GECTurk WEB: An Explainable Online Platform for Turkish Grammatical Error Detection and Correction

Ali Gebeşçe[1,2], **Gözde Gül Şahin**[1,2]

[1]Computer Engineering Department, Koç University, Istanbul, Turkey
[2] KUIS AI Lab, Istanbul, Turkey
https://gglab-ku.github.io/

Abstract

Sophisticated grammatical error detection/correction tools are available for a small set of languages such as English and Chinese. However, it is not straightforward—if not impossible—to adapt them to morphologically rich languages with complex writing rules like Turkish which has more than 80 million speakers. Even though several tools exist for Turkish, they primarily focus on spelling errors rather than grammatical errors and lack features such as web interfaces, error explanations and feedback mechanisms. To fill this gap, we introduce GECTURK WEB, a light, open-source, and flexible web-based system that can detect and correct the most common forms of Turkish writing errors, such as the misuse of diacritics, compound and foreign words, pronouns, light verbs along with spelling mistakes. Our system provides native speakers and second language learners an easily accessible tool to detect/correct such mistakes and *also to learn from their mistakes* by showing the explanation for the violated rule(s). The proposed system achieves 88,3 system usability score, and is shown to help learn/remember a grammatical rule (confirmed by 80% of the participants). The GECTURK WEB is available both as an offline tool [1] or at www.gecturk.net.

1 Introduction

Grammatical Error Correction/Detection (GEC/D) (Bryant et al., 2023) is a well-established NLP task, that aims to detect and correct various errors in text, including grammatical issues like missing prepositions, mismatched subject-verb agreement, as well as orthographic and semantic errors such as misspellings and inappropriate word choices. Tools that can perform GEC/D have recently gained attention due to the rise in digital communication, remote work, and global interactions, which demand clear and professional

writing. With the inclusion of the detection module, GEC/D formulation facilitates the *teaching of grammar rules*, empowering users not only to produce error-free writing but also to enhance their language skills and comprehension gradually.

Therefore, developing open-source GEC/D tools is particularly crucial, yet challenging for languages with complex writing rules, such as Turkish. The writing rules for such languages generally involve multiple linguistic layers—phonetic, syntactic, and semantic—which makes them difficult to follow and remember even for native speakers. While several tools exist for high-resource languages such as GECko+ (Calò et al., 2021) and ALLECS (Qorib et al., 2023), they often suffer from discontinuation of support or lack adaptability for languages such as Turkish. Moreover, while advanced commercial tools such as LanguageTool[2] offer support for 31 languages, yet Turkish is notably absent from their list. Furthermore, as highlighted in § 2, numerous offline tools are accessible for Turkish spelling correction, whereas only two models (not tools) (Uz and Eryiğit, 2023; Kara et al., 2023) are dedicated to Turkish GEC/D.

To bridge this gap, we leverage the state-of-the-art pretrained GEC/D (Kara et al., 2023)[3] and spelling correction models; and, for the first time, provide a user-friendly web-interface to them. Our system does not only correct errors but also display them in different colors, while providing explanations for each correction through interactive elements in the interface. Additionally, the system includes a feedback mechanism to foster continuous improvement and enhance user engagement. Our system is lightweight and flexible, allowing easy adaptation to other languages through pretrained sequence tagging models. The results of

---

[1] https://github.com/GGLAB-KU/gecturkweb

[2] https://languagetool.org/

[3] We use the pretrained sequence tagging model that has been trained on 130,000 high-quality sentences covering more than 20 expert-curated grammar rules (a.k.a., writing rules) implemented through complex transformation functions.
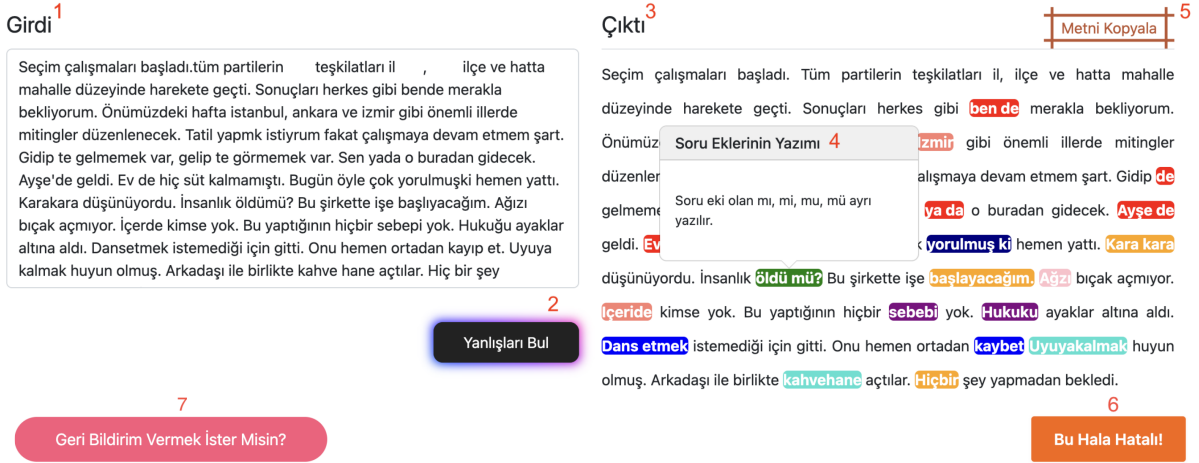
GECTurk

Figure 1: The screenshot of UI after user entering an input. **1-** Girdi (Input): The input area for the user. **2-** Yanlışları Bul (Find Errors): A button which is pressed after entering an input. **3-** Çıktı (Output): The output area for the tagged and corrected text. Note that each error is categorized (colored) according to Table 2. **4-** Pop-up: Each corrected word is represented as button. When clicked the violated rule, i.e., error type, is shown. **5-** Metni Kopyala (Copy Text): A button for copying corrected text. **6-** Bu Hala Hatalı (Still Erroneous): A button for giving feedback in case the user thinks the output still contains errors. When clicked, a pop-up is shown and user is expected to write the corrected version. **7-** Geri Bildirim Vermek İster Misin? (Give Feedback): A button for collecting general suggestions.

the user study (see §4) demonstrate excellent usability and a significant impact on learning and retention of grammar rules. GECTURK WEB, shown in Fig. 1, is accessible both as an offline tool and online at www.gecturk.net and source code licensed with CC BY-SA 4.0 is available at https://github.com/GGLAB-KU/gecturkweb.

2 Previous Systems

High-Resource Languages: Numerous GEC/D models exist for high-resource languages such as English (Lai et al., 2022; Tarnavskyi et al., 2022; Sorokin, 2022; Qorib et al., 2022a) and Chinese (Ren et al., 2018; Qiu and Qu, 2019; Wu and Wu, 2022; Xu et al., 2022). However, these models lack user interfaces, which are crucial for accessibility to non-specialists. Although fewer in number compared to models, several GEC/D tools are available. For instance, GECko+ (Calò et al., 2021) integrates the GECToR XLNet model for sentence-level grammatical correction with a sentence ordering model (Prabhumoye et al., 2020). It processes texts by segmenting them into sentences, applying corrections, and then reordering them. Initially, GECko+ offered a web interface, but it is currently inactive. Now, the only access is through download-

ing the source code and running it locally, which is inconvenient for general users. Similarly, MiSS (Li et al., 2021), a Multi-Style Simultaneous Translation system that includes a GEC/D feature using GECToR XLNet, initially had a web interface which is now inactive.

The most recent non-commercial GEC/D tool is ALLECS (Qorib et al., 2023), which uses GECToR-RoBERTa, GECToR-XLNet, and T5-Large models, alongside two combination methods: ESC (Qorib et al., 2022b) and MEMT (Heafield and Lavie, 2010). ALLECS takes input and displays corrected errors with clickable buttons, and has an easy-to-use web interface. Despite its advantages, ALLECS lacks a feedback mechanism and an enhanced interface that uses color coding to distinguish between different types of errors. Moreover, its implementation is not flexible enough to be extended to other languages, i.e., one cannot simply upload a Turkish GEC/D model and expect the application to function without significant modifications to the source code.

Morphologically Rich Languages: In the case of morphologically rich languages, there are fewer GEC/D models available. Examples include Arabic (Solyman et al., 2022), Bengali (Hossain et al.,

	Spelling	Offline	Open Source	Grammatical	Explanation	Feedback	Web Interface
Google Docs	✓						✓
Microsoft Word	✓	✓					✓
Zemberek (Akın, 2017)	✓	✓	✓				
Hunspell (Zafer, 2017)	✓	✓	✓				
TurkishNLP(Çetinkaya, 2018)	✓	✓	✓				
TrNLP (Bayol, 2018)	✓	✓	✓				
Starlang Yıldız (2019)	✓	✓	✓				
VNLP (Turker, 2021)	✓	✓	✓				✓
Mukayese (Safaya et al., 2022)	✓	✓	✓				
Rule-based (Uz and Eryiğit, 2023)	✓	✓	✓	✓	✓		
GECTurk (Kara et al., 2023)		✓	✓	✓	✓		
GECTurk WEB (Ours)	✓	✓	✓	✓	✓	✓	✓

Table 1: Comparison of features in previous grammatical and spelling error correction tools for Turkish, contrasted with ours. Spelling: Correction of spelling errors. Grammatical: Detection of grammatical errors. Explanation: Explanations for error types. Feedback: User feedback mechanism for model and interface improvement. Web Interface: Availability of a web-based interface.

2024), Czech (Náplava and Straka, 2019; Náplava et al., 2022), and Russian (Rozovskaya and Roth, 2019). However, again these systems lack user interfaces, making them merely as models rather than practical tools, thus limiting their usability for general users. One exception exists in Arabic; however, this tool just underlines mistakes (Alkhatib et al., 2020) and not explain the errors. Also it lacks a web support, making it less suitable for general users.

Commercial Tools: Grammarly[4] offers advanced features for improving writing tone on several aspects like clarity, engagement, and delivery. However, it is not open-source and supports only English. Also, full access to its features requires a paid subscription [5]. LanguageTool, being open-source, supports multiple languages and addresses some of Grammarly's limitations. However, it imposes a 10,000-character limit on inputs, expandable only through a paid subscription [6]. More importantly, despite supporting 31 languages [7], Turkish is not among them.

Turkish: Since aforementioned systems are either commercial or not directly applicable to Turkish GEC/D, we have surveyed commonly available tools and resources that offer support for Turkish, given in Table 1. Google Docs [8] and Microsoft Word [9], widely accessible for their user-friendly interfaces, provide basic spelling error detection. However, they fall short in addressing the specific grammatical nuances of the Turk-

ish language. Additionally, these tools are not open-source, lack explanations for corrections, and do not offer a mechanism for user feedback. There are also open-source tools, such as Zemberek (Akın, 2017), Hunspell (Zafer, 2017), TurkishNLP (Çetinkaya, 2018), TrNLP (Bayol, 2018), StarlangSoftware (Yıldız, 2019), VNLP (Turker, 2021) and MukayeseSpellChecker (Safaya et al., 2022), however they only provide an offline spelling. To the best of our knowledge, there are only two resources for Turkish GEC/D (Uz and Eryiğit, 2023; Kara et al., 2023). Uz and Eryiğit (2023) propose a rule-based, offline GED system that employs common, universal error types (Bryant et al., 2017), while Kara et al. (2023) provide several pre-trained GEC and GED models that can detect expert-curated language specific writing rules and show significant improvements on existing and proposed benchmarks. In this work, we combine the state-of-the-art GEC/D (Kara et al., 2023) and spelling correction models; and, for the first time, provide a user-friendly web-interface to them. Additionally, we provide colorful explanations for a wide range of error types to train the users, and incorporate a feedback mechanism for continuous training of pre-trained models.

3 GECTURK WEB

Our system has four main components: i) frontend, ii) backend, iii) grammatical error correction/detection (GEC/D), and iv) spelling correction modules. GECTURK WEB is based on the Python Django framework,[10] which manages everything related to performance, security, scalability, and database handling. The architecture of our system, incorporating these components along with the data flow,

---

[4] https://www.grammarly.com/
[5] https://www.grammarly.com/plans
[6] https://languagetool.org/premium_new
[7] https://dev.languagetool.org/languages
[8] https://docs.google.com
[9] https://www.microsoft.com/word

[10] https://www.djangoproject.com

Category	Rule ID	Description	Example Correction	Color
-DE/-DA	1. CONJ_DE_SEP	Conjunction "-de/-da" is written separately.	Durumu [oğlunada → oğluna da] bildirdi.	Red
-KI	7. CONJ_KI_SEP	Conjunction "-ki" is written separately.	Bugün öyle çok [yorulmuşki → yorulmuş ki] hemen yattı.	Navy
FOREIGN	9. FOREIGN_R1	Words that start with double consonants of foreign origin are written without adding an "-i" between the letters.	[gıram → gram]	Purple
BISYL	13. BI-SYLL_HAPL_VOW	Some bisyllabic words undergo haplology when they get a suffix starting with a vowel.	[ağızı → ağzı]	Pink
LIGHT VERB	17. LIGHT_VERB_SEP	Light verbs such as "etmek, edilmek, eylemek, olmak, olunmak" are written separately in case of no phonological assimilation	[arzetmek → arz etmek]	Blue
COMPOUND	20. COMP_VERB_ADJ	Compound words formed by knowing, giving, staying, stopping, coming, and writing are written adjacent if they have a suffix starting with -a, -e, -ı, -i, -u, -ü.	[uyuya kalmak → uyuyakalma], [gide durmak → gidedurmak]	Turquoise
SINGLE	22. PRONOUN_EXC	Traditionally, some pronouns are written adjacent.	[hiç bir → hiçbir], [her hangi → herhangi]	Orange

Table 2: A selection of grammatical error types covered in the system from Kara et al. (2023).

is shown in Figure 2.

3.1 Frontend

For the user interface, we use the Bootstrap framework [11] that provides us with modern, responsive, and mobile compatible HTML and CSS. Initially, empty "Input" and "Output" fields are shown. After identifying and correcting grammatical and spelling errors in the input, the output is enriched with error types (see Figure 1). For each correction, HTML snippets are created to wrap the corrected words and transforms them to actionable buttons. These snippets use Bootstrap's pop-over functionality to provide an interactive way to display the error type, an explanation, and the correction. Each correction is highlighted with a specified background color and font size for visibility. Additional information about each error type is retrieved from a predefined set of rules given in Table 2[12]. This information includes a textual explanation and a title for the error, which are both used in the content of the pop-over. For instance, if there is a misspelling of "-de/da", this is displayed as *Conjunction "-de/da" is always written separately*. The tokens within the input text are replaced with the generated HTML snippets, respecting the origi-

nal positions of errors. This involves calculating the offsets to accurately place the HTML snippets within the text, considering the length of the corrected phrases. The corrected tokens are joined back together into strings for each line, and then all lines are combined into a single HTML paragraph (<p> tags).

3.2 Backend

Our system uses Django, a high-level Python web framework, to create a strong backend infrastructure. The architecture of Django, known as Model-View-Template (MVT), supports a clear separation of responsibilities. Here, the Model is responsible for data storage and retrieval. The View handles user requests and provides responses, and the Template dynamically generates HTML pages for user interaction.

View The send_data function is used for accommodating various actions including text submission for correction, feedback submission, and API interactions. Upon receiving a POST request given the input text, the function invokes a text correction process through get_text_corrector. Text correction process starts with sentence tokenization using NLTK's sent_tokenize function (Bird et al., 2009) and continues with the grammatical error correction process, which is described in detail in §3.3. The corrected text, alongside original input and HTML-formatted output for interactive display, is then encapsulated within a TEXT model instance for persistence. Feedback submission, whether specific to text corrections or general website feedback, is similarly processed and stored.

Model Our data model chas two main entities: TEXT and GENERALFEEDBACK. The TEXT model captures the essence of each correction session, storing original and corrected texts, HTML-tagged corrected text for frontend display, and any user feedback. This allows for a comprehensive audit trail of user interactions and system outputs. The GENERALFEEDBACK model, on the other hand, aggregates general user impressions and feedback about the website, enabling continuous improvement based on user insights.

Database and Server Thanks to Django's ORM capabilities, we easily integrate these models with our MySQL[13] database, as the database manage-

---

[11] https://getbootstrap.com

[12] We refer the readers to Kara et al. (2023) for details on each writing rule and how they are handled by the model.

[13] https://www.mysql.com/

ment system. We use AWS Elastic Beanstalk[14] for deployment.

3.3 Grammatical Correction

We employ the state-of-the-art GEC/D model, SequenceTagger, previously described in Kara et al. (2023). Briefly, SequenceTagger finetunes a strong encoder model (e.g., BERTurk (Schweter, 2020)) to classify tokens into grammatical error classes, enabling efficient error detection rather than merely correction. For illustrative purposes, we provide one sample error type from each category in Table 2. Then, corrections are performed with reverse transformations. The model weights and associated files, such as the tokenizer and vocabulary, are securely stored on Amazon S3[15]. Deployment is simplified through the use of AWS Elastic Beanstalk, requiring only the compression of the project (including the model itself) and uploading it to the AWS Elastic Beanstalk application. We have adapted the original code from (Kara et al., 2023) into a class named TEXTCORRECTOR and an API function process_text for performing correction operations with this model. For further details, we encourage consulting the source code of Kara et al. [16] and our implementation [17].

3.4 Spelling Correction

It should be noted that users not only make grammatical mistakes but also commonly commit spelling errors. Since the GEC/D model is not designed for spelling error correction, we employ external tools to extend our system. For mistakes in proper nouns and common typos, we survey external Turkish spelling correction tools. After evaluating different options, we find VNLP (Turker, 2021), StarlangSoftware (Yıldız, 2019), and TurkishNLP (Çetinkaya, 2018) unsatisfactory by means of efficiency and accuracy. As a result, we integrated TrNlp (Bayol, 2018) and ZemberekNLP (Akın and Akın, 2007; Akın, 2017; Uz, 2020) to our system. We apply corrections using TrNlp for proper noun capitalization (e.g., "ankara" → "Ankara")—e.g., any proper noun violating it is capitalized by the tool. Following the proper noun corrections, we leverage ZemberekNLP's TURKISH-SENTENCENORMALIZER for the common typos. Sentences are processed to ensure that the words

are not corrupted (e.g., "yapmk" → "yapmak") and that consistency is maintained across the text. With the combination of TrNlp and ZemberekNLP, our system now not only fixes grammatical errors but also performs spelling correction in Turkish.

4 Evaluation

To evaluate GECTURK WEB, we conduct an in-depth user study. This study aims to assess the usability and effectiveness of the tool in facilitating learning and retention.

The user study is structured into two parts. First, participants are asked to follow a user scenario, where they input 10 short sentences into GECTURK WEB. These sentences are selected to cover all four possible outcomes: True Positives (TP), where the system accurately identifies and corrects an error; True Negatives (TN), where no error exists and the system appropriately refrains from making changes; False Positives (FP), where the system erroneously alters a correct sentence; and False Negatives (FN), where the system overlooks an error. Reflecting on the performance of GECTURK (Kara et al., 2023), which demonstrated a detection precision of 0.89 and a correction F1-score of 0.84, we have designed a representative sample to mirror these results. Therefore, the set of 10 sentences includes 7 True Positives (TPs) and 1 of each other outcome types. It is important to note that the participants are unaware of this distribution. To guide the participants on each potential outcome, we create four videos and present them to participants before they begin experimenting with GECTURK WEB, which is described in detail in §A.1. After viewing these videos, participants are instructed to input each sentence and classify it according to one of the possible outcomes. The complete list of 10 sentences can be found in §A.1. We restrict the average duration of this part to be 45 minutes to align with findings from studies (Lavrakas, 2008; Kost and da Rosa, 2018; Sharma, 2022) on the optimal length for questionnaires.After completing this part, participants are asked several questions to assess the system based on the evaluation metrics. We employ two established metrics to test usability and user satisfaction: the System Usability Scale (SUS) (Brooke, 1995) and the Standardized User Experience Percentile Rank Questionnaire (SUPR-Q) (Sauro, 2015). These metrics are widely recognized for their reliability in assessing user sat-

---

[14] https://docs.aws.amazon.com/elasticbeanstalk/latest/dg
[15] https://aws.amazon.com/s3
[16] https://github.com/GGLAB-KU/gecturk
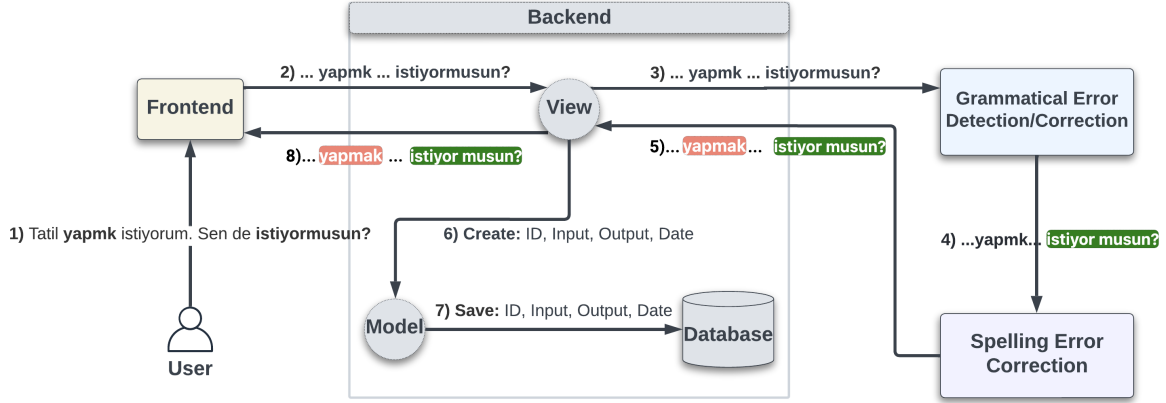[17] https://github.com/GGLAB-KU/gecturkweb

Figure 2: The GECTURK WEB Architecture. **1)** User inputs text containing two errors: a spelling error, "**yapmk**" (shown in red) and a grammatical error, "**istiyormusun**" (shown in green). **2-3)** The view receives the input from the frontend and forwards it to the GEC/D model. **4)** The GEC/D model corrects the grammatical error and adds tags for the frontend to display, as shown in 1. **5)** The SEC module corrects the spelling error, tags it, and sends it back to the View. **6-7)** The model compiles relevant information such as ID, Input, Output, and Date, and records these in the database. **8)** The View sends the prepared output back to the frontend for display.

isfaction and system usability. To understand the effectiveness of GECTURK WEB, we also ask a yes/no question about whether participants learned or remembered a grammatical rule. The SUS questionnaire contains ten five-level Likert scale questions. The SUPR-Q includes seven five-level and one ten-level Likert scale questions. Including our yes/no question, we ask a total of 19 questions. All of these questions are in §A.2.

The evaluation results from 10 users are noteworthy, particularly in terms of usability and user satisfaction. The average SUS score is 88.3 (out of a possible 100; the average benchmark is 69 (Bangor et al., 2009)), indicating an excellent level of usability. Similarly, the average score for the SUPR-Q was 4.34 (out of a possible 5; the average benchmark is 3.93 (Sauro, 2015)), suggesting high user satisfaction with the web interface and functionality. These scores are significantly above the average benchmarks, highlighting the effectiveness of GECTURK WEB in providing a user-friendly and satisfying experience. Notably, 80% of participants report that they learned or remembered a grammatical rule, underscoring the tool's impact on learning and retention. Additionally, we measure the time-efficiency of the system and provide the results in Appendix §B.

5 Extension to Other Languages

As depicted in Figure 2, our system exhibits flexibility and seamless adaptability for multilingual

support. Expanding our system to support other languages merely requires the replacement of the GED/C model and the spelling error correction module. Specifically, the sequence tagger model must be trained to identify the distinct grammatical error patterns of the target language. Similarly, the spelling error correction module can be replaced with an existing spelling corrector for the target language. Both modules can be adjusted by modifying the "text_corrector.py" script and the associated model weights files, facilitating straightforward integration.

6 Conclusion

In this work, we present GECTURK WEB, a practical online platform for Turkish grammatical error detection and correction (GED/C) along with spelling error correction (SEC). Our system aims to not only correct mistakes but also to facilitate learning of complex writing rules via user-friendly rule explanations. Furthermore, the user feedback mechanism allows for continual support and training of the tool. The high SUS and SUPR-Q scores, significantly above average benchmarks, alongside the positive feedback on learning outcomes, validate the platform's design philosophy and its focus on user-centric development. Furthermore, GECTURK WEB is built with a flexible architecture, suggesting that adaptation to additional languages is within reach. Source code and the web-based tool is publicly and freely available.

Limitations

Major limitation of our system is the number of concurrent user interactions it can process. Currently, the system operates on a single AWS i4i.large instance, which can efficiently manage up to ten simultaneous users. Beyond this threshold, performance begins to degrade, necessitating additional instances to preserve service quality. However, it's essential to highlight that this limitation can easily be overcome by enhancing our infrastructure given the budget. Should the GECTURK WEB platform experience a surge in popularity, we are prepared to scale our resources horizontally by incorporating more instances.

Ethics Statement

The development and deployment of GECTURK WEB adhere to ethical considerations crucial for language processing tools. We ensure that user data is handled with the utmost confidentiality and integrity, in accordance with data protection regulations. The feedback system is designed to be non-intrusive and respectful of user privacy.

Acknowledgements

References

Ahmet Afsin Akın and Mehmet Dündar Akın. 2007. Zemberek, an open source nlp framework for turkic languages. *Structure*, 10(2007):1–5.

Ahmet Akın. 2017. Zembereknlp - natural language processing tools for turkish.

Manar Alkhatib, Azza Abdel Monem, and Khaled Shaalan. 2020. Deep learning for arabic error detection and correction. *ACM Trans. Asian Low Resour. Lang. Inf. Process.*, 19(5):71:1–71:13.

Aaron Bangor, Philip Kortum, and James Miller. 2009. Determining what individual sus scores mean: adding an adjective rating scale. *J. Usability Studies*, 4(3):114–123.

Esat Mahmut Bayol. 2018. Trnlp - tr natural language processing.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.".

John Brooke. 1995. Sus: A quick and dirty usability scale. *Usability Eval. Ind.*, 189.

Christopher Bryant, Mariano Felice, and Ted Briscoe. 2017. 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)*, pages 793–805, Vancouver, Canada. Association for Computational Linguistics.

Christopher Bryant, Zheng Yuan, Muhammad Reza Qorib, Hannan Cao, Hwee Tou Ng, and Ted Briscoe. 2023. Grammatical error correction: A survey of the state of the art. *Comput. Linguistics*, 49(3):643–701.

Eduardo Calò, Léo Jacqmin, Thibo Rosemplatt, Maxime Amblard, Miguel Couceiro, and Ajinkya Kulkarni. 2021. GECko+: a grammatical and discourse error correction tool. In *Actes de la 28e Conférence sur le Traitement Automatique des Langues Naturelles. Volume 3 : Démonstrations*, pages 8–11, Lille, France. ATALA.

Kenneth Heafield and Alon Lavie. 2010. CMU multiengine machine translation for WMT 2010. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR, WMT@ACL 2010, Uppsala, Sweden, July 15-16, 2010*, pages 301–306. Association for Computational Linguistics.

Nahid Hossain, Mehedi Hasan Bijoy, Salekul Islam, and Swakkhar Shatabda. 2024. Panini: a transformer-based grammatical error correction method for bangla. *Neural Comput. Appl.*, 36(7):3463–3477.

Atakan Kara, Farrin Marouf Sofian, Andrew Bond, and Gözde Şahin. 2023. GECTurk: Grammatical error correction and detection dataset for Turkish. In *Findings of the Association for Computational Linguistics: IJCNLP-AACL 2023 (Findings)*, pages 278–290, Nusa Dua, Bali. Association for Computational Linguistics.

Rhonda G. Kost and Joel Correa da Rosa. 2018. Impact of survey length and compensation on validity, reliability, and sample characteristics for ultrashort-, short-, and long-research participant perception surveys. *Journal of Clinical and Translational Science*, 2:31 – 37.

Shaopeng Lai, Qingyu Zhou, Jiali Zeng, Zhongli Li, Chao Li, Yunbo Cao, and Jinsong Su. 2022. Type-driven multi-turn corrections for grammatical error correction. In *Findings of the Association for Computational Linguistics: ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 3225–3236. Association for Computational Linguistics.

Paul J Lavrakas. 2008. *Encyclopedia of survey research methods*. Sage publications.

Zuchao Li, Kevin Parnow, Masao Utiyama, Eiichiro Sumita, and Hai Zhao. 2021. Miss: An assistant for multi-style simultaneous translation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, EMNLP 2021, Online and Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 1–10. Association for Computational Linguistics.

Jakub Náplava and Milan Straka. 2019. Grammatical error correction in low-resource scenarios. In *Proceedings of the 5th Workshop on Noisy User-generated Text, W-NUT@EMNLP 2019, Hong Kong, China, November 4, 2019*, pages 346–356. Association for Computational Linguistics.

Jakub Náplava, Milan Straka, Jana Straková, and Alexandr Rosen. 2022. Czech grammar error correction with a large and diverse corpus. *Trans. Assoc. Comput. Linguistics*, 10:452–467.

Shrimai Prabhumoye, Ruslan Salakhutdinov, and Alan W. Black. 2020. Topological sort for sentence ordering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 2783–2792. Association for Computational Linguistics.

Zhaoquan Qiu and Youli Qu. 2019. A two-stage model for chinese grammatical error correction. *IEEE Access*, 7:146772–146777.

Muhammad Reza Qorib, Geonsik Moon, and Hwee Tou Ng. 2023. ALLECS: A lightweight language error correction system. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics. EACL 2023 - System Demonstrations, Dubrovnik, Croatia, May 2-4, 2023*, pages 298–306. Association for Computational Linguistics.

Muhammad Reza Qorib, Seung-Hoon Na, and Hwee Tou Ng. 2022a. Frustratingly easy system combination for grammatical error correction. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2022, Seattle, WA, United States, July 10-15, 2022*, pages 1964–1974. Association for Computational Linguistics.

Muhammad Reza Qorib, Seung-Hoon Na, and Hwee Tou Ng. 2022b. Frustratingly easy system combination for grammatical error correction. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2022, Seattle, WA, United States, July 10-15, 2022*, pages 1964–1974. Association for Computational Linguistics.

Hongkai Ren, Liner Yang, and Endong Xun. 2018. A sequence to sequence learning for chinese grammatical error correction. In *Natural Language Processing and Chinese Computing - 7th CCF International Conference, NLPCC 2018, Hohhot, China, August 26-30, 2018, Proceedings, Part II*, volume 11109 of *Lecture Notes in Computer Science*, pages 401–410. Springer.

Alla Rozovskaya and Dan Roth. 2019. Grammar error correction in morphologically-rich languages: The case of russian. *Trans. Assoc. Comput. Linguistics*, 7:1–17.

Ali Safaya, Emirhan Kurtulus, Arda Göktogan, and Deniz Yüret. 2022. Mukayese: Turkish NLP strikes back. In *Findings of the Association for Computational Linguistics: ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 846–863. Association for Computational Linguistics.

Jeff Sauro. 2015. Supr-q: A comprehensive measure of the quality of the website user experience. *J. Usability Studies*, 10(2):68–86.

Stefan Schweter. 2020. Berturk - bert models for turkish.

Hunny Sharma. 2022. How short or long should be a questionnaire for any research? researchers dilemma in deciding the appropriate questionnaire length. *Saudi journal of anaesthesia*, 16(1):65–68.

Aiman Solyman, Zhenyu Wang, Qian Tao, Arafat Abdulgader Mohammed Elhag, Rui Zhang, and Zeinab Mahmoud. 2022. Automatic arabic grammatical error correction based on expectation-maximization routing and target-bidirectional agreement. *Knowl. Based Syst.*, 241:108180.

Alexey Sorokin. 2022. Improved grammatical error correction by ranking elementary edits. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 11416–11429. Association for Computational Linguistics.

Maksym Tarnavskyi, Artem N. Chernodub, and Kostiantyn Omelianchuk. 2022. Ensembling and knowledge distilling of large sequence taggers for grammatical error correction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 3842–3852. Association for Computational Linguistics.

Meliksah Turker. 2021. Vnlp - nlp library for turkish language.

Harun Uz. 2020. Zemberek-python - python implementation of zembereknlp.

Harun Uz and Gülşen Eryiğit. 2023. 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, pages 134–142, Dubrovnik, Croatia. Association for Computational Linguistics.

Xiuyu Wu and Yunfang Wu. 2022. From spelling to grammar: A new framework for chinese grammatical error correction. In *Findings of the Association for Computational Linguistics: EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 889–902. Association for Computational Linguistics.

Lvxiaowei Xu, Jianwang Wu, Jiawei Peng, Jiayu Fu, and Ming Cai. 2022. FCGEC: fine-grained corpus for chinese grammatical error correction. In *Findings of the Association for Computational Linguistics: EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 1900–1918. Association for Computational Linguistics.

Olcay Taner Yıldız. 2019. Turkish spell checker - starlangsoftware.

Harun Reşit Zafer. 2017. hunspell-tr.

Metehan Çetinkaya. 2018. Turkishnlp - turkish nlp with python.

Appendix

A User Study

A.1 The user scenario

Participants are given 10 short sentences and are requested to input them into GECTurk WEB. To help participants understand the potential outcomes, we produced four instructional videos and showed them to the participants before they started using GECTurk WEB. Figures 3 through 6 display screenshots of each scenario along with its English transcription. Following the video demonstration, participants are directed to input each sentence and categorize it based on the possible outcomes. The full list of the 10 sentences is provided in Table 3.

DURUM - 1: Cümlede hata var ve GECTurk hatayı başarıyla tespit ediyor.

Örnek: "Sonuçları herkes gibi bende merakla bekliyorum."

Bu cümlede GECTurk doğru bir şekilde "bende" kelimesini "ben de" olarak değiştirmektedir. Bu nedenle, aşağıdaki videoda da gösterildiği gibi "Bu Hala Hatalı!" butonuna tıklamanıza gerek bulunmamaktadır.
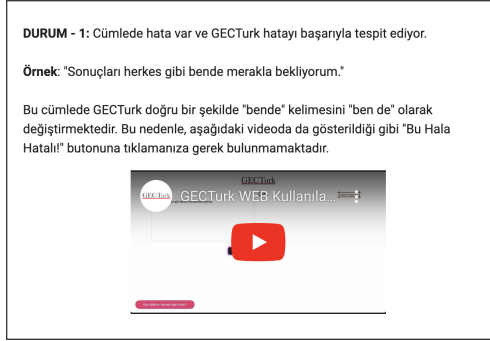
Figure 3: **CASE - 1:** The sentence contains an error and GECTurk successfully detects the error. **Example:** "Sonuçları herkes gibi bende merakla bekliyorum." In this sentence, GECTurk correctly changes "bende" to "ben de". Therefore, there is no need to click on the "This is still incorrect!" button, as shown in the video below.

DURUM - 2: Cümlede hata yok ve GECTurk cümleyi değiştirmiyor.

Örnek: "Lyon, bir milyonu aşan nüfusuyla Fransa'nın üçüncü büyük kenti."

Bu cümlede herhangi bir hata yoktur ve GECTurk cümleyi değiştirmemektedir. Bu nedenle, aşağıdaki videoda da gösterildiği gibi "Bu Hala Hatalı!" butonuna tıklamanıza gerek bulunmamaktadır.
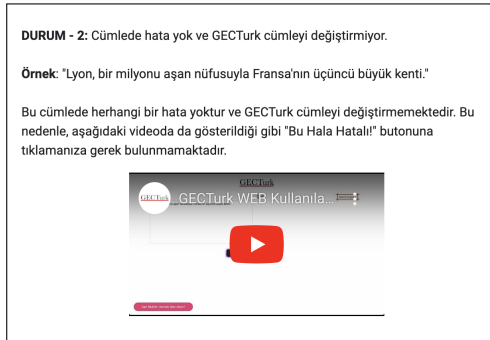
Figure 4: **CASE - 2:** There is no error in the sentence and GECTurk does not change the sentence. **Example:** "Lyon, bir milyonu aşan nüfusuyla Fransa'nın üçüncü büyük kenti." There are no errors in this sentence and GECTurk does not change the sentence. Therefore, there is no need to click on the "This is still incorrect!" button, as shown in the video below.

DURUM - 3: Cümlede hata yok ama GECTurk cümleyi değiştiriyor.

Örnek: "O kadar merhametlidir ki yakın arkadaşları arasında karıncaincitmez olarak anılır."

Bu cümlede bir hata yok ama GECTurk "karıncaincitmez" kelimesini "karınca incitmez" olarak değiştirmektedir. Bu nedenle, aşağıdaki videoda da gösterildiği gibi "Bu Hala Hatalı!" butonuna tıklamalısınız ve cümlenin doğru versiyonunu yazmalısınız.
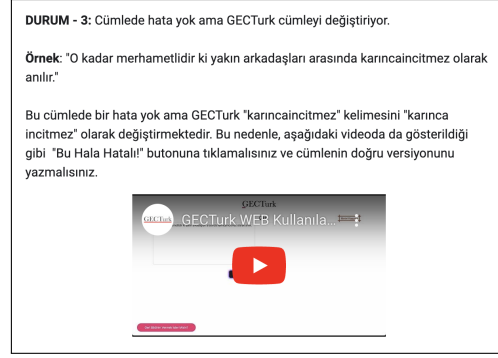
Figure 5: **CASE - 3:** There is no error in the sentence but GECTurk changes the sentence. **Example:** "O kadar merhametlidir ki yakın arkadaşları arasında karıncaincitmez olarak anılır." There is no mistake in this sentence, but GECTurk changes the word "karıncaincitmez" to "karınca incitmez". Therefore, you should click on the "This is still incorrect!" button and type the correct version of the sentence, as shown in the video below.

DURUM - 4: Cümlede hata var ancak GECTurk cümledeki hatayı tespit edemiyor.

Örnek: "Oldum olası kendime çeki düzen vermeyi hiç bilmem."

Bu cümle "çeki düzen" kelimesi "çekidüzen" şeklinde yazılmadığı için hatalıdır ancak GECTurk bu hatayı tespit edememektedir. Bu nedenle, aşağıdaki videoda da gösterildiği gibi "Bu Hala Hatalı!" butonuna tıklamalısınız ve cümlenin doğru versiyonunu yazmalısınız.
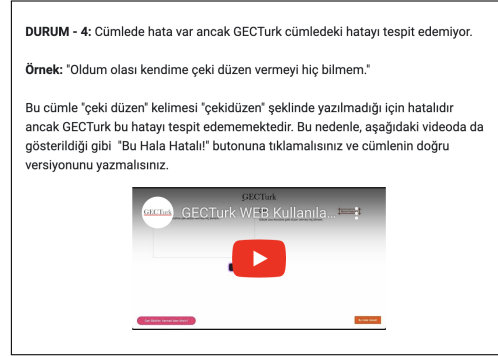
Figure 6: **CASE - 4:** There is an error in the sentence but GECTurk cannot detect it. **Example:** "Oldum olası kendime çeki düzen vermeyi hiç bilmem." This sentence is incorrect because the word "çekidüzen" is incorrectly spelled as "çeki düzen", but GECTurk is unable to detect this error. Therefore, you should click on the "This is still incorrect!" button and type the correct version of the sentence, as shown in the video below.

Input No	Input	GECTurk WEB Output	Ground Truth	Case No
1	Dilin birey ve toplum hayatında taşıdığı önem, **anadili** öğretimini de önemli kılmaktadır.	UNCHANGED	... [anadili → ana dili] ...	4
2	Onu baban görmeden hemen ortadan **kayıp et**.	... [kayıp et → kaybet] ...	... [kayıp et → kaybet] ...	1
3	Tatil yapmak **istiyrum** fakat çalışmaya devam etmem şart.	... [istiyrum → istiyorum] ...	... [istiyrum → istiyorum] ...	1
4	Bugün hep beraber gittiğimiz geziye **Ayşe'de** geldi.	... [Ayşe'de → Ayşe de] ...	... [Ayşe'de → Ayşe de] ...	1
5	**Bir takım** ansiklopediye dünyanın parasını ödedim.	[Bir takım → Birtakım] ...	UNCHANGED	3
6	Düştüğü bu durumdan kurtulmak için **karakara** düşünüyordu.	... [karakara → kara kara] ...	... [karakara → kara kara] ...	1
7	Bugün öyle çok **yorulmuşki** hemen yattı.	... [yorulmuşki → yorulmuş ki] ...	... [yorulmuşki → yorulmuş ki] ...	1
8	Bu yaptığının elle tutulur **sebepi** yok.	... [sebepi → sebebi] ...	... [sebepi → sebebi] ...	1
9	Sanki uyurgezer biri gibi çarşıyı baştan başa adımladı.	UNCHANGED	UNCHANGED	2
10	**İçerde** kimsenin olmadığını gördü ve bağırmaya başladı.	[İçerde → İçeride] ...	[İçerde → İçeride] ...	1

Table 3: The complete list of 10 sentences is given to the participants. For each sentence, participants are required to enter the **Input** and observe the **GECTurk WEB Output**. Based on this output, they decide the **Case No**. Note that participants have no access to the **Ground Truth**.

A.2 User Evaluation

In the second part of the user study, participants are asked to complete the SUS and SUPR-Q questionnaires based on their experience in the first half of the study. Additionally, participants are asked a yes/no question regarding whether they learned or remembered a grammatical rule. The SUS questionnaire comprises ten five-level Likert scale questions, while the SUPR-Q consists of seven five-level Likert scale questions and one ten-level Likert scale question, making a total of 19 questions including the yes/no question.

B Time Efficiency

This section highlights the model's performance in terms of time efficiency, demonstrating a linear relationship between the volume of words processed and the response time. The data suggests that the system can process up to 14,000 words in under 90 seconds, affirming its ability to scale effectively while retaining user engagement. This performance is supported by robust hardware specifications of an AWS i4i.large instance, including 2 vCPUs, 16.0 GiB of memory, and a 3.5 GHz Intel Xeon 8375C
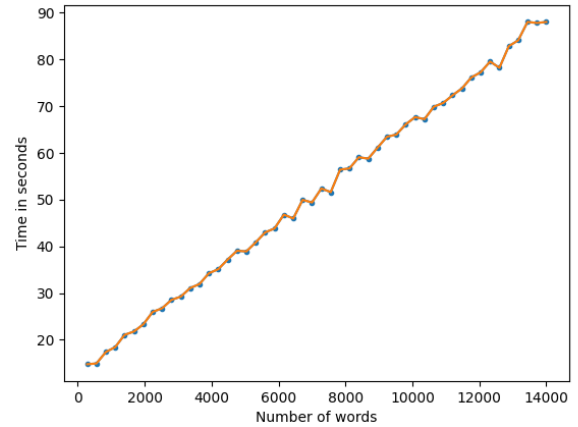

Figure 7: The relationship between the number of words processed by the GECTURK model and the response time, demonstrating the model's time efficiency.

processor, which collectively ensure minimal latency even under significant text processing loads. For visual representation, see Figure 7.