Computational Linguistics and Intellectual Technologies:
Proceedings of the International Conference "Dialogue 2016"

Moscow, June 1–4, 2016

AUTOMATIC SPELLING CORRECTION FOR RUSSIAN SOCIAL MEDIA TEXTS

Sorokin A. A. (alexey.sorokin@list.ru)[1,2,3],
Shavrina T. O. (rybolos@gmail.com)[1,3]

[1]Lomonosov Moscow State University, Moscow, Russia

[2]Moscow Institute of Science and Technology, Dolgoprudny, Russia

[3]General Internet Corpus of Russian, Moscow, Russia

This paper describes an automatic spelling correction system for Russian. The system utilizes information from different levels, using edit distance for candidate search and a combination of weighted edit distance and language model for candidate hypotheses selection. The hypotheses are then reranked by logistic regression using edit distance score, language model score etc. as features. We also experimented with morphological and semantic features but did not get any advantage. Our system has won the first SpellRuEval competition for Russian spell checkers by all the metrics and achieved F1-Measure of 75%.

Key words: automatic spelling correction, Russian spellchecking, non-word errors, real-word errors, spelling correction, spelling correction for Russian, spelling correction on corpora, social media language

АВТОМАТИЧЕСКОЕ ИСПРАВЛЕНИЕ ОПЕЧАТОК И ОРФОГРАФИЧЕСКИХ ОШИБОК ДЛЯ РУССКОЯЗЫЧНЫХ СОЦИАЛЬНЫХ МЕДИА

Сорокин А. А. (alexey.sorokin@list.ru)[1,2,3],
Шаврина Т. О. (rybolos@gmail.com)[1,3]

[1]МГУ им. М. В. Ломоносова, Москва, Россия;

[2]МФТИ, Долгопрудный, Россия;

[3]ГИКРЯ, Москва, Россия

В данной статье исследуется многоуровневый метод исправления опечаток для русскоязычных текстов, взятых из сети Интернет. Исправление опечаток является особенно важной проблемой в связи с повсеместным использованием социальных медиа в качестве источника для лингвистических исследований. Мы используем комбинацию нескольких методов, в частности основанных на расстоянии Левенштейна и словарном поиске, а также контекстном ранжировании гипотез с помощью алгоритмов машинного обучения. Наша система заняла 1 место в первом соревновании SpellRuEval по автоматическому исправлению опечаток для русского языка, достигнув F1-меры в 75%.

Ключевые слова: исправление опечаток, автоматическое исправление опечаток, язык социальных медиа, нормализация текста, словарные опечатки

1. Introduction

Spelling correction is one of the oldest and most important problems of computational linguistics. It has attracted many researchers since the pioneer works of Levenshtein and Damerau in the 60-s [Levenshtein, 1965; Damerau, 1964] through the studies on isolated word correction in the beginning of modern NLP era, such as [Kernighan et al., 1989] and early context-based methods [Golding and Roth, 1999] to sophisticated machine-learning based techniques of last decade used in [Whitelaw, 2009] and [Schaback, 2007]. Automatic spellchecking is a problem of high practical importance, especially concerning actual technical requirements of big corpora [Popescu, Phuoc, 2014]. The most straightforward application of it is query correction and completion, used in search engines, as well as orthography correctors which form a part of any modern text editor. More marginal applications include second language learning [Flor, 2012] and grammatical error correction [Rozovskaya, 2013].

In the next section, we describe present-day situation in the field of spelling correction. Section 3 contains a detailed overview of the system developed. Section 4 describes the problems we faced during the development of our system and some open questions left, while Section 5 discusses the results attained. Finally, in Section 6, we offer a brief conclusion of our research and propose some directions for its future application.

2. Past and present of spellchecking: methods and problems

Different researchers focus on different aspects of spelling correction in their studies. Most of the early works dealt with effective search of candidates for typo correction, addressing the problem of fast dictionary lookup and approximate string matching, which is especially important for agglutinative and polysynthetic languages. The task of candidate search is alleviated by the fact that 80% of time a correct word can be obtained from the mistyped word by one primitive edit operation[1] [Kukich, 1992]. However, orthographic mistakes usually happen on phonetic level, while spelling correction is performed on the graphic one, which complicates the search when phonetic and graphic representations of do not exactly map to each other: the [Toutanova et al., 2002]. Moreover, not all primitive edit operations and orthographic changes are equiprobable which means that a variant of weighted Levenshtein distance should be used instead of the basic one to achieve better performance [Kernighan et al., 1990; Ristad, Yanilos, 1995].

Correction systems used in text editors, such as Notepad++ or MS Word, usually suggest several candidates, compelling the user to select between them. However, the number of variants even for a medium-length sentence is too high, which implies that ideal spellchecker should be able not only to generate corrections, but also to select the best one in the given context. Another difficulty concerns the typos producing another dictionary word (such as *piece/peace* or *компания/кампания*). Such problems are a subject of context-sensitive spelling correction [Golding, Roth, 1999; Carlson, Fette, 2007]. Most studies address this task in a narrow fashion, trying to correct real-word spelling errors only in the groups of several predefined confusion sets [Pedler, Mitton, 2010]. Then for every confusion set the task becomes a usual classification problem which can be solved using standard machine-learning techniques. However, this method cannot be straightforwardly generalized to real-world spelling correction since the dimension of feature space becomes too high (the most standard features for this task are adjacent words, which means that every pair of dictionary words is a separate feature). Another approach which we pursued in our work is to learn a low-dimensional classifier which uses the scores given by error and language models as its features.

The difficulty of spelling correction also depends from its domain of application and the source language. Indeed, the more fine-grained is the morphology system,

[1] Primitive edit operations include a) deletion of a single character, b) insertion of a single character, c) substitution of one character for another, d) permutation of a pair of adjacent characters.

the larger is the dictionary, which complicates candidate search and selection. This also makes the data more sparse implying that larger corpora are necessary to learn the language model. Using World Wide Web as a huge unannotated corpora partially solves the problem, but in this case the training data already contains typos which can deteriorate the performance of correction system. Moreover, the percenttage of out-of-vocabulary words, such as proper names, slang and neologisms, is very high for the Web creating another obstacle for the dictionary-based approach. That's why some authors [Whitelaw, 2009] even refuse to use the dictionary basing their algorithms only on corpora frequency.

We are especially interested in spelling correction when applied to social media texts, such as Live Journal, VKontakte and other blogs and social networks. The percentage of misspelled words in such texts is rather high both due to typos and orthography errors and effective correction of such errors is a necessary preliminary condition for further processing such as morphological and syntactical parsing. The percentage of out-of-vocabulary words is also rather high. Our work is a part of General Internet Corpora of Russian (GICR) Project [Belikov et al, 2013]. There are very few works on spelling correction for Russian [Baytin, 2008], [Panina et al., 2013], [Sorokin and Shavrina, 2015]; moreover, the first two are concerned primarily with correction of mistyped search queries, while the latter addresses only to isolated word correction.

3. Our system

Our system participated in the first competition of spellcheckers for Russian SpellRuEval-2016 and won the first place by all the measures, including precision, recall, F1-measure and percentage of correct sentences. We decided to follow the scheme described in [Schaback, 2007] and [Flor, 2013] by collecting scores from different levels including dictionary model, n-gram language model, a weighted error model and morphological error model and combining them in a single linear classifier. We used the reranking approach [Zhang, 2006] often applied in machine translation [Shen et al., 2006]: the algorithm first created n-best lists of candidate sentences according to the simplest of the models and then reranked these hypotheses using logistic regression classifier. We observed that reranking indeed leads to a consistent gain in performance. Other results were quite surprising for us: we observed that morphological and semantic features does not give any further improvement after applying the error model, which either implies that the features we used are too weak to distinguish good hypotheses from the best ones in comparison with other features or that our model of morphology and semantics was not adequate for this task.

3.1. Multi-level spelling correction

In this section we describe our algorithm of spelling errors correction. The algorithm processes one sentence at a time and consists of two stages. On the first stage we rank the candidate hypotheses according to a baseline model. Then for every candidate correction several scores are calculated. These scores include the number of words corrected, the logarithmic probability of the sentence according to the language model, the logarithmic probability of the source sentence to be obtained from the correction by the error model and several other scores characterizing the adequacy and quality of the corrections. These scores were given to a linear classifier as features and the candidate sentence with the highest score was selected. The weights of the classifier were trained on the development set. We describe all these operations further in the article.

3.2. Candidate generation

When generating the candidate sentences, the first step is to find possible corrections for every word in this sentence. The first part of this list consists of all the words on the edit distance 1 from the source as well as the source word itself, no matter whether it appears in the dictionary or not. We used the list of words from ABBYY Compreno [http://www.abbyy.ru/isearch/compreno/] dictionary, which includes approx. 3.7 million words. We store the dictionary as a prefix tree which allows us to effectively search for the words on the distance *d* or less. We selected *d=1* since larger values lead to a drastic expansion of candidate list. The search procedure follows the algorithm described in [Oflazer, 1996] with the heuristics used in [Hulden, 2009], for a more detailed description we refer the reader to [Sorokin and Shavrina, 2015]. We transformed all the words to lower case but preserve the information about the capitalization of source word since the abbreviations written by all capitals and lowercase common words have different probabilities to be mistyped. All the words which contain non-alphabetic characters such as Latin letters or digits were copied to the output without candidate search except certain special cases like (в4ера → вчера).

However, this error model does not capture several frequent patterns, such as *цца* → *тся/ться* transformation in the verb flexion (*появляцца* → *появляться, появицца* → *появится*), which is very popular in Russian Internet slang. We deal with this problem by using an analogue of Metaphone algorithm [Philips, 2000], mapping the sounds to their phonetic classes. We used the following table of classes:

Table 1. Mapping from symbols to phonetic codes

code	Russian letters	code	Russian letters	code	Russian letters
1	а, о, ы, у, я	8	г, к, х	13	з, с
3	и, е, ё, ю, я, э	9	л	14	й
5	б, п	10	р	15	щ, ч
6	в, ф	11	м	16	ж, ш
7	д, т	12	н	17	ц

The symbols also affect the code of consequent symbols: all the vowels after the *ь, ъ, щ, ч, й* letters were mapped to class 3, as well as to class 1 after *ш, ж* and *ц*. The signs *ь, ъ* were omitted. To deal with multisymbol sequences we mapped the *тс* sequence possibly with sign letters between them to the class 17, we also omitted *т* after a sibilant and before other consonant (for example, in the word *грустный* and compressed consecutive occurrences of the same code to a single one. To use this algorithm in spelling correction, we extended the list of candidates by all the dictionary words having the same phonetic code as the source word. However, these transformation does not capture all irregular patterns, so we handcoded a list of about 50 transformations such as *ваще* → *вообще* and *грит* → *говорит*.

Another frequent error pattern is insertion/deletion of space symbol. When the space is inserted in the middle of the word, it is straightforward to model it by allowing the algorithm to traverse from the terminal node of the dictionary tree to its root and paying the special cost for space insertion on this edge. This modification allows us to recognize all the tuples of dictionary word separated by one or more spaces. An analogous problem arises when considering the deletion/insertion of hyphen (-) symbol in composite words, we solved it by adding '-' to the alphabet. This method cannot handle space deletion since we process the sentence word by word. Therefore we performed the candidate search not only for every single word, but also for the groups of two consecutive words.

Given a sentence of 10 words and 3 dictionary candidates for every word in average, which is more probably an underestimate than an overestimate, we obtain approximately 60,000 candidate sentences for every source sentence, which is obviously impossible to handle. Therefore the procedures of candidate generation and baseline candidate ranking cannot be separated. We rank the candidates by the sum $C_{lm}+C_{err}$ of language model score C_{lm} and basic error model score C_{err} (the least score is the best). We describe language model in the next subsection and now we explain the basic error model.

Given a source sentence $u_1, \ldots, u_n$, we generate the candidate hypothesis $v_1, \ldots, v_m$ by groups, for example (1):

(1)	*кто то*	*ищо*	*сделал*	*тоже*	*предположение*
	кто-то	*ещё*	*сделал*	*то же*	*предположение*

Here we have 5 groups (*кто то, кто-то*), (*ищо ещё*), (*сделал, сделал*), (*тоже, то же*), (*предположение, предположение*) and denote them by $g_1, \ldots, g_5$. If the partition of the source-candidate pair of sentences include r groups $g_1, \ldots, g_r$, each group including the source word group s_i and correction word group t_i, then the overall cost of this transformation is $-\sum_{i=1}^{r} \log C(s_i \to t_i)$ where $C(s_i \to t_i)$ is the cost of transforming s_i to t_i. This cost is calculated by the following euristics table. The row of a table correspond to the property of s_i, while the column describes the way to obtain t_i from s_i. When a source group is fixed, the weights of different hypotheses are taken from the same row, therefore the weights do not need to sum to 1 since the normalizing coefficient after calculating the logarithm yields a constant summand, which is the same form all candidate word groups. The weights were obtained empirically from the development set by calculating the frequencies of different typo-correction transformations.

Table 2. Weights of different word transformations

	$s_i = t_i$	**Levenshtein**	**Phonetic code**	**2 words from 1**	**2 words from 1**
no capitals, dictionary word	1	0.005	0.0005	0.001	0.005
initial capital, dictionary word	1	0.001	0.0001	0.0001	0.0001
all capitals, dictionary word	1	0.001	0.0001	0.0001	0.0001
no capitals, dictionary word	0.15	0.6	0.09	0.15	0.01
initial capital, dictionary word	1	0.05	0.01	0.005	0.005
all capitals, dictionary word	1	0.1	0.01	0.01	0.01

3.3. Language model and generation of candidate sentences

Since basic error model score cannot distinguish between different dictionary words on the same Levenshtein distance, we also took into account the language model to obtain the baseline candidate score $C_{lm}+C_{err}$. Provided k is the order of the language model, the language model score of the candidate sentence $t_1, \ldots, t_m$ equals

$$\begin{aligned} C_{lm} &= -\log p(t_1, \ldots, t_m) = -\log(p(t_1)\, p(t_2|t_1) \ldots p(t_k|t_1 \ldots t_{k-1}) \ldots p(t_m|\, t_{m-k+1} \ldots t_{m-1})) \\ &= -(\log p(t_1) + \log p(t_2|t_1) + \cdots + log\, p(t_m|t_{m-k+1} \ldots t_{m-1}) \\ &= \sum_{i=1}^{i=m} -\log p(t_i|t_{i-k+1} \ldots t_{i-1}) \end{aligned}$$

The logarithmic cost in the language model appears to be additive as well as the error model cost. It permits us to apply beam search for pruning partial hypotheses space. Agenda consists of n+1 hypotheses lists, where i-th list store partial hypotheses after processing i words of the sentence and n is the length of the source sentence. Initial agenda item consists of empty hypotheses with initial cost 0. On i-th step we generate all the candidates for the word s_i to expand the hypotheses on the step i–1, as well as the candidates for the group (s_{i-1}, s_i) to expand the hypotheses from the (i–2)-th item of the agenda. For every partial hypothesis we store its current score and the state of the language model (roughly speaking, last (k–1) words). Using this information, we are able to recalculate the score and the state for the expanded hypotheses. We arrange the list items by the states of language models, storing all the partial hypotheses with the same state together. To prune the hypotheses space we preserve only such hypotheses whose score is not greater than the score of the best hypothesis times some constant α.

3.4. Learning of the reranking model

After the previous step we have a list of candidates together with their baseline scores. Now our task is to rerank these candidates using algorithms of machine learning. For this goal we use a linear classifier and determine the best correction using $\hat{c}$the rule

$$\hat{c} = \ argmax_{c\in C} \sum_{i} w_i f_i(c) + w_0$$

where C is the set of candidate sentences and f_i are features which we specify below. To learn the weights of the classifier we observe that maximum does not depend on the additive term w_0 therefore only the linear coefficients w_i should be learnt. In machine translation literature the usual approach is to learn these coefficients from the ranking of train hypotheses, however, in our disposal are only the corrections for the training sentences. However, it is sufficient to learn the weights: a good decision function should rank the correct hypothesis higher than the incorrect ones, therefore we have $\sum w_i f_i(\hat{c}) \geq \sum w_i f_i(c)$ for any other hypothesis c. Equivalently, we have $\sum w_i (f_i(\hat{c}) - f_i(c)) \geq 0$. Then our task is to find a linear classifier such that all the vectors of the form $[f_1(\hat{c}) - f_1(c), \ldots, f_m(\hat{c}) - f_m(c)\,]$ belong to the positive class and the opposite vectors—to negative. Then our problem is reformulated as usual linear classification problem and can be solved by any of standard algorithms, such as SVM or logistic regression.

In our experiments we used the following list of features (Table 3). When a feature is defined for a single word (say, its capitalization), it means that we sum its values for all the words in the. For example, the unit feature for a single word yields the number of words for the whole sentence.

Table 3. Features used in classification

Feature name	Description
F1	Sentence length in words
F2	Error score C_{err}
F3	Language model score C_{lm}
F4	Number of corrected words
F5	Number of OOV words
F6	Number of corrections in OOV words
F7	Number of corrections in dictionary words
F8	Number of corrections in capitalized words
F9	Number of corrections on edit distance 1
F10	Number of corrections by phonetic similarity
F11	Number of corrections by word lists
F12	Number of 1 → 2 corrections (space insertions)
F13	Number of 2 → 1 corrections (space deletions)
F14	Number of OOV words having dictionary partitions
F15	Morphological model score
F16	Weighted edit distance score
F17, F18	Semantic model score
F19, F20	Prepositional model score

The calculation of features F1–F14 is straightforward: we just memorize their values for single words in the candidate generation phase and sum these values to obtain the aggregate score. Morphological model score is calculated just as usual language model score, except the n-gram model is built on POS tags instead of words. To learn the weights in the edit distance we use the algorithm of [Brill, Moore, 2000]: we align each word in the development set with its correction and extract all the groups of up to 3 alignment tokens. The only refinement we made is that a token containing a space symbol on either of its sides is not joined to any longer group.

The features F17–F20 were not used in the system we submitted for evaluation since they did not improve performance but we describe them for future research. We tried to use cooccurence information in order to grasp semantic relations. For example nothing but semantics can force the system to prefer the correction (2) "*мне снится, что мы в ссоре и ты на меня ругаешься и сердишься*" instead of (3) "*не снится, что мы в море и ты на меня ругаешься и сердишься*" for the source sentence *мне снится, что мы в соре и ты на меня ругаешься и сердишься (*note that the source word is also in the dictionary). To calculate the semantic score we collected a frequency list for dictionary lemmata from a supplementary corpora (by frequency we mean the number of sentences containing the lemma). Then we remove the words occurring more than in 1% of sentences (they are noninformative stopwords). We retain 10,000 most frequent lemmata after this removal and for every such lemma collect the list of lemmata cooccurring with it more than a limited number (say, 10) of times. So, for every word from the list we obtain its potential collocations. Then to calculate the semantic score of the sentence we take as features both the number of words from the list of lemmata occurring in the sentence and the number of collocation pairs between these words.

The aim of the prepositional score is to determine the case of a noun knowing a preposition before it. It is often useful because most of the case flections are on the distance of one edit from each other and often simultaneously appear in the candidate list. When the noun immediately follows the preposition it can be captured by a language model, however, often there are intermediate adjectives or dependent noun phrases between the preposition and the noun. To measure this characteristic we collect the total number of prepositions in the sentence, as well as the number of prepositions which do not have nouns or pronouns of the corresponding case to its right. In future research we plan to use several analogous features, characterizing sentence morphology, such as number of coordinated adjective-noun pairs, subject-verb pairs (using gender and number agreement), as well as the total number of nominative case words and finite verbs in the sentence. However, these features are too noisy when collected from a corpus without morphological disambiguation and we do not have access to disambiguated corpora of sufficient size. Since straightforward addition of such features did not improve performance and even led to slight degradation, we decided not to use them. We rejected from application of morphological and syntactic parsers since there quality on social media texts is moderate especially when these texts contain typos. Therefore the exact role of morphology and semantics in Russian spelling correction is left for future research.

4. Evaluating the system

We tested our system in SpellRuEval-2016 competition of spelling correctors for Russian social media. The development set of the competition consisted of 2,000 sentences from Russian social media texts together with their corrections. The test set included 100,000 sentences only 2,000 of which were used for testing. We used the development set to tune the parameters of baseline error model (see previous section) used in candidate selection as well as to tune the weighted edit distance. To avoid zero probabilities in Brill-Moore method we added 0.1 to the counts of every symbol-to-symbol, symbol-to-space, symbol-to-nothing and nothing-to-symbol corrections, as well as to the counts of each transposition of symbols. Since phonetic similarity corrections such as *ться* → *цца* have a high cost in Levenshtein model which leads to noise and outliers in training data, we bound the obtained weighted distance by a fixed number from above. To train the language model we used a supplementary corpus of 5,000,000 sentences (50,000,000 words) obtained from a sample of GICR. Since these sample contained lemmata and morpho-tags (though only POS tags may be considered as reliable), we also used it to train morphological and semantic models. Our algorithm was implemented in Python language, we used the KenLM toolkit [Heafield et al., 2013] to train the language model and the realization of logistic regression from scikit-learn [Pedregosa et al., 2011] as a linear classifier.

We report the results both for development and test sets. In the development phase we used one half of the set for training and another one for testing. The baseline model in the table before is the model used before the training phase, in the weighted baseline model we train a linear classifier on three features: the number of words, the error model score and the language model score. The Levenshtein model adds as a feature the weighted edit distance, the competition model also uses features F4–F14 from Table 4 and the morphological model also uses the score of the POS-tag n-grams model.

Table 4. Spellchecking quality on development set

Model	Precision	Recall	F1-measure	Sentence accuracy
Baseline	71.68	78.01	74.71	70.70
Weighted baseline	82.83	77.89	80.28	79.40
Levenshtein	87.69	79.15	83.20	81.90
Competition	88.43	81.44	84.79	83.20
Competition+Morpho	88.15	81.79	84.85	83.00

Table 5. Spellchecking quality on test set

Model	Precision	Recall	F1-measure	Sentence accuracy
Baseline	63.11	67.26	65.12	60.06
Weighted baseline	75.55	64.27	69.46	66.93
Levenshtein	80.58	65.94	72.53	68.63
Competition	81.98	69.25	75.07	70.32
Competition+Morpho	81.12	68.98	74.56	70.22

Our system won the first place among 7 participants by all the measures: precision, recall, F-measure and accuracy (percentage of correctly recovered sentences). Moreover, already the baseline model is on the par with the system on the second place. We observe that learning weights of different components of baseline model indeed improves its performance consistently, as well as replacing standard edit distance by its weighted version in the error model. Using additional features also improves performance quality by several percents. However, enriching the model with morphological features does not affect performance on the development set and leads to slight degradation on the test set. Note that all the systems except the baseline have higher precision than recall.

5. Results and discussion

Analyzing the mistakes, we have found two main sources of them: the first are space/hyphen errors and the second—real-word errors. For example, in all the 7 cases when the word "еслиб" should be corrected to "*если б*" (for example, in (4) "*страшно представить еслиб с ней что-то случилось*"), it was erroneously replaced by "*если*". Note that this typo is indeed rather difficult: the system suggestion is also a grammatically correct sentence so language model cannot resolve this ambiguity. Moreover, every word sequence that can follow "*если б*", can potentially follow *если* as well. There only part of our system which can capture such cases is the weighted edit distance model, but its influence is overweighed by other factors.

The second source of errors are real-word errors. In many cases they are in fact grammatical errors like (5) "*у этой девченки одни плюсы и не одного минуса*". Sometimes the system cannot select a correct word between two members of a confusion set such as "*формации/фармации*" in (6) "*если говорить точно то эти две фармации исторически противостоящие есть свойства одного*". Though the trigram *и ни одного* is more frequent than its counterpart *и не одного*, the cost of correction in a dictionary word *не* appears to be too high. Real-word errors of the second type could be potentially resolved using cooccurence statistics ("*формации*" is more likely to appear together with "*исторически*" than the other variant), probably, using larger corpus to train the language model or more powerful semantic representation like Word2Vec could help in this case. A minority of errors is also due to incomplete list of informal variations like сення/сегодня or using wrong wordform of a correct lexeme like in (7) "*мне было очень страшно казалось что по дороге нам встретиться или тигр или егеря или бандиты*". We plan to deal with grammar errors and real-word errors in a separate study. It is not an obvious question, whether they can be resolved without any handcoding of grammar and morphology rules.

Table 6. Final quality attained

scientific group, SpellRuEval-2016	Precision	Recall	F-measure	Accuracy	Place
GICR corpora, MSU	81.98	69.25	75.07	70.32	1 of 7

There is still a large room to improve our current model. First of all, results of [Schaback, 2007] and [Flor, 2012] demonstrate that proper usage of morphology and semantics consistently ameliorates performance, which is not the case for our system. It means that POS tags alone do not carry enough information for reliable disambiguation and more subtle morphological categories should be taken into account. As we have already said, this hypothesis should be tested on high-quality morphologically annotated corpus of sufficiently large size. Our current model of semantics representation is one of the simplest ones, therefore only usage of more fine-grained one could resolve the question, whether semantic and morphological information could be helpful for Russian social media.

6. Conclusion

We have developed a system for automatic spelling correction for Russian social media texts. We have tested it in the competition of spellcheckers SpellRuEval during Dialogue Evaluation-2016, where our system won the first place by all the metrics, reaching the F1-Measure of 75%. We used edit distance together with phonetic similarity to select correction candidates, language model together with error model to score these candidates and linear classification algorithms to rerank them. The features used in the last stage include error model score, weighted Levenshtein distance between the candidate and correction, language model score and several other features like number of corrections in dictionary and non-dictionary words, capitalization, etc. The most straightforward way to improve our system is to use linguistically-oriented features like morphology, cooccurence and collocation scores, grammatical correctness of the sentence and so on. Since our system is rather simple, we hope it could serve as a baseline for future Russian spelling correction systems. We think it could also be useful in similar tasks like grammar correction or normalization of social media texts. The system can also be successfully applied on big data collections from Russian Web, and is likely to become a part of NLP-tools used on GICR—this gives other researchers the advantages of having corpus with more diverse automatic annotation and better POS-tagging and lemmatization.

7. Acknowledgements

Authors of this paper express their sincere thanks to all the students that helped in the annotation of the training set. We also thank Elena Rykunova for her intense assistance in technological support of the web forms and databases for making the golden standard.

References

1. *Bajtin A.* (2008), Search query correction in Yandex [Ispravlenie poiskovykh zaprosov v Yandekse], Russian Internet technologies [Rossijskie Internet-tekhnologii], 2008.
2. *Belikov V., Kopylov N., Piperski A., Selegey V., Sharoff S.,* (2013), Big and diverse is beautiful: A large corpus of Russian to study linguistic variation. Proceedings of Web as Corpus Workshop (WAC-8), Lancaster.
3. *Brill E., Moore R. C.* (2000) An improved error model for noisy channel spelling correction. In ACL '00: Proceedings of the 38th Annual Meeting on Association for Computational Linguistics, p. 286–293. Association for Computational Linguistics
4. *Damerau F. J.* (1964) A technique for computer detection and correction of spelling errors. Communications of the Association for Computing Machinery, Vol. 7, No. 3, pp. 171–176.
5. *Flor M.* (2012) Four types of context for automatic spelling correction //TAL.—Vol. 53.—Vol. 3.—pp. 61–99.
6. *Golding A. R., Roth D.* (1999) A winnow-based approach to context-sensitive spelling correction //Machine learning.—Vol. 34.—№ 1–3.—pp. 107–130.
7. *Heafield K., Pouzyrevsky I., Clark J., Koehn I.* (2013) Scalable Modified Kneser-Ney Language Model Estimation //ACL (2).—p. 690–696.
8. *Huldén, M.* (2009) Fast approximate string matching with finite automata. // Procesamiento del lenguaje natural, Vol. 43, pp. 57–64.
9. *Kukich, K.* (1992) Techniques for automatically correcting words in texts. ACM Computing Surveys 24, pp. 377–439.
10. *Kernighan M. D., Church K. W., and Gale W. A.* (1990) A spelling correction program based on a noisy channel model. In Proceedings of the 13th conference on Computational linguistics, pp. 205–210. Association for Computational Linguistics.
11. *Levenshtein V. A.* (1965) Binary codes capable of correcting deletions, insertions and reversals [Dvoichnye kody s ispravleniem udalenij, vstavok i zamen simvolov], Doklady of the Soviet Academy of Sciences [Doklady Akademij Nauk SSSR], 1965, Vol. 163, No. 4, pp. 845–848.
12. *Loukashevich N. V., Dobrov B. V.* (2006) Ontologies for natural language processing: description of concepts and lexical senses. Computational Linguistics and Intellectual Technologies: Proceedings of the International Conference "Dialog 2006"
13. *Mark D. Kernighan, Kenneth W. Church, and William A. Gale.* (1990) A spelling correction program based on a noisy channel model. In Proceedings of the 13th conference on Computational linguistics, pp. 205–210. Association for Computational Linguistics.
14. *Oflazer K.* (1996) Error-tolerant finite-state recognition with applications to morphological analysis and spelling correction //Computational Linguistics.—Vol. 22.—Vol. 1.—pp. 73–89.

15. *Panina M. F., Baitin A. V., Galinskaya I. E.* (2013) Context-independent autocorrection of query spelling errors. [Avtomaticheskoe ispravlenie opechatok v poiskovykh zaprosakh bez ucheta konteksta], Computational Linguistics and Intellectual Technologies: Proceedings of the International Conference "Dialog 2013" [Komp'yuternaya Lingvistika i Intellektual'nye Tekhnologii: Trudy Mezhdunarodnoy Konferentsii "Dialog 2013"], Bekasovo, pp. 556–568.
16. *Pedler J., Mitton R.* (2010). A large list of confusion sets for spellchecking assessed against a corpus of real-word errors //Proceedings of the Seventh Conference on International Language Resources and Evaluation (LREC'10).
17. *Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E.* (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12, 2825–2830.
18. *Philips L.* (2000) The double metaphone search algorithm // C/C++ users journal.—Vol. 18.—№ 6.—p. 38–43.
19. *Popescu O., Phuoc An Vo N.* (2014) Fast and Accurate Misspelling Correction in Large Corpora. Proceedings of EMNLP 2014: Conference on Empirical Methods in Natural Language Processing. Doha, Qatar.
20. *Ristad E. S., Yianilos P. N.* (1998) Learning string-edit distance //Pattern Analysis and Machine Intelligence, IEEE Transactions on.—Vol. 20.—№ 5.—pp. 522–532.
21. *Rozovskaya A., Roth D.* (2013) Joint learning and inference for grammatical error correction //Urbana.—Vol. 51.—pp. 61–801.
22. *Schaback J., Li F.* (2007) Multi-level feature extraction for spelling correction // IJCAI-2007 Workshop on Analytics for Noisy Unstructured Text Data.—pp. 79–86.
23. *Shavrina T., Sorokin A.* (2015) Modeling Advanced Lemmatization for Russian Language Using TnT-Russian Morphological Parser. Computational Linguistics and Intellectual Technologies: Proceedings of the International Conference "Dialog 2015", RSUH, Moscow
24. *Shen L., Sarkar A., Och F. J.* (2004) Discriminative Reranking for Machine Translation //HLT-NAACL.—p. 177–184.
25. *Toutanova K., Moore R. C.* (2002) Pronunciation modeling for improved spelling correction //Proceedings of the 40th Annual Meeting on Association for Computational Linguistics.—Association for Computational Linguistics—pp. 144–151.
26. *Whitelaw C., Hutchinson B., Chung G. Y., Ellis G.* (2009) Using the web for language independent spellchecking and autocorrection. // Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2, pp. 890–899. Association for Computational Linguistics.
27. *Zhang, Y., He, P., Xiang, W., & Li, M.* (2006) Discriminative reranking for spelling correction. // Proceedings of the 20th Pacific Asia Conference on Language, Information and Computation—pp. 64–71.