Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence

On Correcting Misspelled Queries in Email Search

Abhijit Bhole and **Raghavendra Udupa**
Microsoft Research
9 Lavelle Road, Bangalore - 560001
{a-abhina, raghavu}@microsoft.com

Abstract

We consider the problem of providing spelling corrections for misspelled queries in Email Search using user's own mail data. A popular strategy for general query spelling correction is to generate corrections from query logs. However, this strategy is not effective in Email Search for two reasons: 1) query log of any single user is typically not rich enough to provide potential corrections for a new query 2) corrections generated using query logs of other users are not particularly useful since the mail data as well as search intent are highly specific to the user. We address the challenge of designing an effective spelling correction algorithm for Email Search in the absence of query logs. We propose SpEQ, a Machine Learning based approach that generates corrections for misspelled queries directly from the user's own mail data.

Introduction

We consider the problem of correcting misspelled user queries in Email Search. Automatic query spelling correction can contribute significantly to a positive user experience and an increased productivity in Email Search. However, there have been few efforts to address this pervasive problem. With the exception of Gmail [1] there is no mail client in our knowledge that providing query spelling corrections. This is in contrast with Web Search where every search engine provides a reasonably effective query spelling correction feature using rich logs of session data produced by hundreds of millions of users (Cucerzan and Brill 2004; Duan and Hsu 2011; Chen, Li, and Zhou 2007; Ahmad and Kondrak 2005).

The success of query log based spelling correction in Web Search suggests its adoption to Email Search. However, there are two significant drawbacks. Firstly, the effectiveness of query log based approaches in Web Search is largely due to the availability of rich query logs which embody the "the intelligence of crowd". In contrast, query log of any user in Email Search is substantially smaller in size and is unlikely to provide a good coverage of that user's future queries. Secondly, search intent as well as the document collection that needs to be searched are highly specific to the user in Email Search. Unlike Web Search where the target document collection is the Web and is the same for all users, in Email Search the user is searching the relatively small, primarily private and substantially more familiar collection of emails. Therefore, the search intent of email users is very specific to and scoped by their mail data. Consequently, aggregated query logs of multiple users are unlikely to be useful in correcting the misspelled query of an individual user.

Personalized and context specific corrections are important in Email Search. A scientific collaborator of the eminent computer scientist *Peter Norvig* might type the query "noarvig flies" while intending to type "norvig files" whereas an entomologist who is searching her mail folder for mails on the subject of black fly season in the Alaskan city of *Noorvik* might also type "noarvig flies". As this example illsutrates, spelling correction algorithms that leverage user's mail data can provide personalized corrections to the user. A generic spelling correction algorithm using an open domain dictionary or trained with Web Search query logs would not be helpful as the query has to be corrected differently for different users. Further, even on the same mailbox, the information need of a user can be different for the same misspelled query. To give an example, "july patents" is a more relevant correction to the misspelled query "july parents" than "julie parents" when the user is searching the folder containing mails she has received from her company's intellectual property attorneys. However, "julie parents" would be more appropriate when searching in her personal folder or if she recently received a mail from Julie's parents. Thus, the same query needs to be corrected differently when the user is searching in different folders and at different instances of time.

Contributions

Motivated by the above discussion, we develop SpEQ, a Machine Learning based spelling correction algorithm that has the following capabilities:

- Corrections are given for misspelled queries even when the user's search history is not available
- Corrections are personalized for every user
- Corrections are context specific (i.e. relevant to the current mail state of the user)

[1] http://www.gmail.com

- Mailboxes of vastly varying sizes are handled equally well with respect to efficiency and effectiveness.

We do a systematic empirical study of SpEQ on two public domain email corpora. We compare the performance of SpeQ against two baselines - a source-channel based ranking approach and the spelling correction feature of a popular email service. Our experiments show that SpEQ gives significant improvement in accuracy over the two baselines.

Details

Given the query Q_t and its context $\mathcal{C}$, we employ a ranking framework for spelling correction:

$$Q_c = \arg \max_{Q \in g(Q_t;\mathcal{C})} \sum_{i=1}^{m} w_i \Phi_i \left(Q; Q_t, \mathcal{C}\right) \quad (1)$$

The key components of the ranking framework are a) candidates generator g and b) feature functions $\{\Phi_i\}$.

Candidates Generation We employ a candidate scoring function S and Dynamic Programming to produce a reasonably small number of candidates[2] from the user's mail data. S assigns a score to each candidate using *Idf* and edit distance of constituent tokens. The Dynamic Programming algorithm builds candidates for the whole query from the candidates of the constituent tokens.

Feature Functions We define a set of feature functions $\{\Phi_i\}$ that score the candidate based on a) lexical similarity with the query, b) query context, c) content of the mailbox, and d) search context. Our feature functions are computationally efficient and can be computed for any query, user, language and domain.

We measure lexical similarity using Levenshtein edit distance and fuzzy similarity (Udupa and Kumar 2010). We encode the query context using smoothed bigram language model score and normalized *Idf* score of the candidate.

Content based features aim to rank those candidates higher which contain tokens from subject lines or contacts corresponding with the user or keyphrases from mail body:

- **SubjectCover**: The fraction of mails containing at least one candidate token in its subject line.
- **SubjectMatch**: The maximum of fraction of candidate tokens appearing in any single subject line.
- **KeyPhraseCover**: The score of the maximal set cover of candidate tokens using key phrases. The set is scored by the sum of Jaccard similarity coefficient of all key phrases in it with the candidate, counting a candidate token only once if covered by many key phrases in the cover.
- **ContactMatch** : The fraction of tokens in candidate which are present in at least one user's contact.
- **TooManyMails** A boolean function to evaluate whether the candidate fetches too many emails. If so it is unlikely to be useful to the user.

Contextual feature functions encode the current state of the mail box:

- **RecentSubjectMatch** Similar to SubjectMatch but evaluated in a window of recent emails
- **RecentContactMatch** Similar to ContactMatch but evaluated in a window of recently interacted contacts
- **FrequentContactMatch** Similar to ContactMatch but evaluated on select contacts having more than a certain mails in his current folder.

Data We used two public domain email corpora for our experiments:

- **PALIN**: About $15,000$ mails of the former Alaskan governor Sarah Palin (from the period December 2006 to September 2008 made public in the year 2011) in two folders *Inbox* and *Sent Mail*. [3]
- **ENRON**: All mails (about 2100) in the *All Documents* folder of *rogers-b*, an employee of Enron corporation.

We compiled the query set with the help of volunteers. There were totally 517 misspelled queries in the query set.

Experimental Study We trained the ranker model on the training data set consisting of queries and their corresponding desired corrections. As the feature functions are generic and domain independent, it suffices to train the model offline on the training data only once. The same model goes into the mail client of all users, but the individual user's mail data is used to generate the candidates and compute the feature functions. We did several experiments investigating the effectiveness of SpEQ and found that P@1 of SpEQ was 89.34 on PALIN and 97.17 on ENRON. We get at least 6.4% improvement in P@1 over the baselines. We did a feature ablative study and found that each class of feature functions contributed to the effectiveness of our approach.

References

Ahmad, F., and Kondrak, G. 2005. Learning a spelling error model from search query logs. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, 955–962. Stroudsburg, PA, USA: Association for Computational Linguistics.

Chen, Q.; Li, M.; and Zhou, M. 2007. Improving query spelling correction using web search results. In *EMNLP-CoNLL*, volume 7, 181–189.

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

Duan, H., and Hsu, B.-J. P. 2011. Online spelling correction for query completion. In *Proceedings of the 20th International Conference on World Wide Web*, WWW '11, 117–126. New York, NY, USA: ACM.

Udupa, R., and Kumar, S. 2010. Hashing-based approaches to spelling correction of personal names. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, 1256–1265. Association for Computational Linguistics.

[2] 1000 in our experiments.

[3] We used the version of the data released by the news site msnbc.com originally in PDF format.