T.R.

GEBZE TECHNICAL UNIVERSITY

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

TURKISH TEXT CORRECTOR PLUGIN

ÇAĞRI ÇAYCI

SUPERVISOR
DR. GÖKHAN KAYA

GEBZE
2024

T.R.
GEBZE TECHNICAL UNIVERSITY
FACULTY OF ENGINEERING
COMPUTER ENGINEERING DEPARTMENT

TURKISH TEXT CORRECTOR PLUGIN

ÇAĞRI ÇAYCI

SUPERVISOR
DR. GÖKHAN KAYA

2024
GEBZE

GRADUATION PROJECT
JURY APPROVAL FORM

This study has been accepted as an Undergraduate Graduation Project in the Department of Computer Engineering on 29/02/2024 by the following jury.

JURY

Member (Supervisor)	:	Dr. Gökhan Kaya
Member	:	Prof. Dr. İbrahim Soğukpınar
Member	:	Research Assistant Sibel Gülmez

ABSTRACT

As international relations intensify, brain drain and multicultural structures become more prevalent, notably within companies, schools, and etc. With companies progressively globalizing both internally and externally, using multiple languages in emails and comments through the day becomes commonplace. However, the physicality of this situation presents challenges for employees. For instance, using an English keyboard often leads to spelling errors in Turkish text written with this keyboard due to the different alphabets of the two languages. Additionally, these spelling errors sometimes lead to misunderstandings or failure to understand at all.

This project aims to reduce the physical challenges of multilingualism for English-Turkish by automatically transforming Turkish text typed using an English keyboard into correct Turkish text. A browser plugin has been developed to analyze each sentence written by the user on a word-by-word basis, utilizing a deep learning model to select the sentence with the most accurate word combination. Numerous studies have been conducted to enhance performance and accuracy in word processing.

As a result of this project, a web plugin was developed that can select even difficult words with a success rate of up to 90%. It also has an average processing time of 4 seconds per sentence. These statistics demonstrate that the project is both useful and successful in achieving its goals.

Keywords: Multilingualism, Deep Learning, Browser Plugin, Vowel Harmony.

ÖZET

Uluslararası ilişkiler yoğunlaştıkça, beyin göçü ve çok kültürlü yapılar, özellikle şirketler, okullar vb. içinde daha yaygın hale geliyor. Şirketlerin hem içeride hem de dışarıda giderek küreselleşmesiyle, gün boyunca e-postalarda ve yorumlarda birden fazla dil kullanmak olağan hale geliyor. Ancak bu durumun fizikselliği çalışanlar için zorluklar yaratmaktadır. Örneğin İngilizce klavye kullanmak, iki dilin alfabelerinin farklı olması nedeniyle, bu klavyeyle yazılan Türkçe metinlerde sıklıkla yazım hatalarına yol açmaktadır. Ayrıca bu yazım hataları bazen yanlış anlamalara ya da hiç anlaşılamamaya yol açabilmektedir.

Bu proje, İngilizce klavye kullanılarak yazılan Türkçe metni otomatik olarak doğru Türkçe metne dönüştürerek İngilizce-Türkçe için çok dilliliğin getirdiği fiziksel zorlukları azaltmayı amaçlamaktadır. Kullanıcının yazdığı her cümleyi kelime kelime analiz eden ve derin öğrenme modeli kullanarak en doğru kelime kombinasyonuna sahip cümleyi seçen bir tarayıcı eklentisi geliştirildi. Kelime işlemede performansı ve doğruluğu artırmak için çok sayıda çalışma yapılmıştır.

Bu projenin sonucunda %90'a varan başarı oranıyla zor kelimeleri bile seçebilen bir web eklentisi geliştirildi. Ayrıca cümle başına ortalama 4 saniye işlem süresine sahiptir. Bu istatistikler projenin hem faydalı hem de hedeflerine ulaşmada başarılı olduğunu gösteriyor.

Anahtar Kelimeler: Çok Dillilik, Derin Öğrenme, Tarayıcı Eklentisi, Ünlü Uyumu.

ACKNOWLEDGEMENT

I would like to express my deepest gratitude to Gebze Technical University for providing me with the invaluable opportunity to undertake this project. The academic environment and resources at the university have played a pivotal role in shaping and enhancing my research skills.

I extend my sincere appreciation to my advisor, Dr. Gökhan Kaya, for their unwavering support, guidance, and insightful feedback throughout the entire duration of this project.

Çağrı Çaycı

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Globalization in business is a highly discussed concept today, promoting international trade and cultural exchange within the business world. Consequently, the use of different languages in comments, emails, and other communications has become very common. However, this linguistic diversity can lead to practical challenges. For instance, typing Turkish text using an English keyboard can be difficult because the Turkish alphabet contains additional letters not found on an English keyboard. This difficulty can lead to misunderstandings or even a complete failure to communicate effectively. As a result, business deals may be canceled, and companies may lose significant amounts of money. To avoid these problems, two potential solutions include using multiple keyboards for multiple languages, which is not very practical, and developing a browser extension that can intelligently fix spelling errors for users.

Figure 1.1: Logo

1.1. Project Definition

The project involves developing a Turkish text corrector plugin designed to make necessary adjustments to text typed using an English keyboard. This plugin aims to automatically detect and correct common typing errors where Turkish characters are replaced by their English keyboard counterparts. The plugin should be accurate, efficient, and suitable for business use, ensuring that users can type Turkish text seamlessly even when using an English keyboard layout.

1.2. Objectives

The primary objectives of this project are:

1. **Correction Mechanism:** The extension should offer accurate corrections for misspelled words by mapping English keyboard inputs to their Turkish counterparts. Additionally, it should make necessary corrections automatically if there is no uncertainty.

2. **User-Friendly Interface:** The extension must have an intuitive interface that allows users to easily see necessary corrections and make adjustments if needed.

3. **Performance:** The extension must be accurate and efficient enough for business use. It should handle text corrections swiftly without significant lag or errors, ensuring it integrates seamlessly into professional environments.

1.3. Background Information

Typing in Turkish using an English keyboard can lead to numerous misspelling errors. Common Turkish letters such as ”ç”, ”ş”, ”ğ”, ”ı”, ”ö”, and ”ü” are often replaced by their closest English equivalents, resulting in misunderstandable or incomprehensible text. Native Turkish readers may struggle to understand the text and may perceive it as sloppy. This issue is particularly problematic in professional settings where clear communication is crucial. For example, the sentence ”Yönetim kurulu toplantısı 10:00’da başlayacak, lütfen katılımı onaylayın.” intended to mean ”The board meeting will start at 10:00, please confirm your attendance.” However, the part ”katılımı onaylayın” is problematic in professional settings because it might mean as ”confirm the killer” which is absurd.

1.4. Problem Statement

Although many text correction tools are available, none of them currently correct both structurally and semantically. Take the example sentence ”Buraya cig dustu.” A typical tool might change it to ”Buraya çığ düştü.” However, it fails to recognize the semantic context. ”Cig” could mean both ”raw (çiğ)” and ”avalanche (çığ)” in Turkish, so understanding the intended meaning of the sentence is crucial. Hence, there is a pressing need for a Turkish Text Corrector tool that can accurately correct both structural and semantic errors.

1.5. Scope

The scope of this project includes:

- **Inclusions:**
  - Development of a Turkish text corrector plugin for web browsers.
  - Implementation of a correction mechanism to map English keyboard inputs to Turkish characters.
  - Creation of an interface that allows users to review and select from possible corrections when there is uncertainty.
  - Optimization of the plugin to ensure high performance and reliability.
- **Exclusions:**
  - This project does not handle general typos beyond Turkish-specific corrections. For example, correcting ”gtti” to ”gitti” is not within the scope of this project.
  - It does not address input fields of HTML pages, such as search bars, as it focuses on areas where users write sentences, not keywords.
- **Limitations:**
  - The plugin's performance may be limited by the processing power and memory of the user's device.
  - It does not guarantee 100% accuracy in correcting words. It has some tolerance for errors.
- **Assumptions:**
  - Users provide the necessary permissions for the plugin.

2. TOOLS AND MATERIALS USED

2.1. Software Tools

- ***Visual Studio Code:*** Used for coding and debugging.
- ***Google Colaboratory:*** Used for dataset cleaning and training deep learning model.
- ***TensorFlow Library:*** Used for creating deep learning model.

2.2. Hardware Tools

Development and testing were conducted on desktop computers with the following minimum specifications:

- ***8GB RAM:*** Provided sufficient memory for running and testing the deep learning models and related software tools without significant performance issues.
- ***Intel Core i7 Processor:*** Offered the necessary processing power for efficient model training and execution of computational tasks.
- ***Microsoft Edge Browser:*** The extension was deployed and tested in the Microsoft Edge Browser environment and across multiple websites (Outlook, Gmail, Google, Youtube).

2.3. Online Resources

- ***Turkish Sentence Dataset*** It was created by cleaning and splitting Turkish Wikipedia documents into sentences, which were obtained from an online source [1].
- ***Turkish Word Dataset*** The dataset which were obtained from an online source is used [2].

3. PROJECT STEPS

This project consists of five main steps: generating variations of words, eliminating redundant variations, combining words to form sentences, selecting the most probable sentence, and integrating the system into web browser.

3.1. Generating Variations of Words

Since the English alphabet does not contain some letters found in the Turkish alphabet, users may consider using either the exact Turkish equivalent of the letter or its closest Turkish equivalent when typing on an English keyboard. For example, when a user types ”i”, they may intend to type ”ı” or ”i”. So, the mapping of what the user types and what they intend to type is shown in Table 3.1 below:

Table 3.1: The English letter and users' possible intentions

English letter	User intentions
c	c or ç
s	s or ş
g	g or ğ
o	o or ö
u	u or ü
ı	ı or i

Therefore, each letter entered by the user should be verified according to this table, and new words must be generated by incorporating these variations. The possible variations for the sentence ”Bu program coktu.” are shown in Table 3.2 below:

Table 3.2: Generated variations for a sentence.

Bu	program	coktu.
Bu	program	coktu.
Bü	prögram	çoktu.
		cöktu.
		çöktu.
		coktü.
		çoktü.
		cöktü.
		çöktü.

3.2. Eliminating Redundant Variations

In the step where variations of words are generated, 2^n variations are produced, where n is the number of English letters that have multiple Turkish equivalents. The exponential number of possible variations makes selecting the most probable sentence challenging and time-consuming. Therefore, to mitigate these issues, some words must be eliminated if possible. One solution is to remove invalid Turkish words from the variations.

3.2.1. Eliminating Invalid Turkish Words

Eliminating invalid Turkish words can be accomplished by using a dataset of valid Turkish words. For each variation, if it is not in this dataset, it must be eliminated as shown in the following Table 3.3:

Table 3.3: Eliminating invalid Turkish words

Bu	**program**	**coktu.**
Bu	program	~~coktu.~~
~~Bü~~	~~prögram~~	çoktu.
		~~cöktu.~~
		~~çöktu.~~
		~~coktü.~~
		~~çoktü.~~
		~~cöktü.~~
		çöktü.

However, there is a problem with this solution. The agglutinative nature of the Turkish language, which allows for the creation of an infinite number of words, poses a significant challenge. Despite this characteristic, it is not feasible to create a dataset encompassing an almost infinite number of words. For instance, consider the sentence ”Kütüphanemizi çok severiz.” The dataset may not contain the word ”Kütüphanemizi” or its other variations. In such cases, all variations would be eliminated, leaving no valid options to proceed with.

To address this issue, the initial solution involves utilizing a tool to identify the stem of a word. However, this method yielded somewhat inaccurate results. Consequently, the preferred approach is to search for the stem of the word by iteratively removing the last letter until a match is found within the dataset. Once a match is identified, both major and minor vowel harmony should be applied to the remaining word and its variations. If a word satisfies both conditions, it is not eliminated.

Table 3.4: Finding root of a word.

Word	Match
Kütüphanemiz	None
Kütüphanemi	None
Kütüphanem	None
Kütüphane	Yes

Table 3.5: Root and Vowel Harmony check

Word	Elimination Reason
Kütüphanemizi	No Elimination
Kütüphanemizı	Major&Minor Vowel Harmony
Kütüphanemızi	Major&Minor Vowel Harmony
Kütüphanemızı	Major&Minor Vowel Harmony
Kutuphanemizi	Root not in Dataset
⋮	⋮
Kutuphanemızı	Root not in Dataset

3.3. Combining Words to Form Sentences

After the eliminations are completed, the remaining words are combined to generate one or more sentences. The number of generated sentences depends on the number of non-eliminated variations for each word. Based on the remaining variations shown in Table 3.3, two sentences are generated:

- ”Bu program çöktü.”
- ”Bu program çoktu.”

Combining the remaining variations to form sentences results in sentences that contain only valid Turkish words. Therefore, it is not possible to make further eliminations using primitive solution such as dictionary-based elimination.

3.4. Selecting the Most Probable Sentence

The project's pivotal feature revolves around its mechanism for selecting the most probable sentence among multiple alternatives. Section 3.3 explores scenarios where two or more sentences may be encountered simultaneously. A decisive mechanism is imperative to discern the context of the sentence by identifying common words among the alternatives. To accomplish this, a deep learning model has been developed.

Developing a deep learning model is a multifaceted task, involving the creation of a dataset of sentences, the design of the deep learning model itself, and the subsequent training process.

3.4.1. Creating a Dataset of Sentences

The data utilized in this project is obtained from an online source containing the Turkish Wikipedia dump, which consolidates all Turkish documents on Wikipedia. The

data is then cleaned and segmented into sentences for this specific purpose. However, the vast majority of the dataset comprises valid Turkish sentences, with hardly any invalid ones. Therefore, several approaches are being considered:

1. **Creating artificial invalid sentences by shuffling words in valid sentences**

Table 3.6: First Approach

Invalid Sentences	Valid Sentences
“ değiştirmeyeceğidir. için bile ”	*“ Bir tanesine general rütbesi vermiştir. ”*
“ hareket İzmir Başkan DNA’ya... ”	*“ Bu adın anlamı “Dalak Tepeciği” dir. ”*
“ tekrar eklenti dünyaya ve ”	*“ Nehirde boylu boyunca uzandı. ”*
“ Nazi işgalin Planlanan kötümserdi. ”	*“ Höelin başke seçme şansı olmadığı için... ”*
“ Mançurya’da fethetti. şekilde başkenti ”	*“ Anlaşma gereği Temuçin orada kalacaktı. ”*

This approach involves creating a dataset of sentences as depicted in Table 3.6, designing, training the deep learning model with the dataset, and then testing it. Despite achieving high accuracy in both testing and training phases, the model fails when presented with real sentences. This failure arises from the model assigning high probabilities to all structurally valid sentences, regardless of their semantic coherence. The issue stems from invalid sentences in the dataset often comprising overly simplistic examples, while our objective is to distinguish between sentences that are very similar in structure, as outlined in Section 3.3. Consequently, this approach is not utilized in this project. Therefore, there are two possible approaches left: using a unary classifier or creating more appropriate sentences.

2. **Developing a unary classifier instead of a binary classifier**

   Using a unary classifier model is a common technique, particularly when there is an overabundance of a single class or a complete absence of other classes. In this approach, the model is trained exclusively with samples from one class. The model then classifies new data as belonging to this class if it is similar to the training samples or as an outlier if it is not [3].

   Due to the intricate nature of unary classifiers, attempting to exclusively utilize valid sentences to cover all invalid ones in text classification proves to be quite challenging [4]. Additionally, deploying a highly complex deep learning model may negatively impact the performance of a plugin in a web environment.

   As a result, this approach is often abandoned, and binary classifiers are employed, which are trained with carefully crafted invalid sentences.

3. **Creating carefully crafted invalid sentences**

   After testing previous approaches and not obtaining satisfactory results, it has been discovered a more effective method for creating invalid sentences. The primary goal at this stage is to select the most appropriate sentence from a set of closely related sentences, as indicated in Section 3.3. Therefore, the dataset must consist of these types of closely related sentences. For this reason, the dataset is created as following:

Invalid Sentences	Valid Sentences
Bugün hava çök soğuk.	Bugün hava çok soğuk.
Çiğ, hızla kayan büyük kar kütleleridir.	Çığ, hızla kayan büyük kar kütleleridir.
Bu et neredeyse çığ.	Bu et neredeyse çiğ.
Çök soğuk havalarda don olur.	Çok soğuk havalarda don olur.
Çok soğuk havalarda dön olur.	Aradığını bulamayınca geri döndü.
Çok soğuk havalarda don ölür.	Duvardaki bütün çivileri söktü.
Bütün sunucularımız an itibariyle coktu.	Bütün sunucularımız an itibariyle çöktü.
Anı gelişen ataklarla gol bulmaya çalışıyordu.	Ani gelişen ataklarla gol bulmaya çalışıyordu.

3.4.2. Designing a Deep Learning Model

A general tokenizer from the Keras library is employed to tokenize the dataset, converting all words into tokens because deep learning models cannot directly process raw text. Subsequently, a sequential deep learning model is constructed with the following layers:

- ***Embedding Layer:*** An embedding layer is added to deep learning model which takes tokens as input and maps them to dense vectors of fixed size. These vectors capture semantic information about the tokens in a continuous vector space [5].

- ***Bidirectional LSTM Layers:*** The essential layers of the model, namely Bidirectional LSTM layers, are incorporated sequentially. Bidirectional LSTM Layers are a prevalent technique in text classification. LSTMs (Long Short-Term Memory) are a type of recurrent neural network (RNN) capable of capturing long-range dependencies in sequential data such as text. By processing the input sequence in both forward and backward directions, Bidirectional LSTMs enable the model to learn from both past and future information simultaneously which is very critical for this project goal [6].

- ***Dense Layer:*** The dense layer is the general and often used layer that contains a deeply connected neural network layer. It can be used as it means a hidden layer that is associated to each and every one node in the next layer. It has been proven

the positive effects of dense layers in classification models. Therefore it is added to the deep learning model [7].

- ***Dropout Layer:*** A dropout layer is a common technique used to reduce overfitting in models. When working with very large datasets, models can still overfit the data, so adding a dropout layer helps mitigate this issue. Therefore, it added to the model [8].

- ***Output Layer:*** The output layer of the model takes 64 units as input from the preceding dense layer and produces a single output representing the probability of sentence classification. This output probability indicates the likelihood of the input sentence belonging to a certain class (e.g., valid or invalid sentence).

These layers are combined to create the deep learning model, as visualized in Figure 3.1.

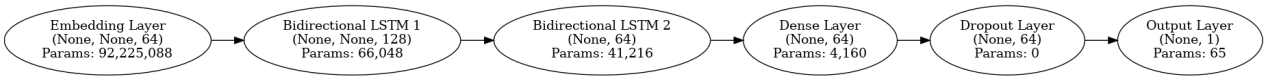

Figure 3.1: Deep Learning Model

3.4.3. Training the Deep Learning Model

The deep learning model is trained with a dataset of sentences as indicated in Table 3. For training purposes, 3 million positive and 3 million negative sentences are used. Additionally, 1.2 million randomly selected sentences among these sentences are used for validation. The model accuracies and losses for training and validation are shown in the following graphs.

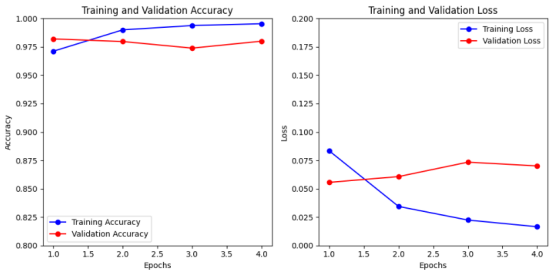

Figure 3.2: Model accuracy and loss per epochs

Although there is a possible overfitting until epoch 3 due to the increasing training accuracy but decreasing validation accuracy, it recover after epoch 3.

3.5. Integrating the System into Web Browser

After previous steps are completed, the system must be integrated into a web browser. For this reason, a folder is created for the plugin as follows:

- ***finalDL folder:*** It contains the Tensorflow.js format of the deep learning model to run it on the web environment.
- ***logo.png:*** The logo of the plugin.
- ***main.js:*** The plugin contains all the necessary code to work correctly. It always checks the page for an editable element. If there is no editable element, it does not load the model, words, and tokens files into the page. After a sentence is typed (a bunch of words ending with ”.”, ”!”, or ”?”), the plugin corrects the sentence in at most 4 seconds. The correction mechanism works as follows:
  - **If there is only one possible sentence:** It corrects it immediately without using the deep learning model.
  - **If there are multiple possible sentences:** It gives the sentences to the deep learning model. After the model calculates each sentence's probability, each word's probability is also calculated by looking at the probabilities of the whole sentences. After that process, each word is written according to the following rule:
    * If the probability of two words is too close (difference ¡ 0.1), the word with the higher probability is written with an underlined orange.
    * Otherwise, write the word without an underline.

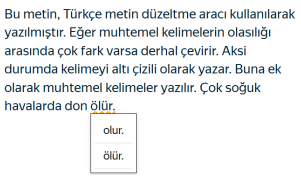

Figure 3.3: Working mechanism of plugin

- ***manifest.json:*** The manifest file of the plugin.
- ***tokens.txt:*** It contains the tokens of each word. It is necessary to tokenize the words before giving them into the model.
- ***words:*** It contains the words to use them as a dictionary as shown in Table 3.3.

4. CONCLUSIONS

4.1. Evaluation of the Project

To evaluate the performance and identify the sources of errors of the developed tool, it was tested using a text containing 8000 words and 500 sentences. The results are as follows:

- For approximately **1000 non-trivial words** (words with multiple valid variations or no variations):
  - **60 errors** were model-related
  - **36 errors** were due to the words not being present in the dataset

The image below shows a section of this text. Words underlined in red indicate model-related errors, while words underlined in orange indicate errors due to the words not being present in the dataset.

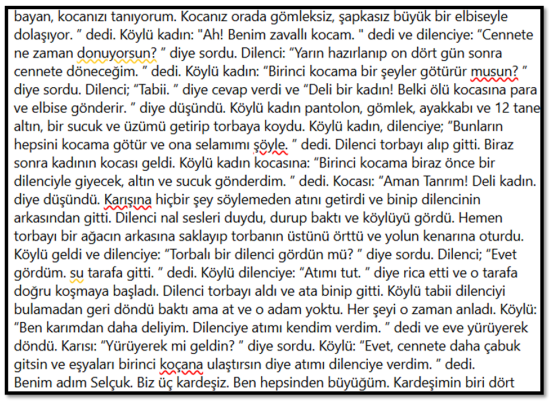
bayan, kocanızı tanıyorum. Kocanız orada gömleksiz, şapkasız büyük bir elbiseyle dolaşıyor. " dedi. Köylü kadın: "Ah! Benim zavallı kocam. " dedi ve dilenciye: "Cennete ne zaman donuyorsun? " diye sordu. Dilenci: "Yarın hazırlanıp on dört gün sonra cennete döneceğim. " dedi. Köylü kadın: "Birinci kocama bir şeyler götürür musun? " diye sordu. Dilenci; "Tabii. " diye cevap verdi ve "Deli bir kadın! Belki ölü kocasına para ve elbise gönderir. " diye düşündü. Köylü kadın pantolon, gömlek, ayakkabı ve 12 tane altın, bir sucuk ve üzümü getirip torbaya koydu. Köylü kadın, dilenciye; "Bunların hepsini kocama götür ve ona selamımı söyle. " dedi. Dilenci torbayı alıp gitti. Biraz sonra kadının kocası geldi. Köylü kadın kocasına: "Birinci kocama biraz önce bir dilenciyle giyecek, altın ve sucuk gönderdim. " dedi. Kocası: "Aman Tanrım! Deli kadın. diye düşündü. Karışına hiçbir şey söylemeden atını getirdi ve binip dilencinin arkasından gitti. Dilenci nal sesleri duydu, durup baktı ve köylüyü gördü. Hemen torbayı bir ağacın arkasına saklayıp torbanın üstünü örttü ve yolun kenarına oturdu. Köylü geldi ve dilenciye: "Torbalı bir dilenci gördün mü? " diye sordu. Dilenci; "Evet gördüm. şu tarafa gitti. " dedi. Köylü dilenciye: "Atımı tut. " diye rica etti ve o tarafa doğru koşmaya başladı. Dilenci torbayı aldı ve ata binip gitti. Köylü tabii dilenciyi bulamadan geri döndü baktı ama at ve o adam yoktu. Her şeyi o zaman anladı. Köylü: "Ben karımdan daha deliyim. Dilenciye atımı kendim verdim. " dedi ve eve yürüyerek döndü. Karısı: "Yürüyerek mi geldin? " diye sordu. Köylü: "Evet, cennete daha çabuk gitsin ve eşyaları birinci kocana ulaştırsın diye atımı dilenciye verdim. " dedi. Benim adım Selçuk. Biz üç kardeşiz. Ben hepsinden büyüğüm. Kardeşimin biri dört

Figure 4.1: A sample from test text.

Based on these test results, it can be observed that the tool corrects with an error rate of approximately **10%**, with **6%** of the errors being model-related and around **4%** being dataset-related. Also, it corrects the sentence within **5 seconds**.

To understand the importance of model-related errors, the confidence levels of the model's predictions were investigated. It was found that the average probability difference between the correct word and the incorrect word in the model's correct predictions is **0.7**, while in the incorrect predictions, the average probability difference between the incorrect word and the correct word is **0.04**. Following examples shows the confidence of the model.

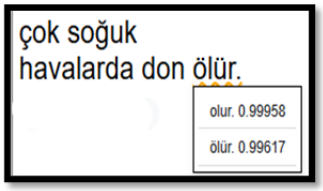

Figure 4.2: Model is not certain

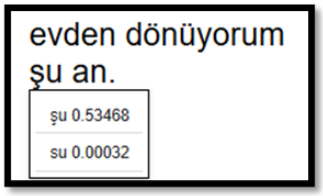

Figure 4.3: Model is almost sure.

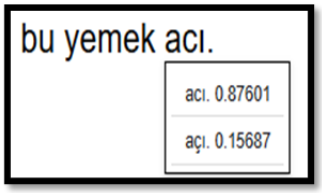

Figure 4.4: Model is sure.

In Sections 4.1.1 and 4.1.2, some of the reasons for incorrectly corrected words were demonstrated. Additionally, in Section 4.2.1, an idea for improving the tool has been put forward.

4.1.1. Eliminating Words

In Section 3.2.1, it is mentioned that the Turkish language has an infinite number of words because it is agglutinative, and a lot of words can be created by adding affixes to the word. Therefore, it is not possible to have a dataset that large of words, and even though vowel harmonies have very good impact on performance of the tool as it shown in Figure 4.5, they cover only some of the words. However, there is still a big gap. To explain, as shown in Figure 4.6, the word ”ciftligimize” cannot be automatically eliminated.

Figure 4.5: Impact of vowel harmony

Figure 4.6: Inadequate Handling of Turkish Word Structure by Dictionary

4.1.2. Special Words

Just as new words are created by adding suffixes to words, different forms arise when suffixes are added to the question particles ”mı” and ”mi”. However, it is impossible for a deep learning model to be trained with all forms of these question

particles. Therefore, as the inflection of the question particle increases, the likelihood of the deep learning model becoming uncertain also increases. For example, while the model correctly analyzes the sentence ”Sen hep buralara gelir miydin?”, it struggles with the following example:

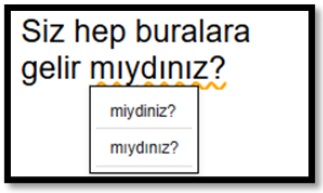

Figure 4.7: Inadequate Handling of Turkish Word Structure by Deep Learning Model

4.2. Recommendations for Future Research

4.2.1. Solution for Eliminating Words

To solve the issue mentioned in Section 4.1.1, additional rules can be added to the system, such as consonant assimilation. Additionally, using a powerful tool for finding stems of Turkish words may help to fix the issue. For instance, if the tool identifies the stem of ”çiftligimize” as ”çiftlik”, then applying consonant assimilation would correct it to ”çiftliğimize” instead of ”çiftligimize”.

4.3. Closing Thoughts

As a result of the project, the Turkish Text Corrector Plugin demonstrates good performance in correcting Turkish text written using an English keyboard. Unlike other correction tools, it makes corrections both semantically and structurally. Although it may not yet achieve the high accuracy required for work environments and critical texts (¡0.95 accuracy), it performs adequately for daily use and informal correspondence.

BIBLIOGRAPHY

[1] M. Keskin, Turkish wikipedia dump, https://www.kaggle.com/datasets/mustfkeskin/turkish-wikipedia-dump, 2018.

[2] E. Tuna, All turkish words dataset, https://www.kaggle.com/datasets/enistuna/all-turkish-words-dataset, 2024.

[3] C. Bellinger, S. Sharma, and N. Japkowicz, "One-class versus binary classification: Which and when?" In 11th International Conference on Machine Learning Applications, vol. 2, 2012, pp. 102–106. DOI: 10.1109/ICMLA.2012.212.

[4] N. Seliya, A. A. Zadeh, and T. M. Khoshgoftaar, "A literature review on one-class classification and its potential applications in big data," Journal of Big Data, vol. 8, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:237474982.

[5] O. Hrinchuk, V. Khrulkov, L. Mirvakhabova, E. Orlova, and I. Oseledets, "Tensorized embedding layers for efficient model compression," arXiv preprint arXiv:1901.10787, 2019.

[6] J. Li, Y. Xu, and H. Shi, "Bidirectional lstm with hierarchical attention for text classification," in IEEE 4th Adv. Inf. Technol., Electron. Autom. Control Conf. (IAEAC), 2019, pp. 456–459. DOI: 10.1109/IAEAC47372.2019.8997969.

[7] V. H. Josephine, A. Nirmala, and V. L. Alluri, "Impact of hidden dense layers in convolutional neural network to enhance performance of classification model," in IOP Conference Series: Materials Science and Engineering, IOP Publishing, vol. 1131, 2021, p. 012 007.

[8] S. Park and N. Kwak, "Analysis on the dropout effect in convolutional neural networks," in Computer Vision – ACCV 2016, S.-H. Lai, V. Lepetit, K. Nishino, and Y. Sato, Eds., Cham: Springer International Publishing, 2017, pp. 189–204, ISBN: 978-3-319-54184-6.

APPENDICES

Appendix 1: Project Trailer

Check out the project trailer on YouTube:
Project Trailer

Appendix 2: Code for the Web Plugin

The code for the web plugin is available on GitHub:
GitHub Repository

Appendix 3: Code for the Deep Learning Model

The code for the deep learning model is available on Google Colaboratory:
Google Colaboratory