Turkish Typo Correction for E-Commerce Search Engines

K.Elif Oral, Koray Mancuhan, Hüseyin Varol Erdem, Ece Hatipoğlu
Hepsiburada, Hepsiburada, Hepsiburada, Hepsiburada
{kadriye.oral, koray.mancuhan, varol.erdem, ece.aktan}@hepsiburada.com

Abstract

Typo correction is a challenging problem when it is developed for morphologically rich languages. The existing approaches in the literature are successful mainly for English, leaving the problem open for such languages. This creates an issue, because the typo correction is a critical component in practice for many systems such as search engines. Especially, the search engines of e-commerce platforms rely heavily on typo correction for product relevancy. A bad performing typo corrector could result in very few number of relevant products when a user is looking for a product on an e-commerce platform, resulting in significant revenue decrease. For the first time in the literature, this paper proposes a modern typo corrector for a morphologically rich language, Turkish; which is integrated to the search engine of one of the leading e-commerce platforms in Turkey, Hepsiburada. Our thorough experiments show that this new typo corrector performs very successful in practice, outperforming the existing Turkish specific propositions in the literature; even if it is applied out of the context of the search engines.

Keywords: typo correction, search engines, e-commerce, deep learning

1. Introduction

The search engines play an important role for e-commerce, providing customers an effective tool for purchasing their desired products. As a first step of the search experience, the typo correction is vital in e-commerce; since it directly affects the precision of the search outcomes in terms of the product relevancy. Users often enter the search queries hastily; this leads to unintentional misspellings. Additionally, they may use different input sources (e.g., keyword customization for different languages), which result in completely incorrect search terms. A typo corrector detects and corrects these errors, ensuring that the search results contain the products that the users are intended to find. Thus, it provides the users a much better online shopping experience.

The traditional spelling corrector systems use statistical techniques (Brill and Moore, 2000; Hasan et al., 2015; Li et al., 2012; Gupta et al., 2019) and edit distances (Damerau, 1964; Whitelaw et al., 2009). These methodologies have limitations in addressing the learning issues, when they are faced with increased data sparsity. In recent years, the deep neural models (Ye et al., 2023; Kuznetsov and Urdiales, 2021; Jayanthi et al., 2020; Etoori et al., 2018a) have gained considerable popularity in this field by improving the accuracy despite an important drawback: the increased inference latency due to the model complexity. Although spelling correction is a well-studied field encompassing languages with different characteristics (Liu et al., 2021; Azmi et al., 2019; Duong et al., 2020; Eryiğit and Torunoğlu-Selamet, 2017; Park et al., 2021); the studies, which focus specifically on the morphologically rich languages (MRLs), are still in an immature state. Especially, their applications in e-commerce are still in early stages.

Turkish, a morphologically rich and agglutinative language with the presence of diacritized letters (e.i, çığüö), is prone to spelling errors. This becomes particularly important in the context of e-commerce search. For example, the users face difficulties in spelling the non-Turkish brands (e.g., "Lenova" instead of "Lenovo"); or, use "ciguo" instead of the Turkish letters and vice versa (e.g., İphone instead of Iphone). Furthermore, the agglutinate nature of the Turkish brings an additional challenge. Correcting considering the inflections is crucial, as the absence of suffixes in the search queries impacts products matched at the retrieval level. The overall search relevancy is also affected significantly. For example, we cannot correct the "banyo muslul" as "banyo musluk" instead of the "banyo musluğu" (*bathroom faucet*)[1]); because, (1) there is a degradation in the meaning due to the absence of the genitive marker **I**; and, (2) the search system could bring different products since there may be exact match between the word "musluk" and other product names, under partially relevant categories such as "banyo musluk aksesuarları" (*bathroom faucet accessories*).

We group the Turkish spelling errors, encountered in e-commerce, under two subcategories: (i) the general spelling errors (e.g., fat finger, insertion/deletion.) which can be dealt with applying general solutions proposed for English, (ii) the language specific spelling errors (e.g., missing diacritics, phonetics) which necessitate the specialized treatments. Figure 1 shows the distribution of

[1]The last consonant of the word "musluk" (*faucet*) becomes ğ when a genitive marker **-I** is attached to it, a phenomenon called the consonant lenition.

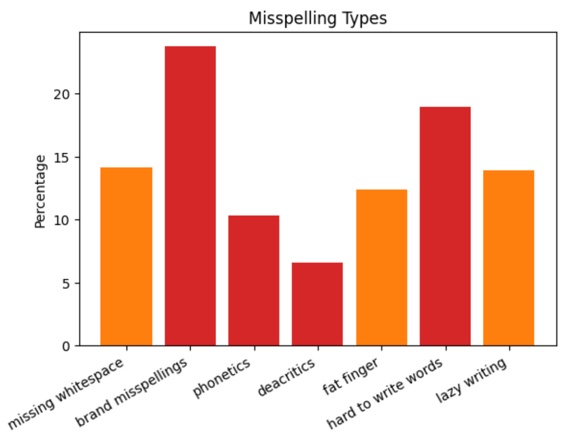

Figure 1: Spelling Error Distribution: The red bars indicate the misspelling types of the language specific spelling errors, the orange ones show the second type of the language specific spelling errors. The misspelling type descriptions are available in the Appendix A.1

.

searches which had bad relevancy due to the misspellings errors [2]. As shown in the figure, 59.58% of total misspellings belong to the latter category.

In the latter subcategory, the proposed approaches, tailored for the languages with limited linguistic characterization, do not fit due to the following factors: (1) high data sparsity due to the rich morphology (a word may have hundreds of different surface forms), (2) substantial edit distances due to the absence of diacritics and phonetic spelling (e.g., the edit distance between the words "başlığı" (*head*) and "basligi" in Turkish is 4 [3]). The deep models are relatively more effective addressing these issues; but, they are not suitable for the search engines due to the high latency costs.

In this paper, we introduce a generalized spelling correction method, which handles the language specific spelling errors along with the general ones, without additional latency. Our simple yet effective method enhances the candidate generation through a morphology-centered approach. The main contributions of our work are:

- A candidate generation method, which relies on using a character level transformer model with low latency; capturing the morphology based relationships between the syntactic word vectors.
- A candidate scoring function that is customized to our morphologically rich language, Turkish.
- A blueprint methodology of forming a training set specific to our morphologically rich language, Turkish.

2. Related Work

The field of spelling correction has evolved from the early work, which was based on the (Wagner and Fischer, 1974) editing distance and the noisy channel model, (Kemighan et al., 2003; Brill and Moore, 2000) to the neural methods based approaches. Early investigations leverage the edit distance and its variations (Wagner and Fischer, 1974); and, the noisy channel model (Brill and Moore, 2000; Kemighan et al., 2003) to rectify the spelling errors. For spelling correction, Sun et al. (2015) propose the convolutional neural networks (CNNs) while Jayanthi et al. (2020) introduce NeuSpell. Ghosh and Kristensson (2017) and Etoori et al. (2018b) extended the neural approach to address the nuanced challenges in the spelling correction tasks. Along with neural models, the spelling correction has been reformulated as a generational task (Zhou et al., 2017; Kuznetsov and Urdiales; Zhang et al., 2019a; Grundkiewicz et al., 2019; Sharma et al., 2023; Zhang et al., 2023).

Spelling correction is a critical component for e-commerce search engines, particularly in the context of e-commerce. Several recent studies have addressed the unique challenges posed by the e-commerce search engines. Yang et al. (2022) propose a generalized spelling correction to address the phonetic errors. Kakkar et al. (2023) and Pande et al. (2022) improve the correction rate on tough spelling mistakes by weakly supervised data. Ye et al. (2023) tailor the pretrained language models for e-commerce search queries.

Non-English languages may need additional treatments for spelling correction due to their specific properties (Zitouni and Sarikaya, 2009; AZMI and ALMAJED, 2015; Liu et al., 2022; Zhang et al., 2019b; Liu et al., 2021). Turkish spelling correction Eryiğit and Torunoğlu-Selamet (2017); Demir and Topcu (2022); Torunoglu-Selamet et al. (2016); Akın and Dündar (2007) has contributed significantly to the field, providing valuable insights and methodologies. Safaya et al. (2022); Koksal et al. (2020) introduced a spelling correction corpus.

3. Spelling Corrector

Our spelling corrector includes the candidate generation and the ranking steps according to the general approach in the literature. For a given input query

[2]We collected 20k queries on our platform and analyzed them based on the search relevancy. We found that 12% of our search results had low relevancy, and 11% of this was due to the misspelling errors.

[3]The last consonant of the word "başlık" (*head*) becomes ğ when a genitive marker **-I** is attached to it due to lenition.

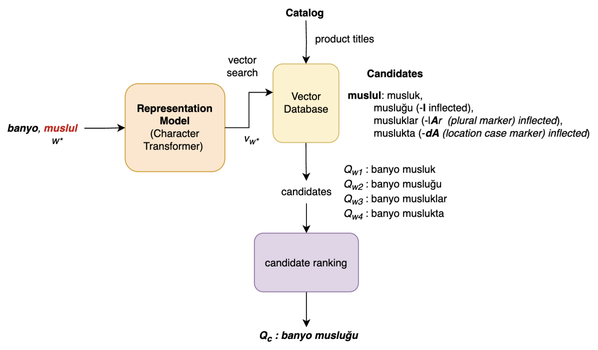

Figure 2: The Correction pipeline

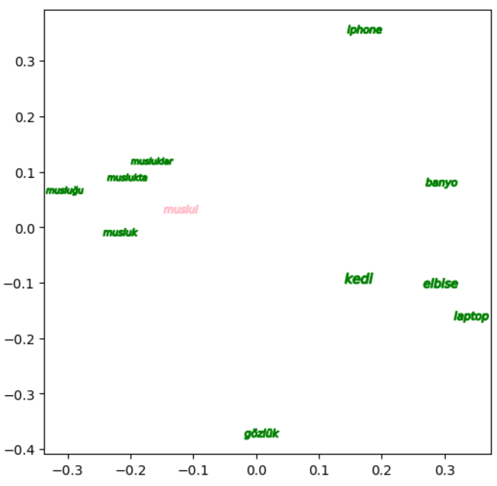

Figure 3: An illustration of our vector space, the similarly spelled words (i.e., word inflections) are represented by the neighbouring vectors.

Q_{w^*} with a misspelled word w^*, the spelling corrector generates a set of words $W = \{w_1, w_2, ..., w_n\}$; where w_i is a candidate word that could be the correction of w^*. Then, a query candidate list is formed by replacing w^* with w_i. The corrected Q_c is obtained by the selection of a candidate with the highest score. The score calculation uses the following formula

$$Q_c = w_i argmax(f(Q_{w_i}|Q_{w^*})) \quad (1)$$

where *f* is the scoring function that is applied on each query candidate (Q_{w_i}) at the ranking stage. Our correction pipeline is depicted in Figure 2

3.1. Candidate Generation

We frame the candidate generation as a vector search where the words are represented by the vectors based on their syntax. The basic idea of this is that the incorrectly spelled words should be substantially close to their correctly spelled counterparts in terms of syntax. Consequently, obtaining the closest n words, which are based on the syntax-aware vector similarity for a misspelled word, automatically forms a candidate list; including the potential corrections.

Syntax-aware representations have the ability to capture the relationship between the root words and their surface forms. Such an ability enables to locate the inflected words close to their nominative forms. This capability provides an opportunity to correct the words, restoring the missing word inflections. It brings the awareness of the morphology; and, allows more accurate corrections since the inflected forms of the words can be included among the candidates. We train a character level transformer (Vaswani et al., 2017) model that learns the syntax-aware representations with a contrastive learning objective. The model aims to create a vector space for the words by capturing the intrinsic character patterns, which enables the model to discover the language specific rules (e.g., vowel harmony, lenition, phone mapping of character sequence such as ’sq‘ to ’su‘, sh to ş, etc.). We avoided complex models due to their latency costs; opted to design the model solely as an encoder layer, incorporating a feed-forward layer with a mean pooler on top of it.

The candidate generation (illustrated in Figure 2) starts with the extraction of v_{w^*}, which is the vector of w^*. Then, the corresponding n vectors similar to v_{w^*} (e.g. corresponding to n similar words) are retrieved from the database; which holds the vectors of the words extracted from the product titles. Figure 3 illustrates the vector space with a few samples[4]. For example, the query "banyo muslul" has a misspelled word "muslul"; and, its correction should be in the inflected form ("musluğu") rather than the nominative form ("musluk" (*faucet*)). Since the vectors of both ($v_{w_{musluk}}$ and $v_{w_{muslugu}}$) are located close to each other, and to v_{w^*}; we can generate the candidate queries "banyo *musluk*" and "banyo *musluğu*".

3.2. Candidate Ranking

The candidates are ranked using a score, which is calculated by using features obtained from three different sources. Each source represents a different aspect of the candidates:

- The first source (s_1) is the *edit distance* between Q_{w^*} and Q_{w_i}. This prevents Q_{w^*} and Q_c from being completely different queries for the cases the input query has more than one misspelled words.
- The second source (s_2) is the vector similarity.

[4]One should note that this figure is only for illustration purposes, and shows approximate distances.

This regularizes the edit distance by considering the similar syntax aware representations. In particular, it boosts the candidates which are dissimilar in terms of the edit distance. The features from the vector similarity play an important role in the correction of the phonetic and diacritic based misspellings.

- The third source (s_3) is the language score, namely perplexity. In this case, we train two statistical language models: one using the search logs, and the other using the product titles. The features derived from the perplexities validate to some extent the candidates, exhibiting a bias towards the selection of the correct inflections.

Based on these various sources, the final form of the scoring function f is

$$f(\mathrm{Q}_{w_i}|\mathrm{Q}_{w^*}) = \sum_{j=0}^{3} \mathbf{w}_{s_j} * \mathbf{x}_{Q_{w_i}|Q_{w^*} s_j} \quad (2)$$

where s_j is a feature source, $\mathbf{x}_{Q_{w_i}|Q_{w^*} s_j} \in R^n$, $n \in [0, N]$ is a feature vector derived form s_i. $\mathbf{x}_{Q_{w_i}|Q_{w^*}}$ are extracted from the generated candidate Q_{w_i} and the user query Q_{w^*}. $\mathbf{w}_{s_j} \in R^m$, $m \in N$ is a weight vector that is learnt by a regression model.

4. Data Generation

We create the training data from the search logs by leveraging the user interactions (e.g., clicks). All the training data is automatically generated, eliminating the need for human annotations. The training data contains samples $(< w_k, w_j >, l_{kj})$, where $< w_k, w_j >$ is a word pair and l_{kj} is the corresponding label of the pair.

We have two sources of obtaining the positive pairs: the users' feedback and the synthetic data. When l_{kj} is equal to 1, namely positive samples, w_k and w_j are expected to possess similar vectors since w_k represent the misspelled version of w_j. To enhance the syntax awareness, we introduce additional morphology-based $< w, w_{inf} >$ pairs; where w_{inf} denotes the inflection of w. This augmentation aims to ensure that the stems and their inflections have similar vectors by providing the syntactic variations of the words. Consequently, it makes our representations more resilient to the syntactic changes. Negative samples (l_{kj} is equal to 0), on the other hand, are expected to have dissimilar vectors for the words w_k and w_j. They are generated by the outer cross join of the positive pairs.

In the following, we explain the used methodologies to form all the pairs.

4.1. User Feedback Chain

We trace the users' specific action sequences and use their interactions as a feedback to generate the positive samples. Q_{w_k} denotes 'the query with the misspelled word w_k', Q_{w_j} denotes 'the query with the corrected word w_j'; and, P_{w_j} denotes 'the product title containing the word w_j'. We extract the correctly spelled-misspelled word pairs from the Q_{w_k} and P_{w_j} (Q_{w_j}), where we add several check conditions to exclude the drastically different word pairs. The considered action sequences are explained below.

4.1.1. Query Refinement Chain

Users may correct their misspelled queries in the consecutive searches. For example, a user could search Q_{w_k}, then correct it himself/herself and make a subsequent search with Q_{w_j}. We leverage such behaviour sequences and match w_k with w_j to form a pair if and only if the elapsed time between two consecutive searches, Q_{w_k} and Q_{w_j}, is less than 10 minutes; and, the edit distance between them is less than 5.

4.1.2. Fuzzy Match and Click

The retrieval systems may bring relevant products with their fuzzy match abilities, even if the queries have misspelled words. In such cases, the users may still click on the relevant products which are not affected by the incorrect search terms. Assuming that a user searched Q_{w_k}, and subsequently clicked on P_{w_j}; we can extract $< w_k, w_j >$ if the edit distance between w_k and w_j is less than 4.

4.1.3. Click on Suggestion or Rejection

Users' clicks can be considered a reliable source for forming pairs, because they serve as a natural labeling mechanism for our system. Specifically, their clicks on 'did you mean' (suggestion of the corrector) or 'return to the initial query' (rejection) automatically labels the outputs of the spelling corrector. A click on the suggestion indicates a successful acceptance of the correction, meaning that the corrector successfully updates the given query. Conversely, the rejection clicks imply that the corrector is unable to achieve the correction. We collect the users' feedback clicks and use the click rates to filter out the unintentional clicks. One should note that this feedback mechanism, complemented with the other twos, plays a crucial role in gradually improving the spelling corrector's performance; as, the training data is continuously updated with the insights from the unsuccessful corrections. However, this is not initially available.

4.2. Synthetic Data Generation

We generate the synthetic data for two purposes: (1) increase the representation of the most common user mistakes in the training data, (2) form effortlessly the morphology based pairs. For the first purpose, we create the artificial misspellings by

- deacritization: removing deacritics of the Turkish characters, e.g., "yılbaşı çam ağacı" (*Christmas pine tree*) to "yilbasi cam agaci"
- insertion: adding adjacent characters on the keyboard, e.g., "yılbaşı çam ağacı" to "yıolbaşı çam ağascı"
- deletion: removing the random vowels[5], e.g., "yılbaşı çam ağacı" to "ylbaşı çam ağacı"
- replacement: replacing the characters with the ones neighbouring on the keyboard, e.g., "yılbaşı çam ağacı" to "yıkbaşı çam ağacu"
- swapping: swapping the adjacent characters, e.g., "yılbaşı çam ağacı" to "yılbaşı çma ağacı"

The meaning of the words might deteriorate if the excessive distortion is applied. Therefore, we allow two artificial distortions on a given word.

4.3. Morphology-based Pairs Generation

Generating the morphology-based pairs ($< w, w_{inf} >$) with the predefined rules is a challenging task due to the grammar rules of Turkish (e.g., lenition, consonant and vowel harmonies, etc.). We employ the n-gram based word embeddings to generate such kind of pairs, leveraging their ability to approximate the morpheme length. As the n-gram size decreases, these embeddings capture the syntactic variation of the words in the agglutinative languages. This allows us to form the meaningful morphology-based pairs that comply with the rules of the Turkish grammar. We train a Fasttext (Bojanowski et al., 2017) model for these embeddings using our search logs and product titles, because it was shown to capture the morphological variations for Turkish (). Thus, the overall morphology-based pair generation process can be summarized in three steps.

1. We collect a set of correctly spelled high impression queries from the search logs.
2. We build a word vocabulary from the former query set.
3. We generate 20 most similar words for each word (w) within the vocabulary using the Fasttext embeddings. If the edit distance between the generated word and the word w is less than 4, they form the pair $< w, w_{inf} >$.

[5]We observed that the users tend to skip the wovels while writing

5. Experiments

We conducted the evaluations for our corrector in both offline and online experiments. We also did a final error analysis to identify the limitations of our corrector, for determining the future steps.

5.1. Offline Experiments

5.1.1. Accuracy Based Evaluations

The first type of evaluation focused on the correction accuracy of the corrector, which was assessed through the in-domain and out-of-domain settings. The in-domain setting was performed using the data from the e-commerce domain, while the out-of-domain setting was performed using the data prepared for general purposes. This allowed us to compare our corrector with others in the literature and evaluate its performance beyond the specific domain.

We adopt the evaluation metrics precision and recall by defining the following terms:

1. True Positive (TP): The successful misspelling identification and correction cases
2. False Positive (FP): The false misspelling identification and correct word distortion cases
3. False Negative (FN): The misspelling identification and correction failure cases
4. True Negative (TN): The correct identification cases of the correctly spelled words

While the precision represents how many of the corrected queries are successfully corrected, the recall represents how many of the misspelled queries are successfully corrected.

For the in-domain evaluation, we created a gold standard test set; which contains 5.6K queries. Here, the data is collected from both short and long tail queries. Then, the data was manually annotated in two iterations by three native-speaking annotators; who have the domain expertise and the NLP background. In the first iteration, the samples were annotated from scratch separately by two annotators. In the second iteration, the third annotator checked the data quality by validating the labeled samples' accuracy. Our corrector achieves an F1-score of 86% on this data, with a precision of 89% and a recall of 83%.

For the out-of-domain evaluation, we evaluated our corrector on trspell-10 (Safaya et al., 2022). To

Corrector	SCA	F1
HUNSPELL-TR (Zafer, 2017)	25.52	86.52
ZEMBEREK (Akın and Dündar, 2007)	62.12	96.56
Safaya et al. (2022)	71.72	**99.62**
ours	**83.3**	98.18

Table 1: Results on trspell10.

be inline with the mentioned study, we reported the spell correction accuracy (SCA) and the macro-averaged F1 score of misspellings detection. Table 1 shows the results. While our corrector performs relatively close to others in detecting the misspellings, it outperforms them with a significant margin in SCA; improving the number of successfully corrected words.

5.1.2. Search Relevancy Based Evaluations

The second type of evaluation focused on the impact of the corrector on the search relevancy. We established an experimental setup, where we compare the change in Normalized Discounted Cumulative Gain (NDCG) for the same dataset after correction intervention. We selected 1000 queries, of which 14% are misspelled to reflect the overall misspelling rate in our search engine. Firstly, top 12 products for each of these queries were retrieved and annotated with Exact/Substitute/Complement/Irrelevant (ESCI) labels. Using these labels, we calculated the NDCG[6] score which is 90.9. Secondly, we fed all the 1000 queries to our corrector model and repeated the same labeling process. After the correction, the NDCG score reached 91.2; where we improved the score of the misspelled samples from 87.8 to 89.7.

5.2. Online Experiments

To be able to correlate our offline evaluation with the main business metric of our search engine (i.e., Conversion Rate (CR)[7]), we launched an A/B test. Our old in-house model was used in the control bucket, while our new in-house model was used in the treatment bucket. 100% of the users were randomly assigned to both control and treatment buckets. The experiment was run for 4 weeks to ensure the statistically significant results. The results showed that the treatment bucket achieved a CR of 4.84%, while the control bucket had a CR of 4.52%; resulting in a 6.99% CR increase between the buckets.

[6]Weights used in our NDCG calculations are 4, 3, 2, 1 for the labels Exact, Substitute, Complement, Irrelevant; respectively

[7]CR is a metric that represents the percentage of users who have completed a desired action, which is the purchase for our domain.

5.3. Error Analysis

We also made an error analysis to identify the weaknesses of our corrector. One limitation is that our corrector operates at the word level, rather than the sequence level. This results in a sub-optimal performance when the context information is crucial for the corrections. Additionally, it has relatively poor performance in the correction of words that are typically Turkish adaptations of the foreign words, deviating from the Turkish harmonies (e.g., palatal harmony, rounding harmony, etc.)

6. Conclusions and Future Work

In this paper, we proposed a typo corrector which was specific to a morphologically rich language, Turkish. We addressed the data related challenges specific to Turkish; in particular, for training and evaluation data creation. We implemented our approach end-to-end, and integrated it to our search engine. The implementation was evaluated using online and offline experiments. In both offline and online experiments, our corrector outperformed the old default corrector that had been deployed. Moreover, our proposed corrector outperformed the existing best Turkish corrector in the literature; when it was used separately from our search engine.

In future, we are planning to modify the candidate generation of our corrector from the word level model to the sequence level model; so, it captures the context in the corrections. We are also planning to adapt our approach to develop the corrector models in Russian and Arabic.

Acknowledgements

The authors would like to offer special thanks to Feride Duman for her valuable feedback, Zeynep Aycan and Şevval Koca for their assistance during the data annotation, and all search-algorithm team for helpful discussions during the development stage.

7. References

Ahmet Afşın Akın and Mehmet Dündar. 2007. Zemberek, an open source nlp framework for turkic languages.

AQIL M. AZMI and REHAM S. ALMAJED. 2015. A survey of automatic Arabic diacritization techniques. *Natural Language Engineering*, 21(3):477–495.

Aqil M Azmi, Manal N Almutery, and Hatim A Aboalsamh. 2019. Real-word errors in Arabic texts:

A better algorithm for detection and correction. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 27(8):1308–1320.

Youssef Bassil. 2012. Parallel spell-checking algorithm based on yahoo! n-grams dataset. *ArXiv*, abs/1204.0184.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the association for computational linguistics*, 5:135–146.

Eric Brill and Robert C Moore. 2000. An improved error model for noisy channel spelling correction. In *Proceedings of the 38th annual meeting of the association for computational linguistics*, pages 286–293.

Qing Chen, Mu Li, and Ming Zhou. 2007. Improving query spelling correction using web search results. In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*, pages 181–189.

Fred J Damerau. 1964. A technique for computer detection and correction of spelling errors. *Communications of the ACM*, 7(3):171–176.

Seniz Demir and Berkay Topcu. 2022. Graph-based turkish text normalization and its impact on noisy text processing. *Engineering Science and Technology, an International Journal*, 35:101192.

Quan Duong, Mika Hämäläinen, and Simon Hengchen. 2020. An unsupervised method for ocr post-correction and spelling normalisation for Finnish. *arXiv preprint arXiv:2011.03502*.

Steffen Eger, Tim vor der Brück, and Alexander Mehler. 2016. A comparison of four character-level string-to-string translation models for (ocr) spelling error correction. *The Prague bulletin of mathematical linguistics*, 105(1):77.

Gülşen Eryiğit and Dilara Torunoğlu-Selamet. 2017. Social media text normalization for turkish. *Natural Language Engineering*, 23(6):835–875.

Pravallika Etoori, Manoj Chinnakotla, and Radhika Mamidi. 2018a. Automatic spelling correction for resource-scarce languages using deep learning. In *Proceedings of ACL 2018, Student Research Workshop*, pages 146–152, Melbourne, Australia. Association for Computational Linguistics.

Pravallika Etoori, Manoj Chinnakotla, and Radhika Mamidi. 2018b. Automatic spelling correction for resource-scarce languages using deep learning. In *Proceedings of ACL 2018, Student Research Workshop*, pages 146–152.

Jianfeng Gao, Chris Quirk, et al. 2010. A large scale ranker-based system for search query spelling correction. In *The 23rd International Conference on Computational Linguistics*.

Shaona Ghosh and Per Ola Kristensson. 2017. Neural networks for text correction and completion in keyboard decoding. *arXiv preprint arXiv:1709.06429*.

Priscila A Gimenes, Norton T Roman, and Ariadne MBR Carvalho. 2015. Spelling error patterns in brazilian portuguese. *Computational Linguistics*, 41(1):175–183.

Roman Grundkiewicz, Marcin Junczys-Dowmunt, and Kenneth Heafield. 2019. Neural grammatical error correction systems with unsupervised pre-training on synthetic data. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 252–263.

Jai Gupta, Zhen Qin, Michael Bendersky, and Donald Metzler. 2019. Personalized online spell correction for personal search. In *The World Wide Web Conference*, pages 2785–2791.

Matthias Hagen, Martin Potthast, Marcel Gohsen, Anja Rathgeber, and Benno Stein. 2017. A large-scale query spelling correction corpus. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1261–1264.

Saša Hasan, Carmen Heger, and Saab Mansour. 2015. Spelling correction of user search queries through statistical machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 451–460, Lisbon, Portugal. Association for Computational Linguistics.

Daniel Hládek, Ján Staš, and Matúš Pleva. 2020. Survey of automatic spelling correction. *Electronics*, 9(10):1670.

Sai Muralidhar Jayanthi, Danish Pruthi, and Graham Neubig. 2020. Neuspell: A neural spelling correction toolkit. *arXiv preprint arXiv:2010.11085*.

Vishal Kakkar, Chinmay Sharma, Madhura Pande, and Surender Kumar. 2023. Search query spell correction with weak supervision in e-commerce. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 5: Industry Track)*, pages 687–694.

Mark Kemighan, Kenneth Church, and William Gale. 2003. A spelling correction program based on a noisy channel model. 2.

Asiye Tuba Koksal, Ozge Bozal, Emre Yürekli, and Gizem Gezici. 2020. # turkishtweets: A benchmark dataset for turkish text correction. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4190–4198.

A Kuznetsov and H Urdiales. Spelling correction with denoising transformer. arxiv 2021. *arXiv preprint arXiv:2105.05977*.

Alex Kuznetsov and Hector Urdiales. 2021. Spelling correction with denoising transformer. *arXiv preprint arXiv:2105.05977*.

Yanen Li, Huizhong Duan, and ChengXiang Zhai. 2011. Cloudspeller: Spelling correction for search queries by using a unified hidden markov model with web-scale resources. In *Spelling Alteration for Web Search Workshop*, pages 10–14. Citeseer.

Yanen Li, Huizhong Duan, and ChengXiang Zhai. 2012. Cloudspeller: query spelling correction by using a unified hidden markov model with web-scale resources. In *Proceedings of the 21st International Conference on World Wide Web*, pages 561–562.

Shulin Liu, Shengkang Song, Tianchi Yue, Tao Yang, Huihui Cai, TingHao Yu, and Shengli Sun. 2022. Craspell: A contextual typo robust approach to improve Chinese spelling correction. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 3008–3018.

Shulin Liu, Tao Yang, Tianchi Yue, Feng Zhang, and Di Wang. 2021. Plome: Pre-training with misspelled knowledge for Chinese spelling correction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2991–3000.

Madhura Pande, Vishal Kakkar, Manish Bansal, Surender Kumar, Chinmay Sharma, Himanshu Malhotra, and Praneet Mehta. 2022. Learning-to-spell: Weak supervision based query correction in e-commerce search with small strong labels. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 3431–3440.

Chanjun Park, Kuekyeng Kim, YeongWook Yang, Minho Kang, and Heuiseok Lim. 2021. Neural spelling correction: translating incorrect sentences to correct sentences for multimedia. *Multimedia Tools and Applications*, 80:34591–34608.

Martin Reynaert. 2004. Multilingual text induced spelling correction. In *Proceedings of the Workshop on Multilingual Linguistic Resources*, pages 110–117, Geneva, Switzerland. COLING.

Ali Safaya, Emirhan Kurtuluş, Arda Goktogan, and Deniz Yuret. 2022. Mukayese: Turkish NLP strikes back. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 846–863, Dublin, Ireland. Association for Computational Linguistics.

Sanat Sharma, Josep Valls-Vargas, Tracy Holloway King, Francois Guerin, and Chirag Arora. 2023. Contextual multilingual spellchecker for user queries. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '23, page 3395–3399, New York, NY, USA. Association for Computing Machinery.

Chengjie Sun, Xiaoqiang Jin, Lei Lin, Yuming Zhao, and Xiaolong Wang. 2015. Convolutional neural networks for correcting english article errors. In *Natural Language Processing and Chinese Computing: 4th CCF Conference, NLPCC 2015, Nanchang, China, October 9-13, 2015, Proceedings 4*, pages 102–110. Springer.

Dilara Torunoglu-Selamet, Eren Bekar, Tugay Ilbay, and Gülsen Eryigit. 2016. Exploring spelling correction approaches for turkish. In *Proceedings of the 1st International Conference on Turkic Computational Linguistics at CICLING, Konya*, pages 7–11.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Robert A Wagner and Michael J Fischer. 1974. The string-to-string correction problem. *Journal of the ACM (JACM)*, 21(1):168–173.

Casey Whitelaw, Ben Hutchinson, Grace Y Chung, and Gerard Ellis. 2009. Using the web for language independent spellchecking and autocorrection.

Fan Yang, Alireza Bagheri Garakani, Yifei Teng, Yan Gao, Jia Liu, Jingyuan Deng, and Yi Sun. 2022. Spelling correction using phonetics in E-commerce search. In *Proceedings of the Fifth Workshop on e-Commerce and NLP (ECNLP 5)*, pages 63–67, Dublin, Ireland. Association for Computational Linguistics.

Dezhi Ye, Bowen Tian, Jiabin Fan, Jie Liu, Tianhua Zhou, Xiang Chen, Mingming Li, and Jin

Ma. 2023. Improving query correction using pre-train language model in search engines. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pages 2999–3008.

Harun Reşit Zafer. 2017. hunspell-tr.

Haoyu Zhang and Qin Zhang. 2017. Embedjoin: Efficient edit similarity joins via embeddings. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 585–594.

Jingfen Zhang, Xuan Guo, Sravan Bodapati, and Christopher Potts. 2023. Multi-teacher distillation for multilingual spelling correction. *arXiv preprint arXiv:2311.11518*.

Shiliang Zhang, Ming Lei, and Zhijie Yan. 2019a. Investigation of transformer based spelling correction model for ctc-based end-to-end mandarin speech recognition. In *Interspeech*, pages 2180–2184.

Shiliang Zhang, Ming Lei, and Zhijie Yan. 2019b. Investigation of Transformer Based Spelling Correction Model for CTC-Based End-to-End Mandarin Speech Recognition. In *Proc. Interspeech 2019*, pages 2180–2184.

Yingbo Zhou, Utkarsh Porwal, and Roberto Konow. 2017. Spelling correction as a foreign language. *arXiv preprint arXiv:1705.07371*.

Imed Zitouni and Ruhi Sarikaya. 2009. Arabic diacritic restoration approach based on maximum entropy models. *Computer Speech Language*, 23(3):257–276.

A. Appendix

A.1. Misspelling Types

- **missing whitespace**: it appears when there is an missing space in the user query, e.g., "banyomusluğu" instead of "banyo musluğu"
- **brand misspellings**: it appears when brand names have a typo, which may occur since users may not know the correct spelling of the non-Turkish brands., e.g., "lenova" instead of "lenovo"
- **phonetic**: it appears when the non-Turkish words are written using their Turkish phones., e.g., "iyfon" instead of "iPhone"
- **missing diacratics**: it appears when Turkish specific letters (ö,ı,ü,ç,ş,ğ) are replaced with their english counterparts., e.g., "banyo muslugu" instead of "banyo musluğu"
- **fat finger**: it appears when keyboard an input mistake occurs, e.g., "bsnyo muskuğı" instead of "banyo musluğu"
- **hard-to-write words**: It appears when users does not know the correct spelling of the words adapted from other languages (e.g., french, persian, arabic) through time, e.g., "hoporlör" instead of " hoparlör"
- **lazy writing**: It appears when there is a missing or extra letter in the user query, e.g., "baanyo msluğu" instead of "banyo musluğu"

A.2. Training Setup & Inference

We trained the transformer model from scratch, employing 8 attention heads with dimensions of 128 for hidden layers and 64 for output layers. The training process utilized the Adam Optimizer and Cosine learning rate scheduler with warm-up. The learning rate was set to 0.001, and $\beta 1$ and $\beta 2$ were configured as 0.8 and 0.998, respectively. The sequence length was determined to be 19, considering the average character length of words within the search queries. For the training phase, we utilized an NVIDIA T4 GPU. It took 29 hours to complete. The inference phase was executed on a CPU setup. The response time <17.3 ms query on average under single concurrency.