Using the Web for Language Independent Spellchecking and Autocorrection

Casey Whitelaw and **Ben Hutchinson** and **Grace Y Chung** and **Gerard Ellis**

Google Inc.

Level 5, 48 Pirrama Rd, Pyrmont NSW 2009, Australia

`{whitelaw,benhutch,gracec,ged}@google.com`

Abstract

We have designed, implemented and evaluated an end-to-end system spellchecking and autocorrection system that does not require *any* manually annotated training data. The World Wide Web is used as a large noisy corpus from which we infer knowledge about misspellings and word usage. This is used to build an error model and an n-gram language model. A small secondary set of news texts with artificially inserted misspellings are used to tune confidence classifiers. Because no manual annotation is required, our system can easily be instantiated for new languages. When evaluated on human typed data with real misspellings in English and German, our web-based systems outperform baselines which use candidate corrections based on hand-curated dictionaries. Our system achieves 3.8% total error rate in English. We show similar improvements in preliminary results on artificial data for Russian and Arabic.

1 Introduction

Spellchecking is the task of predicting which words in a document are misspelled. These predictions might be presented to a user by underlining the misspelled words. Correction is the task of substituting the well-spelled hypotheses for misspellings. Spellchecking and autocorrection are widely applicable for tasks such as word-processing and postprocessing Optical Character Recognition. We have designed, implemented and evaluated an end-to-end system that performs spellchecking and autocorrection.

The key novelty of our work is that the system was developed entirely without the use of manually annotated resources or any explicitly compiled dictionaries of well-spelled words. Our multi-stage system integrates knowledge from statistical error models and language models (LMs) with a statistical machine learning classifier. At each stage, data are required for training models and determining weights on the classifiers. The models and classifiers are all automatically trained from frequency counts derived from the Web and from news data. System performance has been validated on a set of human typed data. We have also shown that the system can be rapidly ported across languages with very little manual effort.

Most spelling systems today require some hand-crafted language-specific resources, such as lexica, lists of misspellings, or rule bases. Systems using statistical models require large annotated corpora of spelling errors for training. Our statistical models require no annotated data. Instead, we rely on the Web as a large noisy corpus in the following ways. 1) We infer information about misspellings from term usage observed on the Web, and use this to build an error model. 2) The most frequently observed terms are taken as a noisy list of potential candidate corrections. 3) Token n-grams are used to build an LM, which we use to make context-appropriate corrections. Because our error model is based on scoring sub-strings, there is no fixed lexicon of well-spelled words to determine misspellings. Hence, both novel misspelled or well-spelled words are allowable. Moreover, in combination with an n-gram LM component, our system can detect and correct real-word substitutions, ie, word usage and grammatical errors.

Confidence classifiers determine the thresholds for spelling error detection and autocorrection, given error and LM scores. In order to train these classifiers, we require some textual content with some misspellings and corresponding well-spelled words. A small subset of the Web data from news pages are used because we assume they contain

890

relatively few misspellings. We show that confidence classifiers can be adequately trained and tuned without real-world spelling errors, but rather with clean news data injected with artificial misspellings.

This paper will proceed as follows. In Section 2, we survey related prior research. Section 3 describes our approach, and how we use data at each stage of the spelling system. In experiments (Section 4), we first verify our system on data with artificial misspellings. Then we report performance on data with real typing errors in English and German. We also show preliminary results from porting our system to Russian and Arabic.

2 Related Work

Spellchecking and correction are among the oldest text processing problems, and many different solutions have been proposed (Kukich, 1992). Most approaches are based upon the use of one or more manually compiled resources. Like most areas of natural language processing, spelling systems have been increasingly empirical, a trend that our system continues.

The most direct approach is to model the causes of spelling errors directly, and encode them in an algorithm or an error model. Damerau-Levenshtein edit distance was introduced as a way to detect spelling errors (Damerau, 1964). Phonetic indexing algorithms such as Metaphone, used by GNU Aspell (Atkinson, 2009), repesent words by their approximate 'soundslike' pronunciation, and allow correction of words that appear orthographically dissimilar. Metaphone relies upon data files containing phonetic information. Linguistic intuition about the different causes of spelling errors can also be represented explicitly in the spelling system (Deorowicz and Ciura, 2005).

Almost every spelling system to date makes use of a lexicon: a list of terms which are treated as 'well-spelled'. Lexicons are used as a source of corrections, and also to filter words that should be ignored by the system. Using lexicons introduces the distinction between 'non-word' and 'real-word' errors, where the misspelled word is another word in the lexicon. This has led to the two sub-tasks being approached separately (Golding and Schabes, 1996). Lexicon-based approaches have trouble handling terms that do not appear in the lexicon, such as proper nouns, foreign terms, and neologisms, which can account for a large proportion of 'non-dictionary' terms (Ahmad and Kondrak, 2005).

A word's context provides useful evidence as to its correctness. Contextual information can be represented by rules (Mangu and Brill, 1997) or more commonly in an n-gram LM. Mays et al (1991) used a trigram LM and a lexicon, which was shown to be competitive despite only allowing for a single correction per sentence (Wilcox-O'Hearn et al., 2008). Cucerzan and Brill (2004) claim that an LM is much more important than the channel model when correcting Web search queries. In place of an error-free corpus, the Web has been successfully used to correct real-word errors using bigram features (Lapata and Keller, 2004). This work uses pre-defined confusion sets.

The largest step towards an automatically trainable spelling system was the statistical model for spelling errors (Brill and Moore, 2000). This replaces intuition or linguistic knowledge with a training corpus of misspelling errors, which was compiled by hand. This approach has also been extended to incorporate a pronunciation model (Toutanova and Moore, 2002).

There has been recent attention on using Web search query data as a source of training data, and as a target for spelling correction (Yang Zhang and Li, 2007; Cucerzan and Brill, 2004). While query data is a rich source of misspelling information in the form of query-revision pairs, it is not available for general use, and is not used in our approach.

The dependence upon manual resources has created a bottleneck in the development of spelling systems. There have been few language-independent, multi-lingual systems, or even systems for languages other than English. Language-independent systems have been evaluated on Persian (Barari and QasemiZadeh, 2005) and on Arabic and English (Hassan et al., 2008). To our knowledge, there are no previous evaluations of a language-independent system across many languages, for the full spelling correction task, and indeed, there are no pre-existing standard test sets for typed data with real errors and language context.

3 Approach

Our spelling system follows a noisy channel model of spelling errors (Kernighan et al., 1990). For an observed word w and a candidate correction s, we compute $P(s|w)$ as $P(w|s) \times P(s)$.
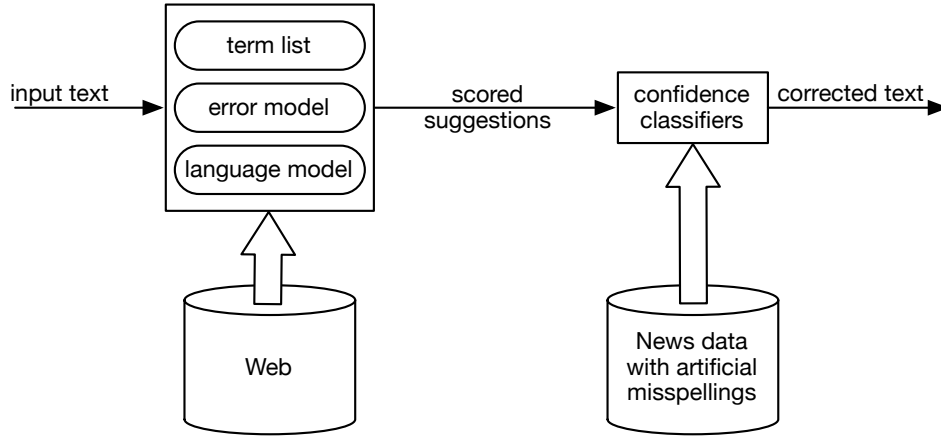
Figure 1: Spelling process, and knowledge sources used.

The text processing workflow and the data used in building the system are outlined in Figure 1 and detailed in this section. For each token in the input text, candidate suggestions are drawn from the term list (Section 3.1), and scored using an error model (Section 3.2). These candidates are evaluated in context using an LM (Section 3.3) and re-ranked. For each token, we use classifiers (Section 3.4) to determine our confidence in whether a word has been misspelled and if so, whether it should be autocorrected to the best-scoring suggestion available.

3.1   Term List

We require a list of terms to use as candidate corrections. Rather than attempt to build a lexicon of words that are well-spelled, we instead take the most frequent tokens observed on the Web. We used a large (> 1 billion) sample of Web pages, tokenized them, and took the most frequently occurring ten million tokens, with very simple filters for non-words (too much punctuation, too short or long). This term list is so large that it should contain most well-spelled words, but also a large number of non-words or misspellings.

3.2   Error Model

We use a substring error model to estimate $P(w|s)$. To derive the error model, let R be a partitioning of s into adjacent substrings, and similarly let T be a partitioning of w, such that $|T| = |R|$. The partitions are thus in one-to-one alignment, and by allowing partitions to be empty, the alignment models insertions and deletions of

substrings. Brill and Moore estimate $P(w|s)$ as follows:

$$P(w|s) \approx \max_{R,\ T\ s.t.\ |T|=|R|} \prod_{i=1}^{|R|} P(T_i|R_i) \quad (1)$$

Our system restricts partitionings that have substrings of length at most 2.

To train the error model, we require triples of (intended_word, observed_word, count), which are described below. We use maximum likelihood estimates of $P(T_i|R_i)$.

3.2.1   Using the Web to Infer Misspellings

To build the error model, we require as training data a set of (intended_word, observed_word, count) triples, which is compiled from the World Wide Web. Essentially the triples are built by starting with the term list, and a process that automatically discovers, from that list, putative pairs of spelled and misspelled words, along with their counts.

We believe the Web is ideal for compiling this set of triples because with a vast amount of user-generated content, we believe that the Web contains a representative sample of both well-spelled and misspelled text. The triples are not used directly for proposing corrections, and since we have a substring model, they do not need to be an exhaustive list of spelling mistakes.

The procedure for finding and updating counts for these triples also assumes that 1) misspellings tend to be orthographically similar to the intended word; Mays et al (1991) observed that 80% of

892

misspellings derived from single instances of insertion, deletion, or substitution; and 2) words are usually spelled as intended.

For the error model, we use a large corpus (up to 3.7×10^8 pages) of crawled public Web pages. An automatic language-identification system is used to identify and filter pages for the desired language. As we only require a small window of context, it would also be possible to use an n-gram collection such as the Google Web 1T dataset.

Finding Close Words. For each term in the term list (defined in Section 3.1), we find all other terms in the list that are "close" to it. We define closeness using Levenshtein-Damerau edit distance, with a conservative upper bound that increases with word length (one edit for words of up to four characters, two edits for up to twelve characters, and three for longer words). We compile the term list into a trie-based data structure which allows for efficient searching for all terms within a maximum edit distance. The computation is 'embarassingly parallel' and hence easily distributable. In practice, we find that this stage takes tens to hundreds of CPU-hours.

Filtering Triples. At this stage, for each term we have a cluster of orthographically similar terms, which we posit are potential misspellings. The set of pairs is reflexive and symmetric, e.g. it contains both (*recieve*, *receive*) and (*receive*, *recieve*). The pairs will also include e.g. (*deceive*, *receive*). On the assumption that words are spelled correctly more often than they are misspelled, we next filter the set such that the first term's frequency is at least 10 times that of the second term. This ratio was chosen as a conservative heuristic filter.

Using Language Context. Finally, we use the contexts in which a term occurs to gather directional weightings for misspellings. Consider a term w; from our source corpus, we collect the set of contexts $\{c_i\}$ in which w occurs. The definition of a context is relatively arbitrary; we chose to use a single word on each side, discarding contexts with fewer than a total of ten observed occurrences. For each context c_i, candidate "intended" terms are w and w's close terms (which are at least 10 times as frequent as w). The candidate which appears in context c_i the most number of times is deemed to be the term intended by the user in that context.

The resulting dataset consists of triples of the original observed term, one of the "intended" terms as determined by the above algorithm, and the number of times this term was intended. For a single term, it is possible (and common) to have multiple possible triples, due to the context-based assignment.

Inspecting the output of this training process shows some interesting patterns. Overall, the dataset is still noisy; there are many instances where an obviously misspelled word is not assigned a correction, or only some of its instances are. The dataset contains around 100 million triples, orders of magnitude larger than any manually compiled list of misspellings . The kinds of errors captured in the dataset include stereotypical spelling errors, such as *acomodation*, but also OCR-style errors. *computationaUy* was detected as a misspelling of *computationally* where the 'U' is an OCR error for 'll'; similarly, *Postmodem* was detected as a misspelling of *Postmodern* (an example of 'keming').

The data also includes examples of 'real-word' errors. For example, 13% of occurrences of *occidental* are considered misspellings of *accidental*; contrasting with 89% of occurrences of the non-word *accidential*. There are many examples of terms that would not be in a normal lexicon, including neologisms (*mulitplayer* for *multiplayer*), companies and products (*Playstaton* for *Playstation*), proper nouns (*Schwarznegger* for *Schwarzenegger*) and internet domain names (*mysapce.com* for *myspace.com*).

3.3 Language Model

We estimate $P(s)$ using n-gram LMs trained on data from the Web, using Stupid Backoff (Brants et al., 2007). We use both forward and backward context, when available. Contrary to Brill and Moore (2000), we observe that user edits often have both left and right context, when editing a document.

When combining the error model scores with the LM scores, we weight the latter by taking their λ'th power, that is

$$P(w|s) * P(s)^{\lambda} \qquad (2)$$

The parameter λ reflects the relative degrees to which the LM and the error model should be trusted. The parameter λ also plays the additional role of correcting our error model's misestimation of the rate at which people make errors. For example, if errors are common then by increasing λ we

can reduce the value of $P(w|w) * P(w)^\lambda$ relative to $\sum_{s \neq w} P(s|w) * P(s)^\lambda$.

We train λ by optimizing the average inverse rank of the correct word on our training corpus, where the rank is calculated over all suggestions that we have for each token.

During initial experimentation, it was noticed that our system predicted many spurious autocorrections at the beginnings and ends of sentences (or in the case of sentence fragments, the end of the fragment). We hypothesized that we were weighting the LM scores too highly in such cases. We therefore conditioned λ on how much context was available, obtaining values $\lambda_{i,j}$ where i, j represent the amount of context available to the LM to the left and right of the current word. i and j are capped at n, the order of the LM.

While conditioning λ in this way might at first appear ad hoc, it has a natural interpretation in terms of our confidence in the LM. When there is no context to either side of a word, the LM simply uses unigram probabilities, and this is a less trustworthy signal than when more context is available.

To train $\lambda_{i,j}$ we partition our data into bins corresponding to pairs i, j and optimize each $\lambda_{i,j}$ independently.

Training a constant λ, a value of 5.77 was obtained. The conditioned weights $\lambda_{i,j}$ increased with the values of i and j, ranging from $\lambda_{0,0} = 0.82$ to $\lambda_{4,4} = 6.89$. This confirmed our hypothesis that the greater the available context the more confident our system should be in using the LM scores.

3.4 Confidence Classifiers for Checking and Correction

Spellchecking and autocorrection were implemented as a three stage process. These employ confidence classifiers whereby precision-recall tradeoffs could be tuned to desirable levels for both spellchecking and autocorrection.

First, all suggestions s for a word w are ranked according to their $P(s|w)$ scores. Second, a spellchecking classifier is used to predict whether w is misspelled. Third, if w is both predicted to be misspelled and s is non-empty, an autocorrection classifier is used to predict whether the top-ranked suggestion is correct.

The spellchecking classifier is implemented using two embedded classifiers, one of which is used when s is empty, and the other when it is non-empty. This design was chosen because the useful signals for predicting whether a word is misspelled might be quite different when there are no suggestions available, and because certain features are only applicable when there are suggestions.

Our experiments will compare two classifier types. Both rely on training data to determine threshold values and training weights.

A "simple" classifier which compares the value of $log(P(s|w)) - log(P(w|w))$, for the original word w and the top-ranked suggestion s, with a threshold value. If there are no suggestions other than w, then the $log(P(s|w))$ term is ignored.

A logistic regression classifier that uses five feature sets. The first set is a scores feature that combines the following scoring information (i) $log(P(s|w)) - log(P(w|w))$ for top-ranked suggestion s. (ii) LM score difference between the original word w and the top suggestion s. (iii) $log(P(s|w)) - log(P(w|w))$ for second top-ranked suggestion s. (iv) LM score difference between w and second top-ranked s. The other four feature sets encode information about case signatures, number of suggestions available, the token length, and the amount of left and right context.

Certain categories of tokens are blacklisted, and so never predicted to be misspelled. These are numbers, punctuation and symbols, and single-character tokens.

The training process has three stages. (1) The context score weighting is trained, as described in Section 3.3. (2) The spellchecking classifier is trained, and tuned on held-out development data. (3) The autocorrection classifier is trained on the instances with suggestions that the spellchecking classifier predicts to be misspelled, and it too is tuned on held-out development data.

In the experiments reported in this paper, we trained classifiers so as to maximize the F_1-score on the development data. We note that the desired behaviour of the spellchecking and autocorrection classifiers will differ depending upon the application, and that it is a strength of our system that these can be tuned independently.

3.4.1 Training Using Artificial Data

Training and tuning the confidence classifiers require supervised data, in the form of pairs of misspelled and well-spelled documents. And indeed we posit that relatively noiseless data are needed to train robust classifiers. Since these data are

Language	Sentences	
	Train	Test
English	116k	58k
German	87k	44k
Arabic	8k	4k
Russian	8k	4k

Table 1: Artificial data set sizes. The development set is approximately the same size as the training set.

not generally available, we instead use a clean corpus into which we artificially introduce misspellings. While this data is not ideal, we show that in practice it is sufficient, and removes the need for manually-annotated gold-standard data.

We chose data from news pages crawled from the Web as the original, well-spelled documents. We chose news pages as an easily identifiable source of text which we assume is almost entirely well-spelled. Any source of clean text could be used. For each language the news data were divided into three non-overlapping data sets: the training and development sets were used for training and tuning the confidence classifiers, and a test set was used to report evaluation results. The data set sizes, for the languages used in this paper, are summarized in Table 1.

Misspelled documents were created by artificially introducing misspelling errors into the well-spelled text. For all data sets, spelling errors were randomly inserted at an average rate of 2 per hundred characters, resulting in an average word misspelling rate of 9.2%. With equal likelihood, errors were either character deletions, transpositions, or insertions of randomly selected characters from within the same document.

4 Experiments

4.1 Typed Data with Real Errors

In the absence of user data from a real application, we attempted our initial evaluation with typed data via a data collection process. Typed data with real errors produced by humans were collected. We recruited subjects from our coworkers, and asked them to use an online tool customized for data collection. Subjects were asked to randomly select a Wikipedia article, copy and paste several text-only paragraphs into a form, and retype those paragraphs into a subsequent form field. The subjects were asked to pick an article about a favorite city or town. The subjects were asked to type

at a normal pace avoiding the use of backspace or delete buttons. The data were tokenized, automatically segmented into sentences, and manually preprocessed to remove certain gross typing errors. For instance, if the typist omitted entire phrases/sentences by mistake, the sentence was removed. We collected data for English from 25 subjects, resulting in a test set of 11.6k tokens, and 495 sentences. There were 1251 misspelled tokens (10.8% misspelling rate.)

Data were collected for German Wikipedia articles. We asked 5 coworkers who were German native speakers to each select a German article about a favorite city or town, and use the same online tool to input their typing. For some typists who used English keyboards, they typed ASCII equivalents to non-ASCII characters in the articles. This was accounted for in the preprocessing of the articles to prevent misalignment. Our German test set contains 118 sentences, 2306 tokens with 288 misspelled tokens (12.5% misspelling rate.)

4.2 System Configurations

We compare several system configurations to investigate each component's contribution.

4.2.1 Baseline Systems Using Aspell

Systems 1 to 4 have been implemented as baselines. These use GNU Aspell, an open source spell checker (Atkinson, 2009), as a suggester component plugged into our system instead of our own Web-based suggester. Thus, with Aspell, the suggestions and error scores proposed by the system would all derive from Aspell's handcrafted custom dictionary and error model. (We report results using the best combination of Aspell's parameters that we found.)

System 1 uses Aspell tuned with the logistic regression classifier. System 2 adds a context-weighted LM, as per Section 3.3, and uses the "simple" classifier described in Section 3.4. System 3 replaces the simple classifier with the logistic regression classifier. System 4 is the same but does not perform blacklisting.

4.2.2 Systems Using Web-based Suggestions

The Web-based suggester proposes suggestions and error scores from among the ten million most frequent terms on the Web. It suggests the 20 terms with the highest values of $P(w|s) \times f(s)$ using the Web-derived error model.

Systems 5 to 8 correspond with Systems 1 to 4, but use the Web-based suggestions instead of Aspell.

4.3 Evaluation Metrics

In our evaluation, we aimed to select metrics that we hypothesize would correlate well with real performance in a word-processing application. In our intended system, misspelled words are autocorrected when confidence is high and misspelled words are flagged when a highly confident suggestion is absent. This could be cast as a simple classification or retrieval task (Reynaert, 2008), where traditional measures of precision, recall and F metrics are used. However we wanted to focus on metrics that reflect the quality of end-to-end behavior, that account for the combined effects of flagging and automatic correction. Essentially, there are three states: a word could be unchanged, flagged or corrected to a suggested word. Hence, we report on error rates that measure the errors that a user would encounter if the spellchecking/autocorrection were deployed in a word-processor. We have identified 5 types of errors that a system could produce:

1. E_1: A misspelled word is wrongly corrected.

2. E_2: A misspelled word is not corrected but is flagged.

3. E_3: A misspelled word is not corrected or flagged.

4. E_4: A well spelled word is wrongly corrected.

5. E_5: A well spelled word is wrongly flagged.

It can be argued that these errors have varying impact on user experience. For instance, a well spelled word that is wrongly corrected is more frustrating than a misspelled word that is not corrected but is flagged. However, in this paper, we treat each error equally.

E_1, E_2, E_3 and E_4 pertain to the correction task. Hence we can define Correction Error Rate (CER):

$$\text{CER} = \frac{E_1 + E_2 + E_3 + E_4}{T}$$

where T is the total number of tokens. E_3 and E_5 pertain to the nature of flagging. We define Flagging Error Rate (FER) and Total Error Rate (TER):

$$\text{FER} = \frac{E_3 + E_5}{T}$$

$$\text{TER} = \frac{E_1 + E_2 + E_3 + E_4 + E_5}{T}$$

For each system, we computed a No Good Suggestion Rate (NGS) which represents the proportion of misspelled words for which the suggestions list did not contain the correct word.

5 Results and Discussion

5.1 Experiments with Artificial Errors

System	TER	CER	FER	NGS
1. Aspell, no LM, LR	17.65	6.38	12.35	18.3
2. Aspell, LM, Sim	4.82	2.98	2.86	18.3
3. Aspell, LM, LR	4.83	2.87	2.84	18.3
4. Aspell, LM, LR (no blacklist)	22.23	2.79	19.89	16.3
5. WS, no LM, LR	9.06	7.64	6.09	10.1
6. WS, LM, Sim	2.62	2.26	1.43	10.1
7. WS, LM, LR	**2.55**	**2.21**	**1.29**	10.1
8. WS, LM, LR (no blacklist)	21.48	2.21	19.75	8.9

Table 2: Results for English news data on an independent test set with artificial spelling errors. Numbers are given in percentages. LM: Language Model, Sim: Simple, LR: Logistic Regression, WS: Web-based suggestions. NGS: No good suggestion rate.

Results on English news data with artificial spelling errors are displayed in Table 2. The systems which do not employ the LM scores perform substantially poorer that the ones with LM scores. The Aspell system yields a total error rate of 17.65% and our system with Web-based suggestions yields TER of 9.06%.

When comparing the simple scorer with the logistic regression classifier, the Aspell Systems 2 and 3 generate similar performances while the confidence classifier afforded some gains in our Web-based suggestions system, with total error reduced from 2.62% to 2.55%. The ability to tune each phase during development has so far proven more useful than the specific features or classifier used. Blacklisting is crucial as seen by our results for Systems 4 and 8. When the blacklisting mechanism is not used, performance steeply declines.

When comparing overall performance for the data between the Aspell systems and the Web-based suggestions systems, our Web-based suggestions fare better across the board for the news data with artificial misspellings. Performance

gains are evident for each error metric that was examined. Total error rate for our best system (System 7) reduces the error of the best Aspell system (System 3) by 45.7% (from 4.83% to 2.62%). In addition, our no good suggestion rate is only 10% compared to 18% in the Aspell system. Even where no LM scores are used, our Web-based suggestions system outperforms the Aspell system.

The above results suggest that the Web-based suggestions system performs at least as well as the Aspell system. However, it must be highlighted that results on the test set with artificial errors does not guarantee similar performance on real user data. The artificial errors were generated at a systematically uniform rate, and are not modeled after real human errors made in real word-processing applications. We attempt to consider the impact of real human errors on our systems in the next section.

5.2 Experiments with Human Errors

System	TER	CER	FER	NGS
English Aspell	4.58	**3.33**	2.86	23.0
English WS	**3.80**	3.41	**2.24**	17.2
German Aspell	14.09	10.23	5.94	44.4
German WS	**9.80**	**7.89**	**4.55**	32.3

Table 3: Results for Data with Real Errors in English and German.

Results for our system evaluated on data with real misspellings in English and in German are shown in Table 3. We used the systems that performed best on the artificial data (System 3 for Aspell, and System 7 for Web suggestions). The misspelling error rates of the test sets were 10.8% and 12.5% respectively, higher than those of the artificial data which were used during development. For English, the Web-based suggestions resulted in a 17% improvement (from 4.58% to 3.80%) in total error rate, but the correction error rate was slightly (2.4%) higher.

By contrast, in German our system improved total error by 30%, from 14.09% to 9.80%. Correction error rate was also much lower in our German system, comparing 7.89% with 10.23% for the Aspell system. The no good suggestion rates for the real misspelling data are also higher than that of the news data. Our suggestions are limited to an edit distance of 2 with the original, and it was found that in real human errors, the average edit distance of misspelled words is 1.38 but

for our small data, the maximum edit distance is 4 in English and 7 in German. Nonetheless, our no good suggestion rates (17.2% and 32.3%) are much lower than those of the Aspell system (23% and 44%), highlighting the advantage of not using a hand-crafted lexicon.

Our results on real typed data were slightly worse than those for the news data. Several factors may account for this. (1) While the news data test set does not overlap with the classifier training set, the nature of the content is similar to the train and dev sets in that they are all news articles from a one week period. This differs substantially from Wikipedia article topics that were generally about the history and sights a city. (2) Second, the method for inserting character errors (random generation) was the same for the news data sets while the real typed test set differed from the artificial errors in the training set. Typed errors are less consistent and error rates differed across subjects. More in depth study is needed to understand the nature of real typed errors.
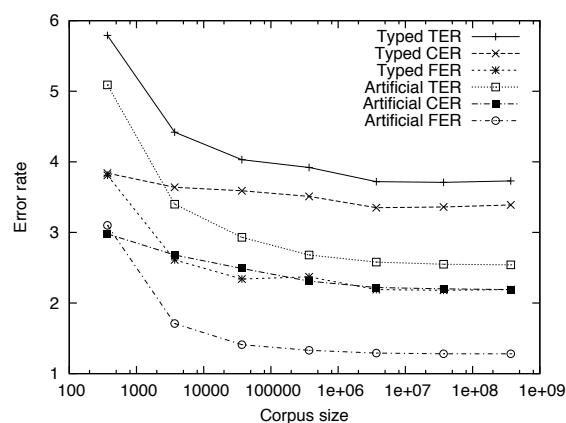

Figure 2: Effect of corpus size used to train the error model.

5.3 Effect of Web Corpus Size

To determine the effects of the corpus size on our automated training, we evaluated System 7 using error models trained on different corpus sizes. We used corpora containing $10^3, 10^4, \ldots, 10^9$ Web pages. We evaluated on the data set with real errors. On average, about 37% of the pages in our corpus were in English. So the number of pages we used ranged from about 370 to about 3.7×10^8. As shown in Figure 2, the gains are small after about 10^6 documents.

5.4 Correlation across data sets

We wanted to establish that performance improvement on the news data with artificial errors are likely to lead to improvement on typed data with real errors. The seventeen English systems reported in Table 3, Table 2 and Figure 2 were each evaluated on both English test sets. The rank correlation coefficient between total error rates on the two data sets was high ($\tau = 0.92$; $p < 5 \times 10^{-6}$). That is, if one system performs better than another on our artificial spelling errors, then the first system is very likely to also perform better on real typing errors.

5.5 Experiments with More Languages

System	TER	CER	FER	NGS
German Aspell	8.64	4.28	5.25	29.4
German WS	4.62	3.35	2.27	16.5
Arabic Aspell	11.67	4.66	8.51	25.3
Arabic WS	4.64	3.97	2.30	15.9
Russian Aspell	16.75	4.40	13.11	40.5
Russian WS	3.53	2.45	1.93	15.2

Table 4: Results for German, Russian, Arabic news data.

Our system can be trained on many languages with almost no manual effort. Results for German, Arabic and Russian news data are shown in Table 4. Performance improvements by the Web suggester over Aspell are greater for these languages than for English. Relative performance improvements in total error rates are 47% in German, 60% in Arabic and 79% in Russian. Differences in no good suggestion rates are also very pronounced between Aspell and the Web suggester.

It cannot be assumed that the Arabic and Russian systems would perform as well on real data. However the correlation between data sets reported in Section 5.4 lead us to hypothesize that a comparison between the Web suggester and Aspell on real data would be favourable.

6 Conclusions

We have implemented a spellchecking and autocorrection system and evaluated it on typed data. The main contribution of our work is that while this system incorporates several knowledge sources, an error model, LM and confidence classifiers, it does not require any manually annotated resources, and infers its linguistic knowledge entirely from the Web. Our approach begins with a very large term list that is noisy, containing both spelled and misspelled words, and derived automatically with no human checking for whether words are valid or not.

We believe this is the first published system to obviate the need for any hand labeled data. We have shown that system performance improves from a system that embeds handcrafted knowledge, yielding a 3.8% total error rate on human typed data that originally had a 10.8% error rate. News data with artificially inserted spellings were sufficient to train confidence classifiers to a satisfactory level. This was shown for both German and English. These innovations enable the rapid development of a spellchecking and correction system for any language for which tokenizers exist and string edit distances make sense. We have done so for Arabic and Russian.

In this paper, our results were obtained without any optimization of the parameters used in the process of gathering data from the Web. We wanted to minimize manual tweaking particularly if it were necessary for every language. Thus heuristics such as the number of terms in the term list, the criteria for filtering triples, and the edit distance for defining close words were crude, and could easily be improved upon. It may be beneficial to perform more tuning in future. Furthermore, future work will involve evaluating the performance of the system for these language on real typed data.

7 Acknowledgment

References

Farooq Ahmad and Grzegorz Kondrak. 2005. Learning a spelling error model from search query logs. In *HLT '05: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 955–962, Morristown, NJ, USA. Association for Computational Linguistics.

K. Atkinson. 2009. Gnu aspell. In *Available at* http://aspell.net.

Loghman Barari and Behrang QasemiZadeh. 2005. Clonizer spell checker adaptive, language independent spell checker. In Ashraf Aboshosha et al., editor, *Proc. of the first ICGST International Confer-*

ence on Artificial Intelligence and Machine Learning AIML 05, volume 05, pages 65–71, Cairo, Egypt, Dec. ICGST, ICGST.

Thorsten Brants, Ashok C. Popat, Peng Xu, Franz J. Och, and Jeffrey Dean. 2007. Large language models in machine translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 858–867.

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. Association for Computational Linguistics.

S. Cucerzan and E. Brill. 2004. Spelling correction as an iterative process that exploits the collective knowledge of web users. In *Proceedings of EMNLP 2004*, pages 293–300.

F.J. Damerau. 1964. A technique for computer detection and correction of spelling errors. *Communications of the ACM 7*, pages 171–176.

S. Deorowicz and M.G. Ciura. 2005. Correcting spelling errors by modelling their causes. *International Journal of Applied Mathematics and Computer Science*, 15(2):275–285.

Andrew R. Golding and Yves Schabes. 1996. Combining trigram-based and feature-based methods for context-sensitive spelling correction. In *In Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 71–78.

Ahmed Hassan, Sara Noeman, and Hany Hassan. 2008. Language independent text correction using finite state automata. In *Proceedings of the 2008 International Joint Conference on Natural Language Processing (IJCNLP, 2008)*.

Mark D. Kernighan, Kenneth W. Church, and William A. Gale. 1990. A spelling correction program based on a noisy channel model. In *Proceedings of the 13th conference on Computational linguistics*, pages 205–210. Association for Computational Linguistics.

K. Kukich. 1992. Techniques for automatically correcting words in texts. *ACM Computing Surveys 24*, pages 377–439.

Mirella Lapata and Frank Keller. 2004. The web as a baseline: Evaluating the performance of unsupervised web-based models for a range of nlp tasks. In Daniel Marcu Susan Dumais and Salim Roukos, editors, *HLT-NAACL 2004: Main Proceedings*, pages 121–128, Boston, Massachusetts, USA, May 2 - May 7. Association for Computational Linguistics.

Lidia Mangu and Eric Brill. 1997. Automatic rule acquisition for spelling correction. In Douglas H. Fisher, editor, *ICML*, pages 187–194. Morgan Kaufmann.

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

M.W.C. Reynaert. 2008. All, and only, the errors: More complete and consistent spelling and ocr-error correction evaluation. In *Proceedings of the sixth international language resources and evaluation*.

Kristina Toutanova and Robert Moore. 2002. Pronunciation modeling for improved spelling correction. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 144–151.

L. Amber Wilcox-O'Hearn, Graeme Hirst, and Alexander Budanitsky. 2008. Real-word spelling correction with trigrams: A reconsideration of the mays, damerau, and mercer model. In Alexander F. Gelbukh, editor, *CICLing*, volume 4919 of *Lecture Notes in Computer Science*, pages 605–616. Springer.

Wei Xiang Yang Zhang, Pilian He and Mu Li. 2007. Discriminative reranking for spelling correction. In *The 20th Pacific Asia Conference on Language, Information and Computation*.