Discriminative Reranking for Spelling Correction*

Yang Zhang[1], Pilian He[1], Wei Xiang[2], Mu Li[3]

[1] School of Computer Science and Technology
Tianjin University, China
{yangzhang, plhe}@tju.edu.cn
[2] Department of Computer Science
Hong Kong University of Science and Technology, China
wxiang@cse.ust.hk
[3] Natural Language Computing Group
Microsoft Research Asia
muli@microsoft.com

Abstract. This paper proposes a novel approach to spelling correction. It reranks the output of an existing spelling corrector, Aspell. A discriminative model (Ranking SVM) is employed to improve upon the initial ranking, using additional features as evidence. These features are derived from state-of-the-art techniques in spelling correction, including edit distance, letter-based n-gram, phonetic similarity and noisy channel model. This paper also presents a new method to automatically extract training samples from the query log chain. The system outperforms the baseline Aspell greatly, as well as previous models and several off-the-shelf spelling correction systems (e.g. Microsoft Word 2003). The results on query chain pairs are comparable to that based on manually-annotated pairs, with 32.2%/32.6% reduction in error rate, respectively.

1. Introduction

Spelling correction is used to suggest one or several hypothetical corrections for the assumed error once a spell checker detects some misspelling, which is typically identified when it cannot been found in a pre-compiled lexicon. In this paper, we focus on the problem of interactively correcting non-word errors (e.g. *teh* for *the*). We do not deal with real word errors (such as *from* is written when *form* is intended). We propose an approach that can be applied to text processing applications, such as the spelling corrector used in Microsoft Word or Aspell [2]. In these applications several suggestions (5 or 10 in most cases) are provided in an interactive manner, letting the user choose the desired one. In most cases the suggestion ranked first is preferred. For this reason top 1 suggestion is our most concern. We also care about the top 5/10 accuracies since they are presented to the user, too. Aspell [2] is a spelling program widely used on different platforms. As shown in [1], the top 5 accuracy of Aspell is more than 85%. However, it suffers from its poor performance on top 1 accuracy, which is less than 60%.

Our motivation is to improve the initial ranking of Aspell's output to gain a higher performance. The main contributions are as follows:

1. We adapt a discriminative model (Ranking SVM) to rerank the output of Aspell's N-best candidates, adding some global features as evidence. The model is general-purposed so that we can integrate state-of-the-art techniques in spelling correction into one model, including edit distance, n-gram, phonetic similarity and noisy channel model. Encouraging results are gained on this task.
2. A novel approach is proposed to actively solicit training pairs from query log chain. The quality of this approach is verified in the evaluation that follows.

The rest of the paper is organized as follows: in Section 2, we conduct a literature research into spelling correction. The reranking problem is formulated in Section 3. Section 4 shows the experimental

* This work was conducted when the first author was visiting Microsoft Research Asia. It is supported by Science-Technology Development Project of Tianjin (04310941R) and Applied Basic Research Project of Tianjin (05YFJMJC11700)

evaluation. In this section we also specify the details to automatically extract training data from query log chain. The last section contains the conclusions and suggestions for possible further developments of the proposed model.

2. Related Work

Comprehensive reviews of the spelling correction literature can be found in [13] and [15]. There are some widely-used algorithms proposed to correct spelling errors. Edit distance algorithm and letter-based n-gram deal with typographic errors due to keyboard proximity. Soundex [15] and metaphone [21] are for phonetic ones (e.g. *fone* for *phone*). Algorithms dealing with phonetic errors map a spelling to a phonetic-encoded string: Soundex uses 7 codes; Phonix uses 9 codes, whereas the number of codes employed in [10] is 14. Metaphone and its descendant double metaphone [22] take not only spelling but also phonetic factors into account. Since the original versions of these algorithms do not work perfectly in real scenarios, variants or hybrid systems have been invented. Phonix [6] [7] is a variant of Soundex and is mainly used for human name detection. The popularity of Aspell is due to the excellent metaphone [21] algorithm and Ispell's [11] near miss strategy which is inserting a space or hyphen, interchanging two adjacent letters, changing one letter, deleting a letter, or adding a letter. Our system takes more than five cutting-edge techniques into account, rather than simply rely on one or two factors.

As for the models in this task, rule-based studies include [18] [19]. In recent years, statistical machine learning also makes contributions to spelling correction task. It falls into two categories: generative and discriminative. The heavily-used generative model in this domain is the noisy channel model: [1] [3] [5] [24]. It considers the phenomenon of making spelling mistakes as the process of sending text through a noisy communication channel, which introduces error in the text. As for each input (a potential misspelling) the candidate with largest posterior, P(cand|input) is chosen. By applying Bayes rule and dropping the constant denominator we can score each candidate by its prior P(cand) multiplying the likelihood, P(input|cand). The language model P(cand) can be estimated using n-gram statistics [3] [14]. The channel model P(input|cand) is generally modeled using letter-to-letter[14] or string-to-string confusion probabilities [3]. [3] has the best performance because it allows generic string-to-string edits. As for the discriminative models, Winnow [8], neural net [10] and maximum entropy [16] are adopted. [8] corrects contextual errors, which resolves ambiguity from the same confusion set. Our model is linearly discriminative, which can integrate some cutting-edge techniques into this model.

Other than applications in text processing, the emerging search engines pose an even bigger challenge on query spelling correction since searching for misspelled query is a waste of time and money. There are a large number of OOV words in queries. Statistics in [5] show that 10%-15% of queries contain one or more errors; the average length of a query is less than 3 words, with little contextual information available. Noisy channel model is dominant in query spelling correction research [1] [5]. Research in [16] exploits to use maximum entropy model for query spelling correction task. It also features the usage of distributional similarity between the correct word and its misspellings based on query log statistics. Although we aim toward single word spelling correction, we believe that our approach can also be adapted to this scenario because of its generality.

3. Problem Formulation

3.1 Definition for the reranking problem

A reranking problem can be formulated as follows: first we use a baseline model to get the N-best

candidates, together with their initial scores. Next a complex model is to rerank these candidates, using additional features as evidence. With these efforts the answer candidate or the approximately correct candidates can be ranked as higher as possible, making the performance enhanced. We consider the following set-up:

- Training data is a set of example pairs $\{q_i, a_i\}$ where each q_i is an input string (a potential misspelling) and each a_i is the correct word for this input.
- We use x_{ij} to denote the j'th candidate for the i'th input in training data, and $C(q_i)=\{ x_{i1}, x_{i2} ...\}$ to denote the set of candidates for q_i. In this paper, the top N outputs from Aspell are used as the set of candidates.
- $R(x_{ij})$ is the rank that the base model assigns to x_{ij}. We will use this information as a feature in the reranking model. If this information is omitted, it turns to be a ranking problem rather than a reranking one.
- We assume a set of m - n additional features, $h_s(x)$ for s = n+1…m. The features could be arbitrary functions of the candidates; any feature which helps discriminate good candidates from bad ones can be included.
- Finally, the parameters of the model form a vector of m parameters, $w = \{w_1, w_2 \ldots w_m\}$. The ranking function is defined as

$$F(x, w) = \sum_{r=1}^{n} w_r R_r(x) + \sum_{s=n+1}^{m} w_s h_s(x) \quad (1)$$

Here $R_1(x)\ldots R_n(x)$ refer to the rank feature functions whereas $h_{n+1}(x)\ldots h_m(x)$ are functions of additional features. Let h(x) be the vector $\{R_1(x)\ldots R_n(x), h_{n+1}(x)\ldots h_m(x)\}$, (1) turns to $F(x, w) = w \cdot h(x)$.

The learning task is to find parameter settings for vector w which lead to good performance on test data; the parameters will be set using training data examples as evidence. We now discuss how to set these parameters using Ranking SVM.

3.2 Ranking SVM

The problem of ranking, in which the goal is to learn an ordering over objects, has gained much attention in recent machine learning research [12] [23]. Different from the standard machine learning tasks of classification and metric regression, Ranking SVM (a generalization of ordinal regression SVMs in [9]) is to predic variables of ordinal scale. This problem arises frequently in social sciences and information retrieval where human preferences play a major role.

The key to Ranking SVM is to find user's preference judgments so that the rank constraints can be induced. In [12] clickthrough data is used to describe the rank constraints while the concept of query chain is employed in [23]. Next we can derive the loss function to be optimized from the preference judgments. If there is a preference judgment over candidate c_i and c_j, for a given input q, the judgment can be expressed as the following form:

$$c_i >_q c_j \quad (2)$$

In Ranking SVM we can rewrite (2) as (here *h* is a function mapping each candidate to a feature vector):

$$w \cdot h(c_i, q) > w \cdot h(c_j, q) \quad (3)$$

This constraint can be expressed in the setting of classification SVM (with bias *b* removed for both sides, this is equivalent to an x_i-x_j instance) as:

$$w \cdot [h(c_i, q) - h(c_j, q)] > 0 \quad (4)$$

After adding a margin and non-negative slack variables to allow some of the preference constraints to be violated, we can minimize an upper bound on the number of violated constraints. Simultaneously maximizing the margin leads to the following convex quadratic optimization problem:

$$\min_{w,\xi_{ij}} \frac{1}{2} w \cdot w + C \sum_{ij} \xi_{ij} \text{ , subject to}$$

$$\begin{aligned} &\forall (q, i, j) : w \cdot h(c_i, q) >_q w \cdot h(c_j, q) + 1 - \xi_{ij} \\ &\forall i, j : \xi_{ij} \geq 0 \end{aligned} \tag{5}$$

With this formulation, the aforementioned constraints fit well into Ranking SVM model.

3.3 Discriminative Reranking for Spelling Correction

Based on the framework described above, we use Ranking SVM to find optimal parameter settings. Note that this reranking framework is similar to [4]. Since it is difficult for us to get the candidate score from Aspell, we just use their relative rank $R(x_{ij})$ as binary features (e.g. Is ranked top-N from Aspell, N=1, 3, …).

Recall that our target application is to present users several suggestions (5 to 10 in most cases) for each assumed error, letting them choose the most preferred one. Our goal is not only rank the answer to the first place of the suggestion list, but also make candidates approximate to the answer (e.g. having the same stemming root w.r.t the answer) to be ranked as higher as possible. When a user encounters a misspelling, in most cases they are not clear which suggestion is desired. For this reason we use some heuristics to simulate users' preference judgments, assigning ranks to different candidates for each input string. In this task, one obvious way is to assign the answer top rank, its inflections with second rank, and all others as third rank. Since the widely-used algorithms (e.g. Perceptron, maximum entropy) can not express these rank constraints effectively, finally we choose Ranking SVMs [12].

3.4 Feature Templates

Table 1. Feature templates

Category	Feature Name	Description
Aspell's initial ranking	IsBaselineRankTop<N>	Binary: Ranked top-N from Aspell candidate list (N = 1, 5, 10…)
Noisy channel score	NoisyChannelScoreTop<N>	Binary: Ranked top-N with noisy channel score (N = 1, 5, 10…)
Frequency	UniFreqRatio_GT_<N>	Binary: The unigram freq. ratio of cand and input > N?
	UniFreq_GT_<N>	Binary: The unigram freq of cand > N?
Edit distance	EditDist_LE_<N>	Binary: edit distance b/w input & cand <= N?
	EditDist_GE_<N>	Binary: edit distance b/w input & cand >= N?
Lexicon	QryinLogLex_&_Namelist	Binary: Is input in log lexicon or name list?
	inTrustLex_&_Namelist	Binary: Is cand in log lexicon or name list?
Phonetic	DMPKeyEQ2Input	Binary: Does cand and input share the same double metaphone key?
Length	Len_LE_<N>	Binary: the length of cand < = N?
	Len_GE_<N>	Binary: the length of cand > = N?
	LenChanged	Binary: length(input) != length(cand)
Letter-based N-gram	NgramSimilar2Input	Binary: Is $Similarity_{n\text{-}gram}$(cand, input) >= N?

The noisy channel score is derived from the P(cand) multiplied by P(input|cand). The context-dependent weighted edit distance is calculated according to [5]. The formula for the letter-based n-gram similarity is a variant of [17].

4. Experimental Results

4.1 Training data (from query chain as well as human annotation)

In a discriminative learning task we need some (input, gold-standard) pairs for training purposes. This work is often conducted by human experts and is really a time-consuming and energy-draining job. In this paper we exploit an innovative approach to actively solicit training pairs from query log data. This approach adopts the concept of *query chain* defined in [23], whose authors observe that users searching the web often perform a sequence, of queries with a similar information need. They refer to a sequence of reformulated queries from the user as a *query chain* and apply this finding to learn ranked retrieval functions for web search results.

We adapt this idea to automatically extract training pairs from query log data. Take figure 1 for illustration. Suppose some user types query "messanger download" to a search engine intending to download the latest version of messenger software. This search engine returns results containing keyword "messanger" or "download" or both, along with a query suggestion "messenger download". At first the user may get confused. After some effort, he realizes that the original query is misspelled and then corrects it. This time he gets web pages he really want, feeling satisfied. After some log mining work we can get training pair (messanger download, messenger download). The pair (resipi, recipe) can be mined in a similar way. As for single word spelling correction task, we care about one-to-one word pairs only. So we just keep (messanger, messenger) and (resipi, recipe) for our use.

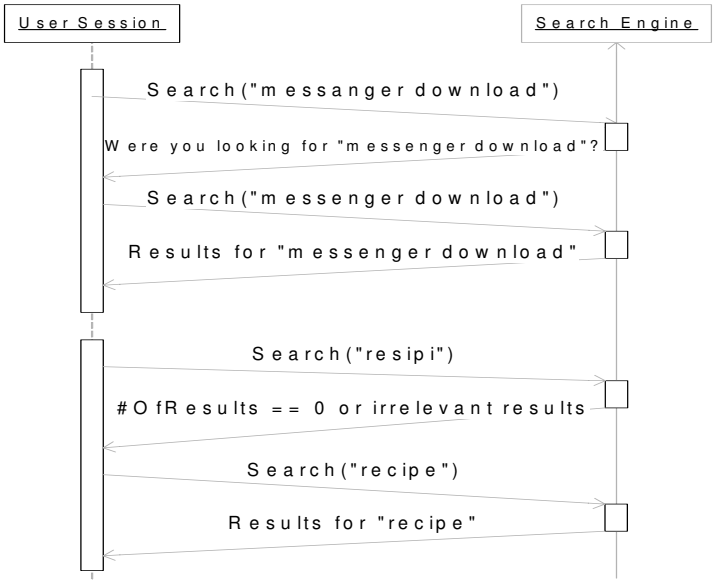

Fig. 1. Query Chain

To the best of our knowledge, this approach is to exploit human knowledge from the query log data. If the results returned by search engine are not what the users really want (few or irrelevant hits, even no results at all), they tend to use their knowledge or resort to external resources (dictionary look-up, clickthrough on the suggestion link, etc.) to reformulate the original query, expecting a satisfying feedback. This procedure goes on and on. It is also common the user gets frustrated and quit, without further reformulation. But in most cases this method works. In this paper the quality of training pairs mined from query chain is assured by its comparable performance with the result trained on manually-annotated pairs.

We randomly sample 5-day query log data from MSN Search [20] in a 5 month period. Finally we choose 120 log files with about 15,000,000 queries in total. Queries from a user session are detected as one-hour timeout sharing the same IP address. Then we group queries according to some criteria (letter-based n-gram overlap, edit distance, etc.). For each group we sort queries in chronological order and assume that lately-typed query is more favorable than the preceding ones. As for each query we make the following constraints: 1) the misspelling does not exist in the dictionary but its correction does; 2) the length of the misspelling is proportional to its edit distance with its correction. We randomly sampled 1000 out of 24,288 extracted query chain pairs as training set. To verify the quality of pairs from query chain, we also use 1000 manually-annotated pairs as a second training set.

4.2 Test Data

We use the test set publicized by Aspell[†] as test data. It consists of 547 misspellings paired with their best correction (such as <*theirselves*, *themselves*>, <*wicken*, *weaken*>) determined by human experts. To compare with the system implemented in [1], we also remove compound words from the test set, leaving 508 misspellings. At present we ignore case error (if any).

4.3 Evaluation Metrics

The evaluation metrics we adopt are top-N accuracy, precision and recall. Note that precision focuses on samples with correction (top-1 suggestion not equal to the input), whereas recall is equivalent to top-1 accuracy. They can be calculates as follows:

$$\mathrm{Pr}\,ecision = \frac{\#OfValidCor\,rection}{\#OfValidCor\,rection + \#OfBadCorre\,ction} \tag{6}$$

$$\mathrm{Re}\,call = \frac{\#OfValidCor\,rection}{\#OfValidCor\,rection + \#OfNoCorrec\,tion + \#OfBadCorre\,ction} \tag{7}$$

$$Accuracy_{TopN\,(N=1,5,10,25,100)} = \frac{\#OfSamplesW\,hoseAnswer\,IsInTopN}{\#OfSamplesI\,nTotal} \tag{8}$$

4.4 Results with Aspell's initial ranking (reranking model)

The N-best candidates are dumped from the latest version of Aspell (0.60.4), together with their initial rankings. There are five suggestion modes for Aspell: *ultra*, *fast*, *normal*, *slow* and *bad-spellers*. We choose the *slow* mode because it has a good trade-off between answer coverage in candidates and the size of candidate set (no more than 100 candidates for each sample). As for Ranking SVM we use SVM^{light} v6.01, with all parameters and loss functions set to default values. A greedy strategy is conducted to do feature selection. At last we choose 16 features in total, according to the feature templates listed in Section 3.4. The word frequency and lexicon statistics is collected using 9-month query log from MSN Search [20]. We use Perl module Lingua::MSWordSpell[‡] from BBC to evaluate the top1/5/10 accuracies of Microsoft Word 2003. We re-implement the error model described in [3] with an optimal window size of 3. We estimate source model using n-gram (degraded to unigram frequency in this setting) from query log statistics.

From Table 2 we can conclude that our systems achieve great improvements over the baseline on top 1/5/10 accuracies. The results based on query chain pairs are comparable to that of based on human-annotated pairs, with 32.2%/32.6% relative reduction in error rate, respectively. It also outperforms previous best performing models [3] and several off-the-shelf spelling correction systems (e.g. Microsoft Word 2003). We think it owes to Aspell's initial ranking and the introduction of complex

[†] Available at http://aspell.net/test/

[‡] http://search.cpan.org/~bbc/Lingua-MSWordSpell-1.010/lib/Lingua/MSWordSpell.pm

features. Since this model involves a relatively large amount of calculation, it may need more processing time. We believe this issue will be improved in future research.

Table 2. Top-N accuracy

	EMBED [1]	Microsoft Word 2003	Brill's error model [3]	Aspell (baseline)	RSVM (query chain)	RSVM (manually)
Total pairs	508	508	508	508	508	508
Top 1	211 (41.5%)	306 (60.2%)	312 (61.4%)	269 (53.0%)	346 (68.1%)	347 (68.3%)
Top 5	331 (65.2%)	377 (74.2%)	429 (84.4%)	410 (80.7%)	434 (85.4%)	440 (86.6%)
Top 10	N/A	382 (75.2%)	445 (87.6%)	443 (87.2%)	448 (88.2%)	452 (89.0%)
Top 25	386 (76.0%)	N/A	460 (90.6%)	456 (89.8%)	460 (90.6%)	461 (90.7%)
Top 100	402 (79.1%)	N/A	462 (90.9%)	462 (90.9%)	462 (90.9%)	462 (90.9%)

4.5 Results without Aspell's initial ranking (ranking model)

A question regarding the above approach is whether the features incorporating state-of-the-art techniques can work well without Aspell's initial ranking. This leads to a ranking model rather than a reranking one, since no baseline information used.

To verify the effectiveness of these features, we carry out two more experiments (training data from query chain pairs and manually-annotated ones) on ranking the N-best candidates, not using Aspell's initial ranking. Table 3 shows the results.

Table 3. Performance of the ranking and reranking model

	Aspell 0.60.4 (baseline)	Ranking SVM query chain ranking	Ranking SVM human expert ranking	Ranking SVM query chain reranking	Ranking SVM human expert reranking
Total pairs	508	508	508	508	508
Precision	53.3%	64.4%	65.4%	70.3%	70.5%
Reduced error rate	0.0%	20.1%	22.2%	32.2%	32.6%
Top 1 accuracy (recall)	269 (53.0%)	317 (62.4%)	322 (63.4%)	346 (68.1%)	347 (68.3%)
Top 5 accuracy	410 (80.7%)	443 (87.2%)	440 (86.6%)	434 (85.4%)	440 (86.6%)
Top 10 accuracy	443 (87.2%)	452 (89.0%)	456 (89.8%)	448 (88.2%)	452 (89.0%)
Top 25 accuracy	456 (89.8%)	461 (90.7%)	462 (90.9%)	460 (90.6%)	461 (90.7%)
Top 100 accuracy	462 (90.9%)	462 (90.9%)	462 (90.9%)	462 (90.9%)	462 (90.9%)

It is clearly shown that in this setting it also gives very credible results without baseline information. The ranking model based on query chain training set gains 20.1% relative reduction in error-rate, whereas ranking model based on human expert training set is 22.2%. However, there is still a 5% gap between the ranking and the reranking model. We think it is due to the effectiveness of Aspell's scoring strategy, which involves a lot of human-tuned rules.

5. Conclusion and Future Work

This paper presents a Ranking SVM-based approach for spelling correction. It reranks the output from Aspell, with a great improvement in top 1/5/10 accuracies. The discriminative model employed is general-purposed so that the state-of-the-art spelling correction techniques, including edit distance, phonetic similarity, n-gram and noisy channel model can be integrated into one single model and they do a great job as a whole. The training pairs extracted from query chain show their quality in the performance evaluation with human-annotated pairs.

However, there are several improvements that can be made to promote this work. One step is to enhance the preceding spell checking module, taking contextual information into account. We think this

effort will not only avoid false alarms but also detect real-word errors. Another improvement is to discover some more discriminant features. For example, users' clickthrough data on the suggestion link posed by search engine is a promising heuristic. Since search engine is a large portal to information, we can take a step further to analyze the distribution of misspelling and its answer in web snippets (even the page from which they are extracted) in search results. Finally, the concept of query chain can be further explored, e.g. query expansion.

References:

[1] Farooq Ahmad and Grzegorz Kondrak. Learning a Spelling Error Model from Search Query Logs. Proceedings of EMNLP 2005, pp. 955-962.

[2] Aspell. Web page http://aspell.net

[3] Eril Brill, Robert C. Moore. An Improved Error Model for Noisy Channel Spelling Correction. In proceedings of 38th ACL, pp. 286-293, 2000.

[4] Michael Collins. Discriminative Reranking for Natural Language Parsing. In proceedings of 17th ICML, pp. 175-182, 2000.

[5] Silviu Cucerzan, Eric Brill. 2004. Spelling Correction as An Iterative Process That Exploits The Collective Knowledge of Web Users. In proceedings of EMNLP 04. 293-300.

[6] Klas Erikson. Approximate Swedish Name Matching - Survey and Test of Different Algorithms, 1997.

[7] T. Gadd. PHONIX: The Algorithm. Program, 24(4):363-366, 1990.

[8] Andrew R. Golding, Dan Roth. Applying Winnow to Context-sensitive Spelling Correction. In proceedings of 13th ICML, pp. 182-190, 1996

[9] R. Herbrich, T. Graepel, K. Obermayer. Large Margin Rank Boundaries for Ordinal Regression. In Advances in Large Margin Classifiers, pp. 115-132, 2000.

[10]Victoria J. Hodge, Jim Austin. A Comparison of Standard Spell Checking Algorithms and a Novel Binary Neural Approach. IEEE TKDE, Vol. 15, No. 5, September 2003.

[11]Ispell. Web page http://www.gnu.org/software/ispell/ispell.html

[12]Thorsten Joachims. Optimizing Search Engines Using Clickthrough Data, In proceedings of the ACM SIGKDD, 2002.

[13]Daniel Jurafsky, James H. Martin, Speech and Language Processing, Prentice Hall, 2000.

[14]Mark D. Kernighan, Kenneth W. Church, Willian A. Gale. A Spelling Correction Program Based on Noisy Channel Model. In proceedings of 13th COLING, Vol. 2:pp 205-210, 1990.

[15]Karen Kukich. Techniques for Automatically Correcting Words in Text, ACM Computing Survey, Vol. 14, No. 4, pp 377-439, 1992.

[16]Mu Li, Muhua Zhu, Yang Zhang, Ming Zhou, Exploring Distributional Similarity Based Models for Query Spelling Correction, In proceedings of 44th ACL, pp. 1025-1032, 2006.

[17]Dekang Lin. An Information-Theoretic Definition of Similarity. In proceedings of 15th ICML 1998: 296-304

[18]Lidia Mangu, Eric Brill. Automatic Rule Acquisition for Spelling Correction, In proceedings of 14th ICML, pp. 187 - 194, 1997

[19]B Martins, M.J. Silva. Spelling Correction for Search Engine Queries, EsTAL 2004

[20]MSN Search. http://search.msn.com

[21]L. Philips. 1990. Hanging on the Metaphone. Computer Language Magazine, 7(12): 39.

[22]L. Philips. 2000. The Double-metaphone Search Algorithm. C/C++ User's Journal, 2000.

[23]Filip Radlinski, Thorsten Joachims. Query Chains: Learning to Rank from Implicit Feedback, Proceeding of 11th ACM SIGKDD, pp. 239-248, 2005.

[24]Kristina Toutanova, Robert C. Moore. Pronunciation Modeling for Improved Spelling Correction. In proceedings of 40th ACL, July 2002, pp. 144-151.