Improved Iterative Correction for Distant Spelling Errors

Sergey Gubanov **Irina Galinskaya** **Alexey Baytin**
Yandex
16 Leo Tolstoy St., Moscow, 119021 Russia
{esgv,galinskaya,baytin}@yandex-team.ru

Abstract

Noisy channel models, widely used in modern spellers, cope with typical misspellings, but do not work well with infrequent and difficult spelling errors. In this paper, we have improved the noisy channel approach by iterative stochastic search for the best correction. The proposed algorithm allowed us to avoid local minima problem and improve the F_1 measure by 6.6% on distant spelling errors.

1 Introduction

A speller is an essential part of any program associated with text input and processing — e-mail system, search engine, browser, form editor etc. To detect and correct spelling errors, the state of the art spelling correction systems use the noisy channel approach (Kernighan et al., 1990; Mays et al., 1991; Brill and Moore, 2000). Its models are usually trained on large corpora and provide high effectiveness in correction of typical errors (most of which consist of 1-2 wrong characters per word), but does not work well for complex (multi-character) and infrequent errors.

In this paper, we improved effectiveness of the noisy channel for the correction of complex errors. In most cases, these are cognitive errors in loan words (*folsvagen* → *volkswagen*), names of drugs (*vobemzin* → *wobenzym*), names of brands (*scatcher* → *skechers*), scientific terms (*heksagidron* → *hexahedron*) and last names (*Shwartzneger* → *Schwarzenegger*). In all these cases, the misspelled word contains many errors and the corresponding error model penalty cannot be compensated by the LM weight of its proper form. As a result, either the misspelled word itself, or the other (less complicated, more frequent) misspelling of the same word wins the likelihood race.

To compensate for this defect of the noisy channel, the iterative approach (Cucerzan and Brill, 2004) is typically used. The search for the best variant is repeated several times, what allows correcting rather complex errors, but does not completely solve the problem of falling into local minima. To overcome this issue we suggest to consider more correction hypotheses. For this purpose we used a method based on the simulated annealing algorithm. We experimentally demonstrate that the proposed method outperforms the baseline noisy channel and iterative spellers.

Many authors employ machine learning to build rankers that compensate for the drawbacks of the noisy channel model: (Whitelaw et al., 2009; Gao et al., 2010). These techniques can be combined with the proposed method by replacing posterior probability of single correction in our method with an estimate obtained via discriminative training method.

In our work, we focus on isolated word-error correction (Kukich, 1992), which, in a sense, is a harder task, than multi-word correction, because there is no context available for misspelled words. For experiments we used single-word queries to a commercial search engine.

2 Baseline speller

2.1 Noisy channel spelling correction

Noisy channel is a probabilistic model that defines posterior probability $P(q_0|q_1)$ of q_0 being the intended word, given the observed word q_1; for such model, the optimal decision rule μ is the following:

$$\begin{gathered}\mu(q_1) = \arg\max_{q_0} P(q_0|q_1); \\ P(q_0|q_1) \propto P_{\text{dist}}(q_0 \to q_1) P_{\text{LM}}(q_0),\end{gathered} \tag{1}$$

where P_{LM} is the source (language) model, and P_{dist} is the error model. Given $P(q_0|q_1)$ defined, to correct the word q_1 we could iterate through

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Short Papers), pages 168–173,
Baltimore, Maryland, USA, June 23-25 2014.

all ever-observed words, and choose the one, that maximizes the posterior probability. However, the practical considerations demand that we do not rank the whole list of words, but instead choose between a limited number of hypotheses $h_1, ..., h_K$:

1. Given q_1, generate a set of hypotheses $h_1, ..., h_K$, such that

$$\sum_{k=1}^{K} P(q_0 = h_k|q_1) \approx 1; \quad (2)$$

2. Choose the hypothesis h_k that maximizes $P(q_0 = h_k|q_1)$.

If hypotheses constitute a major part of the posterior probability mass, it is highly unlikely that the intended word is not among them.

2.2 Baseline speller setup

In baseline speller we use a substring-based error model $P_{\text{dist}}(q_0 \rightarrow q_1)$ described in (Brill and Moore, 2000), the error model training method and the hypotheses generator are similar to (Duan and Hsu, 2011).

For building language ($P_{\text{LM}'}$) and error ($P_{\text{dist}'}$) models, we use words collected from the 6-months query log of a commercial search engine.

Hypotheses generator is based on A* beam search in a trie of words, and yields K hypotheses h_k, for which the noisy channel scores $P_{\text{dist}}(h_k \rightarrow q_1)P_{\text{LM}}(h_k)$ are highest possible. Hypotheses generator has high K-best recall (see Section 4.2) — in 91.8% cases the correct hypothesis is found when $K = 30$, which confirms the assumption about covering almost all posterior probability mass (see Equation 2).

3 Improvements for noisy channel spelling correction

While choosing $\arg\max$ of the posterior probability is an optimal decision rule in theory, in practice it might not be optimal, due to limitations of the language and error modeling. For example, *vobemzin* is corrected to more frequent misspelling *vobenzin* (instead of correct form *wobenzym*) by the noisy channel, because $P_{\text{dist}}(vobemzin \rightarrow wobenzym)$ is too low (see Table 1).

There have been attempts (Cucerzan and Brill, 2004) to apply other rules, which would overcome limitations of language and error models with compensating changes described further.

c	$-\log P_{\text{dist}}$	$-\log P_{\text{LM}}$	$\sum$
vobenzin	2.289	31.75	34.04
wobenzym	12.52	26.02	38.54

Table 1: Noisy-channel scores for two corrections of *vobemzin*

3.1 Iterative correction

Iterative spelling correction with E iterations uses standard noisy-channel to correct the query q repeatedly E times. It is motivated by the assumption, that we are more likely to successfully correct the query if we take several short steps instead of one big step (Cucerzan and Brill, 2004) .

Iterative correction is hill climbing in the space of possible corrections: on each iteration we make a transition to the best point in the neighbourhood, i.e. to correction, that has maximal posterior probability $P(c|q)$. As any local search method, iterative correction is prone to local minima, stopping before reaching the correct word.

3.2 Stochastic iterative correction

A common method of avoiding local minima in optimization is the simulated annealing algorithm, key ideas from which can be adapted for spelling correction task. In this section we propose such an adaptation. Consider: we do not always transition deterministically to the next best correction, but instead transition randomly to a (potentially *any*) correction with transition probability being equal to the posterior $P(c_i|c_{i-1})$, where c_{i-1} is the correction we transition from, c_i is the correction we transition to, and $P(\cdot|\cdot)$ is defined by Equation 1. Iterative correction then turns into a *random walk*: we start at word $c_0 = q$ and stop after E random steps at some word c_E, which becomes our answer.

To turn random walk into deterministic spelling correction algorithm, we de-randomize it, using the following transformation. Described random walk defines, for each word w, a probability $P(c_E = w|q)$ of ending up in w after starting a walk from the initial query q. With that probability defined, our correction algorithm is the following: given query q, pick $c = \arg\max_{c_E} P(c_E|q)$ as a correction.

Probability of getting from $c_0 = q$ to some $c_E = c$ is a sum, over all possible paths, of probabilities of getting from q to c via specific path

$q = c_0 \rightarrow c_1 \rightarrow ... \rightarrow c_{E-1} \rightarrow c_E = c$:

$$P(c_E|c_0) = \sum_{\substack{c_1 \in W \\ ... \\ c_{E-1} \in W}} \prod_{i=1}^{E} P(c_i|c_{i-1}), \quad (3)$$

$$P(c_i|c_{i-1}) = \frac{P_{\text{dist}}(c_i \rightarrow c_{i-1}) P_{\text{LM}}(c_i)}{P_{\text{observe}}(c_{i-1})}, \quad (4)$$

where W is the set of all possible words, and $P_{\text{observe}}(w)$ is the probability of observing w as a query in the noisy-channel model.

Example: if we start a random walk from *vobemzin* and make 3 steps, we most probably will end up in the correct form *wobenzym* with $P = 0.361$. A few of the most probable random walk paths are shown in Table 2. Note, that despite the fact that most probable path does not lead to the correct word, many other paths to *wobenzym* sum up to 0.361, which is greater than probability of any other word. Also note, that the method works only because multiple misspellings of the same word are presented in our model; for related research see (Choudhury et al., 2007).

$c_0 \rightarrow c_1 \rightarrow c_2 \rightarrow c_3$	P
vobemzin→vobenzin→vobenzin→vobenzin	0.074
vobemzin→vobenzim→wobenzym→**wobenzym**	0.065
vobemzin→vobenzin→vobenzim→vobenzim	0.052
vobemzin→vobenzim→vobenzim→**wobenzym**	0.034
vobemzin→wobenzym→wobenzym→**wobenzym**	0.031
vobemzin→wobenzim→wobenzym→**wobenzym**	0.028
vobemzin→wobenzyn→wobenzym→**wobenzym**	0.022

Table 2: Most probable random walk paths starting from $c_0 = q = vobemzin$ (the correct form is in bold).

Also note, that while Equation 3 uses noisy-channel posteriors, the method can use an arbitrary discriminative model, for example the one from (Gao et al., 2010), and benefit from a more accurate posterior estimate.

3.3 Additional heuristics

This section describes some common heuristic improvements, that, where possible, were applied both to the baseline methods and to the proposed algorithm.

Basic building block of every mentioned algorithm is one-step noisy-channel correction. Each basic correction proceeds as described in Section 2.1: a small number of hypotheses $h_1, ..., h_K$ is generated for the query q, hypotheses are scored, and scores are recomputed into normalized posterior probabilities (see Equation 5). Posterior probabilities are then either used to pick the best correction (in baseline and simple iterative correction), or are accumulated to later compute the score defined by Equation 3.

$$\begin{aligned} \text{score}(h_i) &= P_{\text{dist}}(h_i \rightarrow q)^{\lambda} P_{\text{LM}}(h_i) \\ P(h_i|q) &= \text{score}(h_i) \Big/ \sum_{j=1}^{K} \text{score}(h_j) \end{aligned} \quad (5)$$

A standard log-linear weighing trick was applied to noisy-channel model components, see e.g. (Whitelaw et al., 2009). λ is the parameter that controls the trade-off between precision and recall (see Section 4.2) by emphasizing the importance of either the high frequency of the correction or its proximity to the query.

We have also found, that resulting posterior probabilities emphasize the best hypothesis too much: best hypothesis gets almost all probability mass and other hypotheses get none. To compensate for that, posteriors were *smoothed* by raising each probability to some power $\gamma < 1$ and renormalizing them afterward:

$$P_{\text{smooth}}(h_i|q) = P(h_i|q)^{\gamma} \Big/ \sum_{j=1}^{K} P(h_j|q)^{\gamma}. \quad (6)$$

In a sense, γ is like temperature parameter in simulated annealing – it controls the entropy of the walk and the final probability distribution. Unlike in simulated annealing, we fix γ for all iterations of the algorithm.

Finally, if posterior probability of the best hypothesis was lower than threshold α, then the original query q was used as the spell-checker output. (Posterior is defined by Equation 6 for the baseline and simple iterative methods and by Equations 3 and 6 for the proposed method). Parameter α controls precision/recall trade-off (as well as λ mentioned above).

4 Experiments

4.1 Data

To evaluate the proposed algorithm we have collected two datasets. Both datasets were randomly sampled from single-word user queries from the 1-week query log of a commercial search engine. We annotated them with the help of professional analyst. The difference between datasets

is that one of them contained only queries with low search performance: for which the number of documents retrieved by the search engine was less than a fixed threshold (we will address it as the "hard" dataset), while the other dataset had no such restrictions (we will call it "common"). Dataset statistics are shown in Table 3.

Dataset	Queries	Misspelled	Avg. $-\log P_{\text{dist}}$
Common	2240	224 (10%)	5.98
Hard	2542	1484 (58%)	9.23

Table 3: Evaluation datasets.

Increased average error model score and error rate of "common" dataset compared to "hard" shows, that we have indeed managed to collect hard-to-correct queries in the "hard" dataset.

4.2 Experimental results

First of all, we evaluated the recall of hypotheses generator using *K-best recall* — the number of correct spelling corrections for misspelled queries among K hypotheses divided by the total number of misspelled queries in the test set. Resulting recall with $K = 30$ is 91.8% on "hard" and 98.6% on "common".

Next, three spelling correction methods were tested: noisy channel, iterative correction and our method (stochastic iterative correction).

For evaluation of spelling correction quality, we use the following metrics:

- *Precision*: The number of correct spelling corrections for misspelled words generated by the system divided by the total number of corrections generated by the system;
- *Recall*: The number of correct spelling corrections for misspelled words generated by the system divided by the total number of misspelled words in the test set;

For hypotheses generator, $K = 30$ was fixed: recall of 91.8% was considered big enough. Precision/recall tradeoff parameters λ and α (they are applicable to each method, including baseline) were iterated by the grid $(0.2, 0.25, 0.3, ..., 1.5) \times (0, 0.025, 0.05, ..., 1.0)$, and E (applicable to iterative and our method) and γ (just our method) were iterated by the grid $(2, 3, 4, 5, 7, 10) \times (0.1, 0.15, ...1.0)$; for each set of parameters, precision and recall were measured on both datasets. Pareto frontiers for precision and recall are shown in Figures 1 and 2.

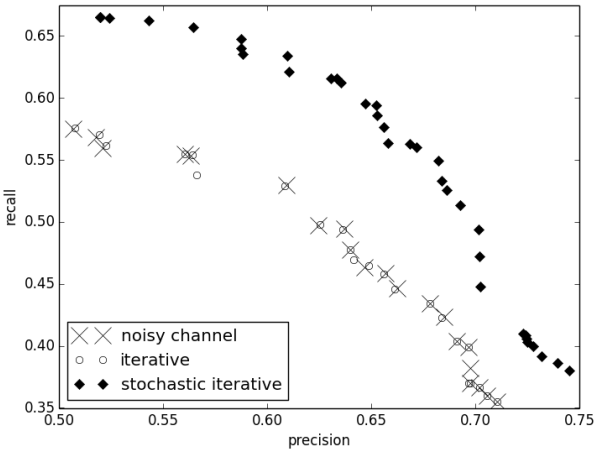

Figure 1: Precision/recall Pareto frontiers on "hard" dataset

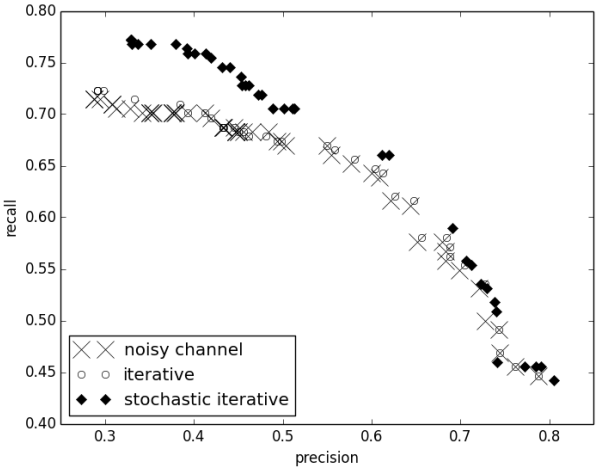

Figure 2: Precision/recall Pareto frontiers on "common" dataset

We were not able to reproduce superior performance of the iterative method over the noisy channel, reported by (Cucerzan and Brill, 2004). Supposedly, it is because the iterative method benefits primarily from the sequential application of split/join operations altering query decomposition into words; since we are considering only one-word queries, such decomposition does not matter.

On the "hard" dataset the performance of the noisy channel and the iterative methods is inferior to our proposed method, see Figure 1. We tested all three methods on the "common" dataset as well to evaluate if our handling of hard cases affects the performance of our approach on the common cases of spelling error. Our method performs well on the common cases as well, as Figure 2 shows. The performance comparison for the "common" dataset shows comparable performance for all considered methods.

Noisy channel and iterative methods' frontiers

are considerably inferior to the proposed method on "hard" dataset, which means that our method works better. The results on "common" dataset show, that the proposed method doesn't work worse than baseline.

Next, we optimized parameters for each method and each dataset separately to achieve the highest F_1 measure. Results are shown in Tables 4 and 5. We can see, that, given the proper tuning, our method can work better on any dataset (but it cannot achieve the best performance on both datasets at once). See Tables 4 and 5 for details.

Method	λ	α	γ	E	F_1
Noisy channel	0.6	0.1	-	-	55.8
Iterative	0.6	0.1	-	2	55.9
Stochastic iterative	0.9	0.2	0.35	3	62.5

Table 4: Best parameters and F_1 on "hard" dataset

Method	λ	α	γ	E	F_1
Noisy channel	0.75	0.225	-	-	62.06
Iterative	0.8	0.275	-	2	63.15
Stochastic iterative	1.2	0.4	0.35	3	63.9

Table 5: Best parameters and F_1 on "common" dataset

Next, each parameter was separately iterated (by a coarser grid); initial parameters for each method were taken from Table 4. Such iteration serves two purposes: to show the influence of parameters on algorithm performance, and to show differences between datasets: in such setup parameters are virtually tuned using "hard" dataset and evaluated using "common" dataset. Results are shown in Table 6.

The proposed method is able to successfully correct distant spelling errors with edit distance of 3 characters (see Table 7).

However, if our method is applied to shorter and more frequent queries (as opposed to "hard" dataset), it tends to suggest frequent words as false-positive corrections (for example, *grid* is corrected to *creed* – Assassin's Creed is popular video game). As can be seen in Table 5, in order to fix that, algorithm parameters need to be tuned more towards precision.

5 Conclusion and future work

In this paper we introduced the stochastic iterative correction method for spell check corrections. Our experimental evaluation showed that the proposed method improved the performance of popular spelling correction approach – the noisy channel model – in the correction of difficult spelling errors. We showed how to eliminate the local minima issue of simulated annealing and proposed a technique to make our algorithm deterministic.

The experiments conducted on the specialized datasets have shown that our method significantly improves the performance of the correction of hard spelling errors (by 6.6% F_1) while maintaining good performance on common spelling errors.

In continuation of the work we are considering to expand the method to correct errors in multi-word queries, extend the method to work with discriminative models, and use a query performance prediction method, which tells for a query whether our algorithm needs to be applied.

	F_1, common			F_1, hard		
	N.ch.	It.	Our	N.ch.	It.	Our
$\lambda = 0.5$	45.3	45.9	37.5	54.9	54.8	50.0
0.6	49.9	50.5	41.5	*55.8*	*55.9*	56.6
0.7	50.4	50.4	44.1	54.5	55.1	59.6
0.8	52.7	52.7	46.0	52.6	53.0	61.5
0.9	53.5	53.5	49.3	50.3	50.6	**62.5**
1.0	**55.4**	*55.0*	50.9	47.0	47.3	61.8
1.1	53.7	53.4	52.7	44.3	44.6	60.8
1.2	52.5	52.5	53.7	41.9	42.3	58.8
1.3	52.2	52.6	54.6	39.5	39.9	56.6
1.4	51.4	51.8	*55.0*	36.8	37.3	53.6
$\alpha = 0$	41.0	41.5	33.0	52.9	53.1	58.3
0.1	49.9	50.6	35.6	55.8	*55.9*	59.7
0.15	59.4	59.8	43.2	*55.8*	55.6	61.6
0.2	*60.8*	**61.3**	49.4	51.0	51.0	**62.5**
0.25	54.0	54.0	54.9	46.3	46.3	61.1
0.3	46.3	46.3	*57.3*	39.2	39.2	58.4
0.4	25.8	25.8	53.9	22.3	22.3	50.3
$E = 2$		*50.6*	*53.6*		*55.9*	60.4
3		50.6	49.4		55.9	**62.5**
4		50.6	46.4		55.9	62.1
5		50.6	46.7		55.9	60.1
$\gamma = 0.1$			10.1			6.0
0.2			49.4			51.5
0.3			*51.4*			61.4
0.35			49.4			*62.5*
0.4			47.5			62.0
0.45			45.8			60.8
0.5			45.2			60.3

Table 6: Per-coordinate iteration of parameters from Table 4; per-method maximum is shown in italic, per-dataset in bold

Query	**Noisy channel**	**Proposed method**
akwamarin	akvamarin	**aquamarine**
maccartni	maccartni	**mccartney**
ariflaim	ariflaim	**oriflame**
epika	**epica**	replica
grid	**grid**	creed

Table 7: Correction examples for the noisy channel and the proposed method.

References

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

Monojit Choudhury, Markose Thomas, Animesh Mukherjee, Anupam Basu, and Niloy Ganguly. 2007. How difficult is it to develop a perfect spell-checker? a cross-linguistic analysis through complex network approach. In *Proceedings of the second workshop on TextGraphs: Graph-based algorithms for natural language processing*, pages 81–88.

Silviu Cucerzan and Eric Brill. 2004. Spelling correction as an iterative process that exploits the collective knowledge of web users. In *EMNLP*, volume 4, pages 293–300.

Huizhong Duan and Bo-June Paul Hsu. 2011. Online spelling correction for query completion. In *Proceedings of the 20th international conference on World wide web*, pages 117–126. ACM.

Jianfeng Gao, Xiaolong Li, Daniel Micol, Chris Quirk, and Xu Sun. 2010. A large scale ranker-based system for search query spelling correction. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 358–366. Association for Computational Linguistics.

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-Volume 2*, pages 205–210. Association for Computational Linguistics.

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

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

Casey Whitelaw, Ben Hutchinson, Grace Y Chung, and Gerard Ellis. 2009. Using the web for language independent spellchecking and autocorrection. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2*, pages 890–899. Association for Computational Linguistics.