Improving Finite-State Spell-Checker Suggestions with Part of Speech N-Grams

Tommi A Pirinen and Miikka Silfverberg and Krister Lindén

University of Helsinki
Department of Modern Languages
University of Helsinki, 00014
tommi.pirinen@helsinki.fi, miikka.silfverberg@helsinki.fi, krister.linden@helsinki.fi

Abstract. In this paper we demonstrate a finite-state implementation of context-aware spell checking utilizing an N-gram based part of speech (POS) tagger to rerank the suggestions from a simple edit-distance based spell-checker. We demonstrate the benefits of context-aware spell-checking for English and Finnish and introduce modifications that are necessary to make traditional N-gram models work for morphologically more complex languages, such as Finnish.

1 Introduction

Spell-checking by computer is perhaps one of the oldest and most researched applications in the field of language technology starting from the mid 20th century [3]. One of the crucial parts of spell-checking—both from an interactive user-interface point of view and for unsupervised correction of errors—is the production of spelling suggestions. In this article we test various finite-state methods for using context and shallow morphological analysis to improve the suggestions generated by traditional edit distance measures or unigram frequencies such as [12].

The spell-checking task can be split into two parts, i.e. *detection* and actual *correction* of the spelling errors. The spelling errors can be detected in text as word forms that are unlikely to belong to the natural language in question, such as writing 'cta' instead of 'cat'. This form of spelling errors is commonly called *non-word (spelling) errors*. Another form of spelling errors is word forms that do not belong to the given context under certain syntactic or semantic requirements, such as writing 'their' instead of 'there'. This form is correspondingly called *real-word (spelling) errors*. The non-word type of spelling errors can easily be detected using a dictionary, whereas the detection of the latter type of errors typically requires syntactic analysis or probabilistic methods [8]. For the purpose of this article we do not distinguish between them, as the same correction methods can be applied to both.

The correction of spelling errors usually means generating a list of word forms belonging to the language for a user to chose from. The mechanism for generating correction suggestions for the erroneous word-forms is an *error-model*. The purpose of an error-model is to act as a filter to revert the mistakes the user typing the erroneous word-form has made. The simplest and most traditional model for making such corrections is the Levenshtein-Damerau edit distance algorithm, attributed initially to [6] and especially in the context of spell-checking to [3]. The Levenshtein-Damerau edit distance assumes that spelling errors are one of insertion, deletion or changing of a single character to another, or swapping two adjacent characters, which models well the spelling errors caused by an accidental slip of finger on a keyboard. It was originally discovered that for most languages and spelling errors, this simple method already covers 80 % of all spelling errors [3]. This model is also language-independent, ignoring the differences in character repertoires of a given language. Various other error models have also been developed, ranging from confusion sets to phonemic folding [5].

In this paper, we evaluate the use of context for further fine-tuning of the correction suggestions. The context is still not commonly used in spell-checkers. According to [5] it was lacking in the majority of spell-checkers and while the situation may have improved slightly for some commercial office suite products, the main spell-checkers for open source environments are still primarily context-ignorant, such as hunspell[1] which is widely used in the open source world. For English, the surface word-form trigrams model has been demonstrated to be reasonably efficient both for non-word cases [2] and for for real-word cases[7]. As an additional way to improve the set of suggestions, we propose to use morphosyntactically relevant analyses in context. In this article, we evaluate a model with a statistical morphological tagger [14].

The system described is fully built on freely available tools and data, available for download and use from `http://hfst.svn.sourceforge.net/viewvc/hfst/trunk/cicling-2011-contextspell/`. The only exception to this is the training data for Finnish, since there is no available morphological training data for Finnish as of yet, the download does not contain the source material for training but only the trigram models compiled into binary format automata.

Furthermore, we test the context-based spelling methods using both English and Finnish language materials to ensure the applicability of the method for morphologically different languages. The reason for doing this is two-fold; firstly the fact that English has rather low morphological productivity may make it behave statistically differently from other languages. On the other hand, English has the largest amount of freely available text corpora. For other languages, the availability of free corpora, especially annotated material, is often seen as a problem.

The article is laid out as follows: In Section 2, we outline the implementation of a finite-state context-aware spell-checker and describe the statistical methods used. In Section 3, we introduce the corpora and dictionaries used for spell-checking and training material as well as the corpora used for obtaining the spelling errors with context. In Section 4, we show how the created spelling correctors improve the results and explain the errors left. In Section 5, we com-

[1] `http://hunspell.sf.net`

pare our work to other current systems and enumerate possible improvements for both.

2 Methods

The spelling correction in this article is performed in several phases: assuming misspelled word *cta* for *cat*, we first apply the error model to the already known incorrect string *cta* to produce candidates for probable mistypings. For this purpose we use the Damerau-Levenshtein edit-distance algorithm in finite-state form. When applied to *cta* we get all strings with one or two typing mistakes, i.e. *ata*, *bta*, ..., *acta*, *bcta*, ..., *ta*, *ca*, ..., *tca*, and the correct *cat*. This set of strings is simultaneously matched against the language model, which will produce a set of corrections, such as *cat*, *act* or *car*. Since both the error-model and the language model contain information on likelihoods of errors and words respectively, the resulting list will be sorted according to a combination of the edit distance measure and the probability of the word in a reference corpus. The rankings based on edit distance alone and the edit distance combined with word probabilities form our two baseline models.

The context-based models we introduce here use the suggestion list gained from a contextless spelling-checker and the context of the words as input to rerank suggestions based on N-gram models. Each of the suggestions is tried against the N-gram models, and the ones with higher likelihoods will be lifted. For example when correcting the misspelling of 'an' as 'anx' in the sentence "this is anx example sentence", as shown in the Table 1, we have the surface trigrams {this, is, _}, {is, _, example}, {_, example, sentence}, and corresponding analysis trigrams {DET, VVBZ, _}, {VVBZ, _, NN}, {_, NN, NN}. The suggestions for anx at edit distance one include 'ax', 'an' (one deletion), 'ant', 'and', 'any' (one change) and so on. To rank the possible suggestions, we substitute s_3 with the suggestions, and calculate the likelihood of their analyses.

Table 1. Example trigram combinations

this$_{s_1}$	is$_{s_2}$	_$_{s_3}$	example$_{s_4}$	sentence$_{s_5}$
DET$_{a_1}$	VVBZ$_{a_2}$	_ $_{a_3}$	NN$_{a_4}$	NN$_{a_5}$

2.1 Weighted Finite-State Interpretation of the Method

In this article we use a finite-state formulation of spell-checking. We assume the standard notation for finite-state algebra and define the language model as a weighted finite-state automaton assigning a weight to each correctly spelled word-form of a language, and an error model automaton mapping a misspelled string to a set of corrected strings and their weights. The probabilistic interpretation of the components is such that the weighted fsa as a language model assigns weight $w(s)$ to word s corresponding to the probability $P(s)$ for the word to be a correct word in the language. The error model assigns weight $w(s:r)$ to string pair s, r corresponding to the probability $P(s|r)$ of a user writing word r when intending to write the word s, and the context model assigns weight $w(s_3 a_3)$ to word s_3 with associated POS tagging a_3 corresponding to the standard HMM estimate $P(a_3 s_3)$ of the analysis being in a 3-gram context given by equation (1).

$$P(a_3 s_3) = \prod_{i=3}^{5} P(s_i|a_i) P(a_i|a_{i-2},\ a_{i-1}) \tag{1}$$

In a weighted finite-state system, the probabilistic data needs to be converted to the algebra supported by the finite-state weight structure. In this case we use the tropical semi-ring by transforming the frequencies into penalty weights with the formula $-\log \frac{f}{CS}$, where f is the frequency and CS the corpus size in number of tokens. If the language model allows for words that are not in the dictionary, a maximal weight is assigned to the unseen word forms that may be in the language model but not in the training corpus, i.e. any unseen word has a penalty weight of $-\log \frac{1}{CS}$.

The spelling corrections suggested by these unigram lexicon-based spell-checkers are initially generated by composing an edit-distance automaton [13] with an error weight corresponding to the probability of the error estimated in a corpus, i.e. $-\log \frac{f_F}{CS+1}$, where f_F is the frequency of the misspelling in a corpus. This weight is attached to the edit distance type error. In practice, this typically still means that the corrections are initially ordered primarily by the edit distance of the correction, and secondarily by the unigram frequency of the word-form in the reference corpus. This order is implicitly encoded in the weighted paths of the resulting automaton; to list the corrections we use the n-best paths algorithm [9]. This ordering is also used as our second baseline.

For a context-based reordering of the corrections, we use the POS tagging probabilities for the given suggestions. The implementation of the analysis N-gram probability estimation is similar to the one described in [14] with the following adaptations for the spelling correction. For the suggestion which gives the highest ranking, the most likely analysis is selected. The N-gram probability is estimated separately for each spelling suggestion and then combined with the baseline probability given by the unigram probability and the edit distance weight. The ideal scaling for the weights of unigram probabilities, i.e. edit distance probabilities with respect to N-gram probabilities, was acquired by performing tests on an automatically generated spelling error corpus.

The resulting finite-state system consists of three sets of automata, i.e. the dictionary for spell-checking, the error-model as described in [12], and the new N-gram model automata. The automata sizes are given in Table 2 for reference. The sizes also give an estimate of the memory usage of the spell-checking system, although the actual memory-usage during correction will rise depending on the actual extent of the search space during the correction phase.

Table 2. Automata sizes.

Automaton	States	Transitions	Bytes
English			
Dictionary	25,330	42,448	1.2 MiB
Error model	1,303	492,232	5.9 MiB
N-gram lexicon	363,053	1,253,315	42 MiB
N-gram sequences	46,517	200,168	4.2 MiB
Finnish			
Dictionary	179,035	395,032	16 MiB
Error model	1,863	983,227	12 MiB
N-gram lexicon	70,665	263,298	8.0 MiB
N-gram sequences	3,325	22,418	430 KiB

2.2 English-Specific Finite-State Weighting Methods

The language model for English was created as described in [10]. [2]. It consists of the word-forms and their probabilities in the corpora. The edit distance is composed of the standard English alphabet with an estimated error likelihood of 1 in 1000 words. Similarly for the English N-gram material, the initial analyses found in the WSJ corpus were used in the finite-state tagger as such. The scaling factor between the dictionary probability model and the edit distance model was acquired by estimating the optimal multipliers using the automatic misspellings and corrections of a Project Gutenberg Ebook[3] *Alice's Adventures in Wonderland*.

2.3 Finnish-Specific Finite-State Weighting Methods

The Finnish language model was based on a readily-available morphological weighted analyser of Finnish language [11]. We further modified the automaton to penalize suggestions with newly created compounds and derivations by adding a weight greater than the maximum to such suggestions, i.e. $-A \log \frac{1}{CS+1}$ in the training material. This has nearly the same effect as using a separate dictionary for suggestions that excludes the heavily weighted forms without requiring the extra space. Also for Finnish, a scaling factor was estimated by using automatic misspellings and corrections of a Project Gutenberg Ebook[4] *Juha*.

In the initial Finnish tagger, there was a relatively large tagset, all of which did not contain information necessary for the task of spell-checking, such as discourse particles, which are relatively context-agnostic [4], so we opted to simplify the tagging in these cases. Furthermore, the tagger used for training produced

[2] The finite-state formulation of this is informally described in `http://blogs.helsinki.fi/tapirine/2011/07/21/how-to-write-an-hfst-spelling-corrector/`

[3] `http://www.gutenberg.org/cache/epub/11/pg11.txt`

[4] `http://www.gutenberg.org/cache/epub/10863/pg10863.txt`

heuristic readings for unrecognized word-forms, which we also removed. Finally, we needed to add some extra penalties to the word forms unknown to the dictionary in the N-gram model, since this phenomenon was more frequent and diverse for Finnish than English.

3 Material

To train the spell-checker lexicons, word-form probabilities can be acquired from arbitrary running text. By using unigram frequencies, we can assign all word-forms some initial probabilities in isolation, i.e. with no spell-checking context. The unigram-trained models we used were acquired from existing spell-checker systems [10, 12], but we briefly describe the used corpora here as well.

To train the various N-gram models, corpora are required. For the surface-form training material, it is sufficient to capture running N-grams in the text. For training the statistical tagger with annotations, we also require disambiguated readings. Ideally of course this means hand-annotated tree banks or similar gold standard corpora.

The corpora used are summarized in Table 3. The sizes are provided to make it possible to reconstruct the systems. In practice, they are the newest available versions of the respective corpora at the time of testing. In the table, the first row is the training material used for the finite-state lexicon, i.e. the extracted surface word-forms without the analyses for unigram training. The second row is for the analyzed and disambiguated material for the N-gram based taggers for suggestion improvement. The third line is the corpora of spelling errors used only for the evaluation of the systems. As we can see from the figures of English compared with Finnish, there is a significant difference in freely available corpora such as Wikipedia. When going further to lesser resourced languages, the number will drop enough to make such statistical approaches less useful, e.g. Northern Sámi in [12].

Table 3. Sizes of training and evaluation corpora.

	Sentences	Tokens	Word-forms
		English	
Unigrams		2,110,728,338	128,457
N-grams	42,164	969,905	39,690
Errors	85	606	217
		Finnish	
Unigrams		17,479,297	968,996
N-grams	98,699	1,027,514	144,658
Errors	333	4,177	2,762

3.1 English corpora

The English dictionary is based on a frequency weighted word-form list of the English language as proposed in [10]. The word-forms were collected from the English Wiktionary[5], the English EBooks from the project Gutenberg[6] and the British National Corpus[7]. This frequency weighted word-list is in effect used as a unigram lexicon for spell-checking.

To train an English morphosyntactic tagger, we use the WSJ corpus. In this corpus each word is annotated by a single tag that encodes some morphosyntactic information, such as part-of-speech and inflectional form. The total number of tags in this corpus is 77 [8].

The spelling errors of English were acquired by extracting the ones with context from the Birkbeck error corpus[9]. In this corpus, the errors are from a variety of sources, including errors made by children and language-learners. For the purpose of this experiment we picked the subset of errors which had context and also removed the cases of word joining and splitting to simplify the implementation of parsing and suggestion.

3.2 Finnish Corpora

As the Finnish dictionary, we selected the freely available open source finite-state implementation of a Finnish morphological analyser[10]. The analyser had the frequency-weighted word-form list based on Finnish Wikipedia[11] making it in practice an extended unigram lexicon for Finnish. The Finnish morphological analyser, however, is capable of infinite compounding and derivation, which makes it a notably different approach to spell checking than the English finite word-form list.

The Finnish morphosyntactic N-gram model was trained using a morphosyntactically analyzed Finnish Newspaper[12]. In this format, the annotation is based on a sequence of tags, encoding part of speech and inflectional form. The total number of different tag sequences for this annotation is 747.

For Finnish spelling errors, we ran the Finnish unigram spell-checker through Wikipedia, europarl and a corpus of Finnish EBooks from the project Gutenberg[13] to acquire the non-word spelling errors, and picked at random the errors

[5] `http://en.wiktionary.org`

[6] `http://www.gutenberg.org/browse/languages/en`

[7] `http://www.kilgarriff.co.uk/bnc-readme.html`

[8] We used a deprecated version of the Penn Treebank where the tag set includes compound tags like `VB|NN` in addition to the usual Penn Treebank tags. For the final version of the article we will rerun the tests on the regular WSJ corpus.

[9] `http://ota.oucs.ox.ac.uk/headers/0643.xml`

[10] `http://home.gna.org/omorfi`

[11] `http://download.wikipedia.org/fiwiki/`

[12] http://www.csc.fi/english/research/software/ftc

[13] `http://www.gutenberg.org/browse/languages/fi`

having frequencies in range 1 to 8 instances; a majority of higher frequency non-words were actually proper nouns or neologisms missing from the dictionary. Using all of Wikipedia, europarl and Gutenberg provides a reasonable variety of both contemporary and old texts in a wide range of styles.

4 Tests and Evaluation

The evaluation of the correction suggestion quality is described in Table 4. The Table 4 contains the precision for the spelling errors. The precision is measured by ranked suggestions. In the tables, we give the results separately for ranks 1—5, and then for the accumulated ranks 1—10. The rows of the table represent different combinations of the N-gram models. The first row is a baseline score achieved by the weighted edit distance model alone, and the second is with unigram-weighted dictionary over edit-distance 2. The last two columns are the traditional word-form N-gram model and our POS tagger based extension to it.

Table 4. Precision of suggestion algorithms with real spelling errors.

Algorithm	1	2	3	4	5	1—10
English						
Edit distance 2 (baseline)	25.9 %	2.4 %	2.4 %	1.2 %	3.5 %	94.1 %
Edit distance 2 with Unigrams	28.2 %	5.9 %	29.4 %	3.5%	28.2 %	97.6 %
Edit distance 2 with Word N-grams	29.4 %	10.6 %	34.1 %	5.9 %	14.1 %	97.7 %
Edit distance 2 with POS N-grams	68.2 %	18.8 %	3.5 %	2.4 %	0.0 %	92.9 %
Finnish						
Edit distance 2 (baseline)	66.5 %	8.7 %	4.0 %	4.7 %	1.9 %	89.8 %
Edit distance 2 with Unigrams	61.2 %	13.4 %	1.6 %	3.1 %	3.4 %	88.2 %
Edit distance 2 with Word N-grams	65.0 %	14.4 %	3.8 %	3.1 %	2.2 %	90.6 %
Edit distance 2 with POS N-grams	71.4 %	9.3 %	1.2 %	3.4 %	0.3 %	85.7 %

It would appear that POS N-grams will in both cases give a significant boost to the results, whereas the word-form N-grams will merely give a slight increase to the results. In next subsections we further disect the specific changes to results the different approaches give.

4.1 English Error-Analysis

In [10], the authors identify errors that are not solved using simple unigram weights, such as correcting *rember* to *remember* instead of *member*. Here, our scaled POS N-gram context-model as well as the simpler word N-gram model, which can bypass the edit distance model weight will select the correct suggestion. However, when correcting e.g. *ment* to *meant* in stead of *went* or *met* the POS based context reranking gives no help as the POS stays the same.

4.2 Finnish Error-Analysis

In Finnish results we can easily notice that variation within the first position in the baseline results and reference system is more sporadic. This can be traced back to the fuzz factor caused by a majority of probabilities falling into the same category in our tests. The same edit-distance and unigram probability leaves the decision to random factors irrelevant to this experiment, such as alphabetical ordering that comes from data structures backing up the program code. The N-gram based reorderings are the only methods that can tiebreak the results here.

An obvious improvement for Finnish with POS N-grams comes from correcting agreeing NP's towards case agreement, such as *yhdistetstä* to *yhdisteistä* ('of compounds' PL ELA) instead of the statistically more common *yhdisteestä* ('of compound' SG ELA). However, as with English, the POS information does fail to rerank cases where two equally rare word-forms with the same POS occur at the same edit distance, which seems to be common with participles, such as correcting *varustunut* to *varautunut* in stead of *varastanut.*

Furthermore we note that the the discourse particles that were dropped from the POS tagger's analysis tag set in order to decrease the tag set size will cause certain word forms in the dictionary to be incorrectly reranked, such as when correcting the very common misspelling *muillekkin* into *muillekokin* ('for others as well?' PL ALL QST KIN) instead of the originally correct *muillekin* ('for others as well' PL ALL KIN), since the analyses QST (for question enclitic) and KIN (for additive enclitic) are both dropped from the POS analyses.

4.3 Performance Evaluation

We did not work on optimizing the N-gram analysis and selection, but we found that the speed of the system is reasonable—even in its current form. Table 5 summarizes the average speed of performing the experiments in Table 4.

Table 5. The speed of ranking the errors.

Material Algorithm	English	Finnish
Unigram (baseline)	10.0 s	51.8 s
	399.1 wps	6.2 wps
POS N-grams	377.4 s	1616.2 s
	10.6 wps	0.14 wps

The performance penalty that is incurred on Finnish spell-checking but not so much on English comes from the method of determining readings for words unknown to the language model, i.e. from the guessing algorithm. The amount of words unknown to the language model in Finnish was greater than for English due to the training data sparseness and the morphological complexity of the language.

5 Future Work and Discussion

We have shown that the POS based N-gram models are suitable for improving the spelling corrections for both morphologically more complex languages such as Finnish and for further improving languages with simpler morphologies like English. To further verify the claim, the method may still need to be tested on a typologically wider spectrum of languages.

In this article, we used readily available and hand-made error corpora to test our error correction method. A similar method as the one we use for error correction should be possible in error detection as well, especially when detecting real-word errors [7]. In future research, an obvious development is to integrate the N-gram system as a part of a real spell-checker system for both detection and correction of spelling errors, as is already done for the unigram based spell checker demonstrated in [12].

The article discussed only the reranking over basic edit distance error models, further research should include more careful statistical training for the error model as well, such as one demonstrated in [1].

6 Conclusion

In this paper we have demonstrated the use of finite-state methods for trigram based generation of spelling suggestions. We have shown that the basic word-form trigram methods suggested for languages like English do not seem to be as useful without modification for morphologically more complex languages like Finnish. Instead a more elaborate N-gram scheme using POS n-grams is successful for Finnish as well as English.

Acknowledgements

We are grateful to Sam Hardwick for making the spell checking software available and the HFST research group for fruitful discussions. We also thank the anonymous reviewers for useful suggestions and pointers.

References

1. Brill, E., Moore, R.C.: An improved error model for noisy channel spelling correction. In: ACL '00: Proceedings of the 38th Annual Meeting on Association for Computational Linguistics. pp. 286–293. Association for Computational Linguistics, Morristown, NJ, USA (2000)
2. Church, K., Gale, W.: Probability scoring for spelling correction. Statistics and Computing pp. 93–103 (1991)

3. Damerau, F.J.: A technique for computer detection and correction of spelling errors. Commun. ACM (7) (1964)
4. Hakulinen, A., Vilkuna, M., Korhonen, R., Koivisto, V., Heinonen, T.R., Alho, I.: Iso suomen kielioppi (2008), referred on 31.12.2008, available from `http://kaino.kotus.fi/visk`
5. Kukich, K.: Techniques for automatically correcting words in text. ACM Comput. Surv. 24(4), 377–439 (1992)
6. Levenshtein, V.I.: Binary codes capable of correcting deletions, insertions, and reversals. Soviet Physics—Doklady 10, 707–710. Translated from Doklady Akademii Nauk SSSR pp. 845–848 (1966)
7. Mays, E., Damerau, F.J., Mercer, R.L.: Context based spelling correction. Inf. Process. Manage. 27(5), 517–522 (1991)
8. Mitton, R.: Ordering the suggestions of a spellchecker without using context*. Nat. Lang. Eng. 15(2), 173–192 (2009)
9. Mohri, M., Riley, M.: An efficient algorithm for the n-best-strings problem (2002)
10. Norvig, P.: How to write a spelling corrector. Web Page, Visited February 28th 2010, Available `http://norvig.com/spell-correct.html` (2010), `http://norvig.com/spell-correct.html`
11. Pirinen, T.A.: Modularisation of finnish finite-state language description—towards wide collaboration in open source development of a morphological analyser. In: Pedersen, B.S., Nešpore, G., Inguna Skadi n. (eds.) Nodalida 2011. NEALT Proceedings, vol. 11, pp. 299—302. NEALT (2011), `http://www.helsinki.fi/\%7Etapirine/publications/Pirinen-nodalida2011.pdf`
12. Pirinen, T.A., Lindén, K.: Finite-state spell-checking with weighted language and error models. In: Proceedings of the Seventh SaLTMiL workshop on creation and use of basic lexical resources for less-resourced languagages. pp. 13–18. Valletta, Malta (2010), `http://siuc01.si.ehu.es/\%7Ejipsagak/SALTMIL2010_Proceedings.pdf`
13. Savary, A.: Typographical nearest-neighbor search in a finite-state lexicon and its application to spelling correction. In: CIAA '01: Revised Papers from the 6th International Conference on Implementation and Application of Automata. pp. 251–260. Springer-Verlag, London, UK (2002)
14. Silfverberg, M., Lindén, K.: Combining statistical models for pos tagging using finite-state calculus. In: Proceedings of the 18th Conference on Computational Linguistics, NODALIDA 2011. pp. 183–190 (2011)