TURKISH AND ENGLISH GRAMMATICAL ERROR CORRECTION

A Thesis

by

Asım Ersoy

Submitted to the
Graduate School of Sciences and Engineering
in Partial Fulfillment of the Requirements for
the Degree of

Master of Science

in the
Department of Computer Science

Özyeğin University
June 2024

Copyright © 2024 by Asım Ersoy

TURKISH AND ENGLISH GRAMMATICAL ERROR CORRECTION

Approved by:

Prof. Olcay Taner Yıldız, Advisor
Dept. of Artificial Intelligence and Data Engineering
Özyeğin University

Prof. Arzucan Özgür
Dept. of Computer Engineering
Boğaziçi University

Asst. Prof. İlknur Karadeniz
Dept. of Artificial Intelligence and Data Engineering
Özyeğin University

Date Approved: May 17, 2024

To the resilient people of Gaza and everyone standing up against the oppression and injustice in the world. To my beloved parents and fiancé, whose unwavering support empowers me every step of the way.

ABSTRACT

Grammatical Error Correction has seen significant progress with the recent advancements in deep learning. As those methods require huge amounts of data, synthetic datasets are being built to fill this gap. Unfortunately, synthetic datasets are not organic enough in some cases and even require clean data to start with. Furthermore, most of the work that has been done is focused mostly on English. In this work, we introduce a new organic data-driven approach, clean insertions, to build parallel Turkish Grammatical Error Correction datasets from any organic data, and to clean the data used for training Large Language Models. We achieve state-of-the-art results on two Turkish Grammatical Error Correction test sets out of the three publicly available ones. We also show the effectiveness of our method on the training losses of training language models. In addition to our work for Turkish Grammatical Error Correction, we build a dataset for English Grammatical Error Correction composed of essays written by native Turkish students from different universities capturing the grammatical mistakes that Turkish native speakers tend to make. We release with this work all the datasets and models developed in this study.

ÖZETÇE

Gramatik Hata Düzeltme, derin öğrenmedeki son gelişmelerle önemli bir ilerleme kaydetmiştir. Bu yöntemler geniş miktarda veri gerektirdiğinden, sentetik veri setleri bu boşluğu doldurmak için oluşturulmaktadır. Ne yazık ki, sentetik veri setleri bazı durumlarda yeterince organik değildir ve hatta temiz veriye ihtiyaç duyarlar. Dahası, yapılan çalışmaların çoğu genellikle İngilizce üzerine odaklanmıştır. Bu çalışmada, herhangi bir organik veriden paralel Türkçe Gramatik Hata Düzeltme veri setleri oluşturmak ve Büyük Dil Modelleri için kullanılan veriyi temizlemek için yeni bir organik veri odaklı yaklaşım olan ”clean insertions”i tanıtıyoruz. Üçü arasında kamuoyunda bulunan iki Türkçe Gramatik Hata Düzeltme test setinde en son teknolojiye dayalı sonuçlar elde ediyoruz. Ayrıca, dil modellerini eğitirken eğitim kayıplarının etkinliğini gösteriyoruz. Türkçe Gramatik Hata Düzeltme için yaptığımız çalışmanın yanı sıra, Türk ana dili konuşanların genellikle yaptığı dilbilgisel hataları yakalayan farklı üniversitelerdeki Türk öğrenciler tarafından yazılmış denemelerden oluşan bir İngilizce Gramatik Hata Düzeltme veri seti oluşturuyoruz. Bu çalışmada geliştirilen tüm veri setlerini ve modelleri de yayınlıyoruz.

ACKNOWLEDGEMENTS

I want to express my sincere gratitude to my thesis advisor, Olcay Taner Yıldız, for his invaluable guidance, support, and encouragement throughout this research journey. His expertise, patience, and insightful feedback have been instrumental in shaping this thesis.

We would like to extend our sincere thanks and appreciation to the Scientific and Technological Research Council of Turkey (TUBITAK) for their invaluable support and funding towards our project, numbered 121E321. Their contributions have been instrumental in making this project a success.

My deepest gratitude is my family's consistent love, encouragement, and understanding during this challenging phase. Their ongoing support and confidence in my capabilities have been a significant source of strength and motivation.

Lastly, I extend my sincere gratitude to God for His guidance, blessings, and steadfast strength that carried me through this journey.

TABLE OF CONTENTS

DEDICATION . . . . . **iii**

ABSTRACT . . . . . **iv**

ÖZETÇE . . . . . **v**

ACKNOWLEDGEMENTS . . . . . **vi**

LIST OF TABLES . . . . . **ix**

LIST OF FIGURES . . . . . **xi**

I INTRODUCTION **1**

II RELATED WORK **4**

2.1 Datasets 4

2.2 Approaches 4

2.3 Computer Assisted Language Learning 5

2.4 Turkish Grammatical Error Correction 7

III DATA METHODOLOGY **8**

3.1 Clean Insertions 8

3.1.1 OSCAR GEC 8

3.1.2 GPT GEC 11

3.1.3 Datasets Overview 12

3.2 English GEC for Turkish Speakers 13

3.2.1 Data Collection and Processing 13

3.2.2 Data Labelling 14

IV EXPERIMENTAL SETUP **18**

4.1 Turkish GEC 18

4.1.1 Models 18

4.1.2 Evaluation 18

4.2 English GEC 20

V RESULTS **22**
5.1 Turkish GEC 22
5.1.1 Correction 23
5.1.2 Detection 25
5.2 English GEC 26
VI LANGUAGE MODELS **28**
6.1 Data Processing 28
6.2 Training 29
6.3 Evaluation 29
VII CONCLUSION **31**
REFERENCES **33**
VITA **39**

LIST OF TABLES

1 Spelling dictionary expansion iterations details, showing for each iteration the size of the spelling dictionary, the number of extracted OSCAR texts and distinct words, and the size difference increase of the spelling dictionary. 11

2 An overview of the datasets utilized in this work. The datasets in the top half are synthetic and the bottom ones, the evaluation sets, are humanly annotated. The error type ERRANT refers to the automatic annotation tool ERRANT [1, 2], which automatically annotates parallel sentences with error-type information. Tokens information is based on OpenAI's tokenizer tiktoken with gpt2 encodings 12

3 Error Types and their percentages in the evaluation sets mentioned in Table 2 classified by ERRANT-TR 13

4 Number of Compositions by University, Level, and Format 15

5 Error Types percentages in checkMate-X and checkMate-Y datasets. 17

6 The list of 25 main error categories in the ERRANT framework with examples and explanations as listed in their work. 19

7 ERRANT-TR's Error Codes, Descriptions, and Examples as they list in their work. An empty cell indicates that the category has no example of being either too wide or not useful for Turkish. 21

8 ERRANT span-based **correction** scores (precision, recall, and F0.5) of every model on all the evaluation sets. All mT5 are trained are ours. GECTurk (Seq Tagger) is trained by the referenced work. The highest value of all models on each metric is bolded per evaluation set. 22

9 ERRANT span-based **detection** scores (precision, recall, and F0.5) of every model on all the evaluation sets. All mT5 are trained are ours. GECTurk (Seq Tagger) is trained by the referenced work. The highest value of all models on each metric is bolded per evaluation set 25

10 Results of the fine-tuned T5 model. checkMate-X and checkMate-Y are the two datasets we introduce each being annotated by a different annotator. The precision, recall, and F0.5 are shown for each experiment. 26

11 Trainig and validation losses of 4 different GPT-2 models of two different sizes trained on two dataset combinations having a common Turkish OSCAR sample and either the original sentences of our OSCAR GEC dataset or the corrected sentences of our OSCAR GEC dataset cleaned using our clean insertions method. Bolded values show lower training and validation losses. 29

12 The average ratings of 50 generated texts sampled per model. The samples are rated from 1 to 5 by five annotators (A1-A5). 30

LIST OF FIGURES

1 A pipeline of the creation process of OSCAR GEC showing all the steps involved in creating the OSCAR GEC dataset 8

2 An annotation tool developed with Python 14

3 An annotation tool developed with Java 16

4 One example from the Turkish Tweets and the output of the three models OSCAR GEC, GPT GEC, and Sequence Tagger. The red segments are incorrect and the green ones are correct. 24

CHAPTER I

INTRODUCTION

Humans naturally tend to make typos for various factors. Those typos and grammatical errors propagate to the data used in Natural Language Processing (NLP) systems and any data-related tasks, which could lead to unexpected behavior. For instance, a sentiment analysis text classifier that has been trained with a frequently occurring misspelled word may produce unexpected results when processing correctly spelled words in the input. Another example that we looked into closely is Large Language Models (LLMs) which are trained on massive amounts of data mostly from the internet such as the OSCAR dataset [3]. We observe a significant percentage of grammatical mistakes in the OSCAR dataset, specifically, in the Turkish OSCAR data, which has an effect on the training losses and causes the models sometimes to generate erroneous text. These examples show the importance of the NLP task Grammatical Error Correction (GEC) in facilitating text-based communications.

Grammatical Error Correction's significance was also noticeable during the COVID-19 outbreak. Grammar-checking tools played a pivotal role in assisting the students who received online education during the outbreak to improve their writing skills. Turkish students studying English, for example, were affected by the outbreak and missed the opportunity to get quick face-to-face feedback.

Given the GEC task's importance, many works addressed the task and, with the advancement and rise of deep learning techniques, achieved significant progress on the task [4]. Unfortunately, most of that work focused on English and slightly on some other common languages. On the other hand, the work done for Turkish is few and limited, which explains how the Turkish GEC task is barely noticed and paid attention to. There are only two open-source evaluation sets available with more than one error type [5, 6]. And, there

is only one open-source synthetic training set with a pre-defined set of error types from [6] utilizing inorganic data such as newspaper data to build GEC datasets, which leads to poor performance on common general errors that an average person could make.

In this work, we tackle the Turkish GEC task with a new organic data-driven approach, addressing the problem of inorganic and artificial datasets utilized for GEC. We introduce a simple method that we call clean insertions. It involves building an incorrect-correct spelling dictionary to be used in replacing commonly made misspellings of words and phrases with their correct versions in any organic text, e.g. text crawled from the internet, which mostly contains grammatical errors. The spelling dictionary and its size are crucial and have a major effect on the produced dataset's quality.

This method leads to a partially correct parallel text since the spelling dictionary probably does not contain all the existing mistakes in the dataset. Despite this fact, our simple method achieves state-of-the-art results on two different test sets out of the three available open-source evaluation sets. In addition to that, we use GPT-4 to automatically build a parallel GEC dataset and compare the models we train on those different datasets. Furthermore, we run experiments to test and show the effectiveness of our method on cleaning training data for LLMs. We open-source different datasets and models with this work for the Turkish GEC task.

We also address the need to make English grammar-checking tools aware of mistakes made by native Turkish speakers. We collect data from universities' English preparatory departments, annotate them, and utilize the annotated data to train GEC models.

Here are our work's contributions:

- We introduce a new organic data-driven approach, clean insertions, to build synthetic GEC datasets from any organic data, which mostly contains grammatical mistakes. No clean data is required!

- We find that partially corrected GEC datasets could be utilized to achieve state-of-the-art results.
- We find that cleaning the data used for training LLMs leads to lower loss values.
- We open-source [7] 1) A manually annotated spelling dictionary consisting of about 150k incorrect-correct word and phrase pairs. 2) The largest Turkish GEC parallel dataset consisting of 2.3m sentences. 3) A Turkish GEC dataset of about 100k sentences annotated by GPT. 4) The largest test set for Turkish GEC consisting of about 2,400 manually corrected sentences. 5) All the best-performing models trained in this work.
- We open source 1) An English GEC dataset [8] specialized with mistakes often made by native Turkish speakers. 2) The Python and Java annotation tools [9] we developed for this work. 3) The English GEC model that we train on our dataset [10].

CHAPTER II

RELATED WORK

Several datasets and models have been developed to address the grammatical error correction task. We review some of those in this chapter.

2.1 Datasets

Datasets that have been utilized for Grammatical Error Correction mostly consist of English academic essays authored by either English learners and native speakers [11, 12, 13, 14]. Other datasets included web data such as in [15] which contains random paragraphs sampled from the Common-Crawl dataset [16]. Some studies put some effort into filling the gap and built non-English datasets for less popular languages in NLP such as Arabic [17], Chinese [18], and Turkish [5].

In addition to the human-labeled datasets mentioned above, the advancements in deep learning surged the need for large synthetic datasets. Mainly, there are two techniques used in building such datasets: noisy injections and back-translation [19]. The Noisy injections technique involves corrupting an already clean text by inserting some pre-defined errors in a rule-based way [20, 21, 22], or by injecting probabilistic error patterns [23, 24, 25]. The back-translation technique, on the other hand, involves training a noisy channel model to predict a probable source text given a correct text [26].

2.2 Approaches

Approaches used in Grammatical Error Correction developed over time, beginning with rule-based approaches [27] due to their straightforwardness. Later on, data-driven approaches emerged such as single-error-type classifiers [28, 29, 30, 31], where each classifier targets a specific error type independently, assuming the surrounding context is correct

[4], which is a limitation of rule-based approaches. Statistical Machine Translation (SMT) approaches [32, 33, 34] come into the picture to address the limitation of the rule-based approaches by correcting all error types simultaneously [4]. SMT systems leverage statistical models trained on parallel corpora to generate translations by estimating the likelihood of different translations and selecting the most probable one, treating the GEC task as a translation task [4]. The complexity of the SMT approaches, e.g. relying on separate translation and language models, is addressed by neural machine translation (NMT), which consists of a single neural network.

NMT approaches, which achieve state-of-the-art results, are encoder-decoder methods [35] where encoders and decoders could be of different possible architectures such as RNNS [36], CNNS [37], or Transformers [38], which were applied successfully on the GEC task [39, 40, 41]. Recent approaches, utilize pre-trained large language models and achieve state-of-the-art results [42, 43] by only fine-tuning them, solving the data bottleneck requirement for large networks.

2.3 Computer Assisted Language Learning

Computer Assisted Language Learning (CALL) [44] combines foreign language learning and technology by using many fields such as psychology, sociology, natural language processing, linguistics, and artificial intelligence.

CALL has been a key element in the online education systems used during the COVID-19 break. Turkey was among the countries that decided to continue their education online through various platforms during the COVID-19 break, and this process affected especially new high school, preparatory, and freshmen students who received their education in English. The absence of possible face-to-face feedback in this situation urged the usage of CALL.

Writing in a foreign language is an important skill for students to acquire in the preparatory and beginning years of their education. Since feedback is crucial for students to improve their writing skills, a computer-assisted online tool could fill in this gap.

The history of automatic proofreading programs dates back to the 1960s, the core functions of prominent programs such as Project Essay Grade [45] or The Writer's Workbench [46] are still limited by misspellings and short and long sentences. It was known that there was a determination of the students, and this showed the need for programs that provide more comprehensive and detailed feedback for those periods. For this reason, with the effect of technological developments, studies have accelerated in the last 20 years and advanced digital education tools and inspection programs have been created. Vantage Learning, Educational Testing Services, and Pearson Knowledge Technologies are leading companies in the field with programs that can provide detailed feedback to users with numerical values [47]. Programs that control discourse analysis, as well as time and cost advantages, have made significant advances in detailed feedback. MY Access designed by Vantage Learning is a widely used program in US public schools. With the artificial intelligence scoring engine called Intellimetric, it can analyze the writings of users in terms of semantic, syntactic, and discourse features [48]. Criterion, released by Educational Testing Services [49], is another web-based tool used by university students for exams such as GMAT and GRE [50]. The software engine of the tool, e-rater, scores and evaluates the articles by making comparisons with the articles controlled and scored by humans and machines in parallel with Intellimetric [51]. Thanks to natural language processing techniques, e-rater has a wide range of linguistic diagnoses in manuscripts, including grammar, language usage, style, or lexical complexity, thus providing feedback not only on form and accuracy but also on writing style and organization.

Existing automatic writing assessment programs can use dictionaries and sample writings thanks to their advanced technologies and provide progress reports for students and instructors. Given the number of students and the time required, automated spell-checking

programs have become essential, especially during the COVID-19 era, to ease the burden of instructors and provide comprehensive feedback to students. Although the programs designed so far (such as Criterion and Grammarly) provide advanced feedback as desired, they cannot be used for the desired purposes in some countries, e.g. Turkey, because they do not offer completely free use and are not fully suitable for the needs of Turkish students learning English. Based on the studies on the common writing mistakes of Turkish students studying English [52, 53, 54, 55], the most common mistakes made by Turkish students are pluralization, prepositions, pronoun and verb tenses. These studies have shown that existing programs fail to detect these particular errors (eg, fail to detect pluralization and verb tense errors) or perform poorly in detecting such errors (Criterion's accuracy rate is around 70% for prepositions and pronouns).

2.4 Turkish Grammatical Error Correction

Turkish Grammatical Error Correction hasn't been paid as much attention as English GEC. For example, recent work, [56] builds a synthetic dataset considering only a single error type. Later, [5] proposed the first public benchmark dataset of manually annotated 2000 Turkish tweets covering different error types. Then, recently, [6] built and open-sourced the first Turkish GEC synthetic large dataset, by making noisy injections into clean newspaper data, covering 25 error types. Additionally, they released a manually curated test set of 300 movie reviews to the public.

CHAPTER III

DATA METHODOLOGY

This chapter is divided into two sections. First, we detail the procedure followed in developing our Turkish Grammatical Error Correction datasets using our clean insertions method. Then, we describe our data collection, processing, and labeling process for the English Grammatical Error Correction task for native Turkish speakers who learn English as a second language.

3.1 Clean Insertions

3.1.1 OSCAR GEC

Figure 1: A pipeline of the creation process of OSCAR GEC showing all the steps involved in creating the OSCAR GEC dataset

Our data pipeline starts with building a manually annotated spelling dictionary of incorrect-correct 148,932 word and phrase pairs. To build this dictionary, we collected Turkish text from various sources and asked native Turkish speakers to extract incorrect words and

write down their correct versions. We open-source with this work the spelling dictionary and its expanded version comprised of 703,938 pairs, which we introduce in the following sections.

Given the manually created spelling dictionary, we expand our spelling dictionary and build our OSCAR GEC dataset following the pipeline shown in Figure 1. We first use the Turkish OSCAR Dataset to create a word-index dictionary, having each word in the corpus as a key and a list of indexes where the corresponding incorrect word occurs. Then, we apply the following steps: 1) Look up each incorrect word in our spelling dictionary in the word-index OSCAR dictionary. 2) Create a dataset of those texts we extract from OSCAR, each may contain one sentence or more. 3) Create a list of the unique words in the extracted OSCAR texts. 4) Run a word-level Deasciifier 3.1.1.1 and Spell Checker 3.1.1.2 to correct each word of the distinct words if possible. 5) Expand our spelling dictionary with these incorrect-correct pairs. The above steps are repeated until the increase in the size of the spelling dictionary stops.

Table 1 shows the details of each iteration. In the first iteration, the manually created spelling dictionary is used. Starting from the 2nd iteration we notice an increase in the size of the spelling dictionary which leads to an increase in the size of the extracted OSCAR text. In the final iterations, the amount of increase in the spelling dictionary size starts to decrease until it becomes zero and the size of the spelling dictionary stabilizes and does not expand. Finally, all the text extracted from OSCAR is tokenized with a sentence tokenizer, merged, and deduplicated forming 2,326,921 sentences (or a text containing more than one sentence that couldn't be tokenized properly). Since we applied the tokenization later and some of the texts we extracted from OSCAR may composed of more than one sentence, some of those sentences do not necessarily contain any incorrect word or phrase and may be completely correct.

To build our OSCAR GEC dataset, we apply our novel approach clean-insertions,

which involves substituting incorrectly spelled words or phrases with their correct counterparts. We iterate over the final 2,326,921 sentences and for every word in each sentence, we perform a lookup in our expanded spelling dictionary of 703,938 incorrect-correct word pairs and replace the incorrect words with their counterparts. We end up with our OSCAR GEC dataset which contains 2,326,921 parallel sentences of incorrect-correct sentence pairs.

3.1.1.1 Deasciifier

Deasciification is a process used in Turkish natural language processing (NLP) to convert text written in the Turkish language using ASCII characters into its proper form with Turkish-specific characters.

Turkish has specific characters such as ”ı,” ”ş,” ”ğ,” ”ç,” and ”ü” that are not present in the standard ASCII character set. However, due to historical reasons, limitations of older computer systems, or simply out of habit, many texts written in Turkish may use ASCII characters as substitutes for these specific Turkish characters. For instance, ”i” might be used instead of ”ı,” ”s” instead of ”ş,” and so on.

Deasciification algorithms aim to detect and correct these substitutions, transforming the text into its correctly spelled Turkish form. We utilize a Deasciification algorithm [57] that works in the following steps: 1) Generates candidates of all the possible combinations of those characters in a word. 2) Uses a morphological analyzer to analyze each candidate version of the word. 3) Returns a candidate from those that pass the morphological analyzer i.e. are analyzable. However, we only include the words that have a single candidate to make sure that the candidate is indeed the correct version of the word.

3.1.1.2 Spell Checker

We make use of a Spell Checker [58] which generates a list of candidate words by performing various operations such as swapping adjacent letters, deleting letters, replacing letters with different characters, and adding new characters. The algorithm passes those

candidates through a morphological analyzer and returns only the analyzable candidates, similar to the Deasciifier algorithm. And, again similar to the Deasciifier, we only consider the words that have one candidate.

Table 1: Spelling dictionary expansion iterations details, showing for each iteration the size of the spelling dictionary, the number of extracted OSCAR texts and distinct words, and the size difference increase of the spelling dictionary.

Iteration Number	Spelling Dictionary Size	Extracted OSCAR Texts	OSCAR Distinct Words	Spelling Dictionary Size Difference (+)
1st	148,932	864,013	1,852,426	-
2nd	463,072	1,220,251	1,025,942	314,140
3rd	670,319	354.423	1,036,973	207,247
4th	698,008	827,72	459,200	27,689
5th	702,887	12,010	143,546	4,879
6th	703,705	2,711	51,348	818
7th	703,901	416	12,887	196
8th	703,937	52	1,341	36
9th	703,938	1	12	1
10th	703,938	0	0	0

3.1.2 GPT GEC

The emergence of ChatGPT [59] has significantly impacted the field of Natural Language Processing (NLP), marking the start of a new era of language generation and understanding. ChatGPT, based on OpenAI's GPT architecture, has demonstrated remarkable capabilities in generating human-like responses to text inputs across various domains. Its versatile applications span from conversational agents and chatbots to content generation, summarization, and translation tasks.

Moreover, researchers and developers leverage ChatGPT as a benchmark for evaluating language understanding and generation models [60]. It has also been used in many cases as an annotation means to annotate unlabeled data [61, 62], which we do in this work. We randomly sample 100k sentences from our OSCAR GEC dataset and prompt ChatGPT to correct the incorrect sentences, generating a parallel GPT GEC dataset of 100k parallel incorrect-correct sentences.

3.1.3 Datasets Overview

Table 2: An overview of the datasets utilized in this work. The datasets in the top half are synthetic and the bottom ones, the evaluation sets, are humanly annotated. The error type ERRANT refers to the automatic annotation tool ERRANT [1, 2], which automatically annotates parallel sentences with error-type information. Tokens information is based on OpenAI's tokenizer tiktoken with gpt2 encodings

Dataset Name	Split	Sentences	Tokens	Error Types	Domain
OSCAR GEC (ours)	Train	2.3m	213.2m	ERRANT	Web
GPT GEC (ours)	Train	100k	3.6m	ERRANT	Web
GECTurk [6]	Train	138k	5.8m	25	Newspapers
OSCAR GEC (ours)	Test	2.4k	142k	ERRANT	Web
Movie Reviews [6]	Test	300	2.7k	25	Movie Reviews
Turkish Tweets [5]	Test	2k	116.2k	13	Tweets

Table 2 presents a summary of the datasets employed in this study. We utilize three datasets, including two internally developed ones named OSCAR GEC and GPT GEC, which we introduced in previous sections, as well as an open-source dataset called GECTurk [6], which we compare with our datasets.

As for evaluation, we randomly pick 2,408 sentences from our OSCAR GEC dataset, which we exclude from training and validation, and manually annotate them forming a manually annotated evaluation set of organic 2,408 sentences. We also test and benchmark on open-source evaluation sets such as [5], which contains 1,996 tweets, and [6]'s curated evaluation set which includes 300 movie reviews. In Table 3, we show the percentages of the frequencies of the error types found in the three evaluation sets OSCAR GEC (22,366 errors), Turkish Tweets (6,201 errors), and Movie Reviews (227 errors) classified by ERRANT-TR. The Table shows that the OSCAR GEC evaluation set has more error types than the other two.

Table 3: Error Types and their percentages in the evaluation sets mentioned in Table 2 classified by ERRANT-TR

Error Type	OSCAR GEC	Turkish Tweets	Movie Reviews
SPELL	0.4442	0.5175	0.0925
ORTH	0.1131	0.2579	0.5727
OTHER	0.1441	0.1116	0.1894
NOUN	0.0160	0.0155	0.0529
NOUN:INFL	0.0180	0.0108	0.0044
NOUN:NUM	-	0.0026	0.0044
PRON	0.0014	0.0037	-
VERB:INFL	0.0133	0.0363	0.0088
ADJ	0.0059	0.0077	0.0220
CONJ	0.0048	0.0103	0.0352
NUM	0.0047	-	-
DET	0.0020	0.0019	-
QUES	0.0008	0.0015	0.0044
ADJ:POSS	0.0003	-	-
ADJ-VERB:INFL:POSS	0.0004	0.0002	-
ADJ-VERB:INFL:CASE	0.0000	-	-
ADV-VERB:INFL:CASE	0.0001	-	-
ADV	0.0043	0.0065	-
PUNC	0.2070	0.0002	0.0132
VERB:SVA	0.0021	0.0011	-
VERB	0.0064	0.0071	-
PREP	0.0040	0.0044	-
NOUN-VERB:INFL:POSS	0.0001	-	-
VERB:TENSE	0.0022	0.0005	-
WO	0.0014	-	-

3.2 English GEC for Turkish Speakers

3.2.1 Data Collection and Processing

We met with representatives from the English preparatory department of Ozyegin, Bogazici, and Nisantasi University. In those meetings, we discussed the level and the format of the essays to be collected. We acquired essays either handwritten or digitally written. The digitally written ones came in either PDF or Word format. We detail in Table 4 the number of essays and their level and format.

We start the processing phase of the data by removing anything other than the body of

the essay. And in parallel, we spend some time digitizing the handwritten essays. We then tokenize all the essays into sentences.

3.2.2 Data Labelling

Figure 2: An annotation tool developed with Python

We follow the M2 format to annotate the data. In the M2 format [2, 1], a line preceded by S denotes an original sentence while a line preceded by A indicates an edit annotation. Each edit line consists of the start and end token offset of the edit, the error type, and the tokenized correction string. The next two fields are included for historical reasons and can be ignored, while the last field is the annotator id.

A noop edit is a special kind of edit that explicitly indicates an annotator/system made no changes to the original sentence. If there is only one annotator, noop edits are optional, otherwise a noop edit should be included whenever at least 1 out of n annotators considered the original sentence to be correct. This is something to be aware of when combining

Table 4: Number of Compositions by University, Level, and Format

University	Level	Format	Number of Compositions
Nişantaşı	A1	docx, rtf	223
Boğaziçi	A1, A2, B1, B2	docx, rtf	184
Boğaziçi Group1	A1, A2, B1, B2	docx, rtf	183
Boğaziçi Group2	A1, A2, B1, B2	docx, rtf	200
Özyeğin	A2	docx, rtf	46
Özyeğin	A2	pdf (Paper scan)	25 (all digitized)
Özyeğin	B1	docx, rtf	88
Özyeğin	B1	pdf (Paper scan)	140 (11 digitized)
Özyeğin	B2	docx, rtf	10
Özyeğin	B2	pdf (Paper scan)	47

individual M2 files, as missing noops can affect results. To annotate the data with the above-mentioned format, we built an interface for the annotators that is easy to use. To get the annotators to start as soon as possible and also get used to the process, we quickly built an interface using Python's Tkinter. Figure 2 shows our Python annotation tool. With this simple annotation tool, we introduced the problem to the annotators. In the meantime, we were working on a more efficient and faster annotation tool in Java [9], which we show in Figure 3. The tool takes as input a path to a folder containing files each representing a sentence formatted in a specific format. Each annotation information of a sentence is registered into the corresponding file of that sentence.

Because the data collection phase took months, we started with other phases of the project as we were receiving the data. Therefore, all the sentences were not annotated in one batch. Besides, because the handwritten essays came late and digitizing them took some time, they haven't been annotated yet.

The data was annotated in 3 batches [8]. The first batch was annotated with the Python annotation tool, and the second and the third batches with the Java interface. We combined

Figure 3: An annotation tool developed with Java

them all later in one place. We had two annotators, and each annotator annotated the same sentences in batches 2 and 3. Batch 1 on the other hand was different for each annotator. Annotator X annotated 4958 sentences, while annotator Y annotated 4473 sentences. We show in Table 5 the error types found in the data and their frequency percentages both by annotator X, 3286 annotated errors, and annotator Y, annotated 3055 errors.

Table 5: Error Types percentages in checkMate-X and checkMate-Y datasets.

Error Type	Freq. (checkMate-X)	Freq. (checkMate-Y)
ADJ	0.0554	0.0700
PRON	0.0603	0.0511
DET	0.1038	0.0867
NOUN:NUM	0.0603	0.0625
CONJ	0.0520	0.0524
VERB:FORM	0.0204	0.0108
PREP	0.0724	0.1106
VERB	0.1549	0.1486
NOUN	0.0660	0.0864
PART	0.0773	0.0399
PUNCT	0.0849	0.0373
VERB:INFL	0.0386	0.0592
VERB:TENSE	0.0110	0.0190
NOUN:INFL	0.0024	0.0007
ADV	0.0116	0.0137
OTHER	0.0100	0.0010
SPELL	0.0530	0.0471
UNK (unknown)	0.0292	0.0069
ORTH	0.0003	0.0209
ADJ:FORM	0.0037	0.0082
CONTR	0.0012	0.0000
MORPH	0.0012	0.0003
VERB:SVA	0.0295	0.0350
NOUN:POSS	0.0006	0.0000
WO (word order)	0.0000	0.0318

CHAPTER IV

EXPERIMENTAL SETUP

4.1 Turkish GEC

4.1.1 Models

We perform several experiments in this work with the mT5 model [63], a multilingual pre-trained encoder-decoder text-to-text transformer trained on a Common Crawl-based dataset covering 101 languages. We fine-tune the model on three different datasets: the OSCAR GEC, GPT GEC, and GECTurk [6]. Even though we train an mT5 model on the GECTurk dataset, we compare our mT5 models to their model, which is a sequence tagger based on a pre-trained Turkish cased Bert model [64] with extra linear and softmax layers similar to the work done in [65].

We train our models on one NVIDIA GeForce RTX 3090 for 10 epochs while only saving the best three checkpoints. Since some of the sentences extracted from OSCAR are long, we truncate the sentences to a max length of 48.

4.1.2 Evaluation

We evaluate our models and GECturk's sequence tagger on three different evaluation sets listed in Table 2: OSCAR GEC, Movie Reviews, and Turkish Tweets. To automatically annotate the parallel sentences of our evaluation sets and model outputs, we use ERRANT-TR [66], a variant of ERRANT [1, 2] developed for the Turkish language. Table 6 shows the error types, descriptions, and examples defined in the original ERRANT framework, while Table 7 shows the mapped error types in the ERRANT-TR framework.

ERRANT-TR outputs the annotations in M2 format [67], which can be evaluated using ERRANT's evaluation scripts that calculate the F0.5 score given two M2 files: a reference file (i.e. the gold M2 file) and a hypothesis file (i.e. the model output M2 file). ERRANT

Table 6: The list of 25 main error categories in the ERRANT framework with examples and explanations as listed in their work.

Code	Meaning	Description / Example
ADJ	Adjective	big → wide
ADJ:FORM	Adjective Form	Comparative or superlative adjective errors. goodest → best, bigger → biggest, more easy → easier
ADV	Adverb	speedily → quickly
CONJ	Conjunction	and → but
CONTR	Contraction	n't → not
DET	Determiner	the → a
MORPH	Morphology	Tokens have the same lemma but nothing else in common. quick (adj)→ quickly (adv)
NOUN	Noun	person → people
NOUN:INFL	Noun Inflection	Count-mass noun errors. informations → information
NOUN:NUM	Noun Number	cat → cats
NOUN:POSS	Noun Possessive	friends → friend's
ORTH	Orthography	Case and/or whitespace errors. Bestfriend → best friend
OTHER	Other	Errors that do not fall into any other category (e.g. paraphrasing). at his best → well, job → professional
PART	Particle	(look) in → (look) at
PREP	Preposition	of → at
PRON	Pronoun	ours → ourselves
PUNCT	Punctuation	!→.
SPELL	Spelling	genectic → genetic, color → colour
UNK	Unknown	The annotator detected an error but was unable to correct it.
VERB	Verb	ambulate → walk
VERB:FORM	Verb Form	Infinitives (with or without "to"), gerunds (-ing) and participles. to eat → eating, dancing → danced
VERB:INFL	Verb Inflection	Misapplication of tense morphology. getted → got, fliped → flipped
VERB:SVA	Subject-Verb Agreement	(He) have → (He) has
VERB:TENSE	Verb Tense	Includes inflectional and periphrastic tense, modal verbs and passivization. eats → ate, eats → has eaten, eats → can eat, eats → was eaten
WO	Word Order	only can → can only

provides different scoring modes such as span-based correction, span-based detection, and token-based detection.

To sum things up, we follow the following steps in evaluating our models: 1) Generate a parallel tab-separated file of the source-gold sentences. 2) Generate a parallel tab-separated file of the source-model_output sentences. 3) Generate M2 files for the previously mentioned two tab-separated files. 4) Calculate the score of each model on every evaluation set given the corresponding gold and model_output M2 files.

We post-process the Turkish Tweets evaluation set in Table 2 and the model outputs of it for a fair evaluation and comparison. We apply two transformations to it. First, to the evaluation set: 1) We capitalize the first letter of each correct sentence in the dataset since most models are trained with the data that way. Second, to the model outputs: 2) We remove the punctuation marks from the model outputs since they were removed from the evaluation set [5]. We apply the previous two transformations for all models making sure our results and comparisons are accurate.

4.2 English GEC

We take advantage of Google's pre-trained model Text-To-Text Transfer Transformer (T5) [68]. The T5 model's architecture is based on the Transformer architecture, which was first introduced in [38]. The architecture consists of a stack of encoder and decoder layers, each layer consisting of a multi-head self-attention sublayer followed by a feed-forward network sublayer.

The encoder layers process and encode the input sequence into a context representation. The decoder layers then use this context representation to generate the output sequence.

The T5 model is trained on several datasets for 18 different tasks such as Text Summarization and Machine Translation. T5 has been utilized for the Grammar Error Correction task as well. Similarly, we use T5 and fine-tune it on our datasets.

Table 7: ERRANT-TR's Error Codes, Descriptions, and Examples as they list in their work. An empty cell indicates that the category has no example of being either too wide or not useful for Turkish.

Error Code	Meaning	Example
ADJ	Wrong choice of adjective	büyük → küçük
ADJ:FORM	Wrong usage of comparative or superlative adjective	
ADV	Wrong choice of adverb	önce → sonra
CONJ	Wrong choice of conjunction	ama → belki
CONTR	Wrong choice of contraction	
DET	Wrong choice of determiner	bu elma → o elma
MORPH	Tokens have the same lemma but nothing else in common	kalem → silgi
NOUN	Wrong choice of nouns	
NOUN:INFL	Count-mass noun errors	
NOUN:NUM	Wrong usage of noun number	elma → elmalar
NOUN:POSS	Wrong usage of noun possessive	hastalarının ilaçları → hastaların ilaçları
ORTH	Case and/or whitespace errors	herşey → her şey
OTHER	Errors that do not fall into any other category	
PART	Wrong choice of particle	
PREP	Wrong choice of preposition	gibi → için
PRON	Wrong usage of pronoun	sen → ben
PUNCT	Wrong usage of punctuation	? → !
SPELL	Misspelling	broblem → problem
UNK	A detected but not corrected error	
VERB	Wrong choice of verbs	geldim → gittim
VERB:FORM	Infinitives, gerunds and participles	gitmek, gitme, giden
VERB:INFL	Wrong usage of tense morphology	(biz) yaptık → (biz) yaptık
VERB:SVA	Subject-verb agreement	sen geliyorum → sen geliyorsun
VERB:TENSE	Wrong choice of inflectional and periphrastic tense, modal verbs and passivization	geliyorum → gelmiştim
WO	Word order	elma kırmızı → kırmızı elma

CHAPTER V

RESULTS

5.1 Turkish GEC

Table 8 and Table 9 show the precision, recall, and F0.5 scores of correction and detection, respectively, for all models tested on three different test sets. We evaluate four different models on every test set. GECTurk (Seq Tagger), is a sequence tagger trained by [6] on their GECTurk training set. We also fine-tune mT5 on the same dataset, which is the model GECTurk (mT5) in the table. The remaining two models GPT GEC and OSCAR GEC are also fine-tuned mT5 models on our GPT GEC and OSCAR GEC training sets respectively.

Table 8: ERRANT span-based **correction** scores (precision, recall, and F0.5) of every model on all the evaluation sets. All mT5 are trained are ours. GECTurk (Seq Tagger) is trained by the referenced work. The highest value of all models on each metric is bolded per evaluation set.

Model	P	R	F0.5
OSCAR GEC (ours)			
GECTurk (Seq Tagger) [6]	49.0	3.9	14.7
GECTurk (mT5)	42.5	5.7	18.2
GPT GEC (mT5)	**69.8**	**44.9**	**62.8**
OSCAR GEC (mT5)	68.7	31.2	55.4
Turkish Tweets [5]			
GECTurk (Seq Tagger) [6]	64.7	19.8	44.5
GECTurk (mT5)	57.2	20.7	42.3
GPT GEC (mT5)	77.7	**68.9**	75.8
OSCAR GEC (mT5)	**85.1**	61.3	**79.0**
Movie Reviews [6]			
GECTurk (Seq Tagger) [6]	**86.5**	**76.2**	**84.2**
GECTurk (mT5)	73.1	71.8	72.8
GPT GEC (mT5)	36.0	46.3	37.6
OSCAR GEC (mT5)	30.0	22.5	28.1

5.1.1 Correction

In Table 8, which shows the ERRANT span-based correction scores, we notice the poor performance, on the OSCAR GEC test set, of both models trained on the GECTurk dataset achieving an F.05 score of 14.7 by the Sequence Tagger and 18.2 by the fine-tuned mT5. Surprisingly, those models' recall is significantly lower than the recall of the other two models being at most 5.7, which can be interpreted by the fact that the GECTurk dataset covers only 25 error types of all possible error types. Besides, the fact that the fine-tuned mT5 on the GECTurk dataset is doing better than the sequence tagger supports our claim that relying on the knowledge in large models such as mT5 is beneficial. For example, the GECTurk (mT5) model transforms the word Yuzune (to your face), which contains a deasciification error, into Yüzüne, the correct version despite the fact this error type is missing in the GECTurk dataset. On the other hand, GPT GEC and OSCAR GEC models perform significantly better with an F0.5 score of 62.8 and 55.4 respectively.

For the Turkish Tweets test set, the OSCAR GEC model is achieving the highest F0.5 score of 79.0, slightly higher than the GPT GEC model, and significantly higher than the models trained on the GECTurk dataset, which score at most an F0.5 score of 44.5. One of the reasons the OSCAR GEC model is slightly better than the GPT GEC is that GPT sometimes considered hashtags as grammatical mistakes in its annotations and replaced them. Perhaps prompting with this in mind could help in overcoming this problem. We show an example from the Turkish Tweets evaluation set in Figure 4. The OSCAR GEC and GPT GEC models correct the text with a slight difference in punctuation. The sequence Tagger from [6], on the other hand, leaves half of the sentence incorrect and even corrupts the last word.

On the Movie Reviews evaluation set, which is annotated by the same authors who built the GECTurk training set, the GECTurk models achieve noticeably a higher F0.5 score, 84.2 by the Sequence Tagger, than the GPT GEC and OSCAR GEC models which at most score an F0.5 score of 37.6. One of the reasons our models score low here is the

ORIGINAL: benım arkadasım dıye benım halletmem gerekmıyo
(Just because he's my friend, I don't have to handle it.)

OSCAR GEC: Benim arkadaşım diye benim halletmem gerekmiyor

GPT GEC: Benim arkadaşım diye benim halletmem gerekmiyor.

Sequence Tagger: Benım arkadasım dıye benım halletmem gerek miyo?

Figure 4: One example from the Turkish Tweets and the output of the three models OSCAR GEC, GPT GEC, and Sequence Tagger. The red segments are incorrect and the green ones are correct.

annotation inconsistencies in the Movie Reviews evaluation set. For instance, the GPT GEC model capitalizes people's names such as Matt Damon while the evaluation set's gold annotations are sometimes capitalized and sometimes left in lowercase. Another reason is that the evaluation set has wrong annotations such as Matrixten (from the Matrix) which the GPT GEC model corrects as Matrix'ten, its correct version. Another wrong annotation example is the word orjinalinde (in the original one) that GPT GEC corrects as orijinalinde, which is its correct version.

5.1.2 Detection

Table 9: ERRANT span-based **detection** scores (precision, recall, and F0.5) of every model on all the evaluation sets. All mT5 are trained are ours. GECTurk (Seq Tagger) is trained by the referenced work. The highest value of all models on each metric is bolded per evaluation set

Model	P	R	F0.5
OSCAR GEC (ours)			
GECTurk (Seq Tagger) [6]	79.2	6.2	23.7
GECTurk (mT5)	73.0	9.6	31.3
GPT GEC (mT5)	**85.7**	**55.1**	**77.1**
OSCAR GEC (mT5)	82.1	37.3	66.2
Turkish Tweets [5]			
GECTurk (Seq Tagger) [6]	90.5	27.6	62.2
GECTurk (mT5)	85.4	30.9	63.1
GPT GEC (mT5)	92.0	**81.7**	**89.7**
OSCAR GEC (mT5)	**95.3**	68.5	88.4
Movie Reviews [6]			
GECTurk (Seq Tagger) [6]	**90.5**	**79.7**	**88.1**
GECTurk (mT5)	78.5	77.1	78.2
GPT GEC (mT5)	43.2	55.5	45.2
OSCAR GEC (mT5)	34.1	25.6	32.0

The detection results in Table 9 mostly follow the trend of the correction results. On the OSCAR GEC test set, the GPT GEC model scores the highest F0.5 score of 77.1. Besides, the GECTurk models also perform poorly here with at most an F0.5 score of 31.3. The gap

between the correction and detection F0.5 scores is high because, for example, the GPT GEC makes word-choice corrections that aren't wrong but unnecessary or missing in the gold annotations.

On the Turkish Tweets evaluation set, GPT GEC and OSCAR GEC models also significantly do better than GECTurk models with scores F0.5 scores of 89.7 and 88.4 respectively. While the GECTurk models, the Sequence Tagger, and the fine-tuned mT5, only scored 62.2 and 63.1 respectively. The OSCAR GEC model here, again, is more precise than the GPT GEC model because, for example, it does not make as many corrections, e.g. contextual ones, as GPT GEC does.

Finally, the GECTurk models again do well on the Movie Reviews evaluation set with at most an F0.5 score of 88.1 and at least a score of 78.2 by the Sequence Tagger and the fine-tuned mT5 models respectively. GPT GEC and OSCAR GEC, on the other hand, struggle again in detection with at most an F0.5 score of 45.2 by the Sequence Tagger. Again, the OSCAR GEC and the GPT GEC models struggle here for many reasons such as the ones we mentioned in the previous section.

5.2 English GEC

Table 10: Results of the fine-tuned T5 model. checkMate-X and checkMate-Y are the two datasets we introduce each being annotated by a different annotator. The precision, recall, and F0.5 are shown for each experiment.

Training Set	Test Set	TP	FP	FN	Precision	Recall	F0.5
checkMate-X	checkMate-X dev	157	436	498	26.48	23.97	25.93
checkMate-Y	checkMate-Y dev	185	500	574	27.01	24.37	26.44
W&I+LOCNESS	W&I+LOCNESS dev	4938	8200	9984	37.59	**33.09**	36.59
W&I+LOCNESS + checkMate-X	W&I+LOCNESS dev	4474	6790	10448	**39.72**	29.98	**37.30**
W&I+LOCNESS + checkMate-Y	W&I+LOCNESS dev	4206	9938	10716	29.74	28.19	29.41

In our experiments, we do not combine what annotators X and Y annotated and hence we have two different datasets which we will call here checkMate-X and checkMate-Y. We split each dataset into train and test sets. We fine-tune T5 on our training set and use the test

set for evaluation. We use ERRANT [1], which is a toolkit used by the NLP community for producing error types scores for the Grammatical Error Correction (GEC) task. Furthermore, we utilize the dataset W&I+LOCNESS [14, 69], which contains 77384 sentences, and combine it with our data, which we detail in Table 10. Our results show that combining our checkMate-X dataset with the W&I+LOCNESS dataset does improve the performance by 0.71. However, surprisingly, combining W&I+LOCNESS with checkMate-Y causes the F0.5 score to significantly decrease.

CHAPTER VI

LANGUAGE MODELS

The rise of large language models marks a transformative era in artificial intelligence and natural language processing. These models, such as GPT-3 [70] and LLaMA[71] have been given significant attention due to their impressive capabilities in generating human-like text and performing various language-related tasks.

Large language models require billions of tokens, one of the bottlenecks of training large language models for low-resource languages, to achieve high performance on NLP tasks. Most large language models depend on web-based multilingual resources and datasets such as CommonCrawl, C4 [68], and OSCAR.

Unfortunately, web-crawled datasets are noisy and therefore need to be cleaned. For instance, we find that Turkish OSCAR [72], which has around 11.6 million text documents, has at least one spelling mistake in 10% of the documents based on our spelling dictionary.

In this chapter, we train four language models with different settings to show the effectiveness of our clean-insertions method for language models.

6.1 Data Processing

We use our OSCAR GEC parallel dataset which contains around 113.9 million training tokens in the original sentences, and 117.2 million training tokens in the parallel corrected sentences. We add a sample of around 2 million text documents from the untouched Turkish OSCAR samples to the original sentences and the parallel corrected sentences. With the added sample, we end up with two different datasets: 1) the original sentences and the OSCAR sample (305 million training tokens) and 2) the corrected sentences and the OSCAR sample (314.4 million training tokens). All token information is obtained from Karpathy's GPT-2 implementation code [73].

6.2 Training

We train four different GPT-2 models, following Karpathy's GPT-2 implementation, with two different sizes: 30M and 124M. We train the models on one NVIDIA GeForce RTX 3090. We train the 30M size model for 300k iterations, and the 124M size model for 8k iterations. Table 11 shows the training and validation losses. We notice lower training and validation losses for both sizes for the models trained on the corrected sentences and the Turkish OSCAR sample. This indicates the effectiveness of cleaning the Turkish OSCAR dataset with clean insertions using our spelling dictionary on the training losses. Maybe to be certain of this effect, we need to train different architectures other than GPT-2, which we leave as a future work.

Table 11: Trainig and validation losses of 4 different GPT-2 models of two different sizes trained on two dataset combinations having a common Turkish OSCAR sample and either the original sentences of our OSCAR GEC dataset or the corrected sentences of our OSCAR GEC dataset cleaned using our clean insertions method. Bolded values show lower training and validation losses.

Model	Train Loss	Val Loss
Original sentences + Turkish OSCAR sample		
GPT-2 (30M)	2.38	2.39
GPT-2 (124M)	1.85	2.11
Corrected sentences + Turkish OSCAR sample		
GPT-2 (30M)	**2.26**	**2.28**
GPT-2 (124M)	**1.77**	**2.01**

6.3 Evaluation

We also evaluate our GPT models manually to see if there is an effect on the generated text. We generate 50 samples for every model using 50 common Turkish-word prompts and ask 5 evaluators to evaluate the total 200 samples from 1 to 5 based only on their cohesiveness, ignoring the spelling mistakes in the generated text since we observe that the models trained with the misspelled words are already biased towards generating misspelled text.

Table 12 shows the average rating by each annotator, from A1 to A5 for every model. As we see in the table, the clean insertions method did not lead to higher ratings for all models despite its obvious effect on the training losses. This could be because this is a small experiment with only 50 prompts. Or, maybe corrupting misspellings, i.e. correcting them, this way corrupts the context or causes imbalances in the training data.

Table 12: The average ratings of 50 generated texts sampled per model. The samples are rated from 1 to 5 by five annotators (A1-A5).

Model	A1	A2	A3	A4	A5
Original sentences + Turkish OSCAR sample					
GPT-2 (30M)	**3.6**	**2.84**	**3.76**	**3.12**	**3.74**
GPT-2 (124M)	2.96	**2.94**	**3.82**	**3.36**	3.54
Corrected sentences + Turkish OSCAR sample					
GPT-2 (30M)	3.06	2.78	3.58	2.64	3.44
GPT-2 (124M)	**3.1**	2.74	3.68	2.96	**3.74**

CHAPTER VII

CONCLUSION

In summary, this study addresses the lack of attention within the research community to the Turkish Grammatical Error Correction task by introducing a method, clean insertions, that helps in creating organic Turkish GEC datasets. We open-source several datasets two of which are training datasets and one evaluation set. In addition to that, we share our models that achieve state-of-the-art results on two evaluation sets out of the three available evaluation sets.

Our method, clean insertions, is simple to understand and apply. Other than the starting spelling dictionary that we build manually, it is fully automated. Normally, a synthetic dataset requires clean data to start with, which may not be available, however, our method works with any organic data, which usually contains grammatical errors. This leads to datasets that contain various types of errors and not only a set of pre-defined injected error types, which could cause the models trained on such datasets to perform poorly on evaluation sets containing error types out of the pre-defined set as we show in chapter 5.

Even though our method yields partially correct parallel GEC datasets since the spelling dictionary would not contain all possible errors, it can be used to obtain state-of-the-art results by relying on the knowledge in the large pre-trained models such as mT5. This finding is surprising and raises the question of whether we can solve other tasks the same way with partially correct or partially correctly annotated datasets. Certainly, such datasets would confuse the models in tasks such as text classification, but it is maybe worth trying for tasks that can be formulated as text-to-text problems.

In addition to the dataset we build using clean insertions, OSCAR GEC, we use GPT as an annotator and build a GEC dataset, GPT GEC, to show the potential of using such

models as annotators. Indeed, the models trained on the GPT GEC show promising results surpassing the other models on most evaluation sets as we show in Table 8 and 9.

While this study provides valuable insights, it is important to acknowledge its limitations, such as the need for a manually annotated spelling dictionary to start with. Another limitation is the lack of a clear and specific error-type annotation. Since we use ERRANT, which automatically classifies the errors, the set of error types is limited and not detailed and specialized enough for the Turkish language.

Future work could focus more on using other components or even more complex and context-aware components in addition to the Spelling Checker and Deasciifier that we utilize in our OSCAR GEC pipeline. With more components, there would be more incorrect-correct pairs to add to the spelling dictionary, which could lead to higher-quality datasets. Besides, applying the approach to datasets other than Turkish OSCAR could enrich the OSCAR GEC dataset with examples that contain missing error types in the current dataset.

We also build a Grammatical Error Correction datasets capturing mistakes Turkish students tend to make when writing in English. We fine-tune with those datasets a pre-trained model, T5. Future work could compare and merge our two datasets checkMate-X and checkMate-Y into a single dataset. Furthermore, investigate the dataset more to highlight the errors that are commonly made by Turkish native students.

REFERENCES

[1] C. Bryant, M. Felice, and T. Briscoe, "Automatic annotation and evaluation of error types for grammatical error correction," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (R. Barzilay and M.-Y. Kan, eds.), (Vancouver, Canada), pp. 793–805, Association for Computational Linguistics, July 2017.

[2] M. Felice, C. Bryant, and T. Briscoe, "Automatic extraction of learner errors in esl sentences using linguistically enhanced alignments," in *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pp. 825–835, 2016.

[3] H. Face, "OSCAR: Open Super-large Crawled ALMAnaCH coRpus." `https://huggingface.co/datasets/oscar`, 2019. Accessed: 2024-05-29.

[4] C. Bryant, Z. Yuan, M. R. Qorib, H. Cao, H. T. Ng, and T. Briscoe, "Grammatical error correction: A survey of the state of the art," *Computational Linguistics*, vol. 49, no. 3, pp. 643–701, 2023.

[5] A. T. Koksal, O. Bozal, E. Yürekli, and G. Gezici, "#turki$hTweets: A benchmark dataset for Turkish text correction," in *Findings of the Association for Computational Linguistics: EMNLP 2020* (T. Cohn, Y. He, and Y. Liu, eds.), (Online), pp. 4190–4198, Association for Computational Linguistics, Nov. 2020.

[6] A. Kara, F. M. Sofian, A. Bond, and G. G. Şahin, "Gecturk: Grammatical error correction and detection dataset for turkish," in *Findings of the Association for Computational Linguistics: IJCNLP-AACL 2023 (Findings)*, pp. 278–290, 2023.

[7] A. Ersoy, "Turkish-GEC: Turkish Grammatical Error Correction." `https://github.com/asimokby/Turkish-GEC`, 2024. Accessed: 2024-05-29.

[8] A. Ersoy, O. Savlak, C. Kart, and O. T. Yıldız, "Checkmate: English grammatical error correction for native turkish speakers," in *2023 31st Signal Processing and Communications Applications Conference (SIU)*, pp. 1–4, IEEE, 2023.

[9] S. Software, "GrammaticalErrorCorrection: A Collection of Resources for Grammatical Error Correction." `https://github.com/StarlangSoftware/GrammaticalErrorCorrection`, 2022. Accessed: 2024-05-29.

[10] A. Ersoy, "Model for english grammatical error correction for native turkish speakers." `https://huggingface.co/asimokby/checkMate-gec`, 2022. Accessed: 2024-05-29.

[11] H. Yannakoudakis and T. Briscoe, "Modeling coherence in esol learner texts," in *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*, pp. 33–43, 2012.

[12] D. Dahlmeier, H. T. Ng, and S. M. Wu, "Building a large annotated corpus of learner english: The nus corpus of learner english," in *Proceedings of the eighth workshop on innovative use of NLP for building educational applications*, pp. 22–31, 2013.

[13] C. Napoles, K. Sakaguchi, and J. Tetreault, "Jfleg: A fluency corpus and benchmark for grammatical error correction," *arXiv preprint arXiv:1702.04066*, 2017.

[14] C. Bryant, M. Felice, Ø. E. Andersen, and T. Briscoe, "The bea-2019 shared task on grammatical error correction," in *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pp. 52–75, 2019.

[15] S. Flachs, O. Lacroix, H. Yannakoudakis, M. Rei, and A. Søgaard, "Grammatical error correction in low error density domains: A new benchmark and analyses," *arXiv preprint arXiv:2010.07574*, 2020.

[16] C. Crawl, "Common crawl dataset." `https://commoncrawl.org/`, 2007. Accessed: 2024-05-29.

[17] B. Mohit, A. Rozovskaya, N. Habash, W. Zaghouani, and O. Obeid, "The first qalb shared task on automatic text correction for arabic," in *Proceedings of the EMNLP 2014 Workshop on Arabic Natural Language Processing (ANLP)*, pp. 39–47, 2014.

[18] L.-H. Lee, Y.-H. Tseng, and L.-P. Chang, "Building a tocfl learner corpus for chinese grammatical error diagnosis," in *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, pp. 1–7, 2018.

[19] S. Kiyono, J. Suzuki, M. Mita, T. Mizumoto, and K. Inui, "An empirical study of incorporating pseudo data into grammatical error correction," *arXiv preprint arXiv:1909.00502*, 2019.

[20] N. Ehsan and H. Faili, "Grammatical and context-sensitive error correction using a statistical machine translation framework," *Software: Practice and Experience*, vol. 43, no. 2, pp. 187–206, 2013.

[21] J. Lichtarge, C. Alberti, S. Kumar, N. Shazeer, N. Parmar, and S. Tong, "Corpora generation for grammatical error correction," *arXiv preprint arXiv:1904.05780*, 2019.

[22] W. Zhao, L. Wang, K. Shen, R. Jia, and J. Liu, "Improving grammatical error correction via pre-training a copy-augmented architecture with unlabeled data," *arXiv preprint arXiv:1903.00138*, 2019.

[23] A. Rozovskaya and D. Roth, "Training paradigms for correcting errors in grammar and usage," in *Human language technologies: The 2010 annual conference of the north american chapter of the association for computational linguistics*, pp. 154–162, 2010.

[24] M. Felice and Z. Yuan, "Generating artificial errors for grammatical error correction," in *Proceedings of the Student Research Workshop at the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 116–126, 2014.

[25] M. Rei, M. Felice, Z. Yuan, and T. Briscoe, "Artificial error generation with machine translation and syntactic patterns," *arXiv preprint arXiv:1707.05236*, 2017.

[26] Z. Xie, G. Genthial, S. Xie, A. Y. Ng, and D. Jurafsky, "Noising and denoising natural language: Diverse backtranslation for grammar correction," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 619–628, 2018.

[27] D. Naber, *A Rule-Based Style and Grammar Checker*. GRIN Verlag Munich, Germnay, 2003.

[28] J. S. Lee, "Automatic article restoration," in *Proceedings of the Student Research Workshop at HLT-NAACL 2004*, pp. 31–36, 2004.

[29] M. Chodorow, J. Tetreault, and N.-R. Han, "Detection of grammatical errors involving prepositions," in *Proceedings of the fourth ACL-SIGSEM workshop on prepositions*, pp. 25–30, 2007.

[30] G. Berend, V. Vincze, S. Zarrieß, and R. Farkas, "Lfg-based features for noun number and article grammatical errors," in *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*, pp. 62–67, 2013.

[31] J. S. Lee and S. Seneff, "Correcting misuse of verb forms," in *Proceedings of ACL-08: HLT*, pp. 174–182, 2008.

[32] C. Brockett, W. B. Dolan, and M. Gamon, "Correcting ESL errors using phrasal SMT techniques," in *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics* (N. Calzolari, C. Cardie, and P. Isabelle, eds.), (Sydney, Australia), pp. 249–256, Association for Computational Linguistics, July 2006.

[33] T. Mizumoto, M. Komachi, M. Nagata, and Y. Matsumoto, "Mining revision log of language learning sns for automated japanese error correction of second language learners," in *Proceedings of 5th International Joint Conference on Natural Language Processing*, pp. 147–155, 2011.

[34] Z. Yuan and M. Felice, "Constrained grammatical error correction using statistical machine translation," in *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*, pp. 52–61, 2013.

[35] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[36] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[37] J. Gehring, M. Auli, D. Grangier, and Y. N. Dauphin, "A convolutional encoder model for neural machine translation," *arXiv preprint arXiv:1611.02344*, 2016.

[38] A. Vaswani, 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*, vol. 30, pp. 1–15, 2017.

[39] Z. Yuan and T. Briscoe, "Grammatical error correction using neural machine translation," in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 380–386, 2016.

[40] Z. Yuan, F. Stahlberg, M. Rei, B. Byrne, and H. Yannakoudakis, "Neural and fst-based approaches to grammatical error correction," in *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pp. 228–239, 2019.

[41] M. Junczys-Dowmunt, R. Grundkiewicz, S. Guha, and K. Heafield, "Approaching neural grammatical error correction as a low-resource machine translation task," *arXiv preprint arXiv:1804.05940*, 2018.

[42] S. Rothe, J. Mallinson, E. Malmi, S. Krause, and A. Severyn, "A simple recipe for multilingual grammatical error correction," *arXiv preprint arXiv:2106.03830*, 2021.

[43] M. Tarnavskyi, A. Chernodub, and K. Omelianchuk, "Ensembling and knowledge distilling of large sequence taggers for grammatical error correction," *arXiv preprint arXiv:2203.13064*, 2022.

[44] M. Thomas, H. Reinders, and M. Warschauer, *Contemporary Computer-Assisted Language Learning*. Contemporary Studies in Linguistics, Bloomsbury Publishing, 2012.

[45] E. B. Page, "Project essay grade: Peg," in *Automated Essay Scoring: A Cross-Disciplinary Perspective* (M. D. Shermis and J. Burstein, eds.), pp. 43–54, Lawrence Erlbaum Associates Publishers, 2003.

[46] N. Macdonald, L. Frase, P. Gingrich, and S. Keenan, "The writer's workbench: Computer aids for text analysis," *IEEE Transactions on Communications*, vol. 30, no. 1, pp. 105–110, 1982.

[47] S. Dikli, "An overview of automated scoring of essays," *The Journal of Technology, Learning and Assessment*, vol. 5, pp. 1–36, Aug. 2006.

[48] S. M. Elliot, "Intellimetric: From here to validity," in *Automated Essay Scoring: A Cross-Disciplinary Approach* (M. D. Shermis and J. Burstein, eds.), pp. 71–86, Lawrence Erlbaum Associates, Inc., 2003.

[49] M. Warschauer and P. Ware, "Automated writing evaluation: Defining the classroom research agenda," *Language teaching research*, vol. 10, no. 2, pp. 157–180, 2006.

[50] J. Burstein, M. Chodorow, and C. Leacock, "Automated essay evaluation: The criterion online writing service," *Ai magazine*, vol. 25, no. 3, pp. 27–27, 2004.

[51] Y. Attali and J. Burstein, "Automated essay scoring with e-rater® v. 2.0," *ETS Research Report Series*, vol. 2004, no. 2, pp. i–21, 2004.

[52] Y. Kırkgöz, "A case study of teachers' implementation of curriculum innovation in english language teaching in Turkish primary education," *Teaching and teacher education*, vol. 24, no. 7, pp. 1859–1875, 2008.

[53] Z. Ç. Köroğlu, "An analysis on grammatical errors of Turkish efl students'written texts.," *Electronic Turkish Studies*, vol. 9, no. 12, pp. 101–111, 2014.

[54] S. Tasci and B. Aksu Ataç, "Written grammatical errors of Turkish adult learners of english: An analysis.," *Online Submission*, vol. 4, no. 1, pp. 1–13, 2018.

[55] O. R. Erkaya, "Vocabulary and l1 interference: Error analysis of Turkish students," *Literacy Issues in Higher Education*, vol. 36, no. 2, pp. 1–11, 2012.

[56] U. Arikan, O. Güngör, and S. Uskudarli, "Detecting clitics related orthographic errors in turkish," in *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*, pp. 71–76, 2019.

[57] S. Software, "Turkish Deasciifier." `https://github.com/StarlangSoftware/TurkishDeasciifier`, 2018. Accessed: 2024-05-29.

[58] S. Software, "Turkish SpellChecker." `https://github.com/StarlangSoftware/TurkishSpellChecker`, 2018. Accessed: 2024-05-29.

[59] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, *et al.*, "Training language models to follow instructions with human feedback," *Advances in neural information processing systems*, vol. 35, pp. 27730–27744, 2022.

[60] J. Wang, Y. Liang, F. Meng, Z. Sun, H. Shi, Z. Li, J. Xu, J. Qu, and J. Zhou, "Is chatgpt a good nlg evaluator? a preliminary study," *arXiv preprint arXiv:2303.04048*, 2023.

[61] F. Gilardi, M. Alizadeh, and M. Kubli, "Chatgpt outperforms crowd workers for text-annotation tasks," *Proceedings of the National Academy of Sciences*, vol. 120, no. 30, p. e2305016120, 2023.

[62] Y. Zhu, P. Zhang, E.-U. Haq, P. Hui, and G. Tyson, "Can chatgpt reproduce human-generated labels? a study of social computing tasks," *arXiv preprint arXiv:2304.10145*, 2023.

[63] L. Xue, N. Constant, A. Roberts, M. Kale, R. Al-Rfou, A. Siddhant, A. Barua, and C. Raffel, "mt5: A massively multilingual pre-trained text-to-text transformer," *arXiv preprint arXiv:2010.11934*, 2020.

[64] S. Schweter, "Berturk - bert models for turkish." https://doi.org/10.5281/zenodo.3770924, Apr. 2020. Accessed: 2024-05-29.

[65] K. Omelianchuk, V. Atrasevych, A. Chernodub, and O. Skurzhanskyi, "GECToR – grammatical error correction: Tag, not rewrite," in *Proceedings of the Fifteenth Workshop on Innovative Use of NLP for Building Educational Applications* (J. Burstein, E. Kochmar, C. Leacock, N. Madnani, I. Pilán, H. Yannakoudakis, and T. Zesch, eds.), (Seattle, WA, USA → Online), pp. 163–170, Association for Computational Linguistics, July 2020.

[66] H. Uz and G. Eryiğit, "Towards automatic grammatical error type classification for Turkish," in *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics: Student Research Workshop* (E. Bassignana, M. Lindemann, and A. Petit, eds.), (Dubrovnik, Croatia), pp. 134–142, Association for Computational Linguistics, May 2023.

[67] D. Dahlmeier and H. T. Ng, "Better evaluation for grammatical error correction," in *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 568–572, 2012.

[68] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *Journal of machine learning research*, vol. 21, no. 140, pp. 1–67, 2020.

[69] S. Granger, "The computer learner corpus: a versatile new source of data for sla research," in *Learner English on computer*, pp. 3–18, Routledge, 2014.

[70] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[71] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, *et al.*, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[72] Hugging Face Datasets, "OSCAR Turkish Unshuffled Deduplicated Dataset." https://huggingface.co/datasets/oscar/viewer/unshuffled_deduplicated_tr, 2019. Accessed: 2024-05-29.

[73] A. Karpathy, "nanoGPT: Minimalistic GPT models and corresponding utilities." https://github.com/karpathy/nanoGPT, 2023. Accessed: 2024-05-29.

VITA

Name: Asım Ersoy

Education:

- Master of Science, Computer Science, Ozyegin University (Expected June 2024)
- Bachelor's Degree, Computer Science, Marmara University (July 2021)

Experience:

- Senior Research Engineer, Huawei, Istanbul, Turkey (Dec. 2023 – Present)
- Research Engineer, Huawei, Istanbul, Turkey (Jun. 2022 – Dec. 2023)
- Research Assistant, Ozyegin University, Turkey (Oct. 2021 – Nov. 2022)

Publications:

- Ersoy, A., Vizcarra, G., Mayeesha, T. T., & Muller, B. (2023). *In What Languages are Generative Language Models the Most Formal? Analyzing Formality Distribution across Languages.* EMNLP Findings 2023
- Ersoy, A., Yıldız, O. T., & Özer, S. (2023, July). *ORTPiece: An ORT-Based Turkish Image Captioning Network Based on Transformers and WordPiece.* 2023 31st Signal Processing and Communications Applications Conference (SIU) (pp. 1-4). IEEE.
- Ersoy, A., Savlak, O., Kart, C., & Yıldız, O. T. (2023, July). *CheckMate: English Grammatical Error Correction for Native Turkish Speakers.* 2023 31st Signal Processing and Communications Applications Conference (SIU) (pp. 1-4). IEEE.