Automatic Spelling Correction for Russian: Multiple Error Approach

Kseniia Varlamova
Antiplagiat Company
Moscow Institute of Physics and Technology
Moscow, Russia
kvarlamova@ap-team.ru

Ildar Khabutdinov
Antiplagiat Company
Moscow Institute of Physics and Technology
Moscow, Russia
khabutdinov@ap-team.ru

Andrey Grabovoy
Antiplagiat Company
Moscow Institute of Physics and Technology
Moscow, Russia
grabovoy@ap-team.ru

***Abstract*—To date, the amount of textual information is consistently expanding and reaching wider audiences, leading to a rise in spelling and typography errors. This further accentuates the Automatic Spelling Correction problem, which remains one of the primary tasks of Natural Language Processing. At the moment this problem is not widely studied for the Russian language and supposed models often have the strict limitation of the number of errors in the word. This paper presents a model for Automatic Spelling Correction in the Russian language that can handle multiple error cases without limits on the number of errors processed. The model is based on a probabilistic approach and consists of multiple stages, including classification of word correctness, preliminary candidate search with shingle-based approach, source model, error model with the application of bigrams and phonetics. We outline the process of obtaining data from open sources and investigate different methods of constructing and utilising dictionaries. By searching for candidates using a shingle-based approach with no limit on the number of errors, the model is resistant to multiple error cases. The shingle-based search is compared with the fixed cut distance candidate generation approach. We use several test samples and obtain a top-5 F_1-score of 0.80 on the real data, which is mostly social media, and 0.91 on the hand-crafted sample with multiple errors.**

***Index Terms*—natural language processing, spelling correction, automatic spelling correction, spelling correction for Russian, error correction, candidate search, shingle index, multiple errors**

I. Introduction

Spelling Correction remains a crucial and enduring issue in Natural Language Processing. Every passing moment sees an increase in the quantity of textual data available on the Internet [1], some of which requires fast and high quality spellchecking to avoid typos, search and correct errors in student papers, etc. This places a demand for automated methods of such work. Moreover, any text-focused tool, whether a grammar or punctuation editor [2], query searcher [3], translator [4], lemmatiser [5], and many more, require an autocorrector. Various approaches and techniques aim to solve this problem, with roots going back to the works of Damerau [6] and Levenshtein [7].

It is also necessary to be able to handle cases of multiple errors in order to correct complex and long words, as well as editing text from non-native speakers [8] and output post-processing for number of sequence-to-sequence tasks, such as optical character or speech recognition. At present, the number of processing errors is usually limited, and the case of multiple errors has not been widely studied.

Models of the Spelling Correction task can be classically divided into two sub-models: Source and Error models. We separate the task for preliminary search and exact matching.

For preliminary search, the authors propose several effective ways to store and search dictionaries: hash tables [9], [10], finite automata [11]–[13], various tree-based structures [14]–[16]. The Prefix Tree, or Trie, structure is widely used because of its speed and simple linguistic validity. Most previous methods for preliminary search are based on the generation of words at a fixed edit distance from the incorrect word. These generated words are matched with correct words from the dictionary. In this way, we cannot find correct candidates for words with several errors, more than the threshold. In our work we use shingle-based approach [17], [18] over dictionary for preliminary search. This approach is based on comparing intersecting sequences of N-grams of an incorrect word, shingles, with corresponding hashes in an index file compiled using words from the dictionary. It is effective, gives a baseline immediately and adapts to the input word: candidates are found even for a badly distorted word.

The development of the Source Model has progressed from utilising frequency of word n-grams [19], [20] to implementing window-based approaches [14] and sentence-level processing [21]. With the advancement of neural networks [22], their usage has also been incorporated. Our objective, however, is not to rely heavily on context and construct a technically difficult model. Currently, our model operates at the word level and considers the frequency of unigrams [23].

The calculation of the edit distance weighted by the number of symbol additions, deletions, and substitutions usually forms the basis of the Error Model [23]. The authors in [24] propose using such errors not only at the symbol level but also at the symbol n-gram level. This concept possesses convincing linguistic justification, especially for the Russian language with its rich affix-word formation system, and aligns with our research.

It is important to note that data availability in Russian is bounded. Firstly, there is limited marked-up data available for Spelling Correction and it has a special format. Therefore, we extract training data from our word collection by identifying both correct and incorrect words in large internet corpora [25]. Secondly, wide enough dictionaries and corpora are not publicly available, leading us to use and process limited data without marking.

Our problem can be divided into several subtasks:

- classification of the word correctness;
- data finding, preprocessing and keeping;
- preliminary search of candidates;
- exact matching: finding the probability of possible correction for current incorrect word.

The structure of the model is shown in Fig. 1.

II. Problem Statement

Given a fixed alphabet of symbols:

$$\mathcal{A} = \{a_1, a_2, ..., a_n\}, n \in \mathbb{N},$$

the space of all words consisting of elements from $\mathcal{A}$:

$$\mathcal{K} = \{a_1, .., a_n, (a_1a_1), ..., (a_na_n), ..., (a_ia_ja_ka_l...), ...\}, \\ i, j, k, l = \overline{1, n}.$$

Subspace of a-priori correct words — dictionary:

$$\mathcal{D} = \{(a_i^1, ..., a_j^r), ..., (a_k^1, ..., a_l^s)\}, i, j, k, l = \overline{1, n}; \\ r, s \in \mathbb{N}; i, j, k, l, r, s - \text{fixed}; \mathcal{D} \subset \mathcal{K}.$$

There is a mapping of the correct words to all words:

$$\phi : \mathcal{D} \to \mathcal{K}, \phi(\mathcal{D}) = Im\phi = \mathcal{W}, \phi^{-1}(\mathcal{D}) = \mathcal{D}.$$

Our goal is to find $\phi^{-1} : \mathcal{W} \to \mathcal{D}$. On the word level: $w \in \mathcal{W}$ — the word at the entrance with possible errors. Our goal becomes the search for

$$d : \phi^{-1}(w) = d \Leftrightarrow P(d|w) = \max_{\hat{d}} P(\hat{d}|w), \hat{d} \in \mathcal{D}\}, \text{ where:}$$

$$P(d|w) = \frac{P(d) \cdot P(w|d)}{P(w)} \approx P(d) \cdot P(w|d),$$

because $P(w) = const$ relative to d. As a result, our goal is to find

$$\arg\max_{d} P(d) \cdot P(w|d).$$

The exact matching subtask is divided into finding the a-priori probability of correct words with the Source Model and the conditional probability of an incorrect word given correct word with the Error Model.

III. Proposed Approach

A. Data Description

We need four main types of data: data for word correctness classification, data for training with pairs of correct and incorrect words, data for dictionary of correct words and data for Source Model.

Regarding the dictionary, we utilise an open-source corpus of texts to retrieve the collection of distinct words, taking into consideration their frequency for the Source Model. Due to the texts not being perfectly linguistically composed, they contain superfluous symbols that are not included in the language's alphabet, as well as inaccurate words. Consequently, data preprocessing is necessary. A threshold is selected to minimise false-positive and false-negative occurrences within the dictionary, facilitating the exclusion of incorrect words through frequency cutting off.

To decide if the input word is in need of correction, we check its inclusion in our dictionary of correct words. There is a problem of false triggering of the model if the word is included in the dictionary but some of its word forms are not. We build a mapping to an extended dictionary of all possible forms of words from our dictionary:

$$\psi : \mathcal{D} \to \hat{\mathcal{D}},$$

where $\mathcal{D}$ — dictionary built from text collection; $\hat{\mathcal{D}}$ — extended dictionary of wordforms. The extended dictionary is used for word correctness classification. We do not use it for the next model work because it has no correct information about the a-priori probability of low-frequency wordforms. In relation to frequency selection of dictionary:

$$\forall \tilde{d} \in \hat{\mathcal{D}} \backslash \mathcal{D} \hookrightarrow P(\tilde{d}) \approx 0$$

$$\Rightarrow \arg\max_{\hat{d} \in \hat{D}} P(\hat{d}) \cdot P(\omega|\hat{d}) \approx \arg\max_{d \in D} P(d) \cdot P(\omega|d).$$

We need a collection of word pairs featuring correct and incorrect spellings for training purposes. There is a scarcity of extensively annotated datasets with misspelt words in Russian, and we supplement these with data obtained from our dictionary before cutting off.

Let $\mathcal{D}$ be our produced resulting dictionary of correct words, s — low-frequency word not included in the resulting dictionary, $s \notin \hat{\mathcal{D}}$.

$$\mathcal{C} = \{c : c \in \mathcal{D}, \quad Distance(s, c) = 1, \\ Frequency(c) > t \cdot Frequency(s)\},$$

where t — hyperparameter, $t > 1$, $t \approx 10$ [11]; $Distance(s, c)$ is the minimum number of operations of insertion, deletion, replacement of symbols required to form s from c; $Frequency(word)$ is the number of occurrences of the $word$ in the corpus. The resulting pair for training:

$$\text{pair} = \{s, \arg\max_{c} Frequency(c \in \mathcal{C})\}. \tag{1}$$

Resulting dictionary and training pairs are not perfect and quite noisy, but are meaningful in the insufficient availability

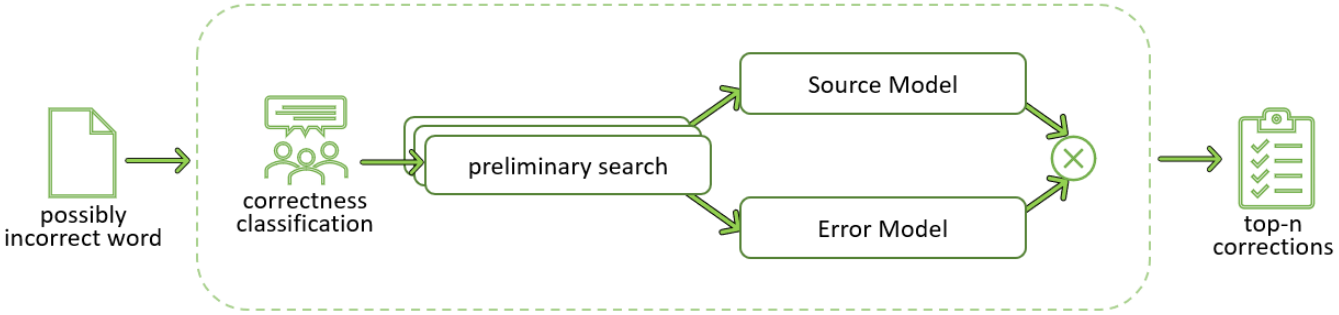

Figure 1. General structure of the model for the Spelling Correction problem

of uncontaminated data. We release the training dataset [26] to expand the available data for Spelling Correction problem in Russian.

B. Preliminary Search

We have to calculate the probability and find the word with the highest probability among the words in the dictionary. However, performing calculations with all the correct words is quite costly. Hence, an approximate list of potential candidates must be obtained and used for exact matching.

We index dictionary of correct words using shingles. Each word in the dictionary goes into the sequence of n-grams of symbols. The sequence is represented as shingles — intersecting sub-sequences of n-grams. We use all values of n from one to the length of a word. The shingles are hashed and written to an index file. The input incorrect word w is also hashed at the preliminary search stage. Then the dictionary index file is searched. If there are matches, the word identifiers found are added to the list of candidate words. For each candidate word d, the number of matches of the hash values with the values of the shingle hashes of w is calculated. The measure of the agreement of w with d is the total length of the fragments that match in both words, related to the length of w. The list of candidates is ranged and truncated by this measure. The number of candidates output is chosen as the hyperparameter.

In this way, the search outcomes exhibit a near-ideal score when the query word is paired with any quantity of supplementary symbols. Thus, the word "там" would have searching results "самолетам", "ракетам", "таможня", "тамбов" etc., and the word "там" itself may be at the end of such list. To avoid this problem, we shingle words with additional special symbols: d — our word for shingling; $<d>$ — resulting variant for shingling. In this way, we include n-grams indicating the beginning and end of the word in the sequence for shingles.

C. Exact Matching

As a prior probability of correct words, we use their frequency in our text collection described in subsection III-A.

The Error Model uses a weighted edit distance. For each pair of training pairs, we build all possible matches at the 1-2 gram level, minimising the Levenshtein distance between correct and incorrect words. The error matrix is filled: each element is the number of substitutions between two symbols, bigrams or empty lines in the training pairs. The resulting edit distance has a probabilistic meaning [24]:

$$P(w|d) \approx max \prod_{i=1}^{N} P(\alpha_i|\beta_i),$$
$$w = \alpha_1\alpha_2...\alpha_N, d = \beta_1\beta_2...\beta_N,$$
$$\alpha_i, \beta_i = a^1a^2, \quad a^j \in \mathcal{A} \cup \{\text{“”}\},$$

$\mathcal{A}$ — our alphabet, “” — empty line, for deletion and insertion of symbols or n-grams. $P(w|d)$ calculates using an algorithm similar to the Wagner-Fischer algorithm [27]: M—matrix of probabilities: $M_i^j = P(w[:i]|d[:j])$, $w[:i]$—prefix of word w of length i, w_i—symbol of word w on the place i

$$M_i^j = max \begin{cases} M_{i-1}^j \cdot P(w_i|\text{“”}), \\ M_i^{j-1} P \cdot (\text{“”}|d_j), \\ M_{i-1}^{j-1} \cdot P(w_i|d_j), \\ M_{i-2}^j \cdot P(w_{i-1}w_i|\text{“”}), \\ M_i^{j-2} \cdot P(\text{“”}|d_{j-1}d_j), \\ M_{i-2}^{j-1} \cdot P(w_{i-1}w_i|d_j), \\ M_{i-1}^{j-2} \cdot P(w_i|d_{j-1}d_j), \\ M_{i-2}^{j-2} \cdot P(w_{i-1}w_i|d_{j-1}d_j), \end{cases}$$

where the probabilities are taken from the Error Matrix, normalised to the number of each character or n-gram in the training set:

$$P(w_i|d_j) = \frac{ErrorMatrix_{w_i}^{d_j}}{Quantity(w_i) + Quantity(d_j)},$$

$Quantity(w_i)$ is number of w_i in incorrect words of training pairs, $Quantity(d_i)$ is number of d_i in correct words of training pairs.

This method includes a phonetic approach as a special addition. Spelling errors are often caused by the sound of a word, which can differ greatly from its correct spelling. Therefore, the previous approach can miss some mistakes: “хочеца” — “хочется”, “щясливый” — “счастливый”. For

this reason, we use a simplified phonetic analysis [21], [28] to improve the relevance of candidates. We create a list of symbols and bigrams of symbols for the Russian language that may have similar sounds. Using this list, we compare the candidates from the preliminary search output and an input incorrect word for phonetic equivalence. If there is a match, we assign a higher score to these candidates.

IV. Experiments

A. Data Description

We use a set of words derived from the Wikipedia text collection [29], taking into account their frequency. This gives us 5.1 million unique words. Remarkably, the Wikipedia data turned out to be quite noisy: we removed 1.5 million words containing Latin symbols, numbers and punctuation marks. The remaining data was divided according to frequency. As a result, the dictionary consists of 0.7 million words with a frequency higher than the selected threshold.

As for the marked-up collections, we use Dialog [30] and RULEC [31] datasets. These datasets have a number of problems:

- small amount: 3.3 thousand pairs before train-test division;
- specific lexicon and mistakes in Dialog data due to their social media source ("мож" — "может", "низзя" — "нельзя", "дооолго" — "долго");
- errors that are not part of our task: cases where the incorrect word is in language dictionaries, but is not correct in this context; ("иностранного" — "иностранному", "возиться" — "возится").

Using the equation III-A we got about a thousand pairs out of 300 thousand low frequency words. This way also has disadvantages: the data is noisy with existing "incorrect" words like "ужасать" — "угасать" and wrong matches like "уходют" — "уходит". Also, it is not perfect to test on this data, because incorrect words have only 1 error.

We use several samples for testing:

- *general*: the test sample consists of a portion of data from marked data and pairs obtained from Wikipedia;
- *marked-up*: the sample of marked-up data only;
- *noisy*: the sample of words with several errors that are common in the Russian language, such as infinitives: "развиваца" — "развиваться", numerals: "пятьюсот" — "пятьюстами", slang: "низя" — "нельзя" and long words, including terms: "индитефекация" — "идентификация";
- *2-errors*: the sample of words from the marked-up sample with 2 errors.

B. Preliminary Search

We use maximum number of n-grams for every word for indexing and matching. We got approximately 11.2 million shingles after dictionary indexing. The top-k accuracy metric is calculated for selection of candidates number. It is equal to the proportion of accurately identified words among the top k candidates, divided by the total number of words searched. The results are shown in the table I with clearing of pairs where correct word does not include in the dictionary — *cl* and without clearing — *ncl*.

Table I
Metric for preliminary search

k	sample	accuracy			k	accuracy	
		cl	*ncl*			*cl*	*ncl*
5000	*general*	1.00	0.93		128	0.80	0.71
	marked-up	0.96	0.90			0.73	0.69
	noisy	1.00	1.00			0.97	0.97
	2-errors	0.86	0.78			0.54	0.54
3500	*general*	1.00	0.93		10	0.47	0.40
	marked-up	0.95	0.90			0.36	0.34
	noisy	0.99	0.99			0.72	0.72
	2-errors	0.84	0.76			0.25	0.23
2048	*general*	1.00	0.92		1	0.18	0.16
	marked-up	0.93	0.88			0.13	0.12
	noisy	0.99	0.99			0.33	0.33
	2-errors	0.83	0.75			0.03	0.03
1024	*general*	0.97	0.89				
	marked-up	0.89	0.84				
	noisy	0.99	0.99				
	2-errors	0.73	0.67				

Table II
Comparing different dictionaries for indexing in preliminary search on marked-up data

dictionary	size of dictionary	mean k	accuracy	work time, s
$\mathcal{D}$	680K	3500	0.90	34
$\hat{\mathcal{D}}$	4.1M	3500	0.85	43
$\mathcal{L}$	520K	5000	0.87	38
$\mathcal{S}$	470K	5000	0.89	40

In order to avoid the problem of false triggering of the model on low frequency wordforms, we build the extended dictionary of wordforms. We also research the use of a narrowed dictionary with a known mapping to an extended dictionary for the preliminary search. The idea is to reduce time and memory costs. We perform experiments comparing the model work on different dictionaries. We measure accuracy, number of candidates and working time when indexing is made using:

- $\mathcal{D}$ — our dictionary;
- $\hat{\mathcal{D}}$ — dictionary of all possible forms of words from $\mathcal{D}$;
- $\mathcal{L}$ — dictionary of lemmatized words from $\mathcal{D}$;
- $\mathcal{S}$ — dictionary of stemmized words from $\hat{\mathcal{D}}$.

Wordforms, stems and lemmas are found effectively due to morphological analyzer [32].

The results are presented in the Table II. According to the experiment, stemmized dictionary is more effective than lemmatized one. It can be explained by the phenomenon of a possible significant difference between the wordform and its lemma: "церемонясь" — "церемониться", "построенный" — "построить". However, stemmized dictionary requires greater number of candidates for appropriate quality in comparison with $\mathcal{D}$. Firstly, stem is usually shorter than wordform, in some cases greatly: "умнейшего" — "умн", "родиться"

— “род”. This means that more candidates are found at the same editing distance. Secondly, part of speech information is often lost, which can lead to irrelevant candidates being found. Thirdly, there is the problem that the incorrect word is lemmatised or stemmed in a non-deterministic way. Using the extended dictionary is also less effective than $\mathcal{D}$ due to surge in the number of candidates.

As a result, we use $\mathcal{D}$ for Preliminary Search and $\hat{\mathcal{D}}$ — for word correctness checking and for possible changes of wordform in the final stages of the model work. We also use dictionary $\mathcal{D}$ for Source Model.

We compare our preliminary search method with the method of word generation and search with threshold equal to 2 using Trie [12], [21] by speed, result working time of the model, top-k accuracy and number of candidates. The *Shingle* method has a fixed number of resulting candidates and cannot change much with dictionary expansion or increasing the threshold. However, the *Trie* method works with an ideal score in the case of a small number of errors, unlike our method. On the other hand, it does not work at all in cases where the number of errors is higher than the threshold. In the same situation, *Shingle* method adapts to the number of errors in the data and continues to function correctly. The results of the experiment are shown in the Table III. They are obtained on the *marked* and *noisy* data.

Table III
COMPARING SHINGLE AND TRIE METHODS FOR PRELIMINARY SEARCH

method	data	mean k	accuracy	time, s	result time, s
Shingle	*marked-up*	3500	0.90	34	112
Trie	*marked-up*	330	0.91	141	148
Shingle	*noisy*	1024	0.99	165	192
Trie	*noisy*	6	0.27	2457	2458

Meanwhile, the *Shingle* method has no formal restrictions on word noise level, so it works much more effectively in cases of quite noisy data. The *Shingle* method is also significantly faster, provided the dictionary is correctly selected.

C. Exact Matching

In our work, we calculate edit distance on the 1 and 2-symbols level.

Tables IV, V show examples of pair matching and Error matrix before normalising and zero processing.

Table IV
WORD PAIR MATCHING EXAMPLE FOR ERROR MODEL TRAINING

	щ	я	с		л	е	в	о
	сч	а	ст		л	и	в	о
	сч	а	с	т	л	и	в	о
с	ч	а	ст		л	и	в	о
				...				
с	ч	а	с	т	л	и	в	о

Possible substitutions: [“ ”, “с”], [“щ”, “сч”], [“щ”, “ч”], [“ща”, “ча”], [“с”, “ст”], [“ ”, “т”], [“л”, “тл”], [“ле”, “ли”], [“е”, “и”], [“ев”, “ив”].

Table V
ERROR MATRIX

	а	б	...	я	аа	...	яя	”
а	0	30		90	5		0	897
б	11	0		4	0		0	387
...								
я	153	8		0	0		5	211
аа	85	0		0	0		0	3
...								
яя	0	0		7	0		0	0
”	382	96		68	0		0	0

For the phonetic accounting, we create clusters with symbols or n-grams of symbols that can sound the same: [а,о,э], [я,йа], [ща,щя], [сч,щ], ..., [жь,ж]. The incorrect word and the candidate are checked for phonetic equivalence according to these clusters and, in the positive case, they are considered with an increased weight with the selected hyperparameter.

We present the results on each test sample in the tables VI, VII, VIII, IX, where *k* is the average number of candidates produced by the final model; *N* is the maximum number of possible symbols in n-grams; *-Ph* with the added weight of phonetic equivalence; *TP* is the normalised set of accurate corrections for incorrect words; *FP* is the normalised amount of wrong correction for incorrect words; *FN* is the normalised number of incorrect words not corrected; F_1 is the F_1-score: $F_1 = \frac{2 \cdot TP}{2 \cdot TP + FP + FN}$

Table VI
METRICS ON GENERAL TEST SAMPLE

k	6			4			1		
N	1	2	2-*Ph*	1	2	2-*Ph*	1	2	2-*Ph*
TP	0.92	0.92	0.92	0.90	0.89	0.89	0.70	0.70	0.68
FP	0.07	0.07	0.07	0.09	0.10	0.10	0.29	0.29	0.31
FN	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
F_1	0.96	0.96	0.96	0.95	0.94	0.94	0.82	0.82	0.81

Table VII
METRICS ON MARKED-UP TEST SAMPLE

k	5			4			1		
N	1	2	2-*Ph*	1	2	2-*Ph*	1	2	2-*Ph*
TP	0.67	0.67	0.67	0.65	0.65	0.66	0.56	0.58	0.58
FP	0.07	0.07	0.07	0.09	0.09	0.08	0.18	0.16	0.16
FN	0.26	0.26	0.26	0.26	0.26	0.26	0.26	0.26	0.26
F_1	0.80	0.80	0.80	0.79	0.79	0.80	0.72	0.73	0.73

Table VIII
METRICS ON NOISY TEST SAMPLE

k	5			3			1		
N	1	2	2-*Ph*	1	2	2-*Ph*	1	2	2-*Ph*
TP	0.83	0.82	0.82	0.79	0.79	0.79	0.72	0.72	0.72
FP	0.17	0.18	0.18	0.21	0.21	0.21	0.28	0.28	0.28
FN	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
F_1	0.91	0.90	0.90	0.88	0.88	0.88	0.84	0.84	0.84

Table IX
METRICS ON 2-ERRORS TEST SAMPLE

k	5			3			1		
N	1	2	2-*Ph*	1	2	2-*Ph*	1	2	2-*Ph*
TP	0.62	0.60	0.60	0.57	0.57	0.57	0.32	0.39	0.41
FP	0.14	0.16	0.16	0.19	0.19	0.19	0.44	0.37	0.35
FN	0.24	0.24	0.24	0.24	0.24	0.24	0.24	0.24	0.24
F_1	0.77	0.75	0.75	0.73	0.73	0.73	0.48	0.56	0.58

FN can be as much of a disadvantage if the dictionary contains incorrect words as if the test data contains correct words that are marked as incorrect. Due to errors irrelevant to our problem, the marked data — tables VII, IX — has a significantly larger *FN*: “встретиться” — “встретится”, “чувстве” — “чувство”, “придел” — “предел”.

The use of bigrams and phonetics is meaningful in the case of data with an a priori sufficiently large number of spelling errors compared to typos. It is particularly useful when we need a top-1 candidate as an output. This can be seen in the Table IX. Results in the Table VIII show high enough quality in the case of multiple error cases.

V. CONCLUSION

Research into the Spelling Correction problem is scarce for Russian, especially when it comes to processing data with many errors. We have built a technically, ideologically simple and effective model of Automatic Spelling Correction for the Russian language. We followed a probabilistic approach with corresponding sub-tasks Error and Source model. We used bigram edit distance with phonetic application for the Error Model and word frequency for the Source Model. The limit on the number of errors in a word has been removed. Applying the shingle-based approach and building the dictionary, we achieved a top-5 F_1-score of 0.91 on noisy data. All methods were applied using open source data only. We also proposed our obtained training dataset.

As for the further work, we suppose to consider the nearest context which will allow us to reduce the number of result candidates keeping the same high quality. We are thinking about making the dictionary cutting off more qualitative by taking into account factors beyond just frequency. We also want to explore the application of our model to real-world data with multiple errors, such as spelling correction for non-native speakers, optical character recognition or speech recognition output.

REFERENCES

[1] D. Rajput, “Review on recent developments in frequent itemset based document clustering, its research trends and applications,” *Int. J. Data Anal. Tech. Strateg.*, vol. 11, pp. 176–195, 2019.

[2] 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)*, 2017, pp. 793–805.

[3] H. Duan, Y. Li, C. Zhai, and D. Roth, “A discriminative model for query spelling correction with latent structural svm,” in *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning*, 2012, pp. 1511–1521.

[4] S. Hasan, C. Heger, and S. Mansour, “Spelling correction of user search queries through statistical machine translation,” in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015, pp. 451–460.

[5] S. Salavati and S. Ahmadi, “Building a lemmatizer and a spell-checker for sorani kurdish,” *arXiv preprint arXiv:1809.10763*, 2018.

[6] F. J. Damerau, “A technique for computer detection and correction of spelling errors,” *Communications of the ACM*, vol. 7, no. 3, pp. 171–176, 1964.

[7] V. I. Levenshtein *et al.*, “Binary codes capable of correcting deletions, insertions, and reversals,” in *Soviet physics doklady*, vol. 10, no. 8. Soviet Union, 1966, pp. 707–710.

[8] K. Shaalan, R. Aref, and A. Fahmy, “An approach for analyzing and correcting spelling errors for non-native arabic learners,” in *2010 The 7th International Conference on Informatics and Systems (INFOS)*, 2010, pp. 1–7.

[9] K. Kukich, “Techniques for automatically correcting words in text,” *ACM computing surveys (CSUR)*, vol. 24, no. 4, pp. 377–439, 1992.

[10] L. Salifou and H. Naroua, “Design of a spell corrector for hausa language,” *international Journal of computational Linguistics (IJCL)*, vol. 5, no. 2, pp. 14–26, 2014.

[11] M. Hulden, “Fast approximate string matching with finite automata,” 2009.

[12] K. Oflazer, “Error-tolerant finite state recognition with applications to morphological analysis and spelling correction,” *arXiv preprint cmp-lg/9504031*, 1995.

[13] A. V. Aho and M. J. Corasick, “Efficient string matching: an aid to bibliographic search,” *Communications of the ACM*, vol. 18, no. 6, pp. 333–340, 1975.

[14] Y. Chaabi and F. A. Allah, “Amazigh spell checker using damerau-levenshtein algorithm and n-gram,” *Journal of King Saud University-Computer and Information Sciences*, vol. 34, no. 8, pp. 6116–6124, 2022.

[15] T. Mardiana, T. B. Adji, and I. Hidayah, “Stemming influence on similarity detection of abstract written in indonesia,” *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, vol. 14, no. 1, pp. 219–227, 2016.

[16] S. W. Bent, D. D. Sleator, and R. E. Tarjan, “Biased search trees,” *SIAM Journal on Computing*, vol. 14, no. 3, pp. 545–568, 1985.

[17] A. Z. Broder, S. C. Glassman, M. S. Manasse, and G. Zweig, “Syntactic clustering of the web,” *Computer networks and ISDN systems*, vol. 29, no. 8-13, pp. 1157–1166, 1997.

[18] A. Z. Broder, “On the resemblance and containment of documents,” in *Proceedings. Compression and Complexity of SEQUENCES 1997 (Cat. No. 97TB100171)*. IEEE, 1997, pp. 21–29.

[19] A. R. Golding and D. Roth, “A winnow-based approach to context-sensitive spelling correction,” *Machine learning*, vol. 34, pp. 107–130, 1999.

[20] E. Mays, F. J. Damerau, and R. L. Mercer, “Context based spelling correction,” *Information Processing & Management*, vol. 27, no. 5, pp. 517–522, 1991.

[21] A. Sorokin and T. Shavrina, “Automatic spelling correction for russian social media texts,” 06 2016.

[22] A. Pal and A. Mustafi, “Vartani spellcheck–automatic context-sensitive spelling correction of ocr-generated hindi text using bert and levenshtein distance,” *arXiv preprint arXiv:2012.07652*, 2020.

[23] M. D. Kernighan, K. Church, and W. A. Gale, “A spelling correction program based on a noisy channel model,” in *COLING 1990 Volume 2: Papers presented to the 13th International Conference on Computational Linguistics*, 1990.

[24] E. Brill and R. C. Moore, “An improved error model for noisy channel spelling correction,” in *Proceedings of the 38th annual meeting of the association for computational linguistics*, 2000, pp. 286–293.

[25] C. Whitelaw, B. Hutchinson, G. Y. Chung, and G. Ellis, “Using the Web for language independent spellchecking and autocorrection,” in *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, P. Koehn and R. Mihalcea, Eds. Singapore: Association for Computational Linguistics, Aug. 2009, pp. 890–899. [Online]. Available: https://aclanthology.org/D09-1093

[26] K. Varlamova, I. Khabutdinov, and A. Grabovoy, “Open access dataset for spelling correction in russian,” *Mendeley Data*, 2023.

[27] R. A. Wagner and M. J. Fischer, “The string-to-string correction problem,” *Journal of the ACM (JACM)*, vol. 21, no. 1, pp. 168–173, 1974.

[28] F. Yang, A. B. Garakani, Y. Teng, Y. Gao, J. Liu, J. Deng, and Y. Sun, "Spelling correction using phonetics in e-commerce search," 2022.
[29] D. Vrandečić and M. Krötzsch, "Wikidata: A free collaborative knowledgebase," *Communications of the ACM*, vol. 57, no. 10, pp. 78–85, 2014.
[30] A. Sorokin, I. Galinskaya, and T. Shavrina, "Spellrueval: The first competition on automatic spelling correction for russian."
[31] A. Rozovskaya and D. Roth, "Grammar error correction in morphologically rich languages: The case of russian," *Transactions of the Association for Computational Linguistics*, vol. 7, pp. 1–17, 2019.
[32] M. Korobov, "Morphological analyzer and generator for russian and ukrainian languages," in *Analysis of Images, Social Networks and Texts: 4th International Conference, AIST 2015, Yekaterinburg, Russia, April 9–11, 2015, Revised Selected Papers 4*. Springer, 2015, pp. 320–332.