A SEQ2SEQ TRANSFORMER MODEL FOR TURKISH SPELLING CORRECTION

by

Şahin Batmaz

B.S., Computer Engineering, Boğaziçi University, 2017

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Computer Engineering

Boğaziçi University

2022

ACKNOWLEDGEMENTS

For all the support and their guidance I am especially grateful to Assoc. Prof. Arzucan Özgür, Assist. Prof. Susan Michele Üsküdarlı and Assoc. Prof. Ahmet Cüneyd Tantuğ. In addition, I would like to thank to Onur Güngör, Bilgin Koşucu and Elif Oral. In many ways, they helped and guided me during my journey through NLP studies. I am grateful for their help and friendship.

ABSTRACT

A SEQ2SEQ TRANSFORMER MODEL FOR TURKISH SPELLING CORRECTION

Natural language processing (NLP) is a fascinating area of artificial intelligence. It allows humans to interact with machines through natural language. There are two main concepts in NLP model architectures, namely input vectorization and contextual representation. The input vectorization process starts with tokenization, where there are three approaches: character-level, word-level, and subword-level. Word-level tokenization results in a large vocabulary, and in agglutinative languages such as Turkish, words derived from the same stem are treated as different words. This makes it difficult for NLP models to understand their relationships and the meaning of the morphological affixes. Furthermore, all NLP models suffer from a common problem: spelling errors in the data. In case of spelling errors, the misspelled tokens become completely different and the models cannot understand them. In this thesis, a character-level seq2seq transformer model is developed for spelling error correction. To train the model, a dataset for Turkish spelling correction is created by collecting correctly spelled Turkish sentences and systematically adding spelling errors to them. Seq2seq models suffer from multiple decoding iterations and have high prediction time. To address this problem, a novel model architecture, one-step seq2seq transformer model, is proposed in which the transformer model predicts the outputs in one iteration. The proposed models are tested with the exact match criteria. The standard seq2seq model and the one-step seq2seq model achieved 68.64% and 42.69% accuracy, respectively. Finally, the standard seq2seq model makes predictions for 160 input characters in 8.47 seconds, while the one-step seq2seq model makes predictions for the same number of characters in 73 milliseconds on CPU and 28 milliseconds on GPU.

ÖZET

TÜRKÇE YAZIM HATASI DÜZELTME İÇİN SEQ2SEQ TRANSFORMER MODELİ

Doğal dil işleme (NLP), yapay zeka içerisindeki ilgi çekici bir alandır. Doğal dil aracılığıyla insanın makinelerle etkileşimini sağlar. NLP model mimarilerinde iki ana kavram vardır; girdi vektörleştirme ve girdinin bağlamsal temsil edilmesi. Girdi vektörleştirme işleminde üç tokenizasyon yaklaşımı bulunmaktadır: karakter düzeyi, kelime düzeyi ve kelime parçaları düzeyi. Kelime düzeyinde tokenizasyon yönteminde, sözcük dağarcığının geniş olması problemi yaşanmaktadır. Ayrıca Türkçe gibi eklemeli dillerde, aynı kökten türeyen sözcüklerin birbirinden tamamen farklı sözcükler olarak ele alınmasına neden olmakta ve modelin bu sözcükler arasındaki ilişkileri ve morfolojik eklerin anlamlarını öğrenmesini zorlaşmaktadır. Ayrıca, tüm NLP modellerin ortak bir sorun vardır: veride bulunan yazım hataları. Yazım hataları ile girdi kelimeleri tamamen farklı hale gelir ve model bunları anlayamaz. Bu tezde, yazım hatalarının düzeltilmesi için karakter düzeyinde bir seq2seq dönüştürücü modeli geliştirilmiştir. Model için doğru yazılmış Türkçe cümleler toplanmış ve toplanan cümlelere farklı türdeki yazım hataları sistematik olarak eklenerek Türkçe yazım düzeltmesi için bir veri seti oluşturulmuştur. Seq2seq modelleri tekrarlanan kod çözme yönteminden dolayı yüksek bir tahmin süresine sahiptir. Bu sorunu çözmek için, transformer modelinin çıktıları tek seferde tahmin ettiği, yeni bir model mimarisi önerilmiştir, tek adımlı seq2seq transformer modeli. Önerilen modeller, tam eşleşme kriterleri ile test edilmiştir. Standart seq2seq modeli ve tek adımlı seq2seq modeli sırasıyla %68.64 ve %42.69 doğruluk oranı elde etti. Son olarak, standart seq2seq modeli 160 giriş karakteri için 8.47 saniyede tahminleme yaparken, tek adımlı seq2seq modeli 160 giriş karakteri için CPU'da 73 milisaniyede ve GPU'da 28 milisaniyede tahminleme yapar.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS . . . iii
ABSTRACT . . . iv
ÖZET . . . v
LIST OF FIGURES . . . viii
LIST OF TABLES . . . ix
LIST OF SYMBOLS . . . xi
LIST OF ACRONYMS/ABBREVIATIONS . . . xii
1. INTRODUCTION . . . 1
2. BACKGROUND . . . 5
2.1. NLP . . . 5
2.2. Text Normalization . . . 8
2.3. Tokenization . . . 9
2.4. Word Embeddings . . . 11
3. RELATED WORK . . . 14
4. DATASET . . . 19
4.1. Data Collection . . . 19
4.2. Spelling Error Generation . . . 20
5. METHOD . . . 23
5.1. Standard Seq2Seq Model . . . 23
5.2. One-Step Seq2Seq Model . . . 25
5.3. Reserved Character . . . 27
5.4. Optimization . . . 27
5.5. Loss . . . 28
6. EXPERIMENTS AND RESULTS . . . 29
6.1. Standard Seq2Seq Model . . . 29
6.1.1. Training . . . 29
6.1.2. Test Dataset Results . . . 32
6.1.3. Prediction Time Performance . . . 36

6.2. One-step Seq2Seq Model 37
6.2.1. Training 37
6.2.2. Prediction Time Performance 39
6.2.3. Test Dataset Results 40
6.3. Comparison Tests 43
7. CONCLUSION AND DISCUSSION 48
8. FUTURE WORK 49
8.1. Turkish Morphological Parser 49
8.2. Turkish Language Model 50
REFERENCES 52

LIST OF FIGURES

Figure 5.1. Standard Transformer Layers. 24

Figure 5.2. One-step Transformer Layers. 26

Figure 5.3. Warmup Learning Rate. 28

Figure 6.1. Standard Seq2Seq Model Training Loss. 31

Figure 6.2. Standard Seq2Seq Model Validation Loss. 31

Figure 6.3. Standard Seq2Seq Model Warmup Learning Rate. 31

Figure 6.4. One-step Seq2Seq Model - 10 Epochs - Training Loss. 38

Figure 6.5. One-step Seq2Seq Model - 10 Epochs - Validation Loss. 38

LIST OF TABLES

Table 4.1. Example sentences from the dataset. 22

Table 5.1. Dataset character vocabulary. 23

Table 6.1. Experimented parameters. 29

Table 6.2. Standard Seq2Seq Model - Exact match accuracies based on number of typos in the sentences. 32

Table 6.3. Standard Seq2Seq Model - Token Accuracy Scores on the Prepared Test Dataset. 33

Table 6.4. Standard Seq2Seq Model Correctly Predicted Text Examples. . . . 34

Table 6.5. Standard Seq2Seq Model Incorrectly Predicted Text Examples. . . 35

Table 6.6. Standard Seq2Seq Model Prediction Durations. 36

Table 6.7. One-step Seq2Seq Model Prediction Durations. 39

Table 6.8. One-step Seq2Seq Model - Exact match accuracies based on number of typos in the sentences. 40

Table 6.9. One-step Seq2Seq Model - Token Accuracy Scores on the Prepared Test Dataset. 41

Table 6.10. One-step Seq2Seq Model Correctly Predicted Text Examples. . . . 42

Table 6.11. One-step Seq2Seq Model Incorrectly Predicted Text Examples. . . 43

Table 6.12. Sentence Accuracy Scores on the Prepared Test Dataset. 44

Table 6.13. Token Accuracy Scores on the Prepared Test Dataset. 45

Table 6.14. Token Accuracy Scores on the TestSmall Test Dataset. 45

Table 6.15. Token Accuracy Scores on the Test2019 Test Dataset. 46

Table 6.16. Error Type Based Token Accuracy Scores on the #Turki$hTweets Dataset. 46

Table 6.17. Accuracy Scores on the 100deda Test Dataset. 47

LIST OF SYMBOLS

α	Learning Rate

LIST OF ACRONYMS/ABBREVIATIONS

BoW	Bag of Words
BPE	Byte Pair Encoding
CBoW	Continuous Bag of Words
NLP	Natural Language Processing
NMT	Neural Machine Translation
OOV	Out of Vocabulary
SMT	Statistical Machine Translation
TFIDF	Term Frequency - Inverse Document Frequency

1. INTRODUCTION

Natural Language Processsing (NLP) is a subfield of Artificial Intelligence whose original goal is to enable machine-human interaction. In this context, NLP can be said to be the domain that enables machines to understand human language. This domain is used for spelling correction, summarization, question answering, information retrieval applications and so on. The full integration of the Internet into human life has made NLP studies a must and has led to developments in fields ranging from medicine to banking.

NLP is a relatively new field of computer science, but it has several cornerstones that have helped it reach its current state. The beginnings of NLP can be traced back to the 1950s. At that time, rule-based methods were prevalent, which were not very effective given the large number of rules [1]. The early studies were statistical and rule-based approaches. While there were many attempts to mimic human language, such as Parry (a program developed to pass the Turing test) [2], the technology at the time did not match this vision. The introduction of artificial neural networks (ANNs) was a very important development. The working mechanism of ANNs was developed based on working mechanism of the human brain, just as it was stated in the Hebbian rule of learning, neurons that fire together also wire together. In the ANNs, the weights between neurons are updated based on the provided data. This development can be considered as one of the important cornerstones that led NLP to its current state. Another important factor has been the increase in available data. Being able to access large corpora paved the way for developments that were previously considered impossible. These immense datasets, especially when it is labelled, provided a much needed source for training models. In addition to the development of ANNs and the availability of large corpora, Deep Learning methods were started to be used in the NLP field in 2012, which is another important cornerstone for the development of this field [1]. Thanks to these three crucial steps, it is now possible to use NLP for machine translation, chatbots, question answering, information retrieval, and so on.

Looking at the current models, it can be seen that a large corpus of text is needed for training. One of the examples is Latent Semantic Analysis (LSA) [3]. This is a text-based model developed by Thomas Landauer and colleagues. LSA is not one of the modern programs, but it is still used as a measure of the relationship between words and text. During the training process of LSA, a large corpus is used, which is divided into parts called "documents" . The words in the documents are obtained and a large matrix is created. This matrix is processed with Singular Value Decomposition to create low-dimensional vectors for each word, called word embeddings [4]. Words that are closely related to each other are given similar vectors due to their similar co-occurrences in the original matrix. Continuous Bag of Words (CBOW) is another important example [5]. Similarly, a corpus of 630 billion words was used for the training. The embedding of a word is created by making use of the surrounding words. Contextual summary is used to predict the masked words. Another important example is BERT [6]. The foremost important difference of this model is the use of a transformer based neural network. This leads to a better representation of contextual information.

In the last two decades, the Machine Learning and Deep Learning fields have had a tremendous impact on many fields, including NLP. Machine Learning is the approximation of output data with input data. Deep Learning is a subfield of Machine Learning where the models are neural networks with different architectures. Deep Learning approaches were developed years ago, but computers were not powerful enough to apply them.

In NLP, the main task is the contextual representation of input data; couple of words, sentence or paragraph. As in natural language, the vocabulary can be considered fixed, but there is no limit to the meanings that can be generated. Therefore, it is not enough to understand a word, but it is needed to understand the context.

There are several neural network architectures for modelling text context. LSTM and CNN based architectures have been used in many domains and have achieved remarkable results. However, the Transformer neural network has raised the success

scores to a higher level. BERT is the first Transformer based model that outperforms almost all NLP tasks on public datasets when released.

In every model architecture, to be able to represent input data, text must be tokenized. There are 3 different approaches to tokenization which are character, word, and subword level. With tokenization outputs as tokens, data can be provided into models.

In model architectures, text is generally provided as words into the models. At this point, spelling errors have dramatic effects on the models, because a single letter difference causes the model to consider the word as a new one. Models learn how to represent a word through examples in the dataset with the interaction of other words. Therefore, each word in the dataset should occur with sufficient frequency. When there is a spelling error, there are not enough occurrences of the word, and it cannot be understood by the model.

Turkish is a member of the Ural-Altaic language family. It is an agglutinative language, so it is possible to form several new words from one stem with a rich morphology. This case puts Turkish to a different position than English in NLP studies for spelling correction [7]. In model studies, any word derived with a new suffix is accepted as a new word. It is important to understand whether a letter is a typo or a morpheme. Spelling errors in the data reduce the performance of the models by increasing the complexity of the data.

When the data has spelling problems, the rest of the model will also be strongly affected by it. Spelling errors can be grouped under two headings; cognitive errors and typing errors. While typos are mechanical errors that occur simply by touching a wrong letter on the keyboard, cognitive errors can be phonetic in nature or result from confusing one word with another. Asciification is also an issue in Turkish, where diacritic characters, “ğüşıöç”, are replaced by “gusioc”.

There are many statistical methods for spelling correction. One of them is the noisy channel algorithm, in which the misspelled word is corrected with one of the candidate words. The words closest to the spelling are considered as candidate words. Then the most likely word is selected based on the surrounding words.

In this study, a character level seq2seq transformer model is developed for spelling correction in Turkish. This research aims to contribute to the literature in several ways and the first contribution is the use of a transformer based architecture, for which there has been no previous transformer based study. Additionally, the seq2seq model approach is used to work at the sentence level, while most spelling error correction studies do not analyze the problem at the sentence level. As the second contribution, a dataset for Turkish spelling correction is created, which can be used for further research in this area. Moreover, the model is trained on a Turkish dataset for which there is no other study on seq2seq models for Turkish spelling correction. In this respect, this study fills an important gap in Turkish NLP studies caused by the spelling correction problem. Lastly, a novel model architecture, one-step seq2seq transformer model, is proposed in which the transformer model predicts the outputs in one iteration to solve the time performance problem of the standard seq2seq transformer model.

2. BACKGROUND

2.1. NLP

Natural Language Processing (NLP) is the computational analysis of human language in text format [8]. There are lots of different tasks in the analysis of text; text categorization, part-of-speech tagging, dependency parsing, named entity recognition, relation extraction, question answering, and so on. Each of these is highly demanded by different use cases and there is a huge volume of text data for them. Even for a single use case, it is impossible for a human to read, understand, and analyse the data in a reasonable time with reliable and high accuracy. NLP aims to handle the analysis of text automatically in a short time. However, there are lots of difficulties in this process. First, the analysis of text starts with the representation of text which requires tokenization and vectorization. Then, problems about spelling errors, size, and vocabulary coverage in dataset affect the performance of NLP systems. Lastly, contextual representation of data is an another difficulty.

Natural languages consist of an alphabet of characters and a vocabulary of words. In digital environments, the text data is stored as sequences of characters. In text representation of NLP, the unit of text is called as token and the process of obtaining tokens is called as tokenization. There are different approaches in tokenization where a token can be a single character, a window of characters, a word or a piece of a word (subword). The main tokenization approach of text is the detection of words from sequences of characters since word is the unit of human languages and contains distinctive semantic information compared to sequence of characters. As words are stored with their characters, computers do not have a semantic information about words. To obtain semantic information and analyse the text documents, different approaches are developed in the history of NLP. As the statistical approaches, co-occurrence metrics are used to learn relations between words. Then word embedding approaches are used to learn semantics of words.

The vocabulary sizes of natural languages are in the scale of tens of thousands. Each word is unique and computers do not have any information about words. It is impossible to manually annotate each and every word. Therefore, the first layers of all NLP models are trained from the raw text documents by utilizing the consecutive co-occurrence of words. These raw documents are called as corpus. Quality of corpora determines the success of NLP models that is why every model's success depends on the corpus used in training.

NLP data representation models can be put into two categories; statistical models and embedding based models. Most popular statistical model approaches can be listed as n-gram, BoW, and tf-idf. Latter two are examples of vector space model approach where documents are represented with same sized vectors. These models produce efficient results in NLP tasks in terms of time and accuracy. However, the language representation capacity of the statistical models is limited. Since vocabulary size and the size of data are immense, there is always space to increase the model capacity. Word embedding approach provided a huge improvement in NLP tasks. Embeddings represent words in a high dimensional space where similar words are close to each other. The language representation capacity of word embeddings is much larger. However, training word embeddings is slower and requires large corpus that contains high number of co-occurrences per word. To solve this problem, transfer learning technique is used where word embeddings are trained on very large datasets for languages and published such as GloVe, fastext, and so on. These are called as pretrained word embeddings. Then, these embeddings can be used on a different dataset but the vocabularies of the datasets must match with each other. There are two usages of word embeddings. The first one is calculating a similarity score between two words and the second one is representing a document with embeddings of words. In the latter usage, padding is needed to have the same length for all the documents and another model is required on top of word embeddings for the representation of the document. In all the models including rule-based, machine learning and deep learning, model capacity increases from left to right, but required data size to outperform the previous approach also increases. With transfer learning of word embeddings, required data

size thresholds shift to lower values. To sum up, embedding based models can achieve better results but require larger size of data for training, require padding, and another model for document representation. On the other hand, the statistical models produce representation for documents, can work better on smaller datasets, but produce lower accuracies.

The performance of pretrained word embeddings changes with task dataset, pretraining corpus, and pretraining model architecture and parameters. Task datasets play role in the evaluation, not in pretraining. Most popular pretrained word embeddings use pretraining corpuses with size of terabytes, so the corpuses cover the language in terms of the words in vocabulary and number of co-occurrences per word. Pretraining model architectures determine the performance of embeddings. word2vec and GloVe use words in text representation and cbow or skipgram approach with feed-forward neural networks as the model and produce a vector for each word. fastext uses character n-gram approach for the representation of text. However, the meaning of a word actually changes with the surrounding words in the sentence, even paragraph. Therefore, word embedding representation of a word should not be a fixed vector and should be determined with the sentence. Additionally, cbow and skipgram algorithms ignore the order of words whereas a word is affected by the both left and right words differently (by different means), and a model should process the sentence directionality accordingly. These ideas has led to the development of contextual word embeddings.

Most common models to process the sentence directionality as a sequence are recurrent and convolutional neural networks. ELMo uses bi-directional LSTM layer in the model. Attention mechanism took place in the encoder-decoder architectures of recurrent and convolutional neural networks and improved the results. With the publication of Transformer structure, a new model based solely on attention mechanism appeared as an alternative sequence processing model which outperformed other models in some NLP tasks. Another issue is the text representation. There are two main concepts; meaning in the token and the number of unique tokens. Token as a single character approach might have around 30 alphabet character but has disadvantage of

losing word meaning. Using words as tokens keeps the meaning however the vocabulary sizes are more than 100 thousands and it is a burden on the model since the model diverges to cover all the words and requires enough examples for each word. Subword based approaches might have more optimal data representations with the relationship among words having similar subwords. GPT uses byte pair encoding (BPE) algorithm for the representation of text as subwords and uses position-wise feed-forward Transformer neural network as the model. In late 2018, BERT was released. BERT uses Wordpiece as the text representation and bidirectional Transformer neural network as the model. BERT dominated most of the state of the art results in NLP tasks in English.

A comprehensive Transformer model on Turkish data is needed for Turkish NLP studies. However, the structure of language is different from English since it is agglutinative. The idea behind Wordpiece algorithm used in BERT can be adjusted to Turkish such that common subwords will be obtained with morphological tokenization since Turkish is a morphologically rich and an additive language. To achieve a successful morphological tokenization, a text normalization model is needed because morphological analysis is also sensitive to typos. Therefore, text normalization plays a critical role in solving spelling errors for other studies.

2.2. Text Normalization

Text normalization is a fundamental area of NLP. There are different kinds of problems in this area; substitute, delete, insert, swap, deasciification, vowelization, repeat. There have been different approaches to solve these problems. Language models with statistical n-gram approaches have limited capacity. There are different approaches in building probabilistic language models. Zemberek [9] has a text normalization functionality with a 2-gram language model. There are also some approaches with deep learning architectures. Lourentzou [10] developed a seq2seq model with RNN architecture.

2.3. Tokenization

In every NLP model, text processing starts with tokenization. The main approach is to split text into list of words. The other approach is subword segmentation. There is a trade-off between approaches; word-based models are more efficient in terms of training since the length of input sequence is smaller but subword based models are more flexible in terms of handling OOV and rare words [11].

On the other hand, the size of vocabulary is another trade-off. It generally increases through subword-based approach to word-based approach. Increased vocabulary size allows words to be expressed more distinctively but requires more training data, more training iterations and more model capacity.

Using subword segmentation is common especially in NMT models since the effect of OOV and rare words are higher. Using a fixed word vocabulary prevents the model to produce unseen words where the translation is an open-vocabulary problem [11, 12]. NMT models with subword segmentation in text representation can solve open-vocabulary translation. Moreover, the problem of OOV and rare words is same for all NLP models that are based on a fixed word vocabulary.

Subword segmentation approaches are developed with different strategies. Character based tokenization is the extreme point where there is no word concept in tokenization; however, it is not efficient because it loses all word information and puts the burden of expressing words to neural network and it produces worse scores than word-based models [13, 14]. Vanilla character-n-gram is another approach. More common approach is to determine the subword vocabulary with information from training dataset. BPE [12] and WordPieceModel [11, 15] algorithms are the main examples of this approach.

BPE subword segmentation [12] is developed by following the idea of Byte Pair Encoding data compression [16]. The vocabulary is initialized with the list of available

characters and an extra end of word symbol. In each iteration of the algorithm, frequencies for all pairs of vocabulary elements are obtained from the dataset and then the most frequent pair is added to the vocabulary and replaced into a single element in the dataset. The pairs that cross word boundaries are not included in the algorithm. The only hyperparameter is the threshold for the vocabulary size where the algorithm continues to merge elements until the desired vocabular size is achieved.

WordPieceModel also uses dataset to determine the subword vocabulary. It is originally developed for the voice search system for Japanese and Korean languages where there are large number of homonyms, and a word segmenter is needed since the concept of spaces between words usually does not exist. Similar to BPE, the vocabulary is initialized with the list of available characters. In the merge operation of vocabulary elements, instead of merging the most frequent pair, a language model is trained on the dataset and then the pair that would increase the likelihood in the dataset the most when added to the language model is merged. The iteration is repeated until the desired vocabulary size is achieved.

Another subword segmentation approach is developed with unigram language model [17]. The algorithm produces multiple subword segmentations with probabilities. To be able to build a language model, the vocabulary is needed to be pre-determined. Opposite to previous studies, vocabulary is initialized with big seed subwords from the dataset. In each iteration, for each subword, how much the likelihood would be reduced when removing the subword is calculated. Lowest n%, for example 20%, of subwords are removed from the vocabulary. The iteration is repeated until the desired vocabulary size is achieved. Single characters in the vocabulary are excluded from this process. With the produced vocabulary, a language model is produced for the dataset. For a given text, with the language model, the most probable segmentation can be obtained. Additionally, an approach called subword regularization [17] is developed where multiple subword segmentations are sampled from the language model and used in a task like NMT.

Last perspective of the tokenization is the opposite of subword segmentation, which is learning phrases where a phrase is a sequence of consecutive words. In the language, there are many phrases that have a meaning which is not a composition of the meaning of individual words [13]. For example, "New York Times" should be represented as single token. A simple approach is to find sequences of words where the ratio of a sequence frequency over individual word frequencies is high.

2.4. Word Embeddings

Distributed representations of words in a vector space, generally called as word embeddings, improve learning algorithms to achieve better performance in NLP systems. The research area has been proposed long ago in [18]. And exemplary model is proposed in [19] as probabilistic feed forward neural network language model.

Word embeddings are learned from a dataset. The straightforward approach is to train an end-to-end model from a task specific dataset and use word embedding layer in the first step; however, task datasets have comparably smaller sizes where the vocabulary size of the dataset does not cover the language enough and the dataset does not contain reasonable word frequencies. Even if the task dataset is large enough, training has high computational cost. For these reasons, as the main approach, word embeddings are trained on a large corpus in an unsupervised manner and then used in task datasets as representations of words. With this approach, word representations are called as pretrained word embeddings. With such pretrained embeddings, high performing models can be trained on small task datasets where pretrained embeddings contain rich information about the language.

In 2013, two novel models for computing word embeddings on large datasets, cbow and skip-gram, are proposed in [5] which gave rise to pretrained word embeddings thus breakthrough improvements in NLP systems. The models use simple feed forward neural networks where the input layer is the words and the hidden projection layer is the embeddings to be learned. The cbow model is short for continous bag-of-words.

In the cbow model, current word is predicted based on the surrounding words. In the skip-gram model, surrounding word are predicted from the current word. The implementation of these models is named as and provided in word2vec [13].

GloVe [20] is another word embedding approach. It claims that the most common two training methods global matrix factorization like LSA and local context window models like skip-gram have drawbacks where the former performs poorly on word analogy task and the latter poorly utilizes the statistics of the corpus. GloVe proposes a new regression model that trains on global word-word co-occurrence counts and consistently outperforms word2vec.

As for the word embedding approaches, word2vec, GloVe and fastText can be categorized as lookup dictionary based word embedding where there is a fixed vector for a given word, the vector does not change with the surrounding words of the given word. To improve the performance of word embeddings, contextualized word embedding models are developed where a sentence is given as input and word embeddings for each token are generated with a model from the sentence.

ELMo [14] is a successful example of contextualized word embeddings and published in 2018 February. As the text representation, ELMo uses characters as tokens and bidirectional LSTM layers as the model. To use it on a task, the model weights are freezed and additional layers on top of it are trained on the task dataset. When published, it outperformed the state of the art results.

In [21], a new architecture Transformer is introduced as an alternative to common CNN and LSTM layers in NLP tasks. Shortly after ELMo, in 2018 July, GPT [22] contextual word embedding model is published where BPE is used as the token representation and deep transformer decoder neural network as the model. GPT outperformed other models on most of the NLP tasks. Later, GPT-2 [23] is published in 2019. Shortly after GPT, in 2018 October, BERT [6] contextual word embedding model is published where WordPieceModel is used as the token representation and

deep transformer encoder neural network as the model which processes the sentence bidirectionally but is not auto-regressive on the other hand. Since the publication of BERT in 2018 October, many other adjustments and models have been appearing where they are mostly variations and ensembles of Transformer, GPT and BERT models like AlBERT, RoBERTA, eRnie, GPT-2, TransformerXL, XLNet as general purpose models, NCBIBERT, BioBERT as domain specific models, DistillBERT, TinyBERT as small-sized models; however, in this project, BERT itself is used because it is the baseline of the current state of the art models in terms of text representation and model architecture.

3. RELATED WORK

Throughout the years, there have been many different approaches on the spelling correction task. Early studies are designed with statistical and probabilistic approaches. With the developments in deep learning studies, deep learning based approaches have become more preferable. The model architectures work on character level inputs. Models try to learn the context around the spelling errors to be able to detect and fix them.

Oflazer [24] studied spelling correction in agglutinative languages. Given an input word, Oflazer uses q-gram approach to have candidate root words. Then, a morphological generator is developed that generates words from candidate root words. During this process, edit distances between generated words and the misspelled word are limited to a threshold. Using morphological generator ensures that predicted words are valid. As ranking, minimum edit distance is used and when distances are similar, edit types are used for selection. Oflazer brought the spelling correction of agglutinative languages to an important milestone; however, there are several drawbacks of the approach. First, it only operates on the misspelled word and does not make use of any other contextual word. Second, it uses edit distance metric as the main mechanism, which ignores the semantics. Therefore, selection from candidate words would result in wrong word.

In 2014, while there were not many precedent studies questioning vowel and diacritic restoration in Turkish, Adalı [25] proposed an approach for this problem. The model architecture contains two parts, a discriminative sequence classifier and language validator. The first part is the model that restores the spelling errors. In the model, CRF mechanism is used to detect errors and predict correct characters. As the model input, target word which is subject to spelling errors is given to the model in character level along with neighbour words. In the second part of the project, predicted words are provided into a morphological analyzer as the language validator to verify that the predicted word is valid. As the morphological analyzer, two-level morphological analyzer by Şahin [26] is used. Compared to this research, the study of Adalı focuses

on two subtasks of spelling error correction and it operates on target words with its neighbours as context; however, this research focuses on wider range of spelling error correction and as the input, it works on sentence level.

Torunoğlu [27] proposed a cascaded approach to spelling error correction. There is a tokenization layer at first and there are two main tasks in this approach, ill formed word detection and candidate word generation. Ill formed detection is controlled via an abbreviation list and a morphological analyzer. For the candidate word generation, it is stated that there are 7 normalization steps; letter case, rules & lexicon lookup, proper noun, deasciification, vowel, accent and spelling. After each normalization step, produced words are checked by a morphological parser and if a word is valid, it is used as the final prediction. The architecture is tested on manually annotated tweets. This approach operates on word level and does not use contextual information. In 2017, Eryiğit [28] proposed an extended version of Torunoğlu [27]. As the spelling corrector, an adapted version of the approximate string search algorithm is used as the error model with a unigram language model to generate and rank candidate words.

Tursun [29] proposed two models for spelling correction of Uyghur; noisy channel model and neural machine translation. An LSTM is used with a source length of 30 is used for the neural network architecture. As for the dataset, news data is crawled from websites and synthetic data is obtained by adding random errors. Both models achieved similar test results. Goker [30] proposed two approaches for Turkish spelling correction; candidate word generation with word embeddings and a neural machine translation model with LSTM. The latter model achieved better results in tests.

Etoori [31] proposed a spelling correction model in 2018. As the authors stated, there was no prior work for Indian languages. As the model architecture, a character level sequence to sequence model is developed using LSTM with Attention mechanism. As the input, sequence length of 50 is used. There are some key points in the model architecture; model operates on character level as it should as the task requires, a sequence to sequence model architecture is developed since the task can be studied as

a translation, a deep learning architecture is designed to be able to understand the context and decode with correct characters. These points are common solution approaches to the spelling error correction problem. Additionally, input sequence design is a parameter in the model architecture. Using target words with neighbours as the input sequence limits the model architecture in the data part. By providing longer sequences like 50 characters or whole sentences, model has to learn more; however, this should be subject of designing model capacity and should not be a limitation to a solution. In addition, sequence length is also a parameter. Having a fixed size like 50 characters or applying padding to training batches would affect the model.

In 2019, Çolakoğlu [32] proposed a contextual model architecture with fewer processing steps and compared SMT and NMT approaches. In general, SMT can perform well even with a smaller amount of data; however, NMT models require large datasets. In the model, spelling error correction is applied at the sentence level and uses contextual information. The proposed SMT model performed better than [28] and the NMT model in the test results.

In one of the recent studies, Büyük [33] proposed a character level seq2seq model for Turkish spelling error correction. As the model architecture, LSTM with Attention mechanism is used. As the dataset, Turkish sentences are collected from Internet, like newspapers. Synthetic errors of substitution, deletion, insertion and swapping are added and 4 million sentences are obtained. As the task, words not in the vocabulary are detected and spelling correction is applied on them. As the input, misspelled word, three consonants of previous word and three consonants of next word are userd. For example, in the sentence “derste oman okuyor”, “drs oman kyr” is given to the model as input and “roman” is expected as the output. This study is one of the recent Turkish spelling error correction studies with a recent model architecture. However, it has several drawbacks compared to this research. First, Transformer architecture is a newer technology with increased learning capabilities in the NLP models. Transformer architecture is based on the Attention mechanism and able to model contextual information of the input much better than LSTMs. Secondly, model directly focuses

on words not in the vocabulary and uses a limited contextual information of three consonants from previous and next word whereas in this research, whole sentence is given as the context. In terms of data preparation, procedures are quite similar. Both studies collected correctly written sentences from internet and introduced synthetic errors to sentences. Although, Büyük proposed a good approach to the problem but the scope is to find the best valid word in terms of edit distance with a word as the input; however, this research focuses on the spelling correction problem from a much larger perspective, it operates on sentence level and fixes error contextually without edit distance limitations.

Guinard [34] proposed a cascaded model architecture for Turkish spelling error correction using Wikipedia edit history data as the dataset. As the types of spelling errors, diacritics, capitalization, spacing and insert, delete, substitute, swap categories are detected. In the model architecture, there are two main blocks; word fusing and single word correction. In the word fusing part, two consecutive words are given and word fusing decides whether to merge them into one word or not. In the single word correction part, there is a pipeline of several steps; deasciification, capitalization correction, word splitting, noisy channel model, ranker. In deasciification, a unigram model is used. In capitalization, also, another unigram model is used. For word merging and splitting, SUMLM [35] is used which uses ngram statistics and their probabilities. In the noisy channel model, error model is trained with processed word pairs and as language model, stupid backoff ngram model is used. In cascading steps, candidates are generated for next step with Damerau-Levenshtein distance 2. As the final step, SUMLM model is used as ranker. Guinard experimented and compared this architecture; however, there are several drawbacks in it. With the cascaded model approach, every step depends on each other since an error in one step would directly cause another error in the next step cumulatively. Word fusing and splitting steps are unable to process multiple space operations at once. All of the steps are developed with statistical and probabilistic approaches, which is not optimal because their capacity is limited and not flexible compared to deep learning architectures. Guinard stated that machine translation models are not promising for Turkish spelling correction. How-

ever, when dataset and model architecture are well designed and optimized, NMT and deep learning architectures can do everything statistical and probabilistic models can do and more.

Kuznetsov [36] proposed a transformer based architecture for spelling correction. The model is trained on English, Arabic, Greek, Russian, Setswana. The study also focuses on generating realistic typos in the dataset. Gao [37] proposed HCTagger for short text spelling correction. A model that uses character level embedding and bidirectional LSTMs is developed. As the output, it predicts edit labels; keep, delete, replace, append. Schmaltz [38] proposed a character level sequence to sequence network that uses LSTM with attention in encoder-decoder blocks.

To sum up, there have been several studies on the spelling error correction task. Models work on character level to fix the spelling errors. The target of the models is to learn the context from input characters and predict corrections. As the model approaches, there are statistical, correction classifier, and sequence to sequence approaches. In most of the studies, target words and surrounding words are given as the input. As the model architectures, noisy channel, HMM, LSTM, Attention mechanisms are used. In this research, spelling error correction for Turkish is studied with a character level sequence to sequence model with Transformer based architecture where whole sentence is given to the model as context.

4. DATASET

In order to train a seq2seq model, a large dataset of sentence pairs is needed where a source sentence has spelling errors and the corresponding target sentence is the correctly written version of the source sentence. Since the model will be trained from scratch without any Turkish knowledge, the dataset must contain a large portion of the vocabulary. Also, the number of occurrences of each word must be high enough so that the model can learn the composition of words from its characters through repetition. These conditions require millions of data pairs.

Manually labeling sentences is not an option because labeling millions of sentences would take too long and is not feasible. Therefore, obtaining a correct sentence from an incorrect sentence is not possible, but an incorrect sentence can be generated from a correct sentence systematically.

4.1. Data Collection

The first part of dataset preparation is to find correctly written Turkish sentences. Since correctness of sentences is a requirement, social media based text can not be used. Correctly written sentences can be obtained from formal sources such as books, newspapers, etc. Wikipedia is a great data source since it is large and formally written. A Turkish article dump of Wikipedia is obtained where it contains 310K articles. Fixy [39] project published several datasets for spelling correction. Datasets for “de”, “mi” and “ki” are obtained, but only “de” dataset is used. Several public book pdfs are obtained.

Wikipedia and pdf data are represented as blocks of text, not sentences. To split them into sentences, spacy [40] python package is used. After processing data into sentences, there are 4.35M, 3.5M and 0.44M sentences from Wikipedia, fixy “de” and book pdfs, respectively. Since sentence length is a parameter that model must cope

with, there should be a limit to sentence length to avoid complexity from unnecessarily long sentences. For this purpose, 80 characters is selected as the limit and sentences longer than 80 characters are filtered. Then, within each Wikipedia article, book and fixy "de" component, the dataset is split into training, validation and test sets. The sizes of the dataset are limited for processing time reasons and is 1M, 76K and 5K sentences, respectively.

4.2. Spelling Error Generation

Spelling errors are highly common in every human written text. It is unavoidable since human knowledge and focus are variable and unreliable. Spelling mistakes can be considered in 4 types; insert, delete, swap and replacement.

- insert : Add a new character to word
- delete : Remove a character from word
- swap : Swap two characters of word
- replacement : Replacing a character of word with another character

When human mistakes are analysed, there are several specialized spelling errors; vowel removal, diacritic replacement, and character repetition.

- vowel removal : Removing vowels from word
- diacritic replacement : Replacing Turkish diacritic characters; "ğüşıöç" - "gusioc"
- character repetition : Repeating same character of word

nlpaug [41] python package supports spelling error generation for insert, delete, swap, replacement types. It randomly selects words from given sentence with a word percentage parameter, and randomly applies spelling errors in words with given character percentage parameter. When a sentence with 7 words is given and word percentage is 0.3, it randomly selects 2 words. For insert spelling error generation, when a word has 7 characters and character percentage is 0.3, it randomly inserts 2 characters. Rest

of the spelling error types are implemented with the infrastructure of nlpaug. Additionally, for replacement type, nlpaug provides “keyboard” feature where new character is selected from original character’s neighbour characters in the keyboard. For example, in word “gemi”, if character “g” is randomly selected, it could be replaced with one of f,v,b,h,y,t.

In Turkish spelling errors, there are common mistakes on specific suffixes mostly in verbs such as -yor, -ecek, -alım. For example, the word “geleceğim” is written with spelling errors as “gelcem”. Spelling error generation functions are developed for this type of errors.

In sentences with spelling errors, there can be more than one spelling error type. Therefore, in spelling error generation, combinations of errors are applied. Vowel delete, vowel repeat, consonant repeat, character swap, ascii character replacement, keyboard distance character replacement and other error generation functions are randomly applied to sentences. Several examples from the prepared dataset are provided in Table 4.1.

Table 4.1. Example sentences from the dataset.

#	**Type**	**Sentence**
1	input	Öyle kı, çok geçmeden ortalıkttta bu sesten başka bir şey duyulmaaz oldddu.
	true	Öyle ki, çok geçmeden ortalıkta bu sesten başka bir şey duyulmaz oldu.
2	input	Istsyon asağıdaki anlamlara geleblir
	true	İstasyon aşağıdaki anlamlara gelebilir
3	input	Bunlarin yanı sıraaaa kitap ve müzik yayıhı da ypar.
	true	Bunların yanı sıra kitap ve müzik yayımı da yapar.
4	input	Köyün gleenek, gorenek ve yemkleri hakoında bilgi yktr.
	true	Köyün gelenek, görenek ve yemekleri hakkında bilgi yoktur.
5	input	Ortasindannn bir de nehir geçiyorddddu.
	true	Ortasından bir de nehir geçiyordu.
6	input	Özellikle vişne yetiştiriciliği köy ekonmisi icin önemllli bi yere samiptir.
	true	Özellikle vişne yetiştiriciliği köy ekonomisi için önemli bir yere sahiptir.

5. METHOD

For spelling correction, a character level seq2seq transformer model is developed. There are encoder and decoder blocks in the model. Sentences with spelling errors are used as the input and correct sentences are expected as the output.

The vocabulary prepared in this research is the character set of the dataset. The size of vocabulary is around 85 depending on the dataset. The details of the character vocabulary are provided in Table 5.1. Every character is provided as a token to the model. The input sentence is splitted into characters, mapped into index values as the preprocessing part, and then index values are provided into embedding layer of the model.

Table 5.1. Dataset character vocabulary.

Vocabulary Characters	**Description**
A-Z	Upper case letters
a-z	Lower case letters
ğüşıöç-ĞÜŞİÖÇ	Turkish diacritic characters
‘ ’	Empty space character
.,!@#$%^&*’()_+=-”	Punctuation characters

In the vocabulary preparation, lowercase normalization and punctuation removal are not applied on the dataset so that model can learn case and punctuation correction; however, the dataset must contain enough examples for comprehensive corrections.

5.1. Standard Seq2Seq Model

In a seq2seq model, there are encoding and decoding blocks. In this research, transformer encoder and transformer decoder architectures are used in the blocks. Pytorch [42] transformer encoder and decoder layer implementations are used.

In standard seq2seq decoding, the model predicts one output token at a time. Therefore, it makes n predictions for an output. Misspelled sentence is processed through embedding layer, positional encoding and encoder block. Then encoder output is provided into decoder block. Decoder input characters are also processed through embedding layer and positional encoding. At first step, besides encoder output, only start token is provided into decoder as input and first output token is predicted. Then decoding is repeated. Previously decoded k elements are provided to decoder as input. By using encoder output and k previous predictions, decoder predicts the $(k+1)$th element. This process continues until end condition is obtained or maximum length is reached.

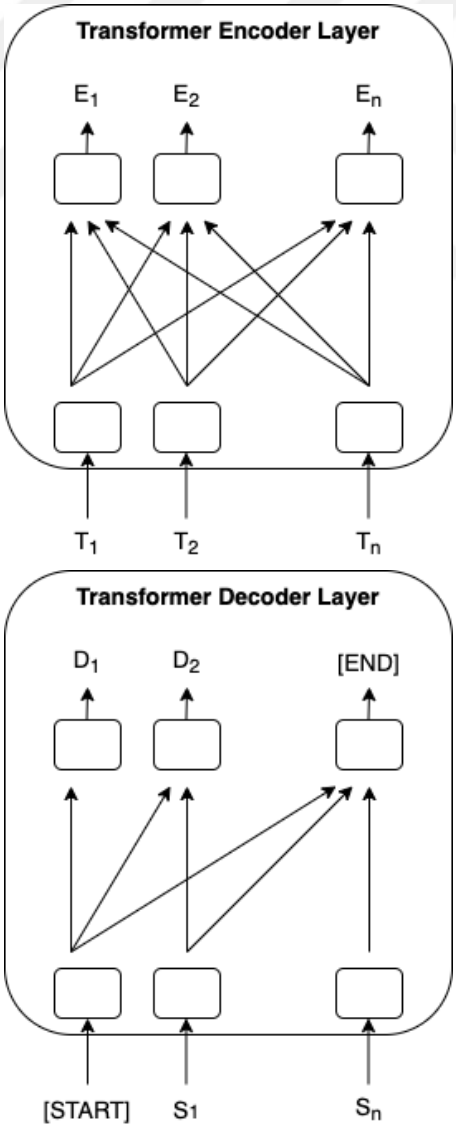

Figure 5.1. Standard Transformer Layers.

In Transformer architecture, training is parallelized where shifted output is used as decoder input and decoder is masked. Decoder masking prevents decoder from mapping nth input to $(n-1)$th output. As shown in Figure 5.1, S1 and D1 are same tokens however S1 does not connection to D1. Therefore, each input can be processed at one step. However, in prediction, the model still makes n predictions.

5.2. One-Step Seq2Seq Model

In this approach, there is no iterative decoding. All output elements are predicted at once. Decoder masking is not used. Since there is no masking and no iterative decoding, any output information could not be provided into decoder block. Therefore, only positional encoding vectors are used as the decoder input. Decoder block is expected to learn all the characters through information from the encoder block.

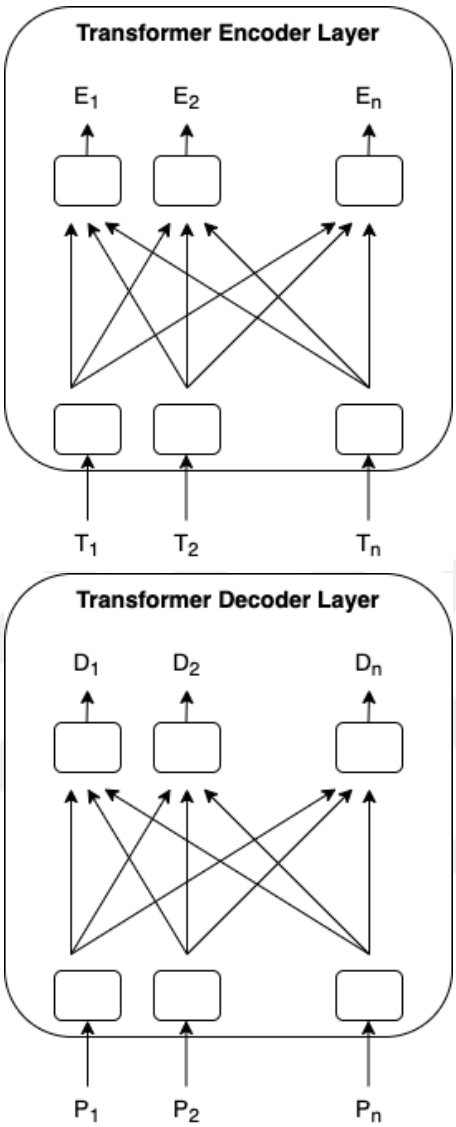

Figure 5.2. One-step Transformer Layers.

Compared to Standard Seq2Seq model, there is an additional complexity, output alignment. This model suffers from wrong aligned output predictions during training. In a 100 character output, if model misses 10th character, remaining 90 characters become different than expected characters and produce a huge loss. Another case is if model misses a character, it tries to compensate it with another wrong character. On the other hand, it can predict much faster in time performance because it predicts outputs at one step and can utilize parallel execution in GPU. Also batch predictions can be done.

5.3. Reserved Character

In language, words defined by a regexp do not contain the characteristics of vocabulary and language rules. Hashtags, mentions, URLs, emails have a specific format. Proper names like personal names or places also do not have the characteristics of vocabulary. These words can be detected with regexp patterns and lookup tables and can be replaced with a reserved character so that the model does not process these words at all. For this reason, the reserved character "¶" is introducted into the model. The character "¶" is inserted into input and output sentences in some training examples and the model is expected to leave this character untouched. During the testing phase, matching words in the input are replaced with "¶" in the preprocessing step, and "¶" characters in the output are mapped backed to the original matching words in the postprocessing step. With this approach, the model is freed from the complexity of regexp patterns and lookup tables, and these reserved words are guaranteed to be preserved.

5.4. Optimization

As for the optimization, Adam optimizer is used. Similar to BERT, warmup and decay learning rate scheduling is used. For the 10% of iterations, learning rate is incremented from 1e-6 to 1e-4 as warmup, then learning rate is linearly decreased back from 1e-4 to 1e-6. The visualization of warmup learning can be seen in Figure 5.3.

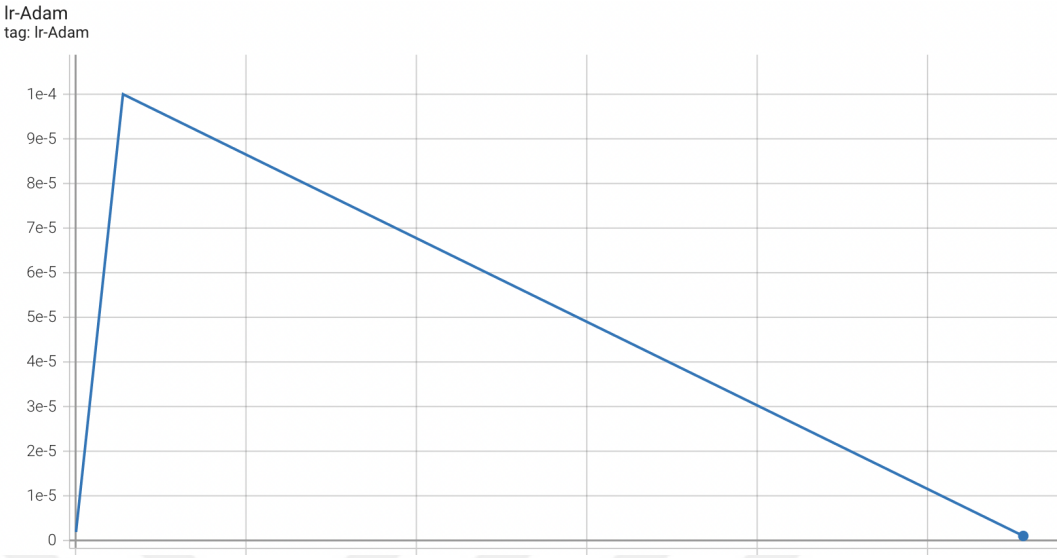

Figure 5.3. Warmup Learning Rate.

5.5. Loss

In input sentences, even if there are misspelled characters, most of the characters are correctly written. Therefore, for a large portion of the task, model just learns to copy input character to output character. Since the model can certainly learn to copy characters, the objective function can be optimized in favor of misspelled characters. To improve the objective function such that model can focus more on areas around misspelled characters, alignment weights are assigned to each output character based on their alignment with input. For example, if input and output characters are same, the weight is 1, and if they are different, the weight is 5. Additionally, neighbour characters of character with a weight of 5 are assigned with weight of 2.

6. EXPERIMENTS AND RESULTS

For the training studies, GeForce RTX 2080 Ti (12GB) and GeForce RTX 3090 (24GB) are used.

6.1. Standard Seq2Seq Model

6.1.1. Training

There have been many model training experiments with different configurations to find the optimum parameters. Transformer encoder and decoder layer counts, transformer size, learning rate, epoch size and batch size are main parameters. Also warmup learning rate and weighted loss calculation approaches can be subject to experiments, but they are not compared.

For the parameter search, one main concern is the size of the model occupied in the GPU. In early studies, experiments were conducted with a capacity of 12 GB GPU. Later, a GeForce RTX 3090 is accessed. The parameters are adjusted accordingly. Experimented parameters can be seen in Table 6.1.

Table 6.1. Experimented parameters.

Parameter	**Values**
transformer encoder layers	3, 6
transformer decoder layers	3, 6
transformer size	32, 64, 128, 256, 512, 1536
batch size	8, 16 and 64
learning rate	1e-4

As a part of deep learning model training, modeling capacity and overfit concepts are analysed. Since the size of dataset is large enough with 1M sentences, it is unlikely

for the model to overfit. Because there are so many different examples of spelling errors on many words and if model can learn to fix them in training dataset, it is probable for model to perform well on other sentences. However, the main problem is having enough capacity so that model can fix spelling errors in the training dataset.

In the first experiments, to analyse and make sure about the model capacity, only random 20k sentences are used as the training dataset. Therefore models with different parameters are experimented on small dataset. With these models, model predictions are observed for sentences in training dataset and validation dataset. From the observations, model parameters are determined.

Loss values are visualized for batch steps with tensorboard. From the experiments, it is observed that lower batch size values 8 and 16 result in large fluctuations in the loss values. Therefore, higher batch values are preferred. As for the transformer size, model predictions are observed. Up to the transformer size of 512, model capacity is not enough to model even 20k data. With transformer size of 512, model could make promising predictions on training and validation datasets. This situation shows that, to increase the model capacity, transformer size can be increased to values such as 768, 1024 and 1536. However, model memory size occupied in GPU increases in quadratic with the transformer size, and the training time also increases accordingly.

In the final model, 3 transformer encoder layers, 3 transformer decoder layers, 1536 transformer size, 64 batch size and 1e-4 learning rate parameters are used. Also, warmup learning rate and weighted loss approaches are applied in the model. The model is trained for 10 epochs on GeForce RTX 3090 and training took 6 hours. In Figures 6.1, 6.2 and 6.3, training loss, validation loss and learning rate are provided.

From Figures 6.1 and 6.2, it is observed that model converged enough, however increasing transformer size and epoch size might lead to better results. However, due to hardware restrictions, these results are obtained.

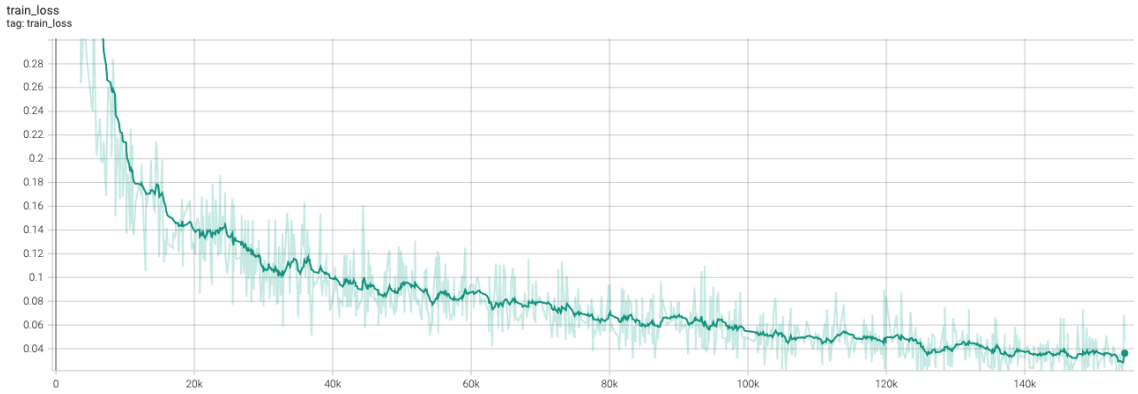

Figure 6.1. Standard Seq2Seq Model Training Loss.

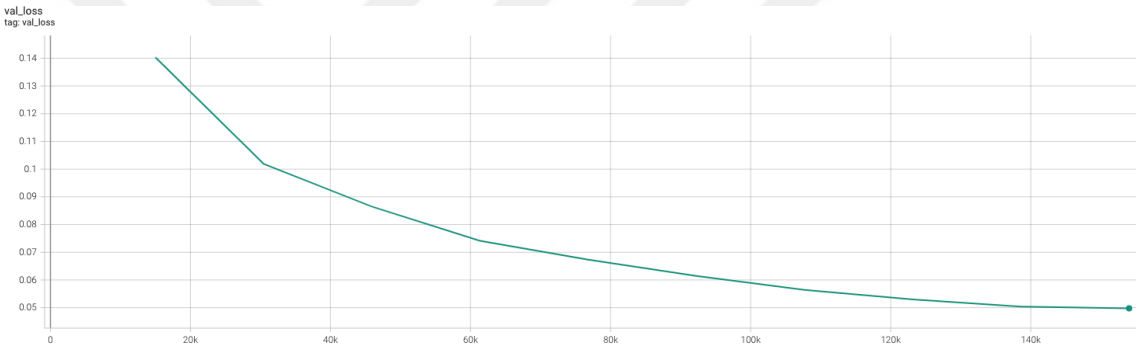

Figure 6.2. Standard Seq2Seq Model Validation Loss.

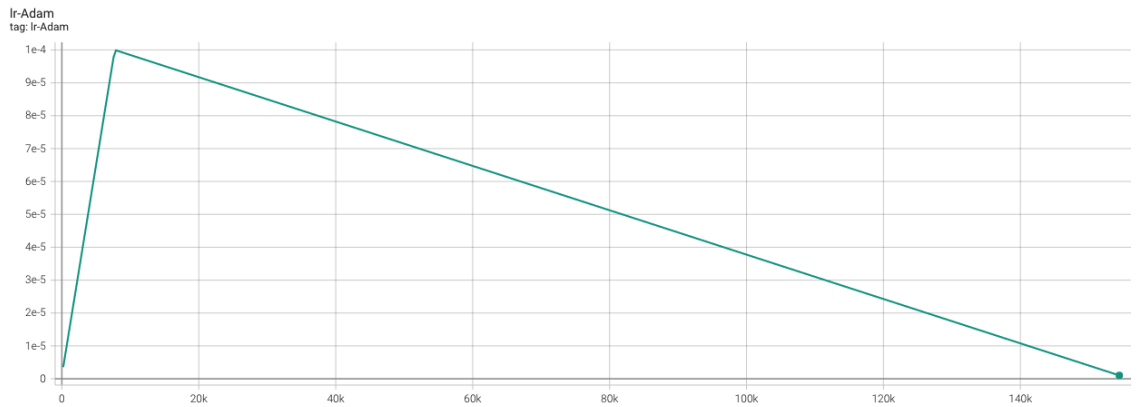

Figure 6.3. Standard Seq2Seq Model Warmup Learning Rate.

6.1.2. Test Dataset Results

In sentences with several spelling errors, for some examples, the model might be able to fix some part of the errors. However, to show numerical scores, exact match metric is used. This means that if there are 3 spelling errors in the sentence and the model only fixes 2 of them, the sentence will be evaluated as wrong.

For the test dataset, there are 3001 examples. Model predictions are obtained for these examples. As the result, 2060 examples are correctly predicted and exact match accuracy of 68.64% is achieved.

Table 6.2. Standard Seq2Seq Model - Exact match accuracies based on number of typos in the sentences.

# of Typos	**# of Sentences**	**Accuracy**
0	38	71.05%
1	227	73.57%
2	601	71.21%
3	789	69.20%
4	662	70.99%
5	401	63.34%
6+	283	59.36%
All	3001	68.64%

In Table 6.2, detailed exact match scores are provided based on the number of spelling errors in the sentences. Since exact match scoring is used, even one character error in the prediction results as false prediction. Therefore, it is expected that, as the number of spelling errors increases in the input sentence, exact match accuracy of the model decreases.

Two cases that stand out from the test results are input sentences without errors and sentences containing one character of spelling error. On sentences without spelling error, the model has 71.05% exact match accuracy score, which means that model corrupts the correctly written sentence with 28.95% of the time. This scenario is not expected from a spelling correction model. Also, the model has 73.57% exact match accuracy on sentences with one character spelling error, which is a direct score since there is only one spelling error and it is promising but needs improvement.

On the test dataset, token based accuracy metrics are calculated for error types. There are 26967 tokens in the test dataset and as the result, 95.49% of the tokens are correctly predicted. In Table 6.3, detailed token based match scores are provided based on the error types in the tokens.

Table 6.3. Standard Seq2Seq Model - Token Accuracy Scores on the Prepared Test Dataset.

	# of Tokens	**Accuracy**
Vowel Delete	2337	84.72%
Vowel Repeat	1322	97.58%
Consonant Repeat	1295	96.99%
Swap	787	91.36%
Ascii	3577	94.74%
Replacement	749	77.57%
Other	231	99.13%
Correct	16669	97.83%
All	26967	95.49%

Table 6.4. Standard Seq2Seq Model Correctly Predicted Text Examples.

#	Type	Sentence
1	input	Bu geniş toprklrı sürebilmek içinsssse ağır sabanalr taşıyan okuzler kullnıldı.
	true	Bu geniş toprakları sürebilmek içinse ağır sabanlar taşıyan öküzler kullanıldı.
	pred	Bu geniş toprakları sürebilmek içinse ağır sabanlar taşıyan öküzler kullanıldı.
2	input	Tropik bölgelwrde, koyunlar dha cok derisi icin yetiştiriliir.
	true	Tropik bölgelerde, koyunlar daha çok derisi için yetiştirilir.
	pred	Tropik bölgelerde, koyunlar daha çok derisi için yetiştirilir.
3	input	Dha sonraları bu aletleriin yapmında bkır kullanildi.
	true	Daha sonraları bu aletlerin yapımında bakır kullanıldı.
	pred	Daha sonraları bu aletlerin yapımında bakır kullanıldı.
4	input	Camının kpisi batıda yer almakta oluuup, bir yonca yprğı şeklindedir.
	true	Caminin kapısı batıda yer almakta olup, bir yonca yaprağı şeklindedir.
	pred	Caminin kapısı batıda yer almakta olup, bir yonca yaprağı şeklindedir.
5	input	Bu süreçfe birkac küçük grup da ortya çkmıştır.
	true	Bu süreçte birkaç küçük grup da ortaya çıkmıştır.
	pred	Bu süreçte birkaç küçük grup da ortaya çıkmıştır.
6	input	Deniz seviyesinden yuksekligiii 885 metredir.
	true	Deniz seviyesinden yüksekliği 885 metredir.
	pred	Deniz seviyesinden yüksekliği 885 metredir.

In Table 6.4, several examples that are correctly predicted with the standard seq2seq model are provided. These examples show that model has potential to be used for spelling correction for the error types provided in the dataset.

Table 6.5. Standard Seq2Seq Model Incorrectly Predicted Text Examples.

#	Type	Sentence
1	input	Popunn krlı hakkında yapilan tartışmalar bununlaaaa da sınırlı klmadı.
	true	Popun kralı hakkında yapılan tartışmalar bununla da sınırlı kalmadı.
	pred	Popun kralı hakkında yapılan tartışmalar bununla da sınırlı kalamadı.
2	input	Bu küçük haliç, boğazdaa seyreden tekneler için önemlii bir doğal barınaktı.
	true	Bu küçük haliç, boğazda seyreden tekneler için önemli bir doğal barınaktı.
	pred	Bu küçük haliç, boğazda seyreden tekneler için önemli bir doğal batınaktı.
3	input	Kpsamlı bi güvnelik kavrami birçok uönden düşünülmelidir.
	true	Kapsamlı bir güvenlik kavramı birçok yönden düşünülmelidir.
	pred	Kapsamlı bir güvenlik kavramı birçok könden düşünülmelidir.
4	input	Oyuncuya odeme yapılırken vergiii düşülrk net tutar ödenır.
	true	Oyuncuya ödeme yapılırken vergi düşülerek net tutar ödenir.
	pred	Oyuncuya ödeme yapılırken vergi düşlürek net tutar ödenir.
5	input	2017 yılına kadarki donm için de bi geçiş süreci öngörüldü.
	true	2017 yılına kadarki dönem için de bir geçiş süreci öngörüldü.
	pred	2017 yılına kadarki donm içinde bir geçiş süreci öngörüldü.
6	input	Erkklr, apandisiteeee, kadinlara oranlla dha vazla yakalanir.
	true	Erkekler, apandisite, kadınlara oranla daha fazla yakalanır.
	pred	Erkekler, apandisite, kadınlara oranla daha vaazla yakalanır.

In Table 6.5, several examples that are incorrectly predicted with the standard seq2seq model are provided. These examples show that model needs improvement to be used in real world applications.

6.1.3. Prediction Time Performance

In standard seq2seq model, model continues decoding the next character until the end token is obtained or maximum length is reached. Therefore, the required time to decode a sentence heavily depends on the input sentence length. Since the model is limited to 80 characters, long sentences are divided into small blocks of 60+ characters and predicted separately. In Table 6.6, model prediction durations for different input sentence lengths are provided. It is observed that prediction duration increases in approximately quadratic.

Table 6.6. Standard Seq2Seq Model Prediction Durations.

Input Length	**Prediction Duration**
10 characters	301 ms
30 characters	1.14 sec
50 characters	2.49 sec
80 characters	4.18 sec
160 characters	8.47 sec

These time performances might still be practical for academic purposes; however, they are not practical for real life applications where spelling correction prediction time should be around 20-30ms and not more than 100ms because there is threshold in human perception after that is considered as slow. Additionally, spelling correction will be a step in some pipeline and overall time performance will be higher. Therefore, this research continued to achieve a model with better prediction time performance and One-step Seq2Seq Model is developed.

6.2. One-step Seq2Seq Model

6.2.1. Training

Model training of one-step seq2seq model is a more challenging task because compared to standard seq2seq model, it must learn the characters and their positions of output sentence at once. Additionally, the decoder block does not have any input from output sentence. Therefore, it is certain that one-step seq2seq model requires more model capacity than standard seq2seq model.

There have been many training experiments to find optimum parameters. Similar to standard seq2seq model, warmup learning rate and weighted loss calculation approaches are applied in the experiments. Transformer encoder and decoder layer counts, transformer size, learning rate, epoch size and batch size are main hyperparameters. In addition, one-step seq2seq model requires output length as an input parameter. However, varying output length leads to higher complexity in the model. To avoid this complexity, input and output vector lengths are fixed to 80 characters during training and prediction.

Since the task is more difficult, model needs a longer training period with more epochs. During these experiments, problems are encountered with the GeForce RTX 2080 Ti GPU and made the training processes difficult. After accessing GeForce RTX 3090, experiments progressed better.

In the training experiments, the decoding task becomes much harder because decoder block does not have any input from output sentence and it must predict all characters at once. Therefore, to increase the capacity of decoder block, number of transfomer decoder layers is increased to 4. To manage the model size stored in the GPU and training duration, number of transformer encoder layers is decreased to 2.

In preliminary experiments, to analyse the feasibility of the one-step seq2seq model, random 100k sentences are used as the training dataset. Models are trained for around 100 epochs to achieve an overfitted model that can perform the desired spelling correction on the training dataset. With these models, model predictions are observed for training and validation datasets. It is observed that there are promising predictions for the spelling correction.

After training models with small dataset, experiments are made with full training dataset. For the transformer size, 256, 512, 1024 and 1536 are experimented. In the experiments, batch size of 64 and learning rate of 1e-4 are used. The configuration with transformer size of 1536 is used as the best model. Model is trained for 10 epochs and it took 6 hours. Training and validation loss values of the model is provided in Figures 6.4 and 6.5.

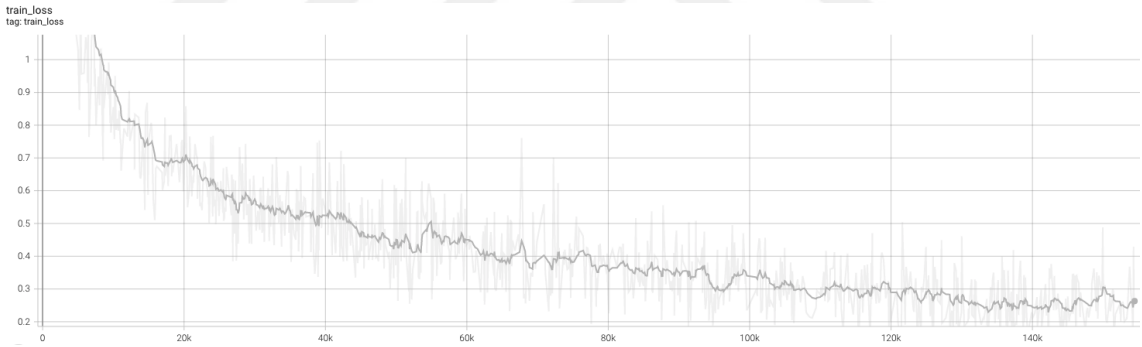

Figure 6.4. One-step Seq2Seq Model - 10 Epochs - Training Loss.

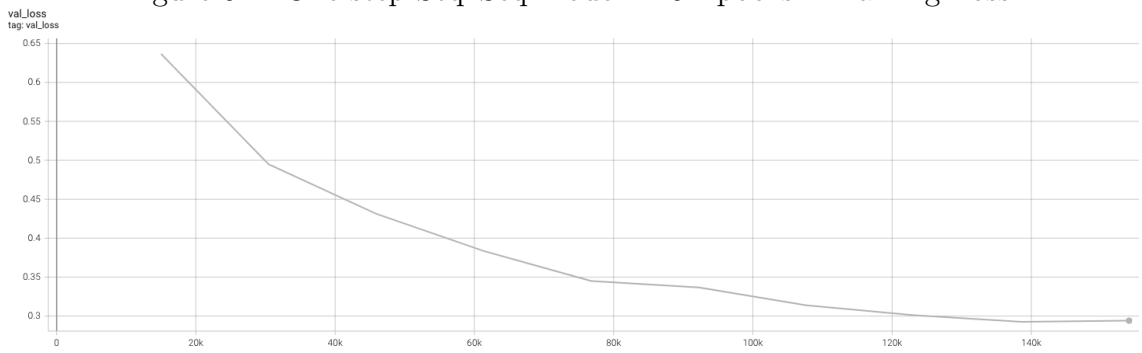

Figure 6.5. One-step Seq2Seq Model - 10 Epochs - Validation Loss.

6.2.2. Prediction Time Performance

In the one-step seq2seq model, model decoding process is done at one step, which is the main feature of the model. With this feature, much better prediction time performance can be achieved. Since the model is limited to 80 characters, long sentences are divided into small blocks of 60+ characters and predicted separately. Also, input and output vector lengths are padded to 80 characters so that each prediction step takes the same amount of time, about 24 milliseconds with CPU. For an input of 80 characters, there are two blocks and for an input of 160 characters, there are three blocks. In Table 6.7, model prediction durations for different input sentence lengths are provided.

Table 6.7. One-step Seq2Seq Model Prediction Durations.

Input Length	**With GPU**	**With CPU**	**Batch of 16 With GPU**	**Batch of 16 With CPU**
10 characters	9 ms	24 ms	32 ms	193 ms
30 characters	9 ms	24 ms	32 ms	193 ms
50 characters	9 ms	24 ms	32 ms	193 ms
80 characters	19 ms	50 ms	65 ms	388 ms
160 characters	28 ms	73 ms	97 ms	583 ms

These time performances have a quite good level for real life applications. One-step seq2seq model satisfied the expectations in terms of prediction time performance. Especially with GPU usage, outstandingly fast results are obtained. Similar GPU usage does not have much effect on the standard seq2seq model because decoding iterations depend on each other. Also, with CPU usage, the model is 100x faster for 160 characters input.

6.2.3. Test Dataset Results

For the test dataset, there are 3001 examples. Model predictions are obtained for these examples. As result, 1281 examples are correctly predicted and exact match accuracy of 42.69% is achieved. In Table 6.8, exact match scores of one-step seq2seq model are provided with the number of spelling errors of input sentences.

Table 6.8. One-step Seq2Seq Model - Exact match accuracies based on number of typos in the sentences.

# of Typos	**# of Sentences**	**Accuracy**
0	38	60.53%
1	227	59.47%
2	601	52.41%
3	789	44.87%
4	662	41.54%
5	401	27.18%
6+	283	24.73%
All	3001	42.69%

On the test dataset, token based accuracy metrics are calculated for error types. There are 26967 tokens in the test dataset and as the result, 85.00% of the tokens are correctly predicted. In Table 6.9, detailed token based match scores are provided based on the error types in the tokens.

Table 6.9. One-step Seq2Seq Model - Token Accuracy Scores on the Prepared Test Dataset.

	# of Tokens	**Accuracy**
Vowel Delete	2337	58.58%
Vowel Repeat	1322	87.44%
Consonant Repeat	1295	85.48%
Swap	787	70.65%
Ascii	3577	81.10%
Replacement	749	54.87%
Other	231	84.85%
Correct	16669	91.35%
All	26967	85.00%

Since the task of one-step seq2seq model is more difficult than the standard seq2seq model, it is expected to have lower accuracies. The model achieved worse but comparable results in return of better time performance, which is promising for the novel model architecture. However, optimizations should still be made for a perfect model.

Table 6.10. One-step Seq2Seq Model Correctly Predicted Text Examples.

#	**Type**	**Sentence**
1	input	Bu küçük haliç, boğazdaa seyreden tekneler için önemlii bir doğal barınaktı.
	true	Bu küçük haliç, boğazda seyreden tekneler için önemli bir doğal barınaktı.
	pred	Bu küçük haliç, boğazda seyreden tekneler için önemli bir doğal barınaktı.
2	input	Oyuncuya odeme yapılırken vergiii düşülrk net tutar ödenır.
	true	Oyuncuya ödeme yapılırken vergi düşülerek net tutar ödenir.
	pred	Oyuncuya ödeme yapılırken vergi düşülerek net tutar ödenir.
3	input	Dha sonraları bu aletleriin yapmında bkır kullanildi.
	true	Daha sonraları bu aletlerin yapımında bakır kullanıldı.
	pred	Daha sonraları bu aletlerin yapımında bakır kullanıldı.

In Table 6.10, several examples that are correctly predicted with one-step seq2seq model are provided. These examples show that model has potential to solve the spelling correction task.

Table 6.11. One-step Seq2Seq Model Incorrectly Predicted Text Examples.

#	Type	Sentence
1	input	Popunn krlı hakkında yapilan tartışmalar bununlaaaa da sınırlı klmadı.
	true	Popun kralı hakkında yapılan tartışmalar bununla da sınırlı kalmadı.
	pred	Popun kraaı hakkında ypııan aattımaalar buunlaa a sınılı kalmaddı
2	input	Ekmeeek, fırından ciktiktan en az 6 saat sonra yenmelidir.
	true	Ekmek, fırından çıktıktan en az 6 saat sonra yenmelidir.
	pred	Ekmek, fırından çıktıktan en az 6 saat sonra yenimlidirr.
3	input	Erkklr, apandisiteeee, kadinlara oranlla dha vazla yakalanir.
	true	Erkekler, apandisite, kadınlara oranla daha fazla yakalanır.
	pred	Erkekler, alandisite, kadınlara oranla daha vazla yakalanır.

In Table 6.11, several examples that are incorrectly predicted with one-step seq2seq model are provided. These examples show that there are issues that must be fixed.

Overall, spelling correction is an important task in NLP, especially in Turkish. These model architectures and experiments show that the task can be successfully done. With optimizations, longer training experiments and dataset improvements, adequate spelling correction models can be achieved.

6.3. Comparison Tests

For the test comparisons of this paper, there are two approaches; the results for the prepared test dataset of this study with other spelling correctors and the results for other datasets published by other researchers. Regarding the test datasets, the "TestSmall" dataset is provided by [28, 43] and the "Test2019" dataset is provided by [32]. Additionally, the "#Turki$hTweets" dataset was published by [44] and the

"100deda" dataset was published by [45]. As for the other spelling correctors, Zemberek [9], and GoogleDocs spelling correction tools are used. The comparison results can be found in 6.12, 6.13, 6.14, 6.15, 6.16, and 6.17.

In Table 6.12, the prepared test dataset contains 3001 sentences, with 38(1.27%) of the sentences containing no error, 227(7.56%) of the sentences containing one error, and 2736(91.17%) of the sentences containing more than one error.

Table 6.12. Sentence Accuracy Scores on the Prepared Test Dataset.

# of Typos	**# of Sentences**	**Standard Seq2Seq**	**Onestep Seq2Seq**	**Zemberek**	**GoogleDocs**
0	38	71.05%	60.53%	0.0%	73.68%
1	227	73.57%	59.47%	0.88%	38.33%
2	601	71.21%	52.41%	0.67%	31.78%
3	789	69.20%	44.87%	0.38%	21.42%
4	662	70.99%	41.54%	0.0%	17.82%
5	401	63.34%	27.18%	0.25%	9.73%
6+	283	59.36%	24.73%	0.0%	4.24%
All	3001	68.64%	42.69%	0.33%	21.46%

Table 6.13. Token Accuracy Scores on the Prepared Test Dataset.

	# of Tokens	**Standard Seq2Seq**	**Onestep Seq2Seq**	**Zemberek**	**GoogleDocs**
Vowel Delete	2337	84.72%	58.58%	39.92%	59.09%
Vowel Repeat	1322	97.58%	87.44%	40.62%	62.33%
Consonant Repeat	1295	96.99%	85.48%	30.97%	58.76%
Swap	787	91.36%	70.65%	50.95%	70.78%
Ascii	3577	94.74%	81.10%	66.45%	40.09%
Replacement	749	77.57%	54.87%	53.94%	68.76%
Other	231	99.13%	84.85%	86.15%	27.27%
Correct	16669	97.83%	91.35%	78.68%	97.51%
All	26967	95.49%	85.00%	68.11%	80.80%

In Table 6.14, the TestSmall dataset contains 509 sentences and 6507 tokens, of which 1171(17.9%) are non-canonical tokens. In Table 6.15, the Test2019 dataset contains 713 tweets and 7948 tokens, of which 2856(35.9%) are non-canonical tokens. In tables 6.14 and 6.15, **(1)** is for case-sensitive results over all tokens, **(2)** is for case-insensitive results over all tokens, **(3)** is for case-sensitive results over non-canonical tokens, and **(4)** is for case-insensitive results over non-canonical tokens.

Table 6.14. Token Accuracy Scores on the TestSmall Test Dataset.

	Standard Seq2Seq	**Onestep Seq2Seq**	**Eryiğit [28] Cascaded**	**Çolakoğlu [32] SMT**	**Çolakoğlu [32] NMT**
(1)	87.75%	79.38%	86.20%	**89.59%**	85.77%
(2)	92.42%	83.65%	92.97%	**93.53%**	89.52%
(3)	49.39%	39.91%	53.80%	**68.40%**	51.84%
(4)	67.07%	53.46%	74.72%	**76.00%**	58.67%

Table 6.15. Token Accuracy Scores on the Test2019 Test Dataset.

	Standard Seq2Seq	**Onestep Seq2Seq**	**Eryiğit [28] Cascaded**	**Çolakoğlu [32] SMT**	**Çolakoğlu [32] NMT**
(1)	76.48%	67.30%	75.39%	**78.10%**	67.87%
(2)	84.27%	73.32%	80.25%	**85.23%**	74.04%
(3)	48.55%	41.21%	56.44%	**66.35%**	45.03%
(4)	65.20%	54.41%	66.18%	**74.02%**	50.84%

In Table 6.16, the #Turki$hTweets dataset contains 2000 sentences and 16878 tokens, of which 9713 of them are unique and 6488 of them are non-canonical tokens. In 77% of the sentences, there are more than one error.

Table 6.16. Error Type Based Token Accuracy Scores on the #Turki$hTweets Dataset.

	Standard Seq2Seq	**Onestep Seq2Seq**	**ZS**	**ZN**	**ED**	**RB**
Accent	42.2%	28.4%	29.5%	60.8%	22.6%	39.9%
Adjacent	3.5%	3.5%	0%	14.3%	53.1%	0%
Deasciification	73%	61.6%	40.7%	87.1%	43.3%	85.8%
Intentional Char.	60.4%	41.5%	66.7%	68.3%	44.8%	36.1%
Phonetic Subs.	8.8%	2.9%	43.5%	39.1%	39.1%	0%
Proper Name	26.7%	27.2%	40.6%	0.9%	0%	0.4%
Separation	27.3%	19.2%	0%	47.9%	0%	0%
Unintentional Char.	21.7%	10.3%	53.4%	50.7%	50.7%	13.7%
Vowel	53.7%	31.7%	4.5%	63.6%	9.1%	18.2%
All	56.22%	45.30%	41.5%	71.3%	37.5%	60.5%

In Table 6.17, the 100deda dataset contains 100 sentences and 540 tokens. In each sentence, there is one error in the spelling of de/da.

Table 6.17. Accuracy Scores on the 100deda Test Dataset.

	Accuracy
Standard Seq2Seq	11%
Onestep Seq2Seq	3%
Arıkan [45]	**71%**
ITU [46]	0%
GoogleDocs	34%
MicrosoftOffice	29%
LibreOffice	0%

7. CONCLUSION AND DISCUSSION

NLP is an important field in artificial intelligence studies with deep learning models. There have been an increasing interest in the domain. Since the life becomes more digital everyday, human interactions with machines become more valuable. With increased demand, more research areas appear in NLP.

In NLP models, vectorization of the input sentences is one of the main challenges because textual data is unstructured in terms of words and sentences. Tokenization is the first step of data processing. There are three tokenization approaches; character, word, and subword level. In word level approach, the vocabulary size become large. In agglutinative languages like Turkish, there are many words from same stem and it is harder for model to learn the relationship between them. Morphological tokenization could perform well on Turkish NLP studies. In addition to these situations, spelling errors make it much harder and unmanageable because there is no rule and limit to spelling errors and a model can cope with only a limited part of spelling errors.

Statistical methods can not cover all cases of spelling errors because by their architecture their modeling capacity is limited. Therefore only a deep learning model can solve spelling error problem. Since the spelling error problem is based on character by definition, the deep learning model should be designed at character level.

In this research, two different character level seq2seq models are developed for Turkish spelling error correction. Models are developed in NMT approach. There have been many training experiments with different model architectures. The models produce promising results on a hard task which is proposing a general solution to spelling error correction. In addition, one-step seq2seq model achieved satisfying results in prediction time performance for other studies.

8. FUTURE WORK

8.1. Turkish Morphological Parser

WordPieceModel offers an approach but it is not guarantee a success since Turkish is different. Vocabulary size is much larger than English due to the additive structure of Turkish. BERT English vocabulary from WordPieceModel has 30k vocabulary size. With WordPieceModel on Turkish with specified vocabulary size, tokens will be like morphemes since statistically morphemes will occur. But this does not guarantee that WordPieceModel tokenized vocabulary will have morphological meaning. Following the main idea of wordpiece which is representing the word with frequently occurring subtokens, this can be utilized better in an additive language like Turkish where words can be tokenized into meaningful morphemes instead of a statistical approach.

For the tokenization, a morphological disambiguator component will be developed. There are existing models for this task like hasim-morph [47], trmorph2 [48] and zemberek [9]. Performance of these models will be analysed. Available datasets for this task will be searched. For this component, there could be an approach such that a training dataset are generated from existing models by majority voting and then several neural network models are trained on this dataset with different architectures like character based neural networks, transformer neural networks. Manual data labeling could be performed if necessary.

The performance of morphological tokenization depends on the input sentence. If the input sentence is misspelled, success of morphological tokenization decreases heavily. For this reason, the text normalization component will be used before processing.

8.2. Turkish Language Model

An important novelty of the BERT model is the WordPieceModel in the vocabulary setting and the tokenization parts. It learns the vocabulary of subwords from a large dataset by utilizing a language model which is used to maximize the likelihood of character sequences. Then the text is tokenized with the vocabulary in a greedy manner. In Turkish, it is expected that subwords would approximate to morphological affixes since Turkish is an additive language. Instead of only depending on dataset, a morphological parser, which directly contains language information, will be used to tokenize the input.

Turkish is an additive language where words are enriched with many affixes. From one root word, many other words could be generated. This situation results in vocabulary problems because of the high vocabulary size where there are more OOV problems compared to English and also it is hard to relate the words with small differences like 'geliyordum' and 'geldim' or 'ben', 'beni' and 'bana'. With increased vocabulary size, increased model capacity would be needed. With increased vocabulary size, larger dataset would be required to have enough word occurences for each word.

Published BERT models are trained on English, Chinese and multilingual datasets. There is no Turkish pretrained BERT model. To train such a model, a Turkish corpus will be collected. Since pretraining is unsupervised, any raw text could be used in pretraining data. The main sources are likely to be books, news, blogs, Wikipedia, articles etc. Other options of raw text can also be used as the source.

The collected corpus will be tokenized with the morphological disambiguator component. For the input format of the Transformer model, an adjustment will be needed to comply with the rest of the model when a state of the art model is used, like GPT, BERT, Roberta. Since the idea of WordPieceModel and morphological tokenization is same, using subwords, similar conventions will be followed to represent and encode words, like keeping word-subword mapping.

After Turkish corpus and data preparation steps are handled, a Transformer model will be trained from scratch. The model hyperparameters will be adjusted for the changed data and vocabulary. Model loss will be analysed through epochs. Since dataset size, vocabulary size and subword approaches will differ, convergence of the model could differ in learning rate, number of epochs, batch size and other possible parameters.

The pretrained model will be finetuned and tested. For testing, an existing dependency parsing end task is selected. For finetuning, proper layer implementation will be added on top of the pretrained model. During finetuning, the weights of the pretrained model could be freezed or could be updated along with finetuning. The hyperparameters of finetuning, like number of finetuning epochs and learning rate, will be analysed and adjusted for the finetuning dataset. The results of the finetuned model will be compared with the state of the art results.

During pretraining and finetuning, “torch” and “transformers” Python packages will be utilized. “transformers” Python package implements the most popular and successful state of the art models that are based on transformer neural networks. BERT is one of the implemented models in the “transformers” package. Most of the required functions for the pretraining and finetuning implementations of this project will be utilized from this package. Additionally needed components will be implemented if any occurs.

REFERENCES

1. Zhou, M., N. Duan, S. Liu and H.-Y. Shum, "Progress in Neural NLP: Modeling, Learning, and Reasoning", *Engineering*, Vol. 6, No. 3, pp. 275–290, 2020.

2. Colby, B. N., "Culture Grammars", *Science*, Vol. 187, No. 4180, pp. 913–919, 1975.

3. Landauer, T. K. and S. T. Dumais, "A Solution to Plato's Problem: The Latent Semantic Analysis Theory of Acquisition, Induction, and Representation of Knowledge.", *Psychological Review*, Vol. 104, No. 2, p. 211, 1997.

4. Martin, D. I. and M. W. Berry, "Mathematical Foundations Behind Latent Semantic Analysis", *Handbook of Latent Semantic Analysis*, pp. 35–56, 2007.

5. Mikolov, T., K. Chen, G. Corrado and J. Dean, "Efficient Estimation of Word Representations in Vector Space", *arXiv preprint arXiv:1301.3781*, 2013.

6. Devlin, J., M.-W. Chang, K. Lee and K. Toutanova, "Bert: Pre-training of Deep Bidirectional Transformers for Language Understanding", *arXiv preprint arXiv:1810.04805*, 2018.

7. Oflazer, K., "Turkish and Its Challenges for Language Processing", *Language Resources and Evaluation*, Vol. 48, No. 4, pp. 639–653, 2014.

8. Cambria, E. and B. White, "Jumping NLP curves: A Review of Natural Language Processing Research", *IEEE Computational Intelligence Magazine*, Vol. 9, No. 2, pp. 48–57, 2014.

9. Akın, A. A. and M. D. Akın, "Zemberek, An Open Source NLP Framework for Turkic Languages", *Structure*, Vol. 10, No. 2007, pp. 1–5, 2007.

10. Lourentzou, I., K. Manghnani and C. Zhai, "Adapting Sequence to Sequence Mod-

els for Text Normalization in Social Media", *Proceedings of the International AAAI Conference on Web and Social Media*, Vol. 13, pp. 335–345, 2019.

11. Wu, Y., M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey *et al.*, "Google's Neural Machine Translation System: Bridging the Gap Between Human and Machine Translation", *arXiv preprint arXiv:1609.08144*, 2016.

12. Sennrich, R., B. Haddow and A. Birch, "Neural Machine Translation of Rare Words with Subword Units", *arXiv preprint arXiv:1508.07909*, 2015.

13. Mikolov, T., I. Sutskever, K. Chen, G. S. Corrado and J. Dean, "Distributed Representations of Words and Phrases and Their Compositionality", *Advances in Neural Information Processing Systems*, pp. 3111–3119, 2013.

14. Peters, M. E., M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee and L. Zettlemoyer, "Deep Contextualized Word Representations", *arXiv preprint arXiv:1802.05365*, 2018.

15. Schuster, M. and K. Nakajima, "Japanese and Korean Voice Search", *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5149–5152, 2012.

16. Gage, P., "A New Algorithm for Data Compression", *C Users Journal*, Vol. 12, No. 2, pp. 23–38, 1994.

17. Kudo, T., "Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates", *arXiv preprint arXiv:1804.10959*, 2018.

18. Hinton, G. E., J. L. McClelland and D. E. Rumelhart, *Distributed Representations*, p. 77–109, MIT Press, Cambridge, MA, USA, 1986.

19. Bengio, Y., R. Ducharme, P. Vincent and C. Jauvin, "A Neural Probabilistic

Language Model", *Journal of Machine Learning Research*, Vol. 3, pp. 1137–1155, 2003.

20. Pennington, J., R. Socher and C. D. Manning, "Glove: Global Vectors for Word Representation", *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, 2014.

21. Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser and I. Polosukhin, "Attention is All You Need", *Advances in Neural Information Processing Systems*, pp. 5998–6008, 2017.

22. Radford, A., K. Narasimhan, T. Salimans and I. Sutskever, *Improving Language Understanding by Generative Pre-training*, 2018, `https://s3-uswest2.amazonaws.com/openaiassets/researchcovers/language-unsupervised/language_understanding_paper.pdf`, accessed in January 2022.

23. Radford, A., J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language Models are Unsupervised Multitask Learners", *OpenAI Blog*, Vol. 1, No. 8, p. 9, 2019.

24. Oflazer, K., "Spelling Correction in Agglutinative Languages", *arXiv preprint cmp-lg/9410004*, 1994.

25. Adali, K. and G. Eryiğit, "Vowel and Diacritic Restoration for Social Media Texts", *Proceedings of the 5th Workshop on Language Analysis for Social Media (LASM)*, pp. 53–61, 2014.

26. Sahin, M., U. Sulubacak and G. Eryigit, "Redefinition of Turkish Morphology Using Flag Diacritics", *Proceedings of The Tenth Symposium on Natural Language Processing, Phuket, Thailand*, 2013.

27. Torunoğlu-Selamet, D. and G. Eryiğit, "A Cascaded Approach for Social Media

Text Normalization of Turkish", *Proceedings of the 5th Workshop on Language Analysis for Social Media (LASM)*, pp. 62–70, 2014.

28. Eryiğit, G. and D. Torunoğlu-Selamet, "Social Media Text Normalization for Turkish", *Natural Language Engineering*, Vol. 23, No. 6, pp. 835–875, 2017.

29. Tursun, O. and R. Cakıcı, "Noisy Uyghur Text Normalization", *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pp. 85–93, 2017.

30. Göker, S. and B. Can, "Neural Text Normalization for Turkish Social Media", *3rd International Conference on Computer Science and Engineering (UBMK)*, pp. 161–166, 2018.

31. Etoori, P., M. Chinnakotla and R. Mamidi, "Automatic Spelling Correction for Resource-scarce Languages Using Deep Learning", *Proceedings of ACL, Student Research Workshop*, pp. 146–152, 2018.

32. Çolakoğlu, T., U. Sulubacak, A. C. Tantuğ *et al.*, "Normalizing Non-canonical Turkish Texts Using Machine Translation Approaches", *The 57th Annual Meeting of the Association for Computational Linguistics Proceedings of the Student Research Workshop*, The Association for Computational Linguistics, 2019.

33. Büyük, O., "Context-Dependent Sequence-to-Sequence Turkish Spelling Correction", *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)*, Vol. 19, No. 4, pp. 1–16, 2020.

34. Guinard, T., *Improving Turkish Spelling Correction with Wikipedia Edit History Data*, Ph.D. Thesis, University of Washington, 2021.

35. Bergsma, S., D. Lin and R. Goebel, "Web-scale N-gram Models for Lexical Disambiguation", *Twenty-First International Joint Conference on Artificial Intelligence*, 2009.

36. Kuznetsov, A. and H. Urdiales, "Spelling Correction with Denoising Transformer", *arXiv preprint arXiv:2105.05977*, 2021.

37. Gao, M., C. Xu and P. Shi, "Hierarchical Character Tagger for Short Text Spelling Error Correction", *arXiv preprint arXiv:2109.14259*, 2021.

38. Schmaltz, A., Y. Kim, A. M. Rush and S. M. Shieber, "Sentence-level Grammatical Error Identification as Sequence-to-sequence Correction", *arXiv preprint arXiv:1604.04677*, 2016.

39. Fixy-TR, *Fixy-TR/fixy*, 2020, `https://github.com/Fixy-TR/fixy`, accessed in January 2022.

40. Honnibal, M. and I. Montani, "spaCy: Natural Language Understanding with Bloom Embeddings, Convolutional Neural Networks and Incremental Parsing", *To appear*, 2017.

41. Makcedward, *makcedward/nlpaug: Data Augmentation for NLP*, 2020, `https://github.com/makcedward/nlpaug`, accessed in January 2022.

42. Paszke, A., S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai and S. Chintala, "PyTorch: An Imperative Style, High-Performance Deep Learning Library", *Advances in Neural Information Processing Systems 32*, pp. 8024–8035, Curran Associates, Inc., 2019.

43. Pamay, T., U. Sulubacak, D. Torunoğlu-Selamet and G. Eryiğit, "The Annotation Process of the ITU Web Treebank", *Proceedings of the 9th Linguistic Annotation Workshop*, pp. 95–101, 2015.

44. Koksal, A. T., O. Bozal, E. Yürekli and G. Gezici, "# Turki $ hTweets: A Benchmark Dataset for Turkish Text Correction", *Findings of the Association for Com-*

putational Linguistics: EMNLP, pp. 4190–4198, 2020.

45. Arikan, U., O. Güngör and S. Uskudarli, "Detecting Clitics Related Orthographic Errors in Turkish", *Proceedings of the International Conference on Recent Advances in Natural Language Processing*, pp. 71–76, 2019.

46. Eryiğit, G., "ITU Turkish NLP Web Service", *Proceedings of the Demonstrations at the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 1–4, 2014.

47. Sak, H., T. Güngör and M. Saraçlar, "Turkish Language Resources: Morphological Parser, Morphological Disambiguator and Web Corpus", *International Conference on Natural Language Processing*, pp. 417–427, Springer, 2008.

48. Çöltekin, Ç., "A Freely Available Morphological Analyzer for Turkish", *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, 2010.