Context Dependent Bag of Words Generation

[1]Swapnil Ashok Jadhav, [2]D. V. L. N. Somayajulu, [3]S.Nagesh Bhattu, [4]R.B.V. Subramanyam
Department of Computer Science and Engineering, NIT Warangal-506004, India
[1]*saj1919@hotmail.com*, [2]*soma@nitw.ac.in*, [3]*nageshbhattu@gmail.com*, [4]*rbvs66@gmail.com*

P. Suresh
E-Commerce Research Lab, Education and Research
Infosys Limited India, Mysore-570018,
Suresh_P01@infosys.com

***Abstract* — Query spelling correction is a crucial component in modern text mining systems such as Question-answering systems and Sentiment Analysis systems where noise can affect the query matching score. In many existing query matching systems Bag of Words (BoW) generation method is used to generate candidates for noisy words. But in these systems candidate generation do not depend upon context of a query sentence. BoW count for each noisy word may vary and selecting correct candidates from such list is not easy and may result in wrong selection. With our context dependent BoW generation method very few but highly probable candidates are generated which are easy for look up and process of query spelling correction would be easier and efficient.**

Keywords — text mining, sentiment analysis, noise removal, spelling correction, faq retrieval, question answering, information retrieval, nlp.

I. INTRODUCTION

Numbers of Internet and mobile users are rapidly increasing. Around 500 million people are accessing these services regularly. People can communicate anytime and anywhere. Text messaging is the prime mode of communication. It is done through traditional SMS systems or through social media services like Twitter and Facebook.

There is a limit for how many characters you can send during each text communication. It is 160 for SMS services and 140 for tweets. For other social media services like Facebook and Google+ there are no such limit, but users tend to write short messages to convey the information. To overcome this limitation on message length users have started using slangs and short forms for real words and sentences. These slangs can be well defined or user generated.

Users generally leaves some characters (consonants most of the time) from words, but phonemically they can be guessed. Words or phrases such as *"omg" (oh my god), "2moro" (tomorrow)* and *"thnx" (thanks)* which may not be found in standard English directory are widely used by web users. Also there are tendencies of spelling errors due to small keypad and small touch screens. Today communication has become considerably secure over communication channels but in few cases noise can alter the text message.

All the issues stated above affect the FAQ Retrieval systems and Web Sentiment Analysis systems in a great way. In FAQ Retrieval (also known as Question - Answering system), question sent by user is matched to existing questions from the dataset and nearest matching question is found out and answers to that question is returned to the user. If there are text errors, then getting the nearest match could be wrong if we use conventional BoW method. Also, in web sentiment analysis, user's sentiments about particular product or person or business are analyzed from their statuses and comments. For analysis each word is categorized in positive, negative or neutral. But if errors exist then many words do get categorized in neutral as they are not dictionary words. Hence, sentiment analysis score changes and it may produce wrong analysis.

In this paper we present Context Dependent Bag of words method which generates efficient list of candidates for each noisy words which are context dependent to other candidate words and also they are ranked according to their possibility of getting selected as a correct word. These lists are then forwarded to text matching systems to get correct form of the noisy text (In case of Web Sentiment Analysis) or question matching system where candidate list of questions is generated to get nearest matching question. Our work contributes in text mining systems where query preprocessing is required to remove spelling errors and incompatible formatting so that mining and eventually text analysis system works to their potential. In this paper we not focusing on query normalization i.e. generating sentence with few or no spelling errors but only part of this procedure, to generate list of ranked candidates efficiently and with high recall.

The rest of the paper is organized as follows. Section II describes the relevant prior work in the area of noise removal and spelling error correction and also about evolution from traditional systems to modern spelling correction systems and also about our specific contributions. In Section III we give the problem formulation. Section IV describes System implementation with G2P (Grapheme to Phoneme Conversion) model and the Context Dependent BoW Algorithm which ranks the best matching candidates for each noisy word for a given text query. Section V provides details about our experiments, results and their analysis. Finally we conclude in Section VI.

978-1-4673-6217-7/13/$31.00 ©2013 IEEE

II. PRIOR WORK

There has been considerable work done in area of spelling correction [18]. But there are some types of spelling errors such as splitting-concatenation errors and replacing abbreviations with its correct form (from multiple possibilities like *"PM"* can be corrected as *"Prime Minister"* or simply just as *"pm"* for its *"Prime Meridian"* form) which never got that much attention and not considerable research work has been done in these areas. Traditional spellers focused on dealing with non-word errors caused by misspelling a known word as an invalid word form. Distance measures like LCS Distance and Levenshtein distance [19] are used to find similarity between noisy word and correct candidate words.

Brill and Moore in their work explained that a better statistical error model is important for improving a speller's accuracy [20]. Error model and n-gram language model are identified as two critical components as a Statistical generative models. But building such an error model requires a large set of word correction pairs, which is expensive to obtain as they are built manually.

This problem was modeled to automatically discover the misspelled or corrected word pairs over the web [21]. With the evolution of the Web, the research on spelling correction has received much more attention. Many research challenges are raised, which are non-existent in traditional settings of spelling correction. There are many more types of spelling errors in queries, such as misspelling, concatenation - splitting of query words, and misuse of legitimate yet inappropriate words. Research in this area requires large web corpus and list of queries [22, 23, 24], training phrase-based error model from click through data [25] and developing additional features for feature dependent spellers [26]. However, one important challenge is under addressed in these approaches, i.e., correcting splitting and concatenation errors.

Query alteration or refinement is a broader topic which naturally depends upon query spelling correction. Besides correcting the misspelled query, query alteration or refinement also need to modify the ineffective query so that it could be more suitable for the query matching or searching or sentiment analysis and most importantly refinement should be done with context dependency. For this purpose, many research topics have been studied. Query expansion expands the query with additional terms to enrich the query formulation [27, 28, 29]. Query segmentation divides a query into semantically meaningful sub-units [30, 31]. There is research attempt [33] to use a unified model to do a broad set of query refinements such as correction, segmentation and even stemming. It has very limited ability for query correction. For example, it only allows one letter difference in deletion, insertion, substitution errors. Other query reformulation methods intend to replace the inappropriate query terms with effective keywords to bridge the vocabulary gaps[32].

Query spelling correction task can be approached with NLP and is similar to tasks like such as speech recognition and machine translation. HMM has been found very useful in these tasks [34, 35]. A significant work has been done using generalized HMM model for query spelling correction [36], and all major spelling errors are approached with unified system of spelling correction. gHMM model is very effective for the task of query spelling correction by discrete training and discriminative approach. Splitting-concatenation errors are removed significantly but there is no mention of these errors on in-word spelling errors as explained in introduction. Else it is the best approach to normalize noisy query with multiple errors.

BoW method has been explained in detail by Langer et al.[3] and Kothari et al.[2]. Both these papers deal with SMS based FAQ retrieval system and explain about candidate generation methods for noisy query. BoW generated for each noisy word varies in length as it depends upon threshold value. For candidate generation domain dictionary and synonym dictionary is build from the given corpus and fuzzy matching is used to match candidates with given noisy word using traditional Edit-Distance and LCS distance methods.

There are many spell-checkers available in the market or online. SpellChecker[1] is one of free online spelling detection and correction tool available. Like other spell-checkers in this tool also context is not checked while generating candidates. A simple example to explain this deficiency is as follows. If we type "hw are you ?", it correctly detects that "hw" is a spelling mistake. Candidates generated for this noise are "he", "h", "w", "hew" and "haw". But the correct replacement should have been "how" which is not included in above list.

A. Our contribution

We introduce Context Dependent BoW method which is improved over BoW method explained in Langer et al. [3] and Kothari et al. [2]. We apply BoW method to find out candidate list for noisy words using less threshold value and then multiple ranking method is used to rank candidates according to similarity measures, number of occurrences in given dataset and phonemic representations matching. Using our approach we are able to produce short and efficient BoW lists of context dependent candidates for noisy words in a given query for a given dataset which can be used later for query normalization (here it means that generating sentence without spelling errors) from sentiment analysis or FAQ retrieval systems.

III. PROBLEM FORMULATION

We see input text S as a sequence of tokens $S = s_1 , s_2 , \ldots , s_n$. Our goal is to find the list of candidates for each s_i which best matches with it and context dependents to the candidate s_j, where j varies from 1 to n and $i \neq j$. Text message may have misspellings, slangs and other distortions, which needs to be taken care of while performing the match.

In the preprocessing stage, we develop a Domain dictionary D consisting of all the terms that appear in the corpus Q. We add

[1] http://www.spellchecker.net/spellcheck/

lemmatized (i.e. Basic forms Exm. "*Make"* is basic form of "*Making"* and "*Made"*) forms of terms to the dictionary. For each sentence S_i in corpus word tokens are w_1, w_2....,w_n. Entry w_i and w_j is added to bi-gram list where $i \neq j$. Also we prepared a slag dictionary[2] Qs to match and replace possible slangs. For slangs t' in Qs exactly matching to noisy word, we include their full forms in BoW.

For each term t in the dictionary Q and each SMS token s_i, we define a similarity measure $\alpha(t, s_i)$ [2, 3] that measures how closely the term t matches the SMS token s_i. Terms t are chosen if weight function [2, 3] $\omega(t, s_i) > y$, where y is the threshold value defined for the problem.

After getting all such candidates c, our ranking method is applied and new ranked BoW is generated. From this ranked BoW such c' are retained if f(c', BoW(1 to n)) > x, where x is a threshold value defined for function f which finds with how many BoW out of n, c' is compatible. Compatibility is counted as 1 for BoW(i) and c' if c' occurs with at least one term t'' of BoW(i) and x < n. After this step we get BoW for noisy words, we term them as CD-BoW (Context Dependent BoW) which contains fewer candidates but has high probability of having correct candidate.

IV. System Implementation

A. Building G2P model

One of the ranking method applied in CD-BoW algorithm depends upon the Grapheme to Phoneme (G2P) translation model. This model is built using Moses [8].

To build the model source language is taken as grapheme words i.e. words given in CMU dictionary and target language is taken as phonemes i.e. phonetic representation of words from CMU dictionary. Total 133,247 English words and their phonetic representations are used as training corpus. Following the procedure given in Moses website [8], MosesDecoder-Master toolkit[3], IRSTLM[4] and GIZA++[5] are used to build HMM for language translation. This model then used to predict the phonetic representation of noisy word at runtime using moses API, if that words is not available in CMU dictionary.

To reduce runtime of finding phonemic representation of noisy word we binarized the language model and also applied tuning to increase the accuracy of the model. Total 13,324 words are used for tuning the language translation model.

B. Context Dependent Bag Of Word (CD-BoW) generation

As explained in problem formulation and in Langer et al.[3] and Kothari et al.[2] BoW is generated for each noisy word in given noisy text. Each list may consist of >100 candidates t from Q or t' from Qs as threshold is set low for weight function.

Now we apply ranking to each BoW separately. We apply three rankings to the same list and each of these rankings is added with weighted function to get final score which defines the final rankings. First Similarity Measure Ranking (SMR) is applied where c'(j) of list BoW(i) are ranked according to their similarity measure i.e weight function $\omega(t, s_i)$. Number of Occurrences Ranking (NOR) is used next, in which c'(j) are ranked according to the number of its occurrences in a dataset Q. Third and final ranking is applied on the BoW(i) is Phonemic Similarity Ranking(PSR). In this ranking method phonemic representation of c'(j) is found out from CMU dictionary and that of s_i is found out from a built language model (G2P) as explained in section IV-A.

Algorithm: CD-B*oW Generation*

Input
For a text S, sequence of tokens are $s_1, s_2, \ldots, s_n$.
BoW(i) is generated for each s_i
c'(j) is candidate in BoW(i), where j = 1 to |BoW(i)|
Output
CD-BoW(i) for each s_i

Start
1. **for** all BoW(i) given i = 1 to n **do**
2. Rank BoW(i) with SMR. c'(j) gets rank r(SMR)(j)
3. Rank BoW(i) with NOR. c'(j) gets rank r(NOR)(j)
4. Rank BoW(i) with PSR. c'(j) gets rank r(PSR)(j)
5. Rank_Score(c'(j)) = p * r(SMR)(j) + q * r(NOR)(j) + s * r(PSR)(j)
6. Rank c'(j) by Rank_Score(c'(j))
7. End **for**

8. **for** all BoW(i) given i = 1 to n **do**
9. count(c'(j)) = 0
10. **for** all BoW(k) given k = 1 to n, k != l **do**
11. **if** c'(j) and c'(l) are bi-grams **then**
12. - count(c'(j)) = count(c'(j)) + 1;
13. End **for**

14. **if** count (c'(j)) > G *|BoW(i)|) **then**
15. - c'(j) is added to CD-BoW(i)

16. Final_Score(c'(j)) = m * Rank_Score(c'(j)) + n * count(c'(j))
17. Rank c'(j) by Final_Score(c'(j)) for CD-BoW(i)
18. End **for**
End

Ranking score for each c'(j) is found out from the formula given in the CD-BoW Generation Algorithm and they are ranked according to it. To get the final list of CD-BoW(i) we check for context dependency of candidates for noisy words within a single noisy query i.e. if c'(j) from BoW(i) is occurring with at least one

[2] http://onlineslangdictionary.com/
[3] https://github.com/moses-smt/mosesdecoder
[4] The IRST Language Modeling Toolkit http://hlt.fbk.eu/en/irstlm
[5] GIZA++ is a statical machine translation toolkit http://code.google.com/p/giza-pp/

c'(l) from BoW(k) where i ≠ k, then as shown in CD-BoW Generation Algorithm, we include such c'(j) to CD-BoW(i).

V. EXPERIMENTS

To validate effectiveness of the CD-BoW method over traditional BoW we generate test cases of the noisy texts. Test cases are generated using corpus *"eng.xml"* (FIREC) from FIRE 2012[6] – SMS Based FAQ Retrieval System. It contains 7497 English sentences. Also another corpus is prepared for similar experiments and for verification and validity of our approach. Using the "twitter4j" Java api 6000 English tweets are collected on the topic *"barack obama"* (BOC). From these 4654 correct tweets are separated manually and treated as training data and remaining as a test data to analyze the effectiveness of our approach.

Test cases are generated for single and double, character spelling error and character missing cases programmatically from corpus. Phonemic errors are put manually by test subjects in the separate test case. Finally test cases having all above errors are generated. To generate BoW we used Stanford-NLP[7] Java api for lemmatization, named entity recognition and moses API [8] to generate phonemic representation of noisy words from G2P model described in section IV-A.

Table 1: %Accuracy for *"FIREC"*

Test Cases	Top 1	Top 3	Top 5	Top 10	Top 15
1 character spelling errors	79.38	92.78	95.05	97.11	97.31
1 character missing errors	75.67	91.95	95.67	96.70	96.90
2 characters spelling errors	78.76	90.30	91.75	93.40	93.40
2 characters missing errors	72.98	85.77	89.89	92.78	93.19
Phonemic errors	91.34	95.87	97.11	97.52	97.52
Spelling Errors on normal text (75% Errors)	82.88	91.34	93.60	94.43	94.43
Spelling Errors on phonemic text (75% Errors)	79.79	88.24	91.34	93.40	93.40

Table 2: %Accuracy for *"BOC"*

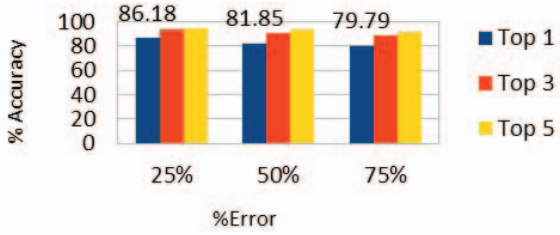
Test Cases	Top 1	Top 3	Top 5	Top 10	Top 15
1 character spelling errors	81.48	95.52	96.34	98.46	98.14
1 character missing errors	79.53	93.98	97.37	98.14	98.96
2 characters spelling errors	83.67	94.37	96.75	97.67	97.89
2 characters missing errors	78.81	88.65	92.33	95.77	96.29
Phonemic errors	94.77	97.72	98.04	98.89	99.37
Spelling Errors on normal text (75% Errors)	86.82	95.88	95.58	96.34	97.07
Spelling Errors on phonemic text (75% Errors)	84.57	93.89	94.75	96.10	96.86

In CD-BoW algorithm p, q and s values are set to 0.4285, 0.2857 and 0.2857 which maximizes the result probability. Also G is set to 0.4. In calculation of Final_Score m and n are used which set to 0.81 and 0.19 which are also experimentally found values which maximizes resultant probability.

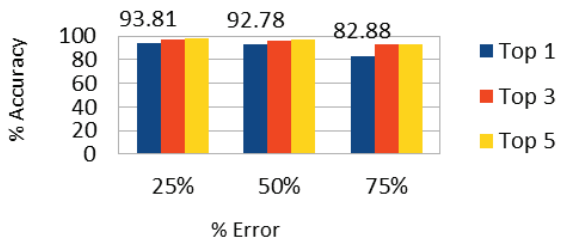
We calculated probabilities of occurrence of correct candidate c" from CD-BoW(i) at various positions for various test cases as given in the Table 1 and Table 2 for corresponding corpus. In Tables, *"Top 1"* means getting correct candidate at position 1. *"Top 3"* means correct candidate at position 1, 2 or 3 and similarly for other column headings.

From above tables we can observe that results are 2-4% better for *"barack obama"* corpus. The reason is that this corpus is more compact and highly subjective and context dependent as all the tweets are for only one topic, unlike FIRE-2012 corpus where QA sentences are with varied topics. Hence, context dependency part in the CD-BoW algorithm from line 8 to line 18 affects the resultant list of candidates positively and improves the accuracy. Hence, our approach is applicable more in Web Sentiment Analysis where one topic is given (For example *"nokia lumia 920" or "syria crisis"*) and users reactions about that topic are to be studied. Also such system will be useful if it is deployed for each topic in question-answering system.

Graph 1 : CD-BoW Accuracy for 75% Spelling Errors



Graph 2 : CD - BoW Accuracy for 75% Spelling Errors



Real time errors are analogous to phonemic errors i.e. Users tend to write text query with words which are phonetically similar to the actual words. E.g. "How are you?" is written as *"hw r u ?"* or in slang language only *"hru"*. Our approach shows significant accuracy in "Phonemic Errors" test case and is better than other test cases. It shows that our approach is feasible with real time SMS queries or comments. But if phonemic errors

[6] http://www.isical.ac.in/~fire/2012/index.html
[7] http://nlp.stanford.edu/index.shtml

exist with noise in the text query or if noise is present over phonemic errors then accuracy decreases compared to that of only phonemic errors as seen in Graph 1 and Graph2 below.

Let us consider the example of noisy sentence, *"tha cntry knwn as land of midn8t sun"*. As we can see from the Table 3 that number of candidates for each word in noisy query is less for CD-BoW than BoW generation method.

Count of candidates for "midn8t" is tricky. It is less in BoW than in CD-BoW. As threshold is considerably high in BoW (which is desired in BoW according to our implementations and understandings) than CD-BoW, nearest matching words do not get listed. Among such unlisted words correct word may reside. In such cases accuracy is compromised. To avoid such cases in CD-BoW Algorithm, less threshold value is used in BoW routine and then CD-BoW is applied.

Table 3: Number of candidates for noisy Query

Words	*tha*	*cntry*	*knwn*	*as*	*land*	*of*	*midn8t*	*sun*
BoW	181	167	60	183	416	24	5	102
CD-BoW	16	13	15	16	16	14	11	15

We continued our experiment and found out original % accuracies for noisy dataset of another different 50 sentences each for BOC and MC (Additional Corpus on topic "Microsoft" comprises of ~9500 queries) . Then we can see the improvement due to our query normalization.

Table 4: %Improvement in spelling accuracy

Data Sets	Without Query Normalization	With Query Normalization	Improvement
BOC	78.64%	91.8%	13.16%
MC	76.74%	92.34%	15.6%

There are limitations to this BoW approach as well as CD-BoW approach. With this approach spelling errors, use of slag words and phonemically similar words are rectified, but some errors such as detecting abbreviation (Exm. *"PM"* stands for *"Prime Minister"* and also for *"Prime Meridian"*) and concatenation - splitting errors (Exm. *"base ball"* should be written as *"baseball"*) are not dealt as while modeling BoW within CD-BoW algorithm as we have only concentrated on technique to reduce the number of candidates for noisy words in a given query, though these errors do affect the results in some considerable amount, as wrong list of candidates will be generated in these cases and matching will not be correct.

VI. Conclusion

In recent times SMS based Question Answering services and web sentiment analysis are on the rise. But noise can be an affecting factor in providing correct and efficient services to the users. In this paper we gave an Context Dependent Bag of Words (CD-BoW) generation algorithm which generates short and effective lists of candidates for noisy words which are ranked and context dependent, which in effect can be used in FAQ retrieval and Web Sentiment Analysis systems, for query normalization where response time and accuracy matters.

Acknowledgment

We would like to thank our whole "Web Sentiment Analysis" Team of NIT Warangal, CSE department and researchers from Infosys Limited for their consistent guidance and support.

References

[1] Kai Wang, Zhaoyan Ming and Tat-Seng Chua. A Syntactic Tree Matching Approach to Finding Similar Questions in Community-based QA Services. SIGIR '09 Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval.

[2] Govind Kothari, Sunit Negi, Tanveer A. Faruquie, V.T. Chakaravarthy and L.V. Subramaniam. SMS based Interface for FAQ Retrieval. ACL '09: Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2.

[3] Akhil Langer, Rohit Banga, Ankush Mittal and L. V. Subramaniam. Variant Search and Syntactic Tree Similarity Based Approach to Retrieve Matching Questions for SMS queries. AND '10: Proceedings of the fourth workshop on Analytics for noisy unstructured text data.

[4] Lawrence R. Rabiner. A Tutorial on Hidden Markov Model and Selected Applications in Speech Recognition. Proceedings of the IEEE, Feb 1989.

[5] Monojit Choudhury et al. Investigation and Modeling of the Structure of Texting Language . International Journal of Document Analysis and Recognition (IJDAR) December 2007, Volume 10.

[6] Paul Taylor. Hidden Markov Models for Grapheme to Phoneme Conversion. Proc. Interspeech 2005

[7] Deepak P and L Venkata Subramaniam . Correcting SMS Text Automatically. CSI communications, May 2012.

[8] Moses.(www.statmt.org/moses/)

[9] Rudy Schusteritsch, Shailendra Rao, Kerry Rodden. 2005. Mobile Search with Text Messages: Designing the User Experience for Google SMS. CHI, Portland, Oregon.

[10] Sunil Kumar Kopparapu, Akhilesh Srivastava and Arun Pande. 2007. SMS based Natural Language Interface to Yellow Pages Directory, In Proceedings of the 4th International conference on mobile technology, applications, and systems and the 1st International symposium on Computer human interaction in mobile technology, Singapore.

[11] Eunghyun Byun, Seung-Wook Lee, Young-In Song, Hae-Chang Rim. 2008. Two Phase Model for SMS Text Messages Refinement, Association for the Advancement of Artificial Intelligence. AAAI Workshop on Enhanced Messaging

[12] Aiti Aw, Min Zhang, Juan Xiao, and Jian Su. 2006. A phrase-based statistical model for SMS text normalization, In Proceedings of COLING/ACL, pages 33−40.

[13] Catherine Kobus, Franois Yvon and Graldine Damnati. 2008. Normalizing SMS: are two metaphors better than one? In Proceedings of the 22nd International Conference on Computational Linguistics, pages 441−448 Manchester.

[14] Wayne Xin Zhao, Jing Jiang, Jing He, Yang Song, Palakorn Achanauparp, Ee-Peng Lim and Xiaoming Li. Topical keyphrase extraction from Twitter. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT'11) (long paper), pages 379-388, 2011.

[15] Wayne Xin Zhao, Jing Jiang, Jianshu Weng, Jing He, Ee-Peng Lim, Hongfei Yan and Xiaoming Li. Comparing Twitter and traditional media

using topic models. In Proceedings of the 33rd European Conference on Information Retrieval (ECIR'11) (full paper), pages 338-349, 2011.

[16] Jansen, Bernard, Zhang, Mimi, Sobel, Kate, and Chowdury, Abdur. (2009). Twitter Power: Tweets as Electronic Word of Mouth. Journal of ASIS&T, 60(9), 1-20.

[17] Zhang, Mimi, Jansen, B. J., & Chowdhury, A. (2011). Business engagement on Twitter: A path analysis. Electronic Markets, 21(3), 161-175.

[18] K. Kukich. Techniques for automatically correcting words in text. ACM computing surveys, 24(4), 1992.

[19] Levenshtein, V I. Binary codes capable of correcting deletions, insertions, and reversals. In Soviet Physics Doklady, 10(8), 707-710, 1966.

[20] E. Brill and R. Moore. An improved error model for noisy channel spelling correction. In Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics, Hong Kong, 2000.

[21] C. Whitelaw, B. Hutchinson, G. Chung, and G. Ellis. Using the web for language independent spellchecking and autocorrection. In EMNLP, pages 890–899. ACL, 2009.

[22] Q. Chen, M. Li, and M. Zhou. Improving query spelling correction using web search results. In EMNLP-CoNLL, pages 181–189. ACL, 2007.

[23] S. Cucerzan and E. Brill. Spelling correction as an iterative process that exploits the collective knowledge of web users. In EMNLP, 2004.

[24] F. Ahmad and G. Kondrak. Learning a spelling error model from search query logs. In HLT/EMNLP. The Association for Computational Linguistics, 2005.

[25] X. Sun, J. Gao, D. Micol, and C. Quirk. Learning phrase-based spelling error models from clickthrough data. In Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, ACL '10, pages 266–274, Stroudsburg, PA, USA, 2010.

[26] J. Gao, X. Li, D. Micol, C. Quirk, and X. Sun. A large scale ranker-based system for search query spelling correction. In C.-R. Huang and D. Jurafsky, editors, COLING, pages 358–366. 2010

[27] Jinxi Xu and W. Bruce Croft. Query expansion using local and global document analysis. In Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval, SIGIR '96. ACM, New York, NY.

[28] Yonggang Qiu and Hans-Peter Frei. Concept based query expansion. In Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval, SIGIR '93. ACM, New York, NY, USA, 160-169.

[29] Mandar Mitra, Amit Singhal, and Chris Buckley. Improving automatic query expansion. In Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval, SIGIR '98.

[30] B. Tan and F. Peng. Unsupervised query segmentation using generative language models and wikipedia. In Proceeding of the 17th international conference on World Wide Web, WWW '08, pages 347–356. 2008.

[31] Y. Li, B.-J. P. Hsu, C. Zhai, and K. Wang. Unsupervised query segmentation using clickthrough for information retrieval. In Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval, SIGIR '11, pages 285–294. 2011.

[32] Xuanhui Wang, ChengXiang Zhai. Mining Term Association Patterns from Search Logs for Effective Query Reformulation. In CIKM'08. 479-488.

[33] J. Guo, G. Xu, H. Li, and X. Cheng. A unified and discriminative model for query refinement. In Proceedings of the 31st annual international ACM SIGIR, SIGIR '08, pages 379–386. 2008.

[34] Stephan Vogel, Hermann Ney, and Christoph Tillmann. HMM-based word alignment in statistical translation. In Proceedings of the 16th conference on Computational linguistics, Volume 2 (COLING '96). Stroudsburg, PA, USA, 836-841.

[35] B.H. Juang. Hidden Markov models for speech recognition. In Technometrics, Vol. 33, No. 3, Aug., 1991.

[36] Yanen Li, Huizhong Duan, and ChengXiang Zhai. 2012. A generalized hidden Markov model with discriminative training for query spelling correction. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval* (SIGIR '12). ACM, New York, NY, USA, 611-620.