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

Moscow, June 1–4, 2016

A COMPLEX APPROACH TO SPELLCHECKING AND AUTOCORRECTION FOR RUSSIAN

Dereza O. V. (oksana.dereza@gmail.com),
Kayutenko D. A. (kayutenko@mail.ru),
Marakasova A. A. (anya.tiva@gmail.com),
Fenogenova A. S. (alenush93@gmail.com)

National Research University Higher School of Economics,
Moscow, Russia

This study discusses a number of methods that can be used jointly for error detection and correction, namely blacklists and pre-compiled dictionaries, a word2vec model, an N-gram language model and a tripartite error model. Our system consists of two standalone modules, an error detection confidence classifier, built with the help of supervised machine learning methods, and a corrector that processes words flagged as misspellings by the classifier. The error detection classifier uses word2vec filtered vector scores as one of the features. Apart from that, to achieve higher accuracy while having little training data, we use a hybrid error model that combines three approaches: the traditional channel model that uses single letter edits, the model introduced by Brill and Moore, and an extended version of the channel model that uses wider context edits. Combining these tools and methods we achieved rather promising results: our system effectively handles both known and unknown words, including difficult cases such as slang.

Keywords: error detection, spelling correction, web as a corpus, noisy channel model

КОМПЛЕКСНЫЙ ПОДХОД К АВТОМАТИЧЕСКОМУ ИСПРАВЛЕНИЮ ОПЕЧАТОК ДЛЯ РУССКОГО ЯЗЫКА

Дереза О. В. (oksana.dereza@gmail.com),
Каютенко Д. А. (kayutenko@mail.ru),
Маракасова А. А. (anya.tiva@gmail.com),
Феногенова А. С. (alenush93@gmail.com)

НИУ ВШЭ, Москва, Россия

В работе описывается комплексный набор методов для обнаружения и исправления ошибок и опечаток в текстах на русском языке. В подходе используются блэклисты, словари, модель word2vec, n-граммная языковая модель и трехчастная модель ошибок. Наша система состоит из двух автономных модулей: классификатор, который обнаруживет потенциальные ошибки, и корректор, который обрабатывает слова, помеченные классификатором. В классификаторе ошибок в качестве признака для обучения использовались результаты векторного представления слов из модели word2vec, обученной на текстах, содержащих как правильные написания, так и опечатки. Мы использовали гибридную модель ошибок, которая объединяет три подхода: традиционную модель зашумленного канала; модель, представленную Э. Бриллом и Р. Муром, и альтернативную версию модели зашумленного канала с расширенным контекстом. Совместив эти инструменты и методы, мы достигли обнадеживающих результатов: наша система эффективно справляется даже с незнакомыми словами, включая сленг, жаргонизмы и обсценную лексику.

Ключевые слова: обнаружение ошибок, исправление опечаток, интернет как корпус, модель зашумленного канала

1. Introduction

With tens and hundreds of new words appearing every day on the Web, spellchecking and autocorrection become issues of ever-growing importance for both end users and NLP systems developers. However, the spelling correction technologies for languages with rich morphology such as Russian are still poorly covered in academic literature. Slang words, neologisms and expletives that developers have to face while processing texts also need more attention at the stage of research. This study presents a methodology of spelling error detection and correction in user-generated texts from the Web.

Our research started from developing a spelling correction system for SpellRuEval-2016 evaluation task. The approach was inspired by the paper 'Using the Web for Language Independent Spellchecking and Autocorrection' [Whitelaw et al., 2009]

where the main idea is to use the Web as a representative data source that can help us learn about typical misspellings, their distribution and word usage (thus, avoiding a costly process of annotating the training data manually). First, our goal was to reimplement the algorithm on Russian data; then the focus moved on how to effectively use limited computational capacity to build a spelling corrector with a high performance.

2. Related works

Most of the state-of-the-art approaches assume two stages. First, the automatic error detection task is performed using dictionaries as a list of correct spellings that help detect a target word. Some systems also incorporate phonetic algorithms (Soundex, Metaphone) and information about keyboard distance. Second, at the stage of error correction, various techniques of calculating similarity (enhanced Damerau-Levenshtein edit distance [Si Lhoussain et al., 2015], Finite-State-Automata [Hassan et al., 2008], their combination [Schulz and Mihov, 2002]) are used to define candidates corrections. Finally, a language model is used to choose the best correction in the given context.

There are various context-sensitive methods: Markov models (mainly trigram models), transformation-based learning, differential grammars, confusion sets, a Bayesian hybrid method, methods based on collocations [Church and Gale 1991, Mayers et al., 1991, Mangu and Brill, 1997, Golding and Roth, 1999, Verberne 2002, Fossati and Di Eugenio 2007]. In [Ringlstetter et al., 2007] a novel approach to compile a language model is introduced. Their main idea is to compute domain dependent web corpora and bigram models for the given input text. It was shown that domain dependent bigram models reflect the language of the input text much better than bigram models obtained from static standard corpora (BNC, Brown). Hirst and Budanitsky present a method aimed at tackling real world errors using insights from lexical semantics. Error detection and correction is performed by identifying tokens that are semantically unrelated to their context and are spelling variations of words that would be related to the context [Hirst and Budanitsky, 2005]. Although context-sensitive spelling correction allows to handle not only typos and homophone errors, but also grammatical errors, grammatical error correction is still quite a challenging task for modern systems.

All the above described techniques have a significant shortcoming since they require misspelled/correct word pairs for training. To overcome this issue a number of unsupervised methods were introduced such as the one combining anomalous pattern initialization and partition around medoids [de Amorin and Zampieri, 2013]. Several approaches of noisy channel paradigm use extremely large training corpora for error and language models, and, thus, achieve very good results. These methods are relatively language independent as no manually annotated data is required [Whitelaw et al., 2009, Richter et al., 2012]. Distributional similarity based models have also proved to outperform traditional models in the Web query spelling correction task [Li et al., 2006], but as far as we know they were not used in text spelling correction task.

There has not been any thorough research of error detection and correction problem for Russian language. The only exception is a morphological dictionary driven spelling correction algorithm [Gelbukh 1993]. Other approaches focus on specific cases such as spell checking of geographic names, web queries, spell checking used for statistical language model refinement or database development [Gnilovskaya 2004, Andreev et al., 2007, Baytin 2008, Karpenko and Protasov 2011, Panina et. al., 2013, Shavrina and Sorokin 2015].

3. Our approach

Initially, our aim was to build an unsupervised spellchecking and autocorrection system, but we had to reject the idea because our resources appeared to be insufficient to crawl a necessary number (up to 3.7×10^8) of public Web pages. This very amount of data, as stated in [Whitelaw et al., 2009], ensures a high performance in spellchecking and autocorrection tasks. We have seen for ourselves that unsupervised algorithms work significantly worse with smaller amount of data. To some extent it might be caused by the complexity of Russian morphology as compared to English. In order to get the same accuracy for Russian as achieved for English we might need an even larger corpus of Web pages.

Eventually, we have developed a hybrid system that performs error detection and autocorrection tasks without a training corpus of typos, orthographic and grammatical mistakes. The system uses dictionaries, hand-crafted rules, a word2vec model, a confidence classifier, a noisy channel model, and the Brill and Moore algorithm.

4. Data

As it was mentioned above, we use a confidence classifier to define which words are to be corrected and which are not. To train a word2vec model which is a part of the classifier, a noisy corpus that contains both well spelled and misspelled words is required. We use a corpus of blog posts of 13.8 million tokens collected for [Skorinkin et al., 2013].

However, it appeared to be not exactly what we expected: the portion of misspelled words in the corpus is rather small. Thus, it was crucial for us to enhance the training data with texts containing a significant portion of typos and other kinds of misspellings. Such texts were drawn from two sources. The first one is the Russian Learner Corpus[1] that comprises texts produced by two categories of non-standard speakers of Russian: learners of Russian as a Foreign language and speakers of Heritage Russian. The corpus size is 686,868 tokens. The other one is a set of blog post extracts and search queries (241,108 tokens in size) with misspellings almost in every sentence, which was collected for the General Internet Corpus of Russian [Belikov et al., 2013].

[1] http://www.web-corpora.net/RLC/

When mining the Web for various language phenomena, the most challenging thing is to achieve the appropriate balance in their distribution. We realize that the training corpus we have for now is still not perfect, and there is a lot to improve, especially with regard that the training data and its size have a great impact on the classifier performance, which, in turn, determines the recall of our system.

Handling neologisms, proper names, swear and slang words seems to be a timeless problem for computational linguistics. Since we had a relatively small training corpus, we could not expect all these words to appear in it. SpellRuEval-2016 shared task was aimed precisely at spell checking of texts from the Web. Given all that, we decided to use pre-compiled dictionaries at the postprocessing stage of error detection module. Firstly, we derived lists of non-dictionary words from public Web pages. Secondly, for every entry in a list (apart from the swear word list) we automatically generated all grammatical forms using Pymorphy2, an open-source morphological analyzer for Russian[2]. At the postprocessing stage, for every input token it is checked if it occurs in any of the dictionaries, and if so, the token is eliminated from the candidates for correction.

To ensure that our classifier is not mistaken by flagging well spelled dictionary words as misspelled, we perform the same strategy as the one for non-dictionary words. In this case a dictionary comprises lemmas from Russian dictionaries and all their forms that are automatically generated by Pymorphy2. There is a huge drawback here because we might wrongly eliminate a misspelled word if it is identical to some well spelled word from a dictionary. All the dictionaries described above are also employed in the error correction module. Finally, for training a language model we use a subcorpus of the Russian National Corpus[3] which consists of newspaper articles (22.2 million tokens) of 'RIA', 'RBC', 'KP' and 'Sovsport' from 2011 to 2014.

5. Error detection

Our system consists of two standalone modules, an error detection module and a corrector that processes words flagged as misspellings by the detector (Figure 1). Error detection is an essential part of the system as the output of this module defines whether the word will be corrected or not.

The major challenge is to define which words have to be corrected. Do we want to correct slang and swear words? Is an intentional misspelling actually a misspelling?

Web texts are quite specific: they usually contain a lot of proper names, etc. that do not have a fixed status in the dictionary. For the most part, such words do not incorporate misspellings and therefore should not be flagged by the classifier; however, they are often treated as misspellings for the simple reason that they do not occur in standard language corpora used for training the classifier.

---

[2] https://pymorphy2.readthedocs.org/en/latest/

[3] http://ruscorpora.ru/search-paper.html

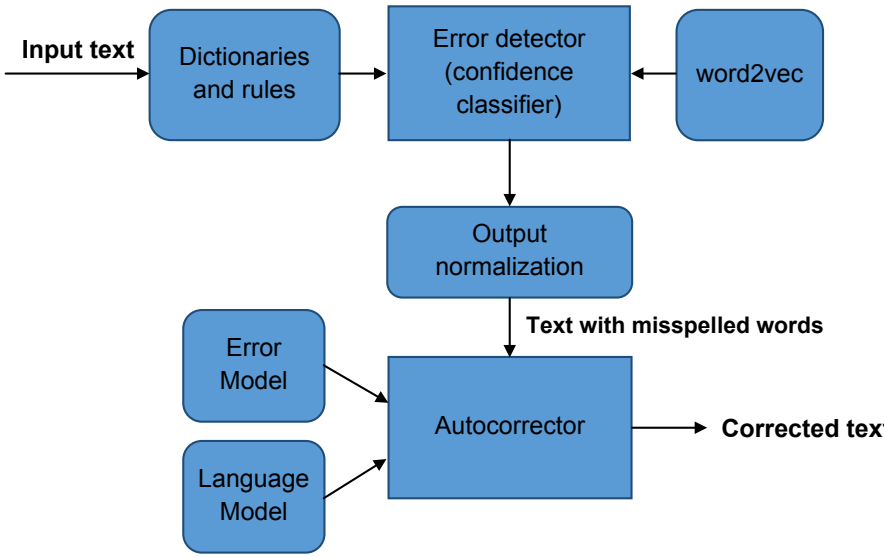

Figure 1

Given that, we exclude certain categories of words from the list of possible misspellings at the preprocessing stage. Tokens are being verified to belong to swear words, slang, or proper names; the ones that occur in any of the corresponding dictionaries will not be processed by the classifier. Furthermore, we derived a list of most frequent and highly probable misspellings that are tricky to correct from the training dataset provided by the organizers of the shared task. The list comprises cases where the edit distance between the misspelling and its correction is extremely large, for instance, *щас — сейчас, грит — говорит*. We exclude the members of this list from the classifier input and apply special handwritten rules to fix them.

Our confidence classifier requires labelled data, namely pairs of misspelled and well spelled tokens. We used 2000 sentences of the aforementioned training dataset to tune our classifier and to determine threshold values and weights. The classifier's task is to predict whether a token is of type 0 (well spelled) or 1 (misspelled). Our algorithm uses logistic regression to obtain not only the class itself, but also its probability. Such a method allows us to configure the thresholds for classes. As we have already eliminated the set of specific words from possible misspellings, the class weights of logistic regression are fitted in accordance with the assumption that class 1 (misspelled) has a very high coefficient. In other words, it is better to catch an extra token than to miss a real misspelling.

Feature sets for the classifier encode the following information:

1. the length of token
2. the amount of left and right context
3. repeated letters
4. belonging to the blacklist
5. word2vec filtered vector scores

The left and right context here means the number of words in the sentence without the input word. The occurrence of repeated letters is a binary feature; if there are more than 2 repeated vowels or 3 repeated consonants in a token, it gets label 1. Certain categories of tokens are blacklisted, so they have to be predicted as non-misspelled. These are numbers, punctuation, latin symbols, and single-character tokens like prepositions and conjunctions. Belonging to the blacklist is also a binary feature.

Word2vec[4] technology is a novel model architecture based on different types of neural networks which computes vector representations of words from a big set of data. A word2vec model is trained to reconstruct linguistic contexts: the network gets a word and must guess the closest words that occurred in adjacent positions in an input text. This module detects similar words and indicates the given word's relation to other ones. In our case, the hypothesis is that a misspelled word and a corresponding well spelled word occur in the same contexts, so the similarity between them in vector representation will be large. To achieve such an effect, a big training dataset of texts having a considerable proportion of misspelled words is required. We trained word2vec on a specially designed corpus described above. For every word flagged as a misspelling, we have to maximize the probability that we get the vector score for the appropriate well spelled form, and not for its synonym or a semantically related word. To filter the output, we choose the top 30 candidates for each token from the model, then select words with vector scores more than 0.4, and check if the sequences' similarity computed with Damerau-Levenshtein algorithm is more than 0.7. The resultant vector score is a feature for the classifier; words that were filtered out get a zero score.

The output of the classifier is then normalized in two stages: 1) the words classified as misspellings are verified in the dictionary of correct words; 2) the words, once classified as misspelled, but missed in another context and flagged as well spelled, are marked as misspelled.

6. Correction module

The second part of our system is the correction module. The approach we use here is the traditional noisy channel model [Shannon 1948], in which the task of error correction is formulated as follows: given a misspelling x we want to find a correction w by taking all words in the dictionary and choosing the one that is most likely to have generated the misspelling x. That is of all words in the dictionary we want to find one that maximizes the probability $P(w|x)$:

$$w = \underset{w \in D}{\operatorname{argmax}} P(w|x) = \underset{w \in Candidates(x)}{\operatorname{argmax}} P(x|w)P(w)$$

The first part of the equation is the channel model, also called the error model. The model requires training data to derive information about the probabilities of misspellings. Since there are no corpora of spelling errors available for Russian, we had to use an online dictionary of misspellings, in which an entry consists of some real word of the

[4] https://code.google.com/archive/p/word2vec/

language and its possible misspelling. Though this method allows us to get basic estimations of the probabilities, it may not be sufficient since the dictionaries do not include the frequency of each misspelling. To get the best results with the limited amount of training data we implemented and tested out three approaches to building the error model.

The first model calculates the probability of each candidate correction as the product of probabilities of each edit in the list of edits obtained from the Damerau-Levenshtein minimum edit distance function for the current candidate. To calculate the probabilities we use a confusion matrix containing the number of times every single letter in the alphabet was deleted, inserted and substituted or switched places with some other letter. The counts are derived from the dictionary described above.

As a second error model we use the approach similar to the one described in Brill and Moore [Brill et al., 2000]. For every candidate we try to find the most probable partitions of the typo and the candidate into substrings of letters, such that the product of conditional probabilities for each part of the typo given the corresponding part of the candidate is the highest:

$$P(w|x) = \max_{\substack{R \in Part(w) \\ T \in Part(x) \\ |T|=|R|}} \prod_{i=1}^{|R|} P(T_i \mid R_i)$$

The algorithm requires an extended confusion matrix that contains counts of substitutions of substrings that are up to three letters long. To get these counts we use the method suggested by Brill and Moore and incorporate up to two adjacent letters into the edits.

The third model is a combination of the first two: here we also use wider context substitutions and an extended confusion matrix, but the probability of the candidate is calculated as the product of conditional probabilities of every single letter substitution as well as wider context substitutions.

The hypothesis behind models 2 and 3 is that using wider context substitutions should give better results for the complex morphology of the Russian language. Table 1 shows most probable candidates yielded by each of the error models for the misspelled word.

Table 1. Most probable candidates according to the three error models. Words in bold indicate the correct variant

Misspelled word	Most probable candidate		
	Model 1 (single edits)	**Model 2 (partitions)**	**Model 3 (extended edits)**
скозал	**сказал**	сковал	сковал
надуфшись	надушусь	надушусь	**надувшись**
сподобалось	**сподобилось**	подобалось	**сподобилось**
орубили	врубили	**отрубили**	врубили
беспорядоченный	**беспорядочный**	**беспорядочный**	неупорядоченный
наверна	наверну	навесна	**наверно**
оэтому	**этому**	**поэтому**	этому
потерялсо	потеряло	потеряло	**потерялся**

As we can see, it turns out that there are misspellings that some of the models correct better than the others. Hence, we decided to use a hybrid model unifying all three as a channel model for our correction system.

At the first stage, we suggest candidate corrections by using Damerau-Levenshtein edit distance to find words that have distance from one to three from the misspelled word and take only the ones that are in the dictionary. Before that, we reduce all repeating letters in the misspelled word to one letter in order to correct intentional misspellings like *ооооченьь*. We assume that even if the word originally contained two letters like in *деревянный*, it can easily be corrected back by the corrector. Apart from that, to be able to correct errors when two space- or hyphen-separated words are mistakenly written as one, we generate additional candidates. To do that we look at every possible partition of the misspelled word into two words and in case both words are in dictionary, we add those pairs separated by a whitespace to the list of candidates. For each of such candidates we then generate another candidate by joining the two words with a hyphen and add the resultant word to the candidate list if the resultant word is in the dictionary. Though this approach can be used to easily correct errors like *чтонибудь*, *помоему* and *потомучто* and turn them into *что-нибудь*, *по-моему* and *потому что*, we still cannot use them as the final corrections. The reason is that some partitions can result in real words even though it was not the initial intention of the person who made the mistake, e.g. *оэтому* > *о этому, сумашедший* > *сума шедший, размылилось* > *размыли лось*. This is why these candidates are passed into the error and language models along with other candidates.

As a result we have four lists of candidates for each typo: edit distance 1 candidates, edit distance 2 candidates, edit distance 3 candidates, space- and hyphen-separated candidates.

In every list each model then selects one candidate with the highest probability, thus the final model returns a maximum of 12 most probable candidates of different types as an output. Normally that number is much smaller since not all words can be separated by white spaces or hyphens and result in real words, moreover, for some misspellings there are only dictionary-word candidates of distance 2 and 3.

Then the list of most probable candidates is passed into the language model. Since the probabilities of the candidates are evaluated by different error models, we cannot directly use them for estimating the final probability. Therefore, we have to drop error model probabilities at this stage and assume that all corrections are equally probable, relying on the language model to select the candidate with the highest context probability. We also add the typo itself to the list of candidates so that the language model could select it in case the classifier made a mistake.

One possible way to actually use the probabilities from the error model is to assign weights to the candidates depending on the probability they were given by the the error model. In the list of candidates returned by each of the error models we can assign the first most probable candidate a weight of 1. Weights for other three candidates can be calculated as the result of dividing their respective probabilities by the probability of the most probable candidate. E.g. If the three most probable candidates for the misspelling *пестня* obtained from the first model have the following probabilities: {(*песня*, 0.0564), (*пестик*, 3.1069e-11), (*шестов*, 1.6639e-16)} they will then

be changed into {(*песня*, 1), (*пестик*, 5.5087e-10), (*шестов*, 2.9502e-15)}. We do that for other models' candidates and then pass the resulting list into the language model, which will select one most probable candidate.

Our trigram language model trained on the 22.2 million corpus of newspapers uses "Stupid Backoff" smoothing [Brants et al., 2007], which consists in going down to lower order N-grams if the higher order N-gram is unknown to the model. For example, if we would like to estimate $P(w_3|w_1w_2)$, but the count of $w_1w_2w_3$ trigram is equal to zero, the model will look for w_2w_3 bigram and return $P(w_3|w_2)$ multiplied by a discount coefficient. If the bigram does not exist either, the model will use unigram probability of w_3 discounted two times.

7. Evaluation

The results of the competition were evaluated with the use of standard precision, recall and f1 measure metrics. With the initial set of features for the classifier our system had the following results on the test set:

Precision	Recall	F1
74.56	28.16	40.88

Thus, we ranked second by precision score and fifth by accuracy, recall and F-measure in the SpellRuEval-2016 evaluation task.

As soon as we got the first results, we realised that such a low recall explained by the unequal distribution of errors in the test set. The majority of sentences contained no misspellings at all, and others had an error almost in every word (*Пацкажыте, люде добрыя—как с ниме барoцца?*), while our classifier was tuned to detect misspellings equally distributed in the text. Removing the "the amount of left and right context" feature from the classifier led to a much larger number of words being selected as misspellings, which resulted in higher recall and lower precision values.

Precision	Recall	F1
60.67	40.24	48.39

8. Discussion

In this section we discuss problems we had to face in our work and various linguistic issues that cannot be ignored by anyone willing to build a modern spelling error detection and correction system.

First of all, the training dataset that we used for tuning our classifier was not comparable with the test dataset. The test set contained much more examples of non-standard language than the training data; moreover, misspelled words were not equally distributed in these two datasets. Indeed, the number and the nature

of spelling errors in different text types are too diverse. They are determined by sociolinguistic factors which are hard to predict, so machine learning techniques alone are powerless to handle them all unless they are given a large representative training corpus. Thus, it might be better to use a specific development set for the classifier to tune the weights and thresholds.

Secondly, there is an interesting linguistic question of what is the correct spelling of a non-dictionary word. For example, shall we change slang words like *инет* and *фотка* to *интернет* and *фотография* respectively? What shall we do with different spellings of a swear word? Shall we standardize it and, in this case, which variant shall we choose if its literary form cannot be found even in modern dictionaries?

Although our system shows rather promising results, there are problems that it cannot solve yet, which is the subject of future work. First, due to the current architecture of the classifier, grammatical error detection is beyond our capacity for now. It can be improved, for example, by adding language-model-based features (such as N-gram language model) to the classifier, especially assuming that the correction module is already capable of correcting some grammatical errors. Furthermore, the cases when several words are separated by a whitespace, but should be written without it or with a hyphen instead, cannot be detected properly by our system, as the detector module examines only one input token and not the context. Finally, there are a lot of diminutive forms for proper and common names, neologisms with unusual paradigms, occasional word formation and other specific cases, which pose a fascinating challenge for automatic systems.

9. Conclusion

Detecting and correcting spelling errors in real user-generated texts on the Web is a challenging task for NLP engineers. We believe, that complex approaches combining statistical methods and sensibly chosen rules are ideally suited for building a spell checking system with a high performance. We implemented such a system and run a number of tests on training and test sets for SpellRuEval-2016 evaluation task. Although our system returned respectable results, there is still a lot to improve both in the algorithm and in its linguistic basis.

References

1. *Andreev, M., Berezkin, D., Nechkin A., Semakov, K., Sharov, Yu.* (2007) The method for unsupervized detection and correction of misprints in geographical names for the system of semantic checking and validation of documents. Digital libraries: advanced methods and technologies, digital collections. The 9th Russian national research conference, Pereslavl, pages 111–118.
2. *Baytin., A.* (2008) Search query correction in Yandex [Ispravlenie poiskovykh zaprosov v Yandekse], Russian Internet technologies [Rossijskie Internet-tekhnologii].

3. *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. По материалам Web as Corpus Workshop (WAC-8), pages 24–29.
4. *Brill, E. and Moore, R.* (2000) An improved error model for noisy channel spelling correction. Proceedings of 38th annual meeting of the ACL, pages 286–293.
5. *Church, K. and Gale, W.* (1991) Probability scoring for spelling correction. In Statistics and Computing, volume 1, pages 93–103.
6. *de Amorim, R. C. and Zampieri, M.* (2013) Effective Spell Checking Methods Using Clustering Algorithms. Proceedings of Recent Advances in Natural Language Processing (RANLP). Hissar, Bulgaria. pages 172–178.
7. *Gelbukh, A.* (1993) Morphological dictionary driven spelling correction (in Russian). J. Nauchno-Tehnicheskaya Informaciya (NTI), ISSN 0548-0027, ser. 2, vol. 5, pages 23–30, Moscow, Russia.
8. *Gnilovskayam, L.* (2004) Automatic correction of orthographic errors [Avtomaticheskaya korrektsiya orfograficheskih oshibok] // Culture of the Pontic people [Kul›tura narodov Prichernomor›ya].—2004.—V. 2, № 48.—pages 171–180.
9. *Golding, A. and Roth, D.* (1999) A winnow-based approach to context-sensitive spelling correction. Machine Learning, 34(1–3), pages 107–130.
10. *Hassan, A., Noeman, S., Hassan, H.* (2008) Language Independent Text Correction using Finite State Automata. IJCNLP-08, pages 913–918.
11. *Hirst, G. and Budanitsky, A.* (2005) Correcting real-word spelling errors by restoring lexical cohesion. Natural Language Engineering, 11(1), pages 87–111.
12. *Karpenko, M., Protasov, S.* (2011) Some methods for language model pruning. Computational Linguistics and Intellectual Technologies: Proceedings of the International Conference “Dialog 2011” [Komp’yuternaya Lingvistika i Intellektual’nye Tekhnologii: Trudy Mezhdunarodnoy Konferentsii “Dialog 2011”], Bekasovo, pages 326–327.
13. *Kukich, K.* (1992) Techniques for automatically correcting words in text. ACM Computing Surveys. 24(4), pages 377–439.
14. *Li M., Zhu M., Zhang Y., Zhou M.* (2006) Exploring Distributional Similarity Based Models for Query Spelling Correction . In: ACL’06. pages 1025–1032.
15. *Mangu, L. and Brill, E.* (1997) Automatic rule acquisition for spelling correction. Proceedings of ICML 1997, pages 734–741.
16. *Mayes, E., Damerau, F. and Mercer, R.* (1991) Context based spelling correction. Information processing and management 27(5), pages 517–522.
17. *Panina, M., Baytin, A., Galinskaya, I.* (2013) Context-independent autocorrection of query spelling errors. 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, pages 556–568.
18. *Richter, M., Straňák, P., Rosen, A.* (2012) Korektor—A System for Contextual Spell-checking and Diacritics Completion In Proceedings of the 24th International Conference on Computational Linguistics (Coling 2012), pages 1–12, Mumbai, India.

19. *Ringlstetter, Ch., Hadersbeck, M., Schulz, K. and Mihov, S.* (2007) Text correction using domain dependent bigram models from web crawls. In Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-2007) Workshop on Analytics for Noisy Unstructured Text Data.
20. *Schulz, K. and Mihov, S.* (2002) Fast string correction with levenshtein-automata. International Journal of Document Analysis and Recognition (IJDAR), 5, pages 67–85.
21. *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
22. *Si Lhoussain, A., Hicham, G., Yusfi, A.* (2015) Adaptating the Levenshtein Distance to Contextual Spelling Correction. International Journal of Computer Science and Applications, Technomathematics Research Foundation Vol. 12, No. 1, pages 127–133.
23. *Skorinkin, D., Temchenko, A., Vybornova, A.* (2013) ConCorT: corpus technologies, digital humanities and contemporary knowledge. The Higher School of Economics, Nizhny Novgorod..
24. *Toutanova, K. and Moore, R.* (2002) Pronunciation modeling for improved spelling correction. Proceedings of the 40th annual meeting of ACL, pages 144–151.
25. *Fossati, D. and Di Eugenio, B.* (2007) A mixed trigrams approach for context sensitive spell checking. In CICLing-2007, Eighth International Conference on Intelligent Text Processing and Computational Linguistics. Mexico City, Mexico.
26. *Verberne, S.* (2002) Context-sensitive spell checking based on word trigram probabilities. Master's thesis, University of Nijmegen.
27. *Whitelaw C., Hutchinson B., Y Chung G., Ellis G.* (2009) Using the Web for Language Independent Spellchecking and Autocorrection. EMNLP '09. Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, 2.