Multilingual Text Induced Spelling Correction

Martin REYNAERT
Induction of Linguistic Knowledge, Computational Linguistics and AI, Tilburg University
Warandelaan 2, 5000 LE Tilburg,
The Netherlands,
reynaert@uvt.nl

Abstract

We present TISC, a multilingual, language-independent and context-sensitive spelling checking and correction system designed to facilitate the automatic removal of non-word spelling errors in large corpora. Its lexicon is derived from raw text corpora, without supervision, and contains word unigrams and word bigrams. The system employs input context and lexicon evidence to automatically propose a limited number of ranked correction candidates. We describe the implemented trilingual (Dutch, English, French) prototype and evaluate it on English and Dutch text, monolingual and mixed, containing real-world errors in context.

1 Introduction

The EAGLES final report on 'Evaluation of Natural Language Processing Systems' lists as a 'dream tool' (EAGLES-I, 1996):

> A multilingual spelling checker which automatically recognizes what language is being dealt with and switches to the appropriate spelling checker for that language.

Our Text Induced Spelling Correction algorithm (TISC) represents such a tool in its current three language version, but we explore the possibility of *not* performing explicit language detection. This was prompted by the observation that language detection in an isolated-word system may easily get confused. Take the recent Dutch newspaper Metro headline 'Crime passionel in Gronings zwembad' [Crime of passion in Groninger swimming-pool (21-10-2003)], which is a typical example of mixed language text, containing a typo **passionel*, which in French should be spelled *passionnel*. The Microsoft Proofing Tools (MPT), for instance, can be set to automatically detect the language. Given the journalist is Dutch, it would typically have Dutch as its default language and so will not switch languages given the headline's first word *crime* is present in the Dutch dictionary, too. It will then encounter **passionel* and propose the correct, Dutch, forms: *passionele* and its lemma *passioneel*. Whereupon the journalist, not being too versant in French, is likely to let his original pseudo-French **passionel* stand. The Dutch part of the web provides many more instances of this same error, as does the English, for that matter. Our system being context-sensitive, we therefore explore whether its word bigrams alone aid the detection and correction of this kind of error, even when no further explicit language detection is done and no switching to another language dictionary occurs, its dictionary containing a mix of its various languages. In order to present our findings, we first describe our novel correction mechanism (section 2), explain how we effect detection in light of a noisy lexicon (section 4), derived from one or more language corpora (section 3) and present the evaluation results obtained on Dutch, English and mixed Dutch-English language texts (section 5).

2 The correction algorithm

We develop the idea of using the corpus itself as the basis on which to build a spelling correction system.

2.1 Anagram hashing

We line up all those word forms present in the corpus that consist of the same set of characters and use that as the basis for a corpus-derived lexicon. A means to do this in a completely unsupervised way was found in the theory of hashing, be it in the 'bad' part of it, in the normally avoided generation of collisions. Hashing has before been applied to spelling checking (Kukich, 1992), but we know of no prior work based on hash collisions. Collisions occur when

Anagram key	anagrams
75123219269	gerti, giert, griet, regit, riget, tiger, tigre
95176774701	ce tigre
95666874202	de griet, de tigre, dreig te, giert de, tigre de
107081254058	dreigt u, du tigre, it urged, u dertig, u dreigt, urged it
115780446077	de gierst, de tigres, gerst die, get rides, griste de, its greed, tigres de
127194825933	de rustig, drug ties, it surged, rustig de, surgit de, tigres du, urged its
129962785833	a stringed, and tigers, art design, dangers it, de ratings, de ratings, drang iets, gradins et, grand site, granted is, gratin des, is granted, its danger, its garden, rating des, ratings de, red giants, sign trade, tigers and, tigre dans

Table 1: Extract from a trilingual (English, Dutch, French) TISC lexicon with the anagram keys and associated, chained anagrams

the mathematical function used to bin the information, puts more than one item of information in a single bin (Knuth, 1981). The mathematically simple function introduced and exploited here does precisely that, for all strings containing the precise same set of characters.

So, for each word type or word type combination (compound or word bigram) to be included in the TISC lexicon, we obtain a numerical value, which will serve as the hash key. The formula represents the mathematical function we devised to do this, where f is a particular numerical value assigned to each character in the alphabet and c_1 to $c_{|w|}$ the actual characters in the input string w.

$$Key(w) = \sum_{i=1}^{|w|} f(c_i)^n$$

In practice, we use the ISO Latin-1 code value of each character in the string raised to a power n. We currently use 5 as the value for n. This was empirically derived, lower values do not produce collisions between anagrams only. The rather large natural number produced by this function in effect inflates the difference between any two characters to such a degree, that all strings containing the same set of characters receive the same natural number. This means that all anagrams, words consisting of a particular set of characters and present in the lexicon, will be identified through their common numerical value. So, in that the collisions produced by this function identify anagrams, we refer to this as an *anagram hash* and to the numerical values obtained as the *anagram keys*.

In the implementation the anagram keys and their associated word forms are stored in a regular hash. The anagram key will enable us to look up immediately whether any string consisting of the same character set as the input string was encountered in the corpus. When not present in the lexicon, close (numerical) neighbours might very well be present, and simple arithmetic will allow us to identify and retrieve these. This representation makes the implementation computationally tractable. The net effect of obtaining anagram hash key values is that it provides a cheap abstraction from the surface sequence of characters which further allows, through simple addition, subtraction or both, for moving from one particular combination of characters to another. The numerical difference between e.g. any verb possibly ending in *-ise* or *-ize* will always be the same. Subtracting the anagram key value of the *s*-variant from the anagram value of the *z*-variant will produce the same numerical result for all these pairs as does subtracting the anagram value for the single character *s* from the anagram value for *z*, namely: $z = 122^5 = 27,027,081,632$ and $s = 115^5 = 20,113,571,875$, difference: $27,027,081,632 - 20,113,571,875 = 6,913,509,757$. The numerical difference between e.g. *randomize* and *randomise* equals $136,483,404,939 - 129,569,895,182 = 6,913,509,757$. The same goes for all systematic spelling variations between e.g. American and British English or in probably any other alphabetic language.

2.2 Anagram key based correction

Anagram key based spelling correction is an inexpensive solution to the string correction problem as it does not entail expensive searching: it uses the non-search strategy implied in hashing. Based on a word form's anagram key it becomes possible to systematically query the lexicon for any variants present, be they morphological, typographical or orthographical. These variants can all be seen as variations of the usual taxonomy in terms of *trasnpositions, *deletons, *inserrtions or *substatutions (Damerau, 1964).

transpositions These we get for free: they have the same anagram key value, so when queried, the lexicon returns the correct form and its anagrams (if any).

deletions We iterate over the alphabet and query the lexicon for the input word anagram value plus each value from the alphabet.

insertions We iterate over the list of anagram values for the character unigrams and bigrams collected from the input type and

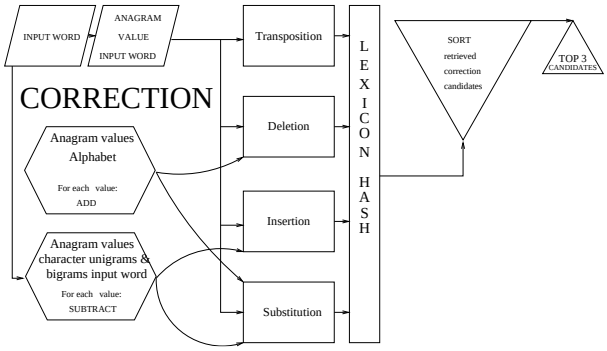

Figure 1: The correction module

query the lexicon for the input word anagram value minus each of these values.

substitutions We iterate over both lists adding each value from the one and subtracting each value of the second to the input word anagram value and repeatedly query the lexicon.

We thus retrieve all numerical near-neighbours (NNNs) from the lexicon and apply standard string matching techniques to retain those that either in front or back match the input type for a specific amount of characters, depending on the input type's length. After doing so, we iterate over the list of NNNs retained and upgrade the actual retrieval counts for those that have the greater substring matches and whose Levenshtein distance (LD)(Levenshtein, 1965) does not exceed 4 (the algorithm is not in itself limited to a particular LD). The elements of this list have thereby been ranked and the top n are then proposed as correction candidates. This ranking is an automatic side effect of the algorithm which produces more hits on the actual nearest NNN's. A deletion error, e.g. such as **cateory*, will return the correct 'category' on the basis of adding the anagram value for 'g' as well as of substituting the value for 'e' with that for 'eg' and substituting the value for 'o' with that for 'go'. The redundancy inherent to our algorithm thereby produces the desirable side-effect of converging on what is usually the best correction candidate by returning it more often than less likely candidates.

3 TISC corpus-derived components

3.1 The Lexicon

The English corpus we used was the New York Times (1994-2002) material available in the LDC Gigaword Corpus (NYT) (Graff, 2003). For Dutch we used both the ILK Corpus[1] and the Twente Corpus[2] (TWC). For French we used 8 years ('91-'98) of Roularta Magazines[3]. Statistics on these corpora are presented in table 2.

Corpus	NYT	ILK-TWC	ROULARTA
language	English	Dutch	French
tokens	1,106,376,695	681,686,340	52,722,253
bigrams	11,246,986	9,927,378	1,270,600
unigrams	672,502	861,604	144,943
keys/anagr.	10,287,826	9,000,131	794,308

Table 2: Statistics of NYT, ILK-TWENTE and ROULARTA corpora and lexicons. Bigrams with frq>2. Unigrams derived from these. French key-anagram ratio based on frq>4.

A TISC lexicon is derived from a large corpus of tokenised, but otherwise raw text, from which all XML or other tags have been discarded. We normalise the corpus by replacing all word-external punctuation by a single unique mark, as well as all digits and numbers by another. We apply a rule-based tokenizer and use the CMU Statistical Toolkit for deriving a bigram frequency list from the corpus (Clarkson and Rosenfeld, 1997). We discard the tail of the bigram list below a given threshold frequency, partly to ensure we do not incorporate the bulk of erroneous types present in the corpus. The effect of varying the threshold frequency is discussed in (Reynaert, 2004).

To make a multilingual version we concatenate the different languages' bigram lists at this point. Next the frequency information is discarded and a unigram list derived from the retained part of the bigram list. We lowercase the unigram list and concatenate the three lists obtained, removing any doubles. We finally compute the anagram key values for the unigram/bigram list. Together, the anagram keys and their lined-up unigrams or bigrams constitute the lexicon. Note that the lexicon will contain names and higher frequency errors.

3.2 The alphabet

Transformations on the word type to be evaluated are necessary in order to identify correction candidates. These transformations occur on the anagram key of the word type under consideration on the basis of numerical, i.e. anagram, values for the alphabet used, which are

[1] http://ilk.uvt.nl/ilkcorpus/

[2] http://wwwhome.cs.utwente.nl/~druid/TwNC/TwNC-main.html

[3] http://www.roularta.be/en/products/default.htm

read in at the start of run time. Our alphabet consists of the anagram key values for all character unigrams (e.g. a = 8,587,340,257, Z = 5,904,900,000) and character bigrams (e.g. ab = ba = 17,626,548,225, $i\ddot{e}$ = $\ddot{e}i$ = 729,465,962,500) we want to work with. The list currently contains 442 anagram values. These have been derived from character unigram and character bigram counts on the corpus.

3.3 The cooccurrence information

From the word bigram and unigram lists we derive cooccurrence information for all the word types present. For each word type we count the number of times it forms the:

- left part of a compound (LPC)
- right part of a compound (RPC)
- left part of a bigram (LPB)
- right part of a bigram (RPB)

Note that these cooccurrence counts (COOC) are counts on word types and not on word tokens. The COOC table contains only the counts per word-type, not the actual cooccurring word types.

4 TISC: the implementation

4.1 Zipf filters

Recall that Zipf stated that the frequency of a word is inversely proportional to its length (Zipf, 1935). This implies that we should expect to see more combinations of any given short word than of longer words. A long compound, e.g. one composed of three or more shorter words, cannot reasonably be expected to combine with very many more words. Short words can be expected to combine in a myriad of ways, be it as part of compounds or of numerous bigrams. It is this idea we exploit in what we would like to call the *Zipf Filters* implemented in our prototype. We make the number of expected cooccurrences of a word dependent on the length of the word form. This then allows to detect anomalies in the COOCs for particular word types. We posit a particular amount of times a string or substring is seen as sufficient to conclude the string is likely well-formed as it is highly productive. To this end we take a constant, which is higher for the shorter strings and lower beyond a particular amount of characters, divided by the number of characters in the string, or the string's length. We compare the COOCs of a string to be evaluated with the outcome of this calculation and accept the string as being well-formed when the COOCs are higher, reject and thus send on to the correction module, when lower.

4.2 Compound splitting

Given that a language such as Dutch to a large degree allows for compounding, any text may contain quite a number of previously unseen compounds. While iterating over the input word string to compute its anagram value, TISC repeatedly queries the lexicon to check for the presence of the substring handled so far. If this is successful for the string as a whole, the substrings, if any, which show the best balance between length and COOCs are stored with their anagram values. If no full parse was possible, the process is repeated from right to left and a decision made over both the left-right and right-left parses and the split deemed most usable stored. TISC proposes a single particular split to be further provided to the checking and correction modules. The implementation currently allows for only a split in a left and right part.

4.3 Checking

The input text is first fully analysed: anagram values are added to the type list, frequencies of types and their compounding parts tallied, track kept of how many times the type was capitalised, recurrent LPC's not in the lexicon stored. Then, all the types are sent to the spelling checking module. Since we cannot content ourselves with simply checking whether a type is present in the dictionary or not, we query the cooccurrence information table to see whether the particular type's COOCs conform to our expectation of how many times a type of the given length should have been incorporated in the lexicon, i.e. the expectancy level or threshold set by the Zipf filter. If this is the case, the type is not further evaluated, which we will refer to as 'let go' . If not, the COOCs for its LPC and the RPC are evaluated against the threshold. We do not, at this stage, want to risk to lose too many of the erroneous types, so the level of expectancy is set rather high. We simultaneously check whether perhaps the lexicon contains possible bigrams based on the type's anagram key value with the value for a space added. All the types which did not conform to the expected levels or were found to be present with an additional space, are further evaluated. Further checks are:

- extra-space cases: If it turns out the lexicon contains only the inverted form with

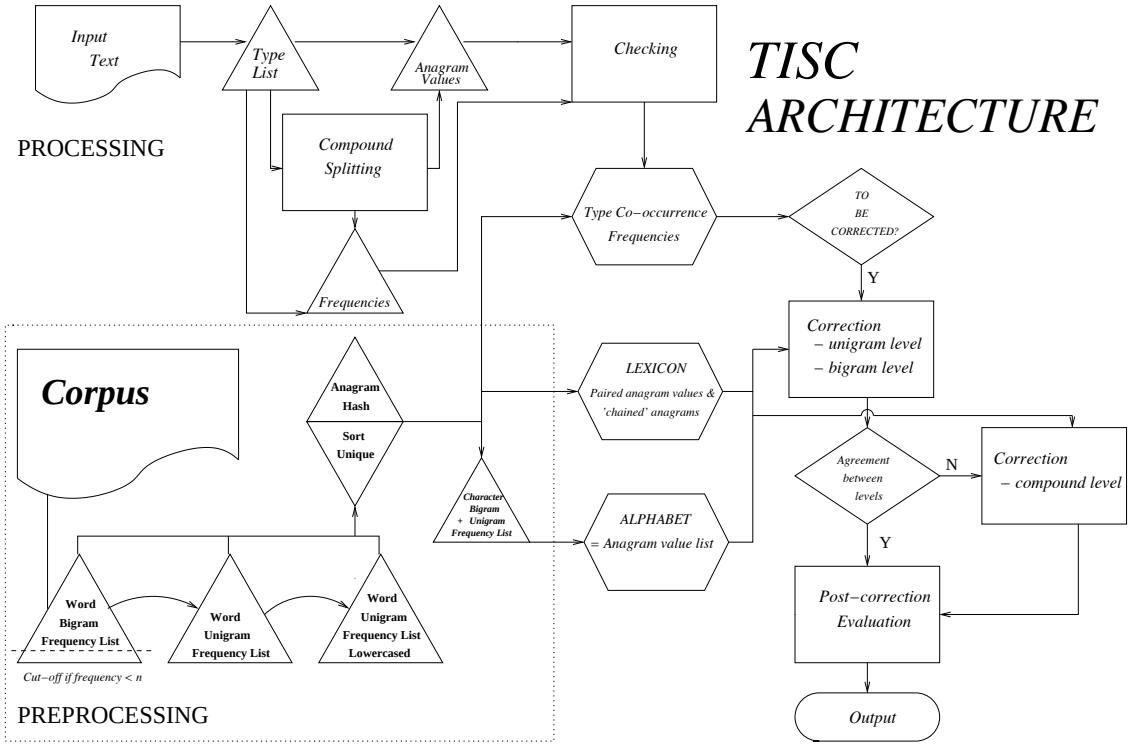

Figure 2: TISC's architecture

the added space (e.g. 'koffiebekertje' [coffee cup]: not in the lexicon, but 'bekertje koffie' [cup of coffee] is present), we accept the form as being correct, the rest are further evaluated.

- whether perhaps the LPC was seen in various other input text compounds or whether the RPC was perhaps seen as a word in its own right with a given frequency in the input text, the other part's COOCs conforming. Again those passing this test are let go.
- whether perhaps the COOCs for the LPC with first or all characters upper-cased conform to expectance.
- if the input type contains a dash, we check whether the COOCs for the type without the dash conform. Or perhaps whether the type without the dash but with an extra space is present in the lexicon.
- finally we check those forms for which the cooccurrence table contains no information at all. If the COOCs for their LPC and RPC exceed a high expectancy threshold, these are let go too.

All types not let go by one of these checks are sent on to the correction module.

4.4 Correction

By default, TISC's correction works on two levels, a third being invoked when these do not return satisfactory results. The unigram level consists of two tiers: unigram correction on the basis of the lexicon and on the basis of the list of input context derived types and compounding parts (with frequency threshold). On the bigram level, TISC performs context-dependent error correction, to some extent. It examines the 4 bigrams contained within a 2-1-2 window around the type in the input text (e.g. the green **bottel* was empty → the **bottel*, green **bottel*, **bottel* was, **bottel* empty). The only difference with the unigram correction module lies in the fact that for the 4 bigrams sent through the correction loop, all the correction candidates retrieved are stored in the same list. This produces more reliable counts after upgrading. After correction on these levels, the output candidates are compared and if both levels concur, i.e. the same candidate(s) were returned, they are accepted if they differ from the input type, or rejected (and 'let go') if not. When no output is returned by the unigram and bigram correction levels, or the results of these do not concur, the type is further checked on the third level, that of its substrings, i.e. the compounding parts returned by the compound

splitter. The compound correction level treats both LPC and RPC as words in their own right, queries the system for correction candidates in the same way as on the unigram level for both parts and finally concatenates the top candidates returned and proposes these as correction candidates. Given a sufficiently high frequency in the input text of the correct form for an incorrect compounding part, this may enable the system to correct the error even if the correct form is not present in the lexicon.

5 Evaluation

5.1 Evaluation method rationale

TISC ought to be compared to other context-sensitive spelling checking and correction systems applied to the task of detecting and correcting non-word errors. Alas, we know of none that have been evaluated on both detection and correction.

Brill and Moore have developed and evaluated an improved noisy channel-based correction system equipped with a language model, therefore context-sensitive, and reported state-of-the-art correction performance (Brill and Moore, 2000). They trained the system on 8,000 erroneous word forms. The system was given another 2000 erroneous word forms to correct under perfect conditions: all correct forms were present in the dictionary. They report an accuracy of 98.8% on the 3-best ranked correction candidates. We think this really constitutes the upper bound their system can reach, rather than its true accuracy. We get no idea of how this system would perform, if it were given both correct and incorrect words not available in the dictionary. In order to evaluate our system in the same way and in order for results to be comparable, we would have to be able to use the same 2,000 error list. This list does not seem to be available.

We therefore tried to next best thing, which is to try and see how an isolated-word spelling checking and correction system, which can easily be equipped with the same bi- and trilingual dictionaries as TISC, performs. ISPELL fulfils these requirements. Unfortunately, it does not perform ranking of the correction candidates. Either it sorts them alphabetically or not. This precludes reporting ranking scores here.

5.2 Test settings

We compare our results with those obtained by ISPELL (version 3.2.06) and MPT (version in Microsoft Office 2000, 9.0.3821 SR-1), as far as possible. For TISC and our trilingual version of ISPELL we varied the threshold at which the corpora's bigram lists were truncated (Frequencies: 4–10, 15, 20, 30, 40, 50 and 100). The TISC implementation used was the same for all tests as it contains no provisions specific to a particular language. For the monolingual tests both ISPELL and MPT were run with their standard US and standard Dutch dictionaries, the first in batch mode, the second manually emulating ISPELL's output for automatic evaluation purposes. For the multilingual test, we declined testing MPT's automatic language detection mode on the 145,100 token file. For both ISPELL and MPT we report the averaged scores of the three monolingual tests in contrast to the trilingual ISPELL and bi- and trilingual TISC test results.

5.3 Composition of the evaluation files

Statistics on the evaluation files are presented in table 3.

Dutch: For evaluation purposes, we proofread the Dutch version of the newspaper *Metro* and collected the non-word errors encountered (typically 0–4 a day). These were extracted from the online version[4] with the full article they appeared in. We used the first batch (Metro1) for development purposes. The second, similar, batch we reserved for testing purposes only (Metro2).

English: We manually collected 1093 erroneous types from the alphabetically sorted unigram frequency list of the Reuters Corpus (Lewis et al., 2003). We then extracted their contexts from the tokenized corpus. The context ran to the paragraph containing the error, as well as the paragraphs preceding and following it. We proofread these manually, which yielded another 105 errors. A preliminary Ispell run finally yielded another 24 overlooked errors. We ran our evaluations with these 1222 known errors. Statistics on the evaluation file are presented in table 3.

Dutch-English: For the bilingual tests, we concatenated both Metro files and the Reuters file and sorted the lines alphabetically, thereby obtaining a mixed language file.

5.4 Scoring and evaluation results

We measure performance in terms of the F-score. Given that the systems are presented

[4] http://www.metropoint.com/cgi-bin/WebObjects/Metropoint.woa/wa/default

	Metro1	Metro2	Reuters	Mixed
context	article	article	3 par.	mix
tokens	21,919	25,750	97,432	145,100
types	5,747	6,441	15,341	24,795
errors	129	123	1,222	1,474
error/type	2.25%	1.9%	8%	5.9%

Table 3: Statistics of the evaluation files

	Rec.	Prec.	F	frq
Dutch:				
MPT	0.66	0.1	0.17	-
ISPELL	0.60	0.07	0.12	-
TISC	**0.67**	0.60	**0.63**	5
TISC-BI	0.64	**0.61**	0.62	5
TISC-TRI	0.64	**0.61**	0.62	5
English:				
MPT	**0.94**	0.38	0.54	-
ISPELL	0.85	0.27	0.41	-
TISC	0.85	0.80	**0.82**	5
TISC-BI	0.81	**0.83**	**0.82**	4
TISC-TRI	0.84	0.81	**0.82**	5
Dutch-English:				
MPT-AVERAGE	0.74	0.19	0.3	-
ISPELL-AVERAGE	0.7	0.14	0.22	-
ISPELL-TRI	0.77	0.59	0.67	6
TISC-BI	**0.80**	0.77	0.78	6
TISC-TRI	0.79	**0.78**	**0.79**	5
D-E Upper bounds				
ISPELL-TRI-UPPER	0.84	0.63	0.72	5
TISC-TRI-UPPER	0.84	0.80	0.82	5

Table 4: Statistics of best test scores

with errors in a context, we do not solely measure their ability to correct incorrect forms (i.e. their accuracy), but also to discern between correct and incorrect input forms. Of the word forms for which correction candidates are returned, we check if the output contains the correct form. If so, the score for successful correction (recall) is augmented by one, no account being taken of the ranking of the correction candidates, because ISPELL does not have a ranking mechanism. For all the forms marked by ISPELL or MPT as 'not in the dictionary' the score for false positives (precision errors) is incremented by one. The same goes for those forms for which the systems return correction candidates, but where the correct one is missing. The results presented in table 4 were obtained on the word types, for all systems.

5.5 Discussion

Monolingual task: For both languages, TISC's lower thresholded lexicons consistently produce the highest precision. Recall rises as the threshold is set higher, to drop again, as does precision, with more and more information not being available. More context causes precision to drop: more words to be checked create more opportunity to report false positives. This is clearly demonstrated by the Dutch results, where the evaluation files contain a lower error to type ratio than the English one. The drop in precision given more context seems to us to be the main cause of current spelling checking systems not being able to attain automatic correction levels of performance, i.e. a level of precision where more errors would be removed than correct words erroneously replaced. The drop in recall for Dutch is certainly a result of its greater morphological diversity.

Bilingual task: Table 4 presents the best results on the bilingual English-Dutch correction task obtained by TISC and ISPELL with dictionaries based on the same bilingual (D-E) (BI) and trilingual (D-E-F) (TRI) bigram lists. These results are contrasted to the average of the monolingual results on the three evaluation sets obtained by ISPELL and MPT. A rather striking result is that ISPELL's performance is drastically improved by providing it with a much larger dictionary. The presence of names alone in the dictionary provided by us must account for the better part of the gain in precision.

We determined the upper bound for both trilingual systems by removing the errors present in the evaluation files from the bigram lists from which the lexicons were derived. Remember that the evaluation files were obtained from disjoint corpora, a number of these errors are therefore recurrent and may obtain relatively high frequencies. It can be seen that ISPELL with its simple dictionary look-up strategy is more sensitive to these than is TISC. This is a clear indication that TISC's error detection strategy based on COOCs and thresholds set by the Zipf filters works. TISC's main gain is due to its context-awareness and to its greater reach in terms of LD covered. So it corrects errors that are beyond ISPELL's scope, but still misses highly recurrent ones.

Simply mixing three languages seems to have no adverse effect on both TISC and ISPELL's capabilities of performing correction to these levels of performance. Nevertheless, the fact remains that this strategy entails that one particular type of errors will go undetected, namely those errors in a specific language that result in a valid word in one of the other languages in

this type of multilingual system. These would have to be called *bilingual or translingual confusables*. Our evaluation files happened to contain a few of them, e.g. polite which should have read 'politie' [police] in the Dutch evaluation set. The fact that these are a lot rarer than errors which do not form a valid word in any of the languages, obscures their effect. Note that these would throw a non-context-aware system which does attempt to do language detection off balance. We think context-awareness here too should help remedy this shortcoming of our non-language-detecting approach. Provided the error detection module is made to take into account the word bigram information in much the same way as the error correction module currently does, it should also be possible to detect these anomalies. And this may be a nice pointer to the way we should direct our future work, in that this at least hints at ways the harder task of detecting and remedying monolingual confusables (Kukich, 1992) may be tackled.

As a final note, we want to draw due attention to the fact, not overly stressed in the above, that we have developed a competitive spelling checking and correction system using nothing besides electronically available collections of text. For Dutch and English, of course, a great deal of natural language processing resources are available. We have deliberately ignored these, as there are a great many languages in this world for which little or no such resources have as yet been developed. The inexpensive approach outlined here, we hope, may help to remedy that.

6 Conclusion

We have presented TISC, a new algorithm for spelling checking and correction. We have outlined how the system is built up from large corpora of raw text. We have introduced a novel representation for lexical information which allows for an exact calculation of the difference between two character strings. Not only does this make the problem computationally tractable, it also allows for building a scaled system. We have shown that incorporating word bigrams, cooccurrence information about individual word types and context information derived from the input text, all combine to make multilingual spelling correction a competitive possibility. We have compared TISC with two state-of-the-art systems and shown that it outperforms both.

Acknowledgements

Heartfelt thanks to my supervisors Prof. Dr. Walter Daelemans and Dr. Antal van den Bosch for their trust and support, as well as to Dr. Sabine Buchholz for providing the tokenizer. This work was funded by the Netherlands Organisation for Scientific Research (NWO/FWO VNC 205-41-119).

References

E. Brill and R.C. Moore. 2000. An improved error model for noisy channel spelling correction. In *Proc. of the 38th Annual Meeting of the ACL*, pages 286–293.

P.R. Clarkson and R. Rosenfeld. 1997. Statistical language modeling using the CMU-Cambridge toolkit. In *Proceedings ESCA Eurospeech 1997*.

Fred J. Damerau. 1964. A technique for computer detection and correction of spelling errors. *Communications of the ACM*, Volume 7, Issue 3 (March 1964):171 – 176.

EAGLES-I. 1996. Final Report. In *Evaluation of Natural Language Processing Systems*, volume EAGLES DOCUMENT EAG-EWG-PR.2.

David Graff. 2003. The New York Times Newswire Service. *English Gigaword LDC-2003T05*.

Donald E. Knuth, 1981. *Sorting and Searching*, volume 2 of *The Art of Computer Programming*, section 6.4, pages 513–558. Addison-Wesley, Reading, Massachusetts, second edition.

Karen Kukich. 1992. Techniques for automatically correcting words in text. *ACM Computing Surveys*, 24(4):377–439.

V.I. Levenshtein. 1965. Binary codes capable of correcting deletions, insertions, and reversals. In *Cybernetics and Control Theory*, volume 10(8), pages 707–710. Original in: Doklady Nauk SSSR 163(4): 845–848 (1965).

D. Lewis, Y.Yang, T.G. Rose, and F. Li. 2003. RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*.

Martin Reynaert. 2004. Text induced spelling correction. In *Proceedings COLING 2004, Geneva*.

George Kingsley Zipf. 1935. *The psycho-biology of language: an introduction to dynamic philology*. The M.I.T. Press, Cambridge, MA, 1965 - 2nd. edition.