A System for Contextual Spell-checking and Diacritics Completion For Czech

Abstract

We present Korektor – a flexible and powerful text correction tool for Czech that goes beyond a traditional spell checker. We use several language models and a morphological analyzer to offer the best ordering of correction proposals and also to find errors that cannot be detected by simple spell checkers, namely spelling errors that happen to be homographs of existing word forms. Our system can also, without any adaptation, generate diacritics for Czech text.

The evaluation demonstrates that the system is a state-of-the-art tool for Czech, both as a spell checker and as a diacritics generator. We also show that these functions combine into a potential aid in the error annotation of a learner corpus of Czech.

1 Introduction

The idea of using the information coming from the context of a misspelled word to improve the performance of a spell checker is not very new (Mays et al., 1991). Recent years have seen the advance of sophisticated spell checkers that use context information. For example *Google Suggest* checks search queries for spelling errors and offers quite reasonable corrections.

These methods are usually based on *noisy-channel* or *winnow-based* approach (Golding and Roth, 1999). The system described here also belongs to the *noisy-channel* class. It makes extensive use of language models based on several morphological factors, exploiting the morphological richness of the target language.

Advanced spell checkers of this type have a natural overlap with abilities of rule-based grammar checkers, because grammar errors also form n-grams with low probabilities.

The purpose of this work was to implement a flexible system capable of performing diverse tasks such as spelling correction, diacritics completion and abbreviated text expansion by a very simple module replacement. In addition, our goal was not only to come up with a scientific prototype. From the beginning we aimed at an end-user system that would provide a better spell checker for Czech than any current system and would be practical in everyday use.

In Section 2, the statistical models used in this work will be introduced together with the description showing how they were utilized for the given tasks of spell-checking and diacritics completion. Section 3 describes the details of the system implementation and Section 4 presents the results of the performance evaluation. Section 6 contains conclusions based of what has been done and outlines possible plans for the future.

2 Statistical Model

The task of context-sensitive spelling correction and diacritics completion can be seen as a problem of sequence decoding which is often formulated in terms of the noisy-channel model. A transmitter sends a sequence of symbols to a receiver. During the transfer, though, certain symbols of the transmitted sequence are garbled due to the deficiencies of the transmission channel. The receiver's goal is to reconstruct the original sequence using the knowledge of the *source* (i.e. how a Czech sentence looks like) and the transmission channel properties.

2.1 Source Modeling

Several feature functions[1] were used to model the source:

- word forms feature - F_f

$$
\begin{aligned}
&F_f(w_1...w_n, w_1^{'}...w_n^{'}) = \\
&\quad \textstyle\sum_{i=1}^{n} f_f((f_{i-2}, f_{i-1}) \rightarrow f_i) = \\
&\quad \textstyle\sum_{i=1}^{n} \log P(f_i|f_{i-2}, f_{i-1})
\end{aligned}
$$

- morphological lemma feature - F_l

$$
\begin{aligned}
&F_l(w_1...w_n, w_1^{'}...w_n^{'}) = \\
&\quad \textstyle\sum_{i=1}^{n} f_l((l_{i-2}, l_{i-1}) \rightarrow (l_i, f_i)) = \\
&\quad \textstyle\sum_{i=1}^{n} \log P(f_i|l_i) + \log P(l_i|l_{i-2}, l_{i-1})
\end{aligned}
$$

- morphological tag feature - F_t

$$
\begin{aligned}
&F_t(w_1...w_n, w_1^{'}...w_n^{'}) = \\
&\quad \textstyle\sum_{i=1}^{n} f_t((t_{i-2}, t_{i-1}) \rightarrow (t_i, f_i)) = \\
&\quad \textstyle\sum_{i=1}^{n} \log P(f_i|t_i) + \log P(t_i|t_{i-2}, t_{i-1})
\end{aligned}
$$

The source feature functions are task independent. They try to ensure the grammaticality of the output sentence. The probability measures were estimated on the basis of n-gram counts collected from a training corpus. Interpolated Kneser-Ney (Kneser and Ney, 1995) smoothing was used.

A large scale text corpus was needed in order to produce well-estimated language models and word emission models. This need was met by *Czech Web Document Collection*(Marek and Pecina, 2007), which is a collection of 111 million words contained in 223 000 articles downloaded from news servers and on-line archives of Czech newspapers. N-gram counts for each morphological factor and counts of `form-lemma` and `form-tag` combinations were collected. For the word forms and lemmas, n-grams up to order 3 were collected. For morphological tags, 4-grams were collected as well.

2.2 Channel Modeling

A single channel feature F_ch estimates the probability of word $w^{'}$ being transferred as word w:

$$
\begin{aligned}
&F_ch(w_1...w_n, w_1^{'}...w_n^{'}) = \\
&\quad \textstyle\sum_{i=1}^{n} f_ch(f_i|f_i^{'}) = \\
&\quad \textstyle\sum_{i=1}^{n} \log P(f_i|f_i^{'})
\end{aligned}
$$

The transmission feature provides the binding between the input words and the output words. The score is assigned according to the similarity of the output words to the input words according to the task specific similarity measure – for the spelling correction problem, it takes into the account the probabilities of specific typing errors. Transmission channel for the diacritics completion is constructed in such a way that it assigns a uniform cost to all variants of an output word with diacritics and infinite cost to all other words.

2.3 Log-linear Model

A log-linear model was used to combine all feature functions into a single statistical model. A general equation capturing a log-linear model for decoding of sequence $w_1^{'}...w_n^{'}$ can be written as follows

$$
(w_1^{*}...w_n^{*}) = \operatorname*{argmax}_{(\hat{w}_1...\hat{w}_n)} \sum_{j=1}^{N} \alpha_j \times F_j(\hat{w}_1...\hat{w}_n, w_1^{'}...w_n^{'})
$$

where $F_{j=1}^{N}$ are feature functions, $\alpha_{j=1}^{N}$ are their weights and $w = (f, l, t)$ denotes the tuple of features representing the given word.

2.4 Hidden Markov Model and Viterbi Algorithm

Exhaustive search for the best hypothesis is not plausible since the number of hypotheses grows exponentially with the length of input sentence.

However, the Viterbi algorithm can be used for sequence decoding since all feature functions are based on statistical estimators with limited history (i.e., they estimate the probability of the next event based on the limited number of past event). This allows us to define a Hidden Markov Model (HMM) where states correspond to the bigrams of triplets $w = (f, l, t)$ and can be denoted as (w_{hist}, w_{act}). Transitions are defined for all pairs of states of the form $(w_1, w_2) \rightarrow (w_2, w_3)$ and the transition costs can be expressed, in conformity with the suggested log-linear model, as

$$
\begin{aligned}
&f\big((wf_1, wf_2) \rightarrow (wf_2, wf_3)\big) = \\
&\quad \alpha_f \times f_f\big((f_1, f_2) \rightarrow f_3\big) \\
&\quad \alpha_l \times f_l\big((l_1, l_2) \rightarrow (l_3, f_3)\big) \\
&\quad \alpha_t \times f_t\big((t_1, t_2) \rightarrow (t_3, f_3)\big)
\end{aligned}
$$

Emission costs of HMM are defined as

$$
g\big(w_2^{'}|(w_1, w_2)\big) = \alpha_{ch} f_c h(f_2^{'}|f_2)
$$

---

[1] For the convenience of the reader, the feature functions are based on trigram statistic in the descriptions. However, higher order n-grams are supported as well.

where $w_i^{'}$ denotes the word at position i in the original sentence.

The Viterbi algorithm (Viterbi, 1967) finds the most probable sequence of states $Q = q_1, q_2, q_3, ...q_T$ of the given HMM when a sequence of observations $S = w_1^{'}, w_2^{'}, w_3^{'}, ..., w_T^{'}$ is given.

The HMM cannot be represented explicitly since the number of possible states and transitions is enormous. Instead, they are built during the evaluation of the Viterbi algorithm. At i-th Viterbi stage, only the states with significant emission probability for the observation $w_i^{'}$ are evaluated.

2.5 Error Model For Spelling Correction

The error model used in this work is based on the model of (Church and Gale, 1991). They consider only candidate words obtained by one edit operation – insertion, deletion, substitution, swap. Edit operations have their distinct probabilities, i.e. the probability of the letter substitution $s \rightarrow d$ may differ from the probability of $e \rightarrow a$. Letter insertion and deletion probabilities are also context-conditioned.

These probabilities were established from the large text corpus. They considered each word that does not appear in the dictionary and is not further than one edit operation from a word included in the dictionary as spelling error. They built their error corpus out of such words. First, they set probabilities of all edit operations uniformly. Later on, they iteratively spell-checked their error corpus, found the best correction for each word and updated edit probabilities according to the proposed *error* $\rightarrow$ *suggestion* pairs.

This method of finding spelling errors was tested on the WebColl corpus (see Section 2.1). However, this method turned out to be useless. The reason was that the vast majority of words identified as spelling errors were correct words or colloquial word forms.

The modified version builds an error corpus out of the words recognized by the spell checker as spelling errors, however there must be a significant evidence that the proposed correction is right, otherwise the spelling error is not added to the error corpus. To be more specific, both bigrams (w_{i-1}, s) and (w_{i+1}, s), where w_{i-1} is the predecessor of a misspelled word e, w_{i+1} is the successive word and s is the correction suggestion, must be present in the language model, otherwise the error-correction pair $e \rightarrow s$ is not included in the error corpus. Recall of this method is rather small, but the precision is quite satisfactory and most of the recognized error-correction pairs were correct. This method identified 12761 words out of 111,000,000 words in WebColl as spelling errors, the classification of these errors is shown in Table 1. The granularity of spelling error types being distinguished is much smaller than in (Church and Gale, 1991). Nevertheless, for languages with straightforward phonology ↔ orthography mapping, such as Czech, error model distinguishing error probabilities mostly on the basis of keyboard layout can be sufficient.

Error Type	Cost
Substitution – horizontally adjacent letters	2.29
Substitution – vertically adjacent letters	2.661
Substitution – $z \rightarrow s$	2.747
Substitution – $s \rightarrow z$	1.854
Substitution – $y \rightarrow i$	3.167
Substitution – $i \rightarrow y$	2.679
Substitution – non-adjacent vocals	3.706
Substitution – diacritic omission	2.235
Substitution – diacritic redundancy	2.250
Substitution – other cases	4.285
Insertion – horizontally adjacent letter	2.290
Insertion – vertically adjacent letter	2.661
Insertion – same letter as previous	1.227
Insertion – other cases	2.975
Deletion	4.14
Swap letters	3.278

Table 1: Spelling Error Types together with their costs ($-log_{10}$ of their probabilities)

2.6 Letter Language Model For Diacritics Completion

It may happen that for a given word of the input sentence, no candidate word is found. The example of such a word is the word *nemeckofrancouzsky* ‘German-French’. In such a case, the word remains untouched and no diacritics is added. However, there is a high probability of error in such case, in the provided example the diacritics should rather be completed as *německofrancouzský* (adjective) or *německofrancouzsky* (adverb).

In order to cut down the number of errors made on unknown words, a custom implementation of the Viterbi decoder was provided. The states on the underlying HMM are tuplets of letters and the transition probabilities are given by a letter n-gram language model (it estimates the probability of next letter on the basis of previous letters).

The aim of this Viterbi decoder is to find the most probable letter sequence given the input letter sequence. The only substitutions allowed are the substitutions that add diacritics.

Using this approach, diacritics can be added correctly even to the unknown words.

Given that the vocabulary of letter n-gram language model is extremely small (the size of the alphabet), it is possible to train letter LMs of a very high order. In this work, letter LMs of the order up to 7 were trained. The letter LMs were trained on the training part of WebColl.

The contribution of using letter LMs was examined during the evaluation, Table 4 shows the significant accuracy improvement when this feature is used.

3 System Implementation

Since the system was developed from the beginning with the goal of wide distribution of an end-user system, not just a proof of concept, a special care was devoted to the efficiency of its implementation, in terms of both low memory and CPU footprint.

The spelling dictionary was implemented as the TRIE (Fredkin, 1960) data structure and the language models were as the ZipTBO structures, using about 4 bytes per n-gram (Whittaker and Raj, 2001)[2]. The system uses caching extensively, statistics for recently used n-grams, correction candidates for recently used words and many other intermediate results are stored in less memory efficient but rapid access speed structures to allow better performance without a significant impact on the memory consumption.

The system was implemented in C++ and can be used either as a command line utility or as a SpellServer and a System Service on Mac OS X. Mac OS X was chosen for practical reasons as the initial distribution platform. Its API allows to make our services (spellchecking, diacritics stripping and completion) easily available to all modern applications on the platform.

4 Evaluation

4.1 Diacritics Completion Results

The diacritics Completion was evaluated on three different datasets, a part of the WebColl corpus devoted for testing and two different books[3]: *Martin Gilbert: A History of the Twentieth Century* (non-fiction) and *Lion Feuchtwanger: Foxes in the Vineyard* (fiction).

The main parameters are the weights α_f, α_l and α_t of features F_f, F_l and F_t.

First, the contributions F_l and F_t were examined separately. In these experiments α_f was ranging from 0 to 1 and the weight $(1 - \alpha_f)$ was given either F_l or F_t, all the language models used were trigrams. The results of such experiments are shown in Tables 2, 3. It is clear from the plots that both features F_l, F_t improve the system performance. However the contribution of F_t is more significant. Surprisingly, it seems to be better to give all the weight to F_t than to give all the weight to F_f.

The performance boost achieved by using F_t is most visible on a comparison of results achieved on history domain and fiction domain data. For baseline setup ($\alpha_f = 1, \alpha_t = 0$), the accuracy is 97.39% on non-fiction data and 96.74% on fiction data, which means that the error rate is 25% bigger on fiction data. Nevertheless, by increasing the weight of F_t the difference in performance was becoming less significant and for the best parameter settings ($\alpha_f = 0.4, \alpha_t = 0.6$), the error rate on fiction data was only 7% bigger (97.72% accuracy on fiction data and 97.89 on non-fiction data).

Next, the estimation of the best parameter setting for each data set was done using a simple hill-climbing algorithm (description can be found in (Russell and Norvig, 2003)). As the starting point, all the weights were set equally. The resulting parameters and the accuracy values are shown in the Figure 4. The results of experiments with the letter LM feature turned on were made as well for the particular settings. It can be seen that the use of letter LM for the completion of the unknown words improves the results significantly.

The comparison between CZACCENT[4], the diacritics completion tool provided NLP Center of Masaryk University Brno, and the diacritics completion provided by Korektor was made on

---

[2] It is possible to go even further and reduce the memory consumption to 3 bytes per n-gram (as proposed by (Church et al., 2007)), but this representation is less CPU-efficient and ZipTBO seemed to be more suitable for the given task.

[3] Czech translation were used for foreign-language originals

[4] http://nlp.fi.muni.cz/cz_accent/index.php

α_l	non-fiction	fiction	WebColl
0.1	97.45%	96.82%	98.16%
0.3	97.49%	96.86%	98.21%
0.5	97.51%	96.85%	98.20%
0.7	97.45%	96.77%	98.09%
0.9	97.18%	96.48%	97.79%

Table 2: Results of *form – lemma* experiments. Only F_l and F_f are used for source modeling and $\alpha_f = (1 - \alpha_l)$.

α_t	non-fiction	fiction	WebColl
0.1	97.66%	97.20%	98.35%
0.3	97.83%	97.60%	98.53%
0.5	97.88%	97.74%	98.57%
0.7	97.85%	97.71%	98.52%
0.9	97.62%	97.53%	98.26%

Table 3: Results of *form – tag* experiments. Only F_t and F_f are used for source modeling and $\alpha_f = (1 - \alpha_t)$.

the *non-fiction* data set. The accuracy achieved by CZACCENT was 95.85% and the accuracy achieved by Korektor was 98.3%, which means that the error rate of Korektor was almost 2.5 times smaller.

4.2 Spell-checking Results

The quality of spell checkers is usually measured by the spelling correction error rate (i.e., the probability that the first given suggestion is correct, or that the correct suggestion is included in the list of first three suggestions etc.) If the context sensitive spell checker is considered and the ability of recognizing the real-word errors is to be tested, *F-measure* based on *precision* and *recall* can be used. It is a good indicator of a quality of a classifier.

During the evaluation of spelling correction, the optimal parameter settings (weights of distinct feature functions) estimated for the diacritic completion task were used based on the assumption that the features F_f, F_l and F_t are task independent and that their weighting obtained for one task will perform well for other tasks as well. The reason why we made no separate parameter tuning was that the size of available annotated spelling error data was too small. The weights were set according to the optimal setting for Diacritic Completion on non-fiction, i.e. $\alpha_f = 0.31$, $\alpha_l = 0.28$ and $\alpha_t = 0.41$. Channel feature F_ch was assigned a weight $\alpha_ch = 1.0$ which assigns the same importance to both source model and channel model.

For the evaluation of spell-checking, three different data sets were used.

- Error corpus *Chyby* (Pala et al., 2003)
- Audio Book Transcription
- WebColl – testing set – semi-automatically recognized spelling errors in the part of WebColl not used during the training
- Learners Corpus

The error corpus *Chyby* (Pala et al., 2003) which is being built in Brno is a collection of essays written by students of Brno University of Technology, annotated for errors including spelling, morphological, syntactic and stylistic errors. The spell-checking was tested on spelling and morphological errors since these types of errors are possibly recognizable by the system. There were 744 such errors and out of this number, 321 were real word errors. The high ratio of real word errors show that most of the student works were already spell-checked.

The Audio Book Transcription (Audio) testing set contains 218 spelling errors (out of this number, 12 errors are real-word errors) and there were 1371 words in total. The dataset was constructed by making transcriptions of an audio version of a Czech novel by Jaroslav Hašek: *Osudy dobrého vojáka Švejka* ‘The Good Soldier Švejk’.[5] There was no post-correction made on the transcribed text and the spelling error rate in the resulting text is relatively high.

WebColl testing set was created out of the part of WebColl not used during the system training. Spelling errors were collected semi-automatically with the use of the Korektor. The words identified by the spell-checker[6] as spelling errors were examined manually and the words that were flagged as spelling errors by mistake were filtered out. The result of this process was the set of sentences containing spelling errors authorized by a human. The

---

[5] The audio extracts can be downloaded for free from the web-sites of Český rozhlas: http://www.rozhlas.cz/ctenarskydenik

[6] The spell-checker made look-up for the out of vocabulary words easier. The correction suggestions given by spell-checker were not taken into the consideration during the creation of golden standard data, so the fact that the spell-checker that is to be tested participated in the creation of the testing set does not invalidate the testing set.

dataset	α_f	α_l	α_t	accuracy – no Letter LM	accuracy – with Letter LM
non-fiction	0.31	0.28	0.41	97.9%	98.3%
fiction	0.31	0.14	0.55	97.7%	97.9%
WebColl	0.34	0.33	0.33	98.6%	99.1%

Table 4: The best accuracy values achieved on each testing set.

golden standard data were created manually in the next step. This approach made the collecting of errors in the WebColl testing data feasible, however all the real-word errors were missed (they were overlooked, because they were not flagged as spelling errors by spell-checker in the first step). Because of this, only the evaluation of suggestion accuracy could be done for this data.

The results of spelling correction accuracy evaluation for Chyby, Audio and Webcoll are shown in Table 5 and the results of real word error detection evaluation are shown in Table 6. For the Audio dataset, comparison with the Microsoft Word 2007 spell checker with grammar checking features turned on was made. Only the accuracy on the first suggestion was made for MS Word spell checker since there is no API that would allow to do the evaluation automatically. The results suggest that accuracy on a single suggestion is much higher for Korektor as well as the ability to detect real word spelling errors. The cases when the MS Word spell checker marked a grammar error were all because of capitalization problems, which suggests that there is no statistical real-word error detection in the Czech version of MS Word [7] Significantly smaller spelling correction rate on Chyby corpus can be caused by the fact that the characteristics of language in Chyby corpus (technical topics) differs significantly from the the training data characteristics (newspapers).

4.3 Correcting Errors in a Learner Corpus (joint task)

A learner corpus consists of texts produced by learners of a second or foreign language. Deviant expressions can be corrected and/or annotated by error type tags. The error tagging system for a learner corpus of Czech (*CzeSL* – Czech as a Second/Foreign Language)[8] is based on a two-stage annotation design, consisting of three levels.

Level 0 includes transcripts of the original handwriting. At the level of orthographical and morphological corrections (Level 1), only forms incorrect in any context are treated, except for obvious misspellings resulting in homographs. All other types of errors are corrected at Level 2. Links connecting forms at different levels are labeled with their error type.

In a preliminary study of 67 short essays, doubly annotated at both levels by correct forms and error codes, the primary aim was to verify the feasibility of the annotation scheme by computing inter-annotator agreement scores. Additionally, Korektor was used to see whether an automatic correction of learner texts is viable as a way to assist the annotator. At the same time, this experiment helped to evaluate its results.

Among the total 9,372 tokens, 918 (10%) were not recognized by a tagger (see *morče* in (Spoustová et al., 2007)). Even more forms were judged as faulty by the annotators: 1,189 (13%) were corrected in the same way by both annotators at Level 1 and 1,519 (16%) at Level 2.

Results of Korektor were compared with those of the tagger and with forms at Level 1 (L1) and Level 2 (L2), provided both annotators were in agreement. Korektor was run in three (batch) modes: (i) “autocorrect” (as proofreader), (ii) “remove-diacritics” followed by “diacritics” (as diacritics assigner), and (iii) same as in (ii), followed by “autocorrect”, the latter two to test the hypothesis that diacritics is a frequent source of errors.

Although the tagger includes a guesser, it makes no attempt to correct an unknown word form, only guesses its morphosyntactic tag and lemma. Korektor is deemed to be successful if it agrees with the tagger in the correct/incorrect status of the form.

Table 7 shows figures for the tagger. The rows give results for the three modes: **cor** for mode (i), **dia** for (ii), and **c+d** for (iii). The column **c’cted** gives the counts for forms corrected by Korek-

---

[7] However, the MS Word spell checker for Czech is equipped with other capabilities that Korektor does not possess such as punctuation checking.

[8] See (Hana et al., 2010). The project is funded by the European Social Fund and the government of the Czech Republic (project no. CZ.1.07/2.2.00/07.0259).

Number of suggestion	WebColl	Chyby 1	Chyby 2	Audio (Korektor)	Audio (MS Word)
1	91.4%	73.5%	82.3%	91.6%	71.2%
2	95.1%	80.1%	80.9%	97.2%	-
5	96.3%	80.9%	90.5%	98.6%	-

Table 5: Spelling correction rates achieved on the different datasets. For the Chyby corpus, two measurements were taken. In *Chyby 1*, all spelling errors are considered. For the *Chyby 2*, only those spelling errors for which an appropriate correct version is in the lexicon are taken into account.

	Chyby	Audio (Korektor)	Audio (MS Word)
Precision	0.41	1.0	0.5
Recall	0.24	0.77	0.08
F-measure	0.31	0.87	0.14

Table 6: Real word error correction statistics for Audio dataset and Chyby corpus.

tor. The column **unkn** gives the number of cases where the tagger flags a form corrected by Korektor as unknown. The results of the tagger are assumed as truth for precision (**unkn**/**c'cted**) and recall (**unkn**/918).

Precision is not a fair measure here, because the tagger never flags real-word errors, while Korektor often manages to use local context to replace a form with an orthographically close but morphosyntactically quite different form: *podlé→podle, jejích→jejich, žit→žít, libí→líbí, ze→že, divá→dívá, drahy→drahý, mel→měl, jích→jich, čine→číně*. Diacritics represents a substantial share of problems in learners' writings, and the preprocessing of the input by the diacritics remover and assigner (iii) means a significant improvement.

mode	c'cted	unkn	prec	recall	F-msr
cor	1151	888	0.77	0.97	0.86
dia	1176	795	0.68	0.87	0.76
c+d	1315	906	0.69	0.99	0.81

Table 7: Comparison with tagger, which identified the total of 918 unknown forms

Corrections made by the annotators can be compared verbatim with those proposed by Korektor. Korektor scores whenever it matches the form at L1 or L2, respectively. The two annotators must agree about the corrected form.

At L1 the total number of corrections (1189) is higher than the number of forms unknown to the tagger (918) because the annotators correct some real words. The result is a lower recall. Precision stays the same because L1 is similar to the tagger: it still largely abstracts from context. E.g., the annotators are instructed to leave errors due to missed grammatical concord for L2. The data are shown in Table 8 – the column **c'cted** is identical to that in Table 7, but the **wrong** column shows the number of cases where the annotators agree with Korektor about a correction.

mode	c'cted	wrong	prec	recall	F-msr
cor	1151	846	0.74	0.71	0.72
dia	1176	780	0.66	0.66	0.66
c+d	1315	904	0.69	0.76	0.72

Table 8: Comparison with corrections at L1, where annotators agreed on the total of 1189 wrong forms

It some cases where Korektor does not agree with the annotators, but both Korektor and the annotators indicate an error (170 such cases at L1 for mode c+d), mode (i) without the diacritics component fares better (in 30 cases out of 170). Here removing and reassigning diacritics takes Korektor too far (Table 9). In some cases the L1 and L2 versions differ and none of the methods matches the contextually correct version of L2 (*pláž, lépe*).

In 150 cases Korektor suggests a correction when L1 prefers the original, but in 37 cases Korektor agrees with an annotator at L2 (in 16 cases with both), which means that the real precision is higher. The rest of the cases are mostly inflectional issues, often due to misassigned diacritics, but also annotation errors (shared by both annotators).

L2 is problematic for evaluation in its own right. Some error types handled here are due to wrong word order, style, phraseology and a few others

R0	c+d	R1=cor	R2
plaži	*plážží*	*pláži*	*pláž*
tydnů	*týdnů*	*týdnu*	*týdnu*
lepšé	*lepše*	*lepší*	*lépe*
jide	*lidé*	*jde*	*jde*
vždicky	*vodičky*	*vždycky*	*vždycky*
všihny	*viny*	*všichni*	*všichni*
zmřel	*zmrzl*	*zemřel*	*zemřel*
sejdime	*šmejdíme*	*sejdeme*	*sejdeme*

Table 9: Where simple autocorrect mode is better

that go beyond simple spell checking, even in a broader sense of some degree of contextual sensitivity. The figures in Table 10, otherwise similar to Table 8, should be interpreted accordingly.

mode	c'cted	wrong	prec	recall	F-msr
cor	1151	687	0.60	0.45	0.51
dia	1176	640	0.54	0.42	0.47
c+d	1315	745	0.57	0.49	0.53

Table 10: Comparison with corrections at L2, where annotators agreed on the total of 1519 wrong forms

The two-stage annotation scheme seems to be well-suited to test the grammar-checking capabilities of Korektor on real-word errors. However, the annotation was not designed to test a spell checker, let alone any advanced features of this kind. Still it is possible to find cases of successful corrections of missed agreement or case government. The mode combining diacritics remover, assigner and proofreader is the best scenario.

Even though its suggestions may not quite match the two annotation levels, the results support the idea to integrate Korektor into the learner corpus annotation workflow, either as suggestions to the annotator or as a solution to obtain large-scale annotation at the cost of a higher error rate.

5 Discussion

The results achieved for spelling correction accuracy are not as good as the results reported in Brill and Moore (2000), where the accuracy around 95% for the first suggestion was reported. However, those results were achieved for English and are thus not directly comparable. The rich morphology of Czech is expected to lower the results somewhat. For the Chyby corpus, significantly lower performance (73% on the first suggestion) was probably caused by the heavy usage of technical terminology, such as names of software products and their declination variants. The fact that Korektor clearly outperformed Microsoft Word 2007 spell checker with the grammar checker included is the best direct indicator of system qualities available.

6 Conclusion and Future Work

A context-sensitive method of spell-checking and diacritics completion was designed and implemented. The resulting spell checker is freely available and ready to use.

Regarding the spell-checking task, emphasis was put on the ability of the system to recognize real-word spelling errors and also on the ability to suggest the most probable corrections for spelling errors.

In the spell-checking evaluation, Korektor achieved much better performance than the MS Word 2007 spell checker. However, the performance of the spell checker shows significant dependence on the domain of the testing data.

Using the implementation of the spell checker, diacritics completion was implemented. The accuracy of diacritics completion was about 98% with training and testing data coming from different domains. Such performance is already acceptable for many tasks. Some of the authors have been using the diacritics completion for emails typed in ASCII in daily use.

To the best of our knowledge both the spell checker and diacritics completion service of Korektor are the most efficient writing aids for Czech currently available.

Korektor was already used during the anotation of CzeSL corpus, which resulted in a significant improvement of annotation efficiency.

In the future, we would like to adapt Korektor to other languages by estimating the language-specific language and error models. A possible improvement of the spell-checking capabilities of Korektor could be achieved by utilization of more fine-grained error models as proposed by Brill and Moore (2000). In standard Czech it has a limited value, but the experiments on a learner corpus show it could still be useful for non-native speakers. For languages with less straightforward orthography it is even more valuable.

References

Eric Brill and Robert C. Moore. 2000. An improved error model for noisy channel spelling correction. In *ACL '00: Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 286–293, Morristown, NJ, USA. Association for Computational Linguistics.

K. Church and W. Gale. 1991. Probability scoring for spelling correction. *Statistics and Computing*, 1(7):93–103.

Ken Church, Redmond Wa, Ted Hart, and Jianfeng Gao. 2007. Compressing trigram language models with golomb coding. In *In Proceedings of EMNLP-CoNLL 2007*, Prague, Czech Republic.

Edward Fredkin. 1960. Trie memory. *Commun. ACM*, 3:490–499, September.

Andrew R. Golding and Dan Roth. 1999. A winnow-based approach to context-sensitive spelling correction. *Machine Learning*, 34:107–130. 10.1023/A:1007545901558.

Jirka Hana, Alexandr Rosen, Svatava Škodová, and Barbora Štindlová. 2010. Error-tagged learner corpus of Czech. In *Proceedings of the Fourth Linguistic Annotation Workshop*, Uppsala, Sweden, July. Association for Computational Linguistics.

R. Kneser and H. Ney. 1995. Improved backing-off for M-gram language modeling. In *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, volume 1, pages 181–184 vol.1.

Michal Marek and Pavel Pecina. 2007. Web page cleaning with conditional random fields. In *CleanEval*. submitted to CleanEval 2007.

Eric Mays, Fred J. Damerau, and Robert L. Mercer. 1991. Context based spelling correction. *Information Processing & Management*, 27(5):517 – 522.

Karel Pala, Pavel Rychlý, and Pavel Smrž. 2003. Text corpus with errors. In *TEXT, SPEECH AND DIALOGUE*, volume 2807/2003 of *Lecture Notes in Computer Science*, pages 90–97. Springer.

Stuart Russell and Peter Norvig. 2003. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Englewood Cliffs, NJ, 2nd edition edition.

Drahomíra Spoustová, Jan Hajič, Jan Votrubec, Pavel Krbec, and Pavel Květoň. 2007. The best of two worlds: Cooperation of statistical and rule-based taggers for Czech. In *Proceedings of the Workshop on Balto-Slavonic Natural Language Processing 2007*, pages 67–74, Praha, Czechia. Association for Computational Linguistics.

A. J. Viterbi. 1967. Error bounds for convolutional codes and an asymptotically optimal decoding algorithm. *IEEE Transactions on Information Theory*, 13:260–269.

E. W. D. Whittaker and B. Raj. 2001. Quantization-based language model compression. In *European Conference on Speech Communication and Technology*.