Andrew LISHCHYTOVYCH[1], Olexandr SHMATOK[2], Olga VESELSKA[3], Yuriy FINENKO [4]

Opiekun naukowy: Volodymyr PAVLENKO[5]

PRAKTYCZNE ASPEKTY STOSOWANIA SYSTEMÓW SPELLCHECKING W DZIEDZINIE MEDYCZNEJ

Streszczenie: Nowoczesne systemy informatyczne w dziedzinie opieki zdrowotnej współpracują z EMR, darmowymi tekstami i komunikatami dla użytkowników, aby zapewnić pacjentom lepsze usługi zdrowotne. Kluczowym elementem systemu jest właściwa identyfikacja leków, objawów i innych informacji klinicznych, które mogą zawierać literówki lub błędy ortograficzne. Na rynku dostępnych jest wiele sprawdzarek pisowni (systemów spellchecking), ale prawie żaden z nich nie jest w stanie poprawić tekstu medycznego i wymaga znacznych zasobów obliczeniowych. Jednym z najlepszych rozwiązań w zakresie korygowania błędów ortograficznych tekstów medycznych jest CSpell, który zapewnia wysoką jakość i specyfikę poprawek, ale działa wolniej niż wymagają tego nowoczesne aplikacje. Problem ten został rozwiązany poprzez przeniesienie bazującego na Javie CSpell do .Net 5.

Słowa kluczowe: CSpell, aspell, word2vec, .Net, korekta słów

PRACTICAL ASPECTS OF THE SPELLCHECKING SYSTEMS USAGE IN THE MEDICAL DOMAIN

Summary: Modern healthcare domain IT systems work with the EMR, free texts and user messages to provide better healthcare services for consumers. Key element of the system is proper identification of the drugs, symptoms and other clinical information that may consist typos or orthographic errors. There are many spellcheckers available on the market, but almost none of them are able to correct medical text and require significant computational resources. One of the best solutions for the medical texts spelling errors correction is CSpell that provides high quality and specificity of the corrections but works slower that required by modern applications. This was overcome by migrating the Java based CSpell to .Net 5.

Keywords: spellchecker, CSpell, aspell, word2vec, .Net, words correction

---

[1] Open International University of Human Development "Ukraine", Kyiv, Ukraine,email: AL@sors.me
[2] Open International University of Human Development "Ukraine", Kyiv, Ukraine,email: sh_al_st@ukr.net
[3] University of Bielsko - Biała, oveselska@ath.bielsko.pl
[4] Open International University of Human Development "Ukraine", Kyiv, Ukraine,email: talaveryuriy@gmail.com
[5] PhD, Open International University of Human Development "Ukraine", Kyiv, Ukraine,email: pavlenko.v@i.ua

1. Introduction

Many modern healthcare solutions process free texts such as medical records, lab results or conversations between physicians and patients. Natural language processing (NLP) subsystems require automatic spelling correction as a base component. For example, MedicalBrain solution provides multiple venues for consumers to communicate health-related information: an online chat inside the mobile application or working with the electronic medical records (EMR) or processing lab results. User messages submitted to such application are usually short, whereas EMRs are much longer and consist of several pages, usually they contain clinical history, diagnosis and prescriptions. Another example is Consumer Health Information (CHI) question answering system that works with the online user messages in real time and processes forms and email questions as well. All such systems require fast spellchecker tool to correct typos, misprints, concatenations and other errors in the text.Typical errors in the user messages and documents are: non-dictionary words, not-intended words, agglutination and splitting, informal language and abbreviations, errors in the brand names of drugs.

2. Research objective representation

Research primary objective was to overview available spellcheckers, select the best one and make it suitable for the real-time medical domain usage patterns.

2.1. Related works

Automated spellchecking got significant development since 1960s. Damerau, Mays and Levenshtein proposed to use edit distance for isolated word error corrections [1, 2]. Other approaches such as word frequency [3,4,5], noisy filtering [6,7] and phonetic algorithms were developed. Nowadays, neural networks and n-gram [4] are used to incorporate context information and correct context-dependent errors.

2.2. Available spellcheckers

There are multiple spellcheckers available for the developers – Ispell/Aspell, Jazzy, JamSpell, SymSpell, CSpell and others: Ispell and its successor Aspell provide with very basic spellchecking and correction features, Jazzy is abandoned many years ago, so the short list [5] is: JamSpell – it is fast and takes context into consideration, but can't handle agglutinations ("whiteblood count" is corrected into "whiteboard count") and is memory consuming; SymSpell – handles long texts and short messages, corrects agglutinations, returns technical information with regards to corrects, but context is not considered; CSpell – the newest Java-based spellchecker, that introduced multiple correction algorithms, respects the context and provides the best in class correction quality, but slow. Based on the features the CSpell spellchecker was chosen and optimized for the production use.

3. CSpell general details

The Ensemble method was chosen by CSpell authors [6] as a strong baseline because it was intended to correct errors in consumer health questions, and it outperformed

publicly available tools by the margin over 30% [1 - 6], although many spelling correction techniques have been developed, no publicly available spelling tools provide corrections needed for consumer health questions. Kilicoglu et al [10] developed an ensemble method (henceforth Ensemble) of combining knowledge sources to correct errors in consumer health questions. Ensemble outperformed publicly available tools by a large margin (over 30%) on a set of consumer health questions. CSpell outperforms Ensemble by 14.03% on error detection and 12.33% on error correction, and with 11.4 times the speed of processing. CSpell authors observed that characteristics of nonword errors, real-word errors, and word boundary infractions are different, and different errors require different correction strategies. Dictionaries, corpora, knowledge sources, and heuristic rules were evaluated and developed individually for each specific type of correction and then integrated together to correct various errors in consumer health questions.CSpell authors applied a novel approach of dual embedding using the Word2vec continuous bag-of-words (CBOW) model [7, 8]. Both input and output matrices of the CBOW were used for dual embedding to calculate context scores. The dual-embedding model shows a noticeable improvement in F_1 score as compared with single embedding [5, 6] for context ranking. Second, a 2-stage ranking system was developed to best utilize knowledge-based orthographic similarity, noisy channel, and context scores together for nonword spelling and split corrections. The result shows an improvement in F_1 score compared with the best single-stage ranking system. Third, a multilayer design of spelling correction was implemented to correct various spelling errors in consumer health questions. Each layer is a stand-alone spelling correction module for a specific type of error. As a result, CSpell achieves a significant improvement in F_1score compared with Ensemble for both spelling error detections and corrections.

The Ensemble method was chosen by CSpell authors [6] as a strong baseline because it was intended to correct errors in consumer health questions, and it outperformed publicly available tools by the margin over 30% [1]. They used both the training set and the test set from the baseline as a training set to develop CSpell and then tested on a newly annotated test set collected from consumer health questions submitted to the NLM customer services. Their training set consists of 471 consumer health questions with 24 837 tokens, 1008 annotation tags, and 774 of 964 instances of nonword or real-word corrections. This training set covers a good variety of lengths and errors for consumer health questions. The word count ranges from 5 to 328, with an average of 52.49 words per question. The number of errors ranges from 0 to 27, with an average of 2.14 errors per question. The distribution of spelling errors is shown in Table 1.

They used both the training set and the test set from the baseline as a training set to develop CSpell and then tested on a newly annotated test set collected from consumer health questions submitted to the NLM customer services. Their training set consists of 471 consumer health questions with 24 837 tokens, 1008 annotation tags, and 774 of 964 instances of nonword or real-word corrections. This training set covers a good variety of lengths and errors for consumer health questions. The word count ranges from 5 to 328, with an average of 52.49 words per question. The number of errors ranges from 0 to 27, with an average of 2.14 errors per question. The distribution of spelling errors is shown in Table 1.

Table 1. Distribution of errors in the training set

Correction needed	Non-words	Real words	Non-dictionary	Multiple	Total
Spelling	348	153	113	n/a	614
Merge	10	38	0	n/a	48
Split	24	10	281	n/a	315
Multiple	n/a	n/a	n/a	31	31
Total	382	201	394	31	1008
Percentage	38%	20%	39%	3%	100%

Figure 1 lists examples of errors corrected by CSpell in the training set. Example 2 shows split errors on "*trichorhinophalangeal*" that require multiple merge corrections on the nonword, "*tricho*". CSpell handles multiple errors through multiple corrections. Example 6 - "*shuntfrom2007. How*" is first corrected to "*shuntfrom 2007. How*" (nondictionary split correction) then to "*shunt from 2007. How*" (nonword split correction). In Example 7, "*anti depressants*" is corrected to "*anti depressants*" (nonword spelling correction), then to "*antidepressants*" (real-word merge correction).

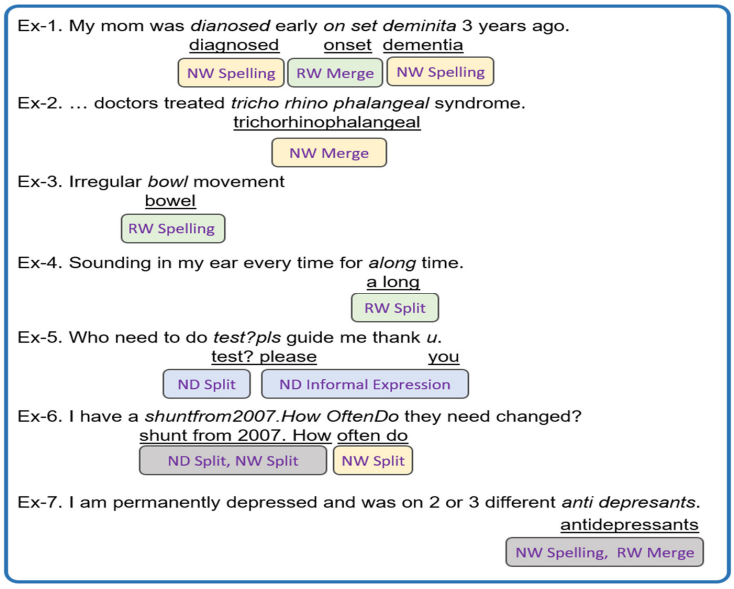

Figure 1. CSpell error corrections

Authors' test set [6] was generated by finding consumer health questions with the highest count of out of vocabulary (OOV) terms. The SPECIALIST Lexicon 2017 release was used as the dictionary to identify OOVs. This test set includes 224 questions, 16 707 tokens, 1946 annotation tags, and 974 of 1178 instances of nonword or real-word corrections. The errors were manually annotated by 2 annotators (A.R.A., S.E.S.) independently. The disagreements were reconciled by the annotators with arbitration by D.D.-F. as needed.

4. CSpell architecture

Different types of errors have different characteristics and require specific strategies for corrections. A multilayer design consisting of models for non–dictionary-based and dictionary-based corrections was implemented in CSpell. It integrates several stand-alone spelling correction models combined in the sequential order as shown in Figure 2. The nondictionary correction model includes handlers and splitters. They were arranged as a chain of intermediate operators to handle HTML/XML tags introduced by the software that consumers use to ask questions, informal expressions, and missing spaces on adjacent punctuation or digits. For example, "*test? pls*" was corrected to "*test? pls*" by a punctuation splitter, then it was corrected to "*test? please*" by the informal expression handler. The dictionary-based correction model includes 4 modules: (1) the detector (to detect errors), (2) the candidate generator (to generate correcting candidates), (3) the ranker (to rank candidates and find the best correction), and (4) the corrector (to replace the detected error with the best correction). The corrector is needed to cope with single-token (spelling and split) and multi-token (merge) corrections.

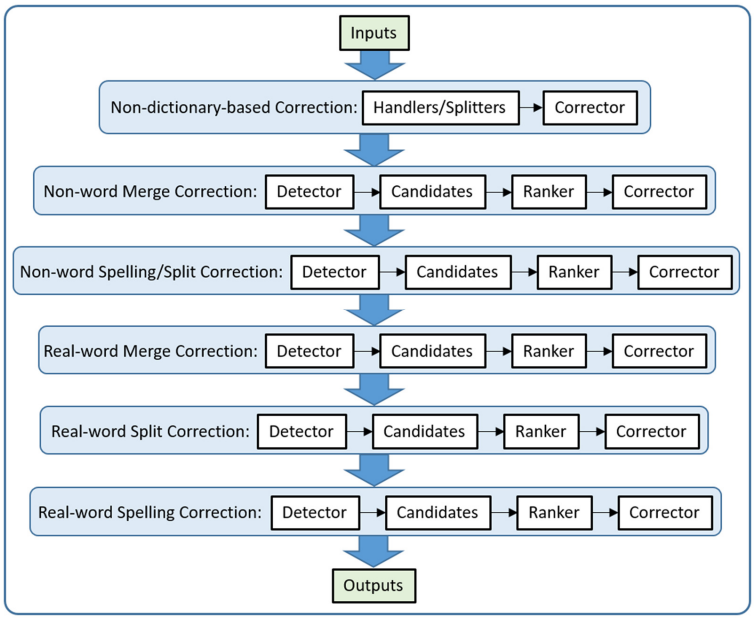

Figure 2. CSpell workflow

The input text is tokenized to words and processed sequentially. A lazy implementation of tokenization on punctuation (delay tokenizing on punctuation until the last moment) was used to avoid unnecessary computation for tokenization and assembly on punctuation.

5. Dictionary-based correction model

5.1. Detectors

The SPECIALIST Lexicon was used as the dictionary because all of its records are manually validated by linguists, and it targets both general English and biomedical terminology. Different word collections, such as numerals, abbreviations or acronyms, proper nouns, single words, multi-words and element words (unigrams in multi-words) were retrieved from Lexicon records. In addition, consumer-related medical terms retrieved from the Unified Medical Language System Metathesaurus were added to the dictionary. Words that are not in the dictionary are detected as nonword errors. Digits, punctuation, URLs, email addresses, and measurements are identified as error exceptions that require no corrections. Abbreviations or acronyms are excluded from the dictionary for the nonword merges. For example, "*dur*" and "*ing*" are considered nonwords and merged to "*during*", which would not have happened had the acronyms been included because "*dur*" stands for drug use review and "*ing*" stands for isotope nephrogram in the Lexicon.

Detection and correction for real-word errors in CSpell is computed on the fly, based on context scores, word frequency scores, and other heuristic rules, as detailed in the section on ranking knowledge sources. No confusion sets or assumptions on the number of real-word errors were used. A real-word error is detected by 6 rules, when the token (1) is in the dictionary, (2) is not an error exception, (3) was not corrected previously in the CSpell pipeline, (4) has a context score, (5) has a word count greater than a threshold, and (6) has length greater than a threshold. The values of threshold in criteria 5 and 6 are configurable in CSpell. Empirical best values of thresholds and other variables described in the paper are provided in the default configuration of CSpell.

5.2. Candidate generators

For real-time spelling correction, Church and Gale's [2-6] reverse minimum edit distance technique was used to generate nonword spelling and split candidates. In CSpell's training set, 67.74%, 91.24%, and 96.37% of errors are within edit distances of 1, 2, and 3, respectively. This result is similar to Flor's [4] reports on error severity. An edit distance of 2 was, therefore, chosen as the edit distance threshold to cover over 91.24% of errors for fast candidate generation. To avoid expensive edit distance computations between a misspelled word and all 0.6 million words in the dictionary,16 only words within 2 edits and in the dictionary were generated as candidates. The maximum length of misspellings and number of splits are configurable. Multi-words, element words, and abbreviations or acronyms were checked to validate split candidates. For example, "*se i ng*" is not a split candidate from "*seing*" because "*se*" and "*ng*" are abbreviations in the Lexicon. Nonword merge candidates include removing spaces and replacing spaces with hyphens among the original token and its adjacent tokens within a specified window (eg, "*non prescription*" has two merge candidates of "*nonprescription*" and "*non-prescription*"). Finally, a merge candidate must be in the dictionary and not an abbreviation or acronym.

A real-word candidate must have a context score greater than an empirically defined threshold. Candidates for real-word spelling corrections have to be at least 3

characters long. Additional orthographic and phonetic rules are used for real-word spelling correction to ensure candidates look and sound like the original token. Heuristic rules based on the total number of short words are used for real-word splits and merges. For example, "*a not her*" is an invalid real-word split candidate of another because the count of short words (words that are less than 4 characters long) is above the default threshold of 2 words, while "*an other*" is a valid real-word split candidate with 1 short word. Other rules such as checking on units, proper nouns, multi-words, and inflectional variants are also used. For example, hu man is an invalid real-word split candidate of human because Hu is a proper noun.

5.3. Ranker

Rankers are used to rank the candidates and find the best candidates for corrections. CSpell authors [6] developed a novel approach to calculate context scores and a 2-stage ranking system to effectively utilize knowledge sources.

5.4. Ranking knowledge sources

CSpell authors enhanced Ensemble's orthographic similarity score by using a weighted sum of edit distance, phonetic similarity, and leading or trailing character overlap similarity scores, with weighting factors of 1.0, 0.7, and 0.8 (empirically determined), respectively. For example, the orthographic similarity score for "*truely*" and "*truly*" is 2.27 (1.0*0.904 + 0.7*1.0 + 0.8*0.83). The edit distance score is 0.904, obtained by deducting the cost of normalized edit distance (1 delete) 0.096 from 1. The phonetic score is 1.0 because both terms have the same phonetic representation as [TRL] in Double Metaphone. Double Metaphone was selected for phonetic representation because it has the best performance compared with Metaphone, Caverphone 2, and Refined Soundex. Leading or trailing character overlap similarity score calculates the overlap of matching characters at the beginning and the end of 2 terms, divided by the length of the longer term, 0.83 = (3 + 2)/6. A consumer health corpus was established by collecting health related articles from 16 consumer-facing National Institutes of Health websites [4] that were used for answering consumer health questions. This corpus includes 17 139 articles, 10 228 699 tokens, and 192 818 unique words. Word frequency score is normalized between 0 and 1. For each word, it is calculated as its frequency in the consumer health corpus divided by the number of occurrences of the most frequent word in this collection. The orthographic similarity score and word frequency score were used as the error model and the language model to calculate the noisy channel score [7, 8, 9, 10]. These techniques were used for isolated-word error ranking.

The Word2vec CBOW model was used for context-dependent ranking. The CBOW is a shallow machine learning neural network model with a single hidden layer (Figure 3). It is used to predict a target word at the output layer from a given context in the input layer. Two matrices, the input matrix (IM) and output matrix (OM), are used to calculate the hidden layer ($[H]_{1*n} = [C]_{1*w} * [IM]_{w*n}$) and target words ($[T]_{1*w} = [H]_{1*n} * [OM]_{w*n}$), respectively, where W is the total number of words in the corpus and N is the dimension of hidden layer. Finally, the softmax function is used to convert the output layer to probabilities (P_w) for updating OM and IM through backpropagation during the training process. The IM, known as the word vector [1,5] is used almost exclusively on its own in all Word2vec applications, whereas the OM is disregarded. CSpell uses both the IM and OM matrices to compute context scores

of the predicted target word with given contexts ($[T]_{1*w} = [C]_{1*w} * [IM]_{w*n} * [OM]_{n*w}$). Namely, treat the hidden layer and target words as the first and second embedding, respectively. The softmax function was not used because backpropagation is not needed (after training) in the application. The above consumer health corpus was used to train the CBOW model to generate IM and OM. CSpell uses modified version of the Word2vec code [5] to generate both the IM and OM using window size of 5 and embedding size of 200. Context scores might be positive, zero, or negative. A zero context score means the target word does not have a word vector.

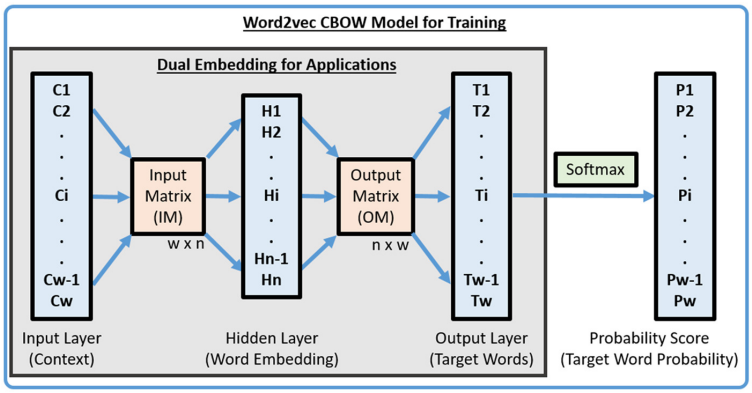

Figure 3. CBOW model

5.5. Utilizing ranking knowledge sources

CSpell authors developed a 2-stage ranking system to efficiently utilize the above knowledge sources for nonword single-token corrections. The nonword spelling and split candidate generator that relies on edit distance measure alone generates irrelevant candidates. Orthographic similarity scores are used to exclude irrelevant candidates. All candidates were ranked by orthographic similarity scores first (stage 1), and those with highest scores are selected for the second-stage ranking. CSpell uses chain comparators to rank the selected candidates by the context score, then the noisy channel score in a sequential order in stage 2. Ranking by orthographic similarity scores in stage 1 was disregarded in stage 2. For example, of 441 candidates to correct "*havy*", 14 candidates with the top orthographic scores (shown in parenthesis) are selected in stage 1: "*heavy*" (2.25), "*have*" (2.20), "*hay*" (2.13), and "*wavy*" (2.13), etc. Lower-scoring candidates, such as "*happy*" (1.92), "*hair*" (1.83), and "*lady*" (1.56), are filtered out. In stage 2, the selected candidates are ranked using context information (eg, "*heavy duty*", "*have diabetes*", "*hay fever*", "*wavy lines*"). A 1-stage ranking system with chain comparators comparing context scores followed by word frequency scores was used for nonword merge corrections.

The best candidate with the highest context score is used for real-word correction only if it had a positive context score and the original token had a negative context score. Additionally, for real-word merges and splits, a confidence factor with a value between 0 and 1 is used when both the best candidate and the original token had positive context scores. The confidence factor multiplied by the best candidate's score must be greater than the original token's score for a real-word correction. In our experiments, confidence factors were 0.6 and 0.01 for real-word merge and split

corrections, respectively. More heuristic rules for using the edit distance score, phonetic similarity, overlap similarity, orthographic similarity, and word frequency scores were implemented for real-word spelling corrections.

5.6. Corrector

The spelling and split correctors replace the original token with the best candidate. The merge corrector modifies the input text by going through all merge operations sequentially. Cases of overlapping candidates were handled by selecting the longest string. For example, implement and implementation are 2 adjacent merges for "*imple ment ation*". "*Implementation*" is used for merge correction because "*implementation*" contains "*implement*".

6. Evaluation and results

On the test set, CSpell outperforms Ensemble by 14.03% and 12.33% to achieve 80.93% and 69.17% on F1 scores for error detection and corrections, respectively. The performance of error detection (80.98% F_1) and error correction (73.38% F_1) on the training set is slightly better than the performance on the test set. The test set is a harder set for spelling correction because it was sampled from questions with the highest OOV rate. The error rate (error corrections / tokens of the test set = 0.07) is much higher than the training set (0.04). Accordingly, both CSpell and Ensemble had worse performance on the test set than on the training set. In addition to evaluating CSpell on all misspellings, the evaluation of its performance was done on only those terms that are important for question understanding. The important terms were identified during manual annotation. The results on the important terms are very close to the overall results.

Table 2. Ensemble vs CSpell results

Method	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**
	Detection: non-words only			**Correction: non-words only**		
Ensemble	76%	76%	76%	62%	61%	62%
CSpell	88%	87%	88%	77%	76%	76%
	Detection: real-word included			**Correction: real-word included**		
Ensemble	82%	56%	67%	70%	48%	57%
CSpell	89%	74%	81%	76%	63%	69%

6.1. CSpell speed and migration to .Net

CSpell was developed using pure Java 8 SDK. Based on the real-world scenarios , the following dataset was used to test the speed: EMR record of 140 sentences including diagnoses, drugs, dosage and treatment. Java based CSpell has corrected all the entries of the dataset in 24.84 seconds, that is 177 ms per record. CSpell source code was migrated to C# / .Net 5. During the migration Apache SoundEx library was replaced by the Phonix package. Original configuration approach was changed to the standard AppSettings pattern. Migration required significant changes and improvements for Java's String.Split method, Java Regex, replacements of RectangularArray, HashSet and HashMaps. The CSpell engine was configured as a singleton service and and access was wrapped by the WebAPI controller.

Table 3. CSpell Java vs .Net speed comparison

CSpell version	New text	New line	Cached responses
CSpell Java	22.145 sec	177 ms	18.35 sec
CSpell.Net	15.76 sec	127 ms	16.88 sec

Migration significantly speed ups CSpell and makes it better solution for the chat bots or other conversation systems.
CSpell.Net utilizes async/await approach to handle multiple requests and was successfully tested under simultaneous corrections by 10 parallel requestors with 10.000 corrections each.

7. Conclusion

CSpell was originally developed as a multilayer spelling correction model for correction of spelling and word boundary infraction errors. A novel approach of dual embedding within the Word2vec CBOW model was proposed for context-dependent corrections. A 2-stage ranking system was developed to best utilize different knowledge sources. The combination of the previously mentioned enhancements with the migration to the modern .Net 5 platform resulted in CSpell.Net with the improved performance for spelling detections and corrections in real time. CSpell.Net could be utilized for the efficient production usage.

REFERENCES

1. DAMERAU F, MAYS E.: A technique for computer detection and correction of spelling errors. Commun ACM 7(1964)3, 171–6.
2. LEVENSHTEIN V.: Binary codes capable of correcting deletions, insertions and reversals. Soviet Phys Doklady (1966)10, 707–10.
3. CROWELL J, ZENG Q, NGO LH et al.: A frequency-based technique to improve the spelling suggestion rank in medical queries. J Am Med Inform Assoc 11(2004)3, 179–85.
4. TURCHIN A, CHU JT, SHUBINA M et al.: Identification of misspelled words without a comprehensive dictionary using prevalence analysis. AMIA Annu Symp Proc 2007, 751–5.
5. MITTON R.: Ordering the suggestions of a spellchecker without using context. Nat Lang Eng 2009; 15 (02): 173–92.
6. WILBUR WJ, KIM W, XIE N.: Spelling correction in the PubMed search engine. Inf Retr Boston 9(2006)5, 543–64.
7. NORVIG P.: Chapter 14: natural language corpus data. In: Segaran T, Hammerbacher J, eds. Beautiful Data. Sebastopol, CA: O'Reilly Media; 2009: 219–42.
8. FLOR M, FUTAGI Y.: On using context for automatic correction of non-word misspellings in student essays. In: Proceedings of the Seventh Workshop on Building Educational Applications Using NLP. Stroudsburg, PA: Association for Computational Linguistics; June 7, 2012: 105–15; Montréal, Canada.