Contextual Spellchecking Based on N-grams

Ivan Srdić
University of Zagreb
Faculty of Electrical Engineering and Computing
Unska 3, Zagreb, Croatia
ivan.srdic@fer.hr

Gordan Gledec
University of Zagreb
Faculty of Electrical Engineering and Computing
Unska 3, Zagreb, Croatia
gordan.gledec@fer.hr

Abstract. *Croatian Academic Spellchecker is an online web-service used for almost 20 years by thousands of users every day. In recent years, the service enabled rudimentary contextual spellchecking, based on pattern matching. In this paper we describe how it is possible to perform n-gram based contextual spellchecking of texts written in Croatian, regardless of the orthographic complexity of the Croatian language. Simple upgrade of the existing implementation was achieved by separating the system into several components. Using a well-known classifier, tweaking the frequency estimator and separating errors into confusion sets resulted in a contextual spellchecking system with a high score of* $F_1 = 0.95$ *on the examined example.*

Keywords. Contextual spellchecking, statistical approach; n-grams

1 Introduction

With the rise of remote communication, there is an increased need for a system that corrects orthographic and grammatical errors. This paper demonstrates an enhancement to the contextual spellchecking system of Haschek, a Croatian online spellchecker (service available at https://ispravi.me/) designed by Šandor Dembitz and described in Dembitz et al. (2011).
Contextual spelling errors, being the most complicated type of errors, have always been difficult to detect and correct. A spelling error in an intended word may result in the wrong real-word; that change will go undetected in a traditional spellchecker (*sljedeći* vs. *slijedeći*, *zahtijeva* vs. *zahtjeva*, etc.).
For both detection and correction of contextual errors, a statistical language model is needed. For every word that is suspected to be an error, a word with higher probability of occurrence in the given context must be chosen. The substitute word can in be any word in the language, making this computationally theoretically impossible for a smaller system.
The remainder of this paper is organized as follows: Chapter 2 describes the theoretical background behind our research; Chapter 3 explains the data sets used – n-grams collected in the more than 20 years of the usage of Croatian spellchecker and describes the system architecture. Chapter 4 explains the results and Chapter 5 gives detail about future research. Final chapter concludes the paper.

2 Theoretical background

In this section, we describe the theoretical background behind our proposed solution to the problem of contextual spellchecking for Croatian language.

2.1 Confusion sets

To solve the problem of detecting and correcting spelling errors, this paper proposes a solution modeled on Kim et al. (2013). A confusion set is a set of words for which there is a high probability of replacement due to either a typographical error or lack of knowledge about the language. An example of a confusion set is {*zahtjeva*, *zahtijeva*} or {*sljedeći*, *slijedeći*}.
The confusion sets can be generated manually or programmatically by using Levenshtein distance. Levenshtein distance is a measure of similarity of two texts (Martin, Jurafsky, 2000). There are four kinds of operations that can be made on a word – insertion of a letter, deletion of a letter, substitution of a letter or transposition of two letters.
While using an edit distance higher than one is possible, due to the nature of the Croatian language, the most common errors are within edit distance of 1.
Using an edit distance of 2 or more would increase the number of words in confusion sets and thus decrease the results of the classifier. In addition to that, the probability of making two errors in one word is very low.

2.2 Classifier

The classifier used in this paper is based on the well-known Naive Bayes classifier.

$$argmax_{TW} P(TW|CW)$$
$$= argmax_{TW} P(CW|TW)P(TW)$$
$$\approx argmax_{TW} \prod_{cw \in CW} P(cw|TW)P(TW)$$

$$TW = \{tw_1, ..., tw_m\}$$
$$CW = \{cw_1, ..., cw_n\}$$

Figure 1. Naive Bayes classifier

In the formula above (Fig. 2) TW (target words) are words from confusion sets and CW (context words) are words from the context of the target words. The context of the target words consists of words and their size depends on the chosen window size, i.e. if the window size is 5 and the third word is the target word, then the first, second, fourth and fifth word form the context.
Probability *P(CW|TW)* can be calculated from the conditional probability between target and context words, but it can only be approximated because CW in theory represents all words in the input except the target word, while $\{cw_1, ..., cw_n\}$ are words within the chosen window size. *P(TW)* is the probability of occurrence of target words.
Since the Naive Bayes classifier uses the probability of occurrence of the target word, the words that occur more frequently will be considered as correct. Because of that, the Naive Bayes classifier must be modified – instead of the probability of occurrence of the target word, we use credibility reliability (Kim et al., 2013).

$$argmax_{TW} P(TW|CW)$$
$$= argmax_{TW} P(CW|TW)P(TW)$$
$$\approx argmax_{TW} \prod_{cw \in CW} P(cw|TW)CR(TW)$$

$$CR(TW) = \begin{cases} 1-\varepsilon, & TW = \text{target word} \\ \frac{\varepsilon}{m-1}, & TW \neq \text{target word} \end{cases}$$

Figure 2. Modified Naive Bayes classifier.

In the formula above (Fig. 3), CR (credibility reliability) is the reliability of words for which the value of $1 - \varepsilon$ is given to the target word and the other value is given to a word from the confusion set.
ε is the typing error rate for the confusion set ($0 \leq \varepsilon \leq 1$) and *m* is the number of words in the confusion set.

2.3 Simple Good-Turing frequency estimation

The training dataset never contains all the words from the language. In this case, that means that there is a set of n-grams that haven't appeared in the training set, but may appear after the training of the classifier. If no method for frequency estimations is used, the system would assign a probability of 0 to unknown words, which would make the spellchecking of n-grams that contain a previously unseen context word impossible. To avoid that problem, a frequency estimator is used. In this research we use the Simple Good-Turing frequency estimation method (Gale, Sampson, 1995).

3 System architecture and n-grams

Our process of n-gram collection relies on the Croatian Academic online spellchecker Hascheck, and involves collecting n-grams from texts received for spellchecking. Our n-gram filtering is based on dictionary criteria. The texts received for spellchecking are used to create the the n-gram database (n = 1, 2, 3, …, 7). We choose to use 5-grams because they give us enough context while not overloading the system.

3.1 System architecture

The architecture of the contextual spellchecking system (Fig. 1) is relatively simple.

Figure 3. System architecture.

It consists of three separate subsystems. First is the data processing subsystem that takes the submitted text, extracts n-grams and calculates probabilities from the number of occurrences of the n-grams.
The second part is the classifier subsystem which uses the base probabilities to calculate the probabilities of the occurrence of each word in the context of the target words.
The last subsystem's task is frequency estimation and it uses the Simple Good-Turing method to calculate the probability of previously unseen words. The single most important characteristic of this system is that all three subsystems are completely decoupled.
The classifier can be completely changed and only the format of storing the final probabilities must remain the same. That means that the subsystems can be replaced without interfering with the rest of the system.

4 Results

In our research, we used the confusion set {*zahtjeva, zahtijeva}*. The confusion set contains approximately 810.000 5-grams split randomly into two parts – 1000 5-grams used as the testing dataset and the rest used as the training dataset. For consistency of the results, the random split was done only once and the same split was used for every improvement.

Table 1. Examples of extracted 5-grams

n-gram
zahtjeva potrebnih za igranje u
zahtjeva za nadzor i čelnika
oštar pad broja prvih zahtjeva
internetske stranice i zahtjeva korisnika
podnositelja zahtjeva na što je
osobine sustava poput zahtjeva za
zahtijeva pisanje programskog koda prema
zahtjeva u informatičke sustave na
ili prosječnu brzinu dolazaka zahtjeva
u javnosti prolazi set zahtjeva
apartmana ne zahtijeva nužno otvaranje

Table 1. shows a small subset of 5-grams extracted from the Hascheck 5-gram data set.

Table 2. Format of result examples

Original n-gram	Corrected n-gram	Probabilities
5-gram from the data set	5-gram corrected by the classifier	probability of *zahtjeva*, probability of *zahtijeva*

Table 2. contains the explanations of the format of tables shown in the following chapters.

4.1 Using modified Naive Bayes classifier

We start by presenting the results that are achieved by just using the modified Naive Bayes classifier.
Precision is the ability of the classifier not to label a negative sample as positive and the formula is $\frac{tp}{tp+fp}$, where *tp* is the number of true positives and *fp* the number of false positives.
Recall is the ability of the classifier to find all positive examples and the formula is $\frac{tp}{tp+fn}$, where *tp* is the number of true positives and *fn* is the number of false negatives.
The F_1 score is the harmonic mean of precision and recall.

Table 3. Results of modified Naive Bayes

Precision	Recall	F_1
0.92	0.88	0.9

As seen in Table 3., the system achieved a high F_1 score of 0.9. This is significant because, as previously stated, a frequency estimator hasn't been used yet and the classifier is based on the simple Naive Bayes classifier.

Table 4. Sample of results of modified Naive Bayes

Original n-gram	Corrected n-gram	Probabilities
zahtjeva potrebnih za igranje u	zahtijeva potrebnih za igranje u	0.49, 0.51
zahtjeva za nadzor i čelnika	zahtijeva za nadzor i čelnika	0.41, 0.59
oštar pad broja prvih zahtjeva	oštar pad broja prvih zahtijeva	0.0, 1.0
internetske stranice i zahtjeva korisnika	internetske stranice i zahtijeva korisnika	0.33, 0.67
podnositelja zahtjeva na što je	podnositelja zahtijeva na što je	0.33, 0.67
osobine sustava poput zahtjeva za	osobine sustava poput zahtijeva za	0.48, 0.52
zahtijeva pisanje programskog koda prema	zahtjeva pisanje programskog koda prema	0.7, 0.3
zahtjeva u informatičke sustave na	zahtijeva u informatičke sustave na	0.43, 0.57
ili prosječnu brzinu dolazaka zahtjeva	ili prosječnu brzinu dolazaka zahtijeva	0.29, 0.71
u javnosti prolazi set zahtjeva	u javnosti prolazi set zahtijeva	0.38, 0.62
apartmana ne zahtijeva nužno otvaranje	apartmana ne zahtjeva nužno otvaranje	0.82, 0.18

Examples from the Table 4. give us some insight into the current state of the system. We see that there are many examples in which the system is very close to the correct decision and only a few which we probably won't be able to correct by fine-tuning.

4.2 Application of the Simple Good-Turing method

By applying the Simple Good-Turing frequency estimator, the system can assign a default probability to previously unseen words. In addition to that, the probabilities of seen words have also been adjusted.

Table 5. Simple Good-Turing results

Precision	Recall	F_1
0.95	0.94	0.95

The results have increased by a significant margin (Table 5.). The F_1 score has gone up from from 0.9 to 0.95 because of the application of a relatively simple frequency estimator.

Table 6. Sample of Simple Good-Turing results

Original n-gram	Corrected n-gram	Probabilities
internetske stranice i zahtjeva korisnika	internetske stranice i zahtijeva korisnika	0.29, 0.71
osobine sustava poput zahtjeva za	osobine sustava poput zahtijeva za	0.43, 0.57
zahtijeva pisanje programskog koda prema	zahtjeva pisanje programskog koda prema	0.67, 0.33
u javnosti prolazi set zahtjeva	u javnosti prolazi set zahtijeva	0.33, 0.67
apartmana ne zahtijeva nužno otvaranje	apartmana ne zahtijeva nužno otvaranje	0.8, 0.2

Table 6. shows that approximately half of the errors present before the application of the Simple Good-Turing method were corrected.

4.3 Finding the optimal ε value

Results of the modified Naive Bayes classifier depend on the value of ε, as discussed in chapter 2.2. By changing the value of ε from 0 to 1 in 0.05 increments, we calculated the corresponding precision, recall and F_1. The subset of results (0 to 0.4) is shown in Table 7.

Table 7. Simple Good-Turing results

ε	Precission	Recall	F_1
0.0	0.78610288	0.79679378	0.79002095
0.05	0.92286562	0.92286562	0.92286562
0.1	0.95099250	0.94581310	0.94828566
0.15	0.94831142	0.94422327	0.94619281
0.2	0.94668353	0.93717405	0.94154730
0.25	0.95371447	0.94114861	0.94679921
0.3	0.94852399	0.93551566	0.94133470
0.35	0.93908864	0.92749797	0.93272569
0.4	0.78610288	0.79679378	0.79002095

Considering that the test set contained 1000 examples of which 200 were errors, the expected highest value of F_1 was for ε around 0.2. In this case, the highest F_1 value was for ε=0.1.

There are two things we can conclude from the results. First, ε can be set to an approximate value, i.e. the average error rate for Croatian. Second, ε can later be optimized for each confusion set separately, further improving the overall results of the system.

5 Future research

In this section, we describe the next steps that are needed to enhance the system described in this paper, in order to be used in a production environment.

5.1 Word parsing

Like all languages, the Croatian language is based on orthographic and grammatical rules. In the case of the Croatian language, the number of grammatical rules is above average, making it more complicated than i.e. English. In addition to many rules, Croatian has a lot of exceptions from those rules. Considering the size of the Hascheck n-gram data set, there is a good chance of "learning" most of those rules only based on statistics. By separating words into parts of speech and specifying the exact form of the word (i.e. past simple verb) it is possible to further increase the quality of the system. The word parser could be added as a separate subsystem when the word type database will be richer.

5.2 Optimizing data storage

Considering that the size of the 5-gram dataset is currently about 20 GB, optimizing data storage plays an important part in the future development of the system. Considering also that Hascheck is a relatively small project with limited funding, the prospects of providing enough system memory without rigorous data optimization is small. Some of the more basic data storage optimizations have been implemented in this research – extracting words from n-grams and converting words into integers.

5.3 Replacing the subsystems

Even though the Naive Bayes classifier paired together with the Simple Good-Turing frequency estimator had good results, there are other classifiers and frequency estimators that would give even better results (Chen, 1996). This means that regardless of the high F_1 score of 0.95, choosing a better classifier and a frequency estimator optimized for this specific data distribution could make the F_1 score get much closer to 1.

Conclusion

The goal of our research was to create a proof of concept contextual spellchecking system using simple and fast components. It was proven that a contextual spellchecking system with fast training times and high F1 score is possible, even for a language as complex as Croatian, by using confusion sets to reduce the number of possible options. The contextual spellchecking system offers a fast and highly efficient way of correcting most contextual errors, especially in categories that are most frequent, i.e. *ije*/*je*.
The specific execution time is dependent on hardware, but the fact that the spellchecking of the 1000 test n-grams takes less than 0.1s is promising. Considering that there is an ever-growing need for high quality contextual spellchecking, further optimization and implementation of our research will be prioritized in the online spellchecking service.

References

Chen, S. F., & Goodman, J. (1996, June). An empirical study of smoothing techniques for language modeling. In Proceedings of the 34th annual meeting on Association for Computational Linguistics (pp. 310-318). Association for Computational Linguistics.

Dembitz, Š., Randić, M., & Gledec, G. (2011). Advantages of online spellchecking: a Croatian example. Software: Practice and Experience, 41(11), 1203-1231.Gale, W. A., & Sampson, G. (1995). Good-turing frequency estimation without tears. Journal of Quantitative Linguistics, 2(3), 217-237.

Kim, M., Jin, J., Kwon, H. C., & Yoon, A. (2013, December). Statistical context-sensitive spelling correction using typing error rate. In Computational Science and Engineering (CSE), 2013 IEEE 16th International Conference on (pp. 1242-1246). IEEE.

Martin, J. H., & Jurafsky, D. (2000). Speech and language processing. International Edition, 710, 25.