of inference.

ChatGPT one-shot learning example outputs with different prompting methods	

Vanilla	ikisi =>iki+Pron+Quant+A3sg+Pnon+Nom
	nin =>(bağlaç ”ki”nin iyelik eki)
	de =>de+Postp
	gözleri =>göz+Noun+Pl+A3sg+P3sg+Nom
	ıslanmıştı =>ıslan+Verb^DB+Verb+Pass+Pos+Past+A3sg
	. =>.+Punct

With Dictionary	ikisi+Pron+Det+A3pl+P3pl+Gen =>”iki” + ”sinin” de+Postp =>”de” göz+Noun+A3sg+Pnon+Nom =>”gözleri” ıslan+Verb+Pos+Narr+A3sg =>”ıslanmıştı” .+Punct =>”.”

Example selection	ikisi =>ik+Pron+Pnon+Nom^DB
	nin =>nin+Postp+Gen
	de =>de+Postp
	gözleri =>göz+Noun+A3pl+P3sg+Nom
	ıslanmıştı =>ıslan+Verb+Pos+Past+A3sg
	. =>.+Punct

Table 9: A predicted sample with different prompting methods on one-shot learning with ChatGPT.

9 Conclusion

This work provided a benchmark for morphological analysis and disambiguation of Turkish by experimenting with a wide range of models such as LSTMs, regular and more generic Transformer models, and Large Language Models (LLMs) with different prompting and fine-tuning techniques. The most challenging part with the project was the resource problems. We experimented with 8 models that required extensive training procedures and resources, and 3 different prompting techniques using the ChatGPT API that necessitates payment per generated token. In the first part of our work, we tried to develop transformer models to capture the contextual information in a sentence. Although we achieved good results in doing so, the Morse++ model in particular needs further development, that is the particular point that will be pursued by attempting for different fusing techniques of the Word and Context encoder outputs. In the LLM part, our results were surprising as the ChatGPT model performed unexpectedly well on the TrMor2018 dataset. Although the one-shot learning results were much worse than the supervised models, this is not a fair comparison since the supervised models are focused on the task of morphological analysis with hundreds of training epochs, while the LLMs are trained on much larger corpus with different content.

10 AI Disclosure

- Did you use any AI assistance to complete this proposal? If so, please also specify what AI you used.
  - Only while writing the ethics statement, we used OpenAI’s ChatGPT-3.5.

If you answered yes to the above question, please complete the following as well:

- If you used a large language model to assist you, please paste *all* of the prompts that you used below. Add a separate bullet for each prompt, and specify which part of the proposal is associated with which prompt.
  - Prompt: ”Our work consists of two separate parts; first part involves the development of supervised models with an emphasis on Transformer models for Turkish language morphological analysis. Whereas the second part involves utilizing Large Language Models (LLMs) with different prompt engineering and fine-tuning methods, applied on morphological analysis and disambiguation as a downstream task. Please write an explicit ethics statement on the broader impact of the work, or other ethical considerations”
- **Free response:** For each section or paragraph for which you used assistance, describe your overall experience with the AI. How

helpful was it? Did it just directly give you a good output, or did you have to edit it? Was its output ever obviously wrong or irrelevant? Did you use it to generate new text, check your own ideas, or rewrite text?

  – The AI output was very guideful while writing the ethics statement. But most of its output needed comprehensive editing. The output was very common and it was more related to the ethics statements of works that release open-source language models and potentially raise many ethical concerns due to the gender and racial biases. Also, the ChatGPT output mostly touched on the data privacy concerns of dataset collection, which is not present in our work.

Limitations

The project was hugely dependent on GPU resources and it is acknowledged that limited resources were a major challenge throughout the project. Running experiments with eight different models that involved extensive training procedures required significant computational resources. To be able to perform analysis on many models, we could only report our results on 3 random splits. It would be statistically more reliable to do at least 5 splits. In addition, using the ChatGPT API for prompting techniques required token generation costs, and the experiments could only be performed on 200 samples. These constraints may have limited the scope of the experiments and prevented exploration of additional models or techniques. It is important to consider these resource constraints when evaluating the generalizability and scalability of the results. Furthermore, although the overall results of the transformer models were encouraging, the Morse++ model needs further development, which is one of the main contributions of this project. This admission suggests that there are potential areas where the performance of the model can be improved. In particular, it highlights the need to explore different fusing techniques for the word and context encoder outputs. If this limitation is addressed, it is possible to improve the accuracy and effectiveness of the model in capturing contextual information in sentences.

Ethics Statement

The research and work described includes the development of supervised models with a focus on transformer models for the morphological analysis of Turkish, and the use of Large Language Models (LLMs) with various prompt engineering and fine-tuning methods applied to morphological analysis and disambiguation. The work does not raise ethical considerations and potentially broader implications, as there is no data collection that could compromise user privacy. Testing of the language models is limited to morphological analysis and disambiguation of languages and cannot raise any potential unfairness, stereotyping, or discrimination in the models and results.

Transparency of the research process and behavior of the models is a priority. We strive for clear documentation, open-source code, and accessible information about the methods used. By promoting transparency and explainability, we aim to promote understanding and enable review by the broader research community. We encourage open collaboration, knowledge sharing, and engagement.

By adhering to these ethical considerations, we aim to ensure that our work on developing supervised models for Turkish language morphological analysis and using large-scale language models for downstream tasks contributes positively to the scientific community, respects user privacy, promotes fairness, does not cause potential ethical risks associated with AI technologies.

References

Acikgoz, E. C., Chubakov, T., Kural, M., Sahin, G. G., and Yuret, D. (2022). Transformers on multilingual clause-level morphology. *CoRR*, abs/2211.01736.

Akyürek, E., Dayanik, E., and Yuret, D. (2019). Morphological analysis using a sequence decoder. *Trans. Assoc. Comput. Linguistics*, 7:567–579.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language models are few-shot learners. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Chrupala, G. (2006). Simple data-driven context-sensitive lemmatization. *Proces. del Leng. Natural*, 37.

Chrupala, G., Dinu, G., and van Genabith, J. (2008). Learning morphology with morfette. In *Proceedings of the International Conference on Language Resources and Evaluation, LREC 2008, 26 May - 1 June 2008, Marrakech, Morocco*. European Language Resources Association.

Devlin, J., Chang, M., Lee, K., and Toutanova, K. (2019). BERT: pre-training of deep bidirectional transformers for language understanding. In Burstein, J., Doran, C., and Solorio, T., editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Hakkani-Tür, D. Z., Oflazer, K., and Tür, G. (2002). Statistical morphological disambiguation for agglutinative languages. *Comput. Humanit.*, 36(4):381–410.

Heigold, G., Neumann, G., and van Genabith, J. (2017). An extensive empirical evaluation of character-based morphological tagging for 14 languages. In Lapata, M., Blunsom, P., and Koller, A., editors, *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 1: Long Papers*, pages 505–513. Association for Computational Linguistics.

Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., de Laroussilhe, Q., Gesmundo, A., Attariyan, M., and Gelly, S. (2019). Parameter-efficient transfer learning for NLP. *CoRR*, abs/1902.00751.

Li, X. L. and Liang, P. (2021). Prefix-tuning: Optimizing continuous prompts for generation. In Zong, C., Xia, F., Li, W., and Navigli, R., editors, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 4582–4597. Association for Computational Linguistics.

Liu, J., Shen, D., Zhang, Y., Dolan, B., Carin, L., and Chen, W. (2022). What makes good in-context examples for gpt-3? In Agirre, E., Apidianaki, M., and Vulic, I., editors, *Proceedings of Deep Learning Inside Out: The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures, DeeLIO@ACL 2022, Dublin, Ireland and Online, May 27, 2022*, pages 100–114. Association for Computational Linguistics.

Nivre, J., de Marneffe, M., Ginter, F., Hajic, J., Manning, C. D., Pyysalo, S., Schuster, S., Tyers, F. M., and Zeman, D. (2020). Universal dependencies v2: An evergrowing multilingual treebank collection. In Calzolari, N., Béchet, F., Blache, P., Choukri, K., Cieri, C., Declerck, T., Goggi, S., Isahara, H., Maegaard, B., Mariani, J., Mazo, H., Moreno, A., Odijk, J., and Piperidis, S., editors, *Proceedings of The 12th Language Resources and Evaluation Conference, LREC 2020, Marseille, France, May 11-16, 2020*, pages 4034–4043. European Language Resources Association.

Oflazer, K. and Tür, G. (1997). Morphological disambiguation by voting constraints. In Cohen, P. R. and Wahlster, W., editors, *35th Annual Meeting of the Association for Computational Linguistics and 8th Conference of the European Chapter of the Association for Computational Linguistics, Proceedings of the Conference, 7-12 July 1997, Universidad Nacional de Educación a Distancia (UNED), Madrid, Spain*, pages 222–229. Morgan Kaufmann Publishers / ACL.

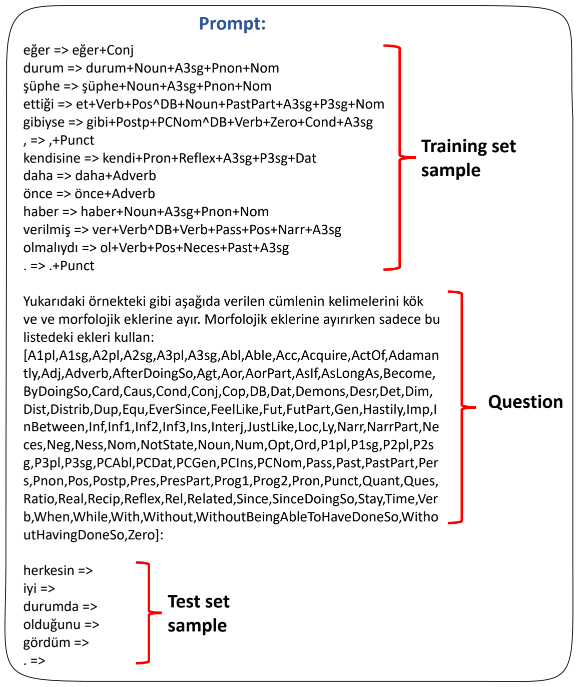
Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E. Z., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 8024–8035.

Shliazhko, O., Fenogenova, A., Tikhonova, M., Mikhailov, V., Kozlova, A., and Shavrina, T. (2022). mgpt: Few-shot learners go multilingual. *CoRR*, abs/2204.07580.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. In Guyon, I., von Luxburg, U., Bengio, S., Wallach, H. M., Fergus, R., Vishwanathan, S. V. N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J. G., Salakhutdinov, R., and Le, Q. V. (2019). Xlnet: Generalized autoregressive pretraining for language understanding. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 5754–5764.

Appendix


Training set sample
Question
Test set sample

Prompt:

```
eğer => eğer+Conj
durum => durum+Noun+A3sg+Pnon+Nom
şüphe => şüphe+Noun+A3sg+Pnon+Nom
ettiği => et+Verb+Pos^DB+Noun+PastPart+A3sg+P3sg+Nom
gibiyse => gibi+Postp+PCNom^DB+Verb+Zero+Cond+A3sg
, => ,+Punct
kendisine => kendi+Pron+Reflex+A3sg+P3sg+Dat
daha => daha+Adverb
önce => önce+Adverb
haber => haber+Noun+A3sg+Pnon+Nom
verilmiş => ver+Verb^DB+Verb+Pass+Pos+Narr+A3sg
olmalıydı => ol+Verb+Pos+Neces+Past+A3sg
. => .+Punct

Yukarıdaki örnekteki gibi aşağıda verilen cümlenin kelimelerini kök
ve ve morfolojik eklerine ayır. Morfolojik eklerine ayırırken sadece bu
listedeki ekleri kullan:
[A1pl,A1sg,A2pl,A2sg,A3pl,A3sg,Abl,Able,Acc,Acquire,ActOf,Adaman
tly,Adj,Adverb,AfterDoingSo,Agt,Aor,AorPart,AsIf,AsLongAs,Become,
ByDoingSo,Card,Caus,Cond,Conj,Cop,DB,Dat,Demons,Desr,Det,Dim,
Dist,Distrib,Dup,Equ,EverSince,FeelLike,Fut,FutPart,Gen,Hastily,Imp,I
nBetween,Inf,Inf1,Inf2,Inf3,Ins,Interj,JustLike,Loc,Ly,Narr,NarrPart,Ne
ces,Neg,Ness,Nom,NotState,Noun,Num,Opt,Ord,P1pl,P1sg,P2pl,P2s
g,P3pl,P3sg,PCAbl,PCDat,PCGen,PCIns,PCNom,Pass,Past,PastPart,Per
s,Pnon,Pos,Postp,Pres,PresPart,Prog1,Prog2,Pron,Punct,Quant,Ques,
Ratio,Real,Recip,Reflex,Rel,Related,Since,SinceDoingSo,Stay,Time,Ver
b,When,While,With,Without,WithoutBeingAbleToHaveDoneSo,Witho
utHavingDoneSo,Zero]:

herkesin =>
iyi =>
durumda =>
olduğunu =>
gördüm =>
. =>
```

Figure 8: The ”With dictionary” prompt that is used in the one-shot learning experiments with ChatGPT. The first part is the Train set sample, second part is the question that asks: ”Considering the above example, separate the words in the sample below to its root and morphological tags only by using the tags from this given set of morphological tags: [A1pl,A1sg,A2pl, ...]”, and the last part is the test sample under analysis.