A Comparative Study of Bing Web N-gram Language Models for Web Search and Natural Language Processing

Jianfeng Gao, Patrick Nguyen, Xiaolong Li, Chris Thrasher, Mu Li, Kuansan Wang
Microsoft Research
One Microsoft Way
Redmond, WA 98052 USA
{jfgao; panguyen; xiaolli; cthrash; muli; kuansanw}@microsoft.com

ABSTRACT

This paper presents a comparative study of the recently released Microsoft Web N-gram Language Models (MWNLM)[1] on three web search and natural language processing tasks: search query spelling correction, query reformulation, and statistical machine translation. MWNLM, as well as the corresponding web services, called Microsoft Web N-gram Services, are much more accessible and easier to use than the previously released text corpora used for large language model training, including the LDC English Gigaword corpus and the Google Web 1T N-gram corpus, because the Microsoft Web N-gram Services provide the access to the smoothed *n*-gram probabilities based on a set of language models trained from the different text fields from the web documents as well as search queries. Our results show that MWNLM outperform the *n*-gram models trained on the Gigaword corpus and the Google Web 1T N-gram corpus on all the three tasks. In particular, the significant improvements on search query spelling correction and search query reformulation, resulting from MWNLM, demonstrate the benefit of training multiple language models on different portions of web data and search queries in a principled way with zero count cutoff.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval;

General Terms

Experimentation, Measurement

Keywords

Language Model, N-gram, Spelling Correction, Query Reformulation, Statistical Machine Translation

[1] An earlier version of MWNLM is described in Huang et al. [10]. This paper provides updated information regarding the most recent development of MWNLM.

1. INTRODUCTION

The goal of a statistical language model (LM) is to predict the probability of a word string. This is fundamental to a wide variety of web search and natural language processing (NLP) applications. The technique that is still dominating the research communities is the *n*-gram model, thanks to its simplicity and effectiveness. Let $w_1^L = (w_1, \dots, w_L)$ denote a string of L words over a fixed vocabulary. An *n*-gram language model assigns a probability to w_1^L according to

$$P(w_1^L) = \prod_{i=1}^{L} P\left(w_i | w_1^{i-1}\right) \approx \prod_{i=1}^{L} P\left(w_i | w_{i-n+1}^{i-1}\right) \qquad (1)$$

where the approximation is based on a Markov assumption that each word depends only upon the immediately preceding *n*-1 words.

N-gram models have been extensively studied in both the research communities and the industry from different perspectives for decades. While the people in the research communities, such as natural language processing, speech and information retrieval, try to figure out a richer and smarter model via better smoothing [6] or capturing more linguistic structures [5]; industry people recently found that simply using more data is far more effective [e.g., 2]. For example, the Google machine translation group trades the mathematical soundness to the scalability of LMs, and use the *n*-gram models that are not properly normalized but can be efficiently trained on very large amounts of text corpora [2].

In this paper we strike a better balance between the mathematical soundness and scalability, and demonstrate that it is possible to scale the *n*-gram LMs without scarifying their nice probabilistic properties. We present a distributed LM platform, based on which a set of web scale smoothed LMs are trained on different portions of web document fields, such as body text, title text and anchor text, as well as search queries. We have also developed web services, called Microsoft Web N-gram Services [13], to make these models accessible to the research communities. We demonstrate the effectiveness of these web scale *n*-gram LMs and the use of the services through three web search and NLP applications: search query spelling correction, query reformulation, and statistical machine translation. We show that in comparison with other text corpora previously released for large LM training, including the LDC English Gigaword corpus and the Google Web 1T N-gram corpus, MWNLM, as well as the corresponding web services, are much easier to use because users can access the smoothed probability

Dataset	Body	Anchor	Title	Query
Total tokens	1.3T	11.0B	257.2B	28.1B
Unigrams	1.2B	60.3M	150M	251.5M
Bigrams	11.7B	464.1M	1.1B	1.3B
Trigrams	60.0B	1.4B	3.1B	3.1B
4-grams	148.5B	2.3B	5.1B	4.6B
5-grams	238.0B	N/A	N/A	N/A
Size on disk#	15.8TB	183GB	395GB	393GB

N-gram entries as well as other statistics and model parameters are stored.

Table 1: Statistics of the Microsoft Web *n*-gram LMs collection (count cutoff = 0 for all models).

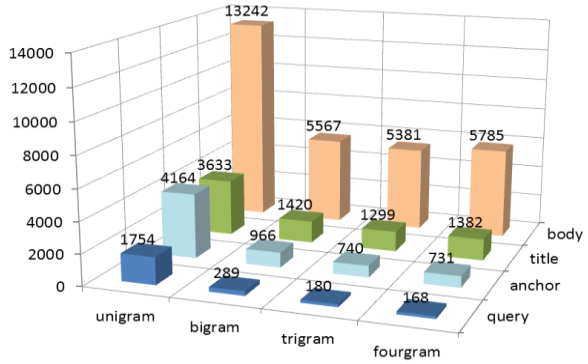

Figure 1. Perplexity results on test queries, using *n*-gram LMs with different orders, derived from different data sources.

of any word string directly without wondering how to train the LMs from the large amounts of text, an engineering intensive task by itself. We also show that MWNLM outperform the *n*-gram models trained on Gigaword corpus and Google Web 1T N-gram corpus on all the three tasks. In particular, the significant improvements on search query spelling correction and query reformulation, resulting from MWNLM, demonstrate the benefit of training multiple LMs on different portions of web data and search queries in a principled way without count cutoff.

2. MICROSOFT WEB N-GRAM SERVICES

This section first describes the Microsoft Web *n*-gram LM collection, and then presents a distributed *n*-gram LM platform based on which these LMs are built, and finally describes the Microsoft Web N-gram services.

2.1 Web N-gram LM collection

Table 1 summarizes the data sets and Web scale *n*-gram LMs used in this study. The collection is built from high quality English Web documents containing trillions of tokens, served by a popular commercial search engine. The collection consists of several data sets built from different Web sources, including the different text fields from the Web documents (i.e., body, title, and anchor texts) and search query logs. The raw texts extracted from these different sources were pre-processed in the following manner: texts are tokenized based on white-space and upper case letters are converted to lower case. Numbers are retained, and no stemming/inflection is performed. The *n*-gram LMs are word-based backoff models, where the *n*-gram probabilities are estimated using Maximum Likelihood Estimation (MLE) with smoothing. Specifically, for a trigram model, the smoothed probability is computed as

$$P(w_i|w_{i-2}w_{i-1}) = \quad (2)$$

$$\begin{cases} \dfrac{C(w_{i-2}w_{i-1}w_i) - D\big(C(w_{i-2}w_{i-1}w_i)\big)}{C(w_{i-2}w_{i-1})} & \text{if } C(w_{i-2}w_{i-1}w_i) > 0 \\ \alpha(w_{i-2}w_{i-1})P(w_i|w_{i-1}) & \text{otherwise} \end{cases}$$

where $C(\cdot)$ is the raw count of the *n*-gram in the training corpus and α is a normalization factor. $D(C)$ is a discount function for smoothing. We use modified absolute discounting as the discount function [12], whose parameters can be efficiently estimated and performance converges to that of more elaborate state-of-the-art techniques like Kneser-Ney smoothing in large scale data [14].

The quality of *n*-gram LMs depends on the order of the model, the size of the training data, and more importantly how well the training data match the test data. Figure 1 illustrates the perplexity results of the four LMs trained on different data sources tested on a random sample of 733,147 queries from the search engine's May 2009 query log. The results suggest several conclusions. First, higher order LMs in general produce lower perplexities, especially when moving beyond unigram models. Second, as expected, the query LMs are most predictive for the test queries, though they are from independent query log snapshots. Third, it is interesting to notice that although the body LMs are trained on much larger amounts of data than the title and anchor LMs, the former lead to much higher perplexity values, indicating that both title and anchor texts are quantitatively much more similar to queries than body texts.

As are in many applications of LMs, the perplexity measure is not the ultimate metric for applications. In other words, models with lower perplexities do not necessarily lead to a better performance. However, the perplexity analysis is still informative in that higher perplexity models can seldom outperform the lower perplexity ones, as we will show in Section 4.

2.2 Distributed N-gram LM platform

The platform is developed on a distributed computing system designed for storing and analyzing massive data sets, running on large clusters consisting of hundreds of commodity servers connected via high-bandwidth network.

We use the SCOPE (Structured Computations Optimized for Parallel Execution) programming model [3] to train the Web scale *n*-gram LMs shown in Table 1. The SCOPE scripting language resembles SQL which many programmers are familiar with. It also supports C# expressions so that users can easily plug-in customized C# classes. SCOPE supports writing a program using a series of simple data transformations so that users can simply write a script to process data in a *serial* manner without wondering how to achieve parallelism while the SCOPE compiler and optimizer are responsible for translating the script into an efficient, parallel execution plan. We illustrate the usage of SCOPE for building LMs using the following ex-

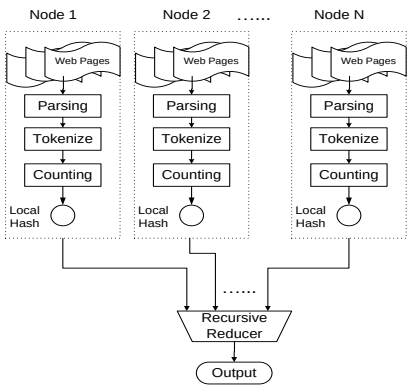

Figure 2. Distributed 5-gram counting.

ample of counting 5-grams from the body text of English Web pages. The flowchart is shown in Figure 2.

The program is written in SCOPE as a step-by- step of computation, where a command takes the output of the previous command as its input.

```
ParsedDoc=SELECT docId, TokenizedDoc
   FROM @"/shares/…/EN_Body.txt"
   USING DefaultTextExtractor;

NGram=PROCESS ParsedDoc
   PRODUCE NGram, NGcount
   USING NGramCountProcessor(-stream TokenizedDoc
   -order 5 -bufferSize 20000000);

NGramCount=REDUCE NGram
   ON NGram
   PRODUCE NGram, NGcount
   USING NGramCountReducer;

OUTPUT TO @"Body-5-gram-count.txt";
```

The first SCOPE command is a SELECT statement that extracts parsed Wed body text. The second command uses a build-in Processor (NGramCountProcessor) to map the parsed documents into separate *n*-grams together with their counts. It generates a local hash at each node (i.e., a core in a multi-core server) to store the (*n*-gram, count) pairs. The third command (REDUCE) aggregates counts from different nodes according to the key (*n*-gram string). The final command (OUTPUT) writes out the resulting to a data file.

The smoothing method can be implemented similarly by implementing the customized smoothing Processor/Reducer. They can be imported from the existing C# codes (e.g., developed for building LMs in a single machine) with minor changes.

2.3 Microsoft Web N-Gram Services

The Microsoft Web N-Gram Services is a data service for providing smoothed LM probability information. Requests are serviced using SOAP over HTTP. The detailed service description can be found at http://web-ngram.research.microsoft.com/Lookup.svc/mex?wsdl. During the public beta period, the service is open to accredited colleges and universities.

The data is originally collected in the datacenters of Microsoft Bing. Counts of word sequences are collected for each stream (body, title, and anchor) and from these counts smoothed word probabilities are computed, as described in Sections 2.1 and 2.2. From the lexicon a hash is created; this hash maps word strings to word IDs. For every word *n*-gram, two values are maintained: the conditional probability, and the backoff parameter. These values are stored in a binary search tree. The trees are distributed over multiple machines, so as to serve as much of this information from RAM as possible.

To access the service, users must first request an access token from Microsoft by sending mail to webngram@microsoft.com. The token is simply a GUID. Once a token is obtained, users can get joint or conditional probability value of words in a phrase. The application also provides a batch mode where multiple phrases can be submitted at once, returning an array of probability values. The probability values are single-precision floating point values in base-10 log.

Dataset	Number
Total tokens	4.2 B
Unigrams	1 M
Bigrams	60 M
Trigrams	246 M
4-grams	447 M
5-grams	558 M

Table 2: Statistics of the LDC English Gigaword corpus.

3. PREVIOUS WORK

This section does not intend to provide a comprehensive survey of the previous work on large LMs for web search and NLP. Instead, we will briefly describe two of the previously released datasets that have been widely used for LM training: LDC English Gigaword corpus and the Google Web 1T N-gram corpus. We will compare MWNLM with the LMs trained on the two corpora in our experiments in Section 4.

The English Gigaword corpus was produced by Linguistic Data Consortium (LDC) in 2008 [12]. This is a comprehensive archive of newswire text data in English that has been acquired over several years by LDC. For comparison, we built n-gram models, referred to as GW models afterwards, as follows. The corpus is tokenized following the Penn Treebank tokenization. After tokenization, the corpus contains about 4.2 billion tokens. The sentence boundaries are marked with two separate tokens <S> and </S>. We then build a vocabulary with 1M high frequent words extracted from the corpus. All *n*-gram (n = 1 to 5) probabilities are computed via MLE with modified absolute discounting and count cutoff 1 for all *n*-grams (n = 2 to 4), using the public toolkit MSRLM [14]. The statistics of the Gigaword models are shown in Table 3.

The Google Web 1T data set [1], contributed by Google Inc., contains English word *n*-grams (n = 1 to 5) and their observed frequency counts calculated over 1 trillion words from web page text collected by Google in January 2006. The text was tokenized following the Penn Treebank tokenization, except that hyphenated words, dates, email addresses and URLs are kept as single tokens. The sentence boundaries are marked with

Dataset	Number	Size on disk (MB)
Total tokens	1 T	N/A
Unigrams	13.6 M	185
Bigrams	314.8 M	5,213
Trigrams	977.1 M	19,979
4-grams	1.3 B	32,041
5-grams	1.2 B	33,679

Table 3: Statistics of the Google Web 1T *n*-gram data collection.

<S> and </S>. Words that occurred fewer than 200 times were replaced with the special token <UNK>. Table 4 shows the data sizes of the Web 1T corpus. The n-grams themselves must appear at least 40 times to be included in the corpus. In this study, we build LMs from the Web 1T corpus using MLE with the dubbed Stupid Backoff smoothing method proposed by Brants et al. [2] such that α in Equation (2) is no longer a normalization factor computed so that the sum of the probabilities is 1, but a predefined constant, i.e., $\alpha = 0.4$ for all orders in all our experiments. As a result, the LMs can be trained much more efficiently than other state-of-the-art smoothing methods such as Kneser-Ney smoothing in the MapReduced environment, but the LMs are no longer statistic models, thus lose all of the nice probabilistic properties.

4. EVALUATIONS

We follow the re-ranking experimental paradigm to evaluate the performance of different LMs on three web search and NLP tasks: query spelling correction, query reformulation, and Chinese to English statistical machine translation. In each of these tasks, we assume that for each test sample, (i.e., an input query in the first two tasks, and a Chinese sentence in the third task) we have been given a list of candidates, generated using a *baseline* system. Then the candidates are re-ranked using the log probabilities produced by LMs. We compare the effectiveness of different LMs using application-specific measures. We also use t-test to test the statistical significance of different LMs. A significant difference should be read as significant at the 95% level. In what follows, we describe for each evaluation the baseline system, the measure, and the re-ranking results.

4.1 Query Spelling Correction

Search queries present a particular challenge for traditional spelling correction methods that are based on dictionaries because query language is changing constantly, and many search query terms, such as names and proper nouns, are not well-established in the language and are not included in any dictionaries. Therefore recent research has been focused on inferring knowledge about spellings and word usage in search queries from large amounts of web data and search logs. In addition, there are many real-word errors in search queries, whose correction depends to a large degree upon the use of local context, such as the *n*-gram features captured by *n*-gram LMs.

This makes the query spelling correction task an ideal test bed to evaluate the effectiveness of MWNLM, which are trained on web data and search queries. In our experiments, we used the baseline system described in Gao et al. [9] to produce for each input query 20-best candidate corrections, which are found with small edit distance, similar morphology or alternative word breaking. We then use the LMs to re-rank these spelling candidates, and consider the top-ranked candidate as the correction generated by the system.

#	System	Accuracy	Precision	Recall	OOV rate
1	MWNLM (query)	82.53	53.30	10.90	0.13
2	MWNLM (body)	83.83	62.04	18.55	0.18
3	MWNLM (title)	84.86	68.04	23.95	0.69
4	MWNLM (anchor)	83.59	57.55	19.80	0.77
5	Google	81.91	79.52	2.28	6.53
6	GW model	81.55	60.01	0.10	18.27

Table 4. Summary of spelling correction results. All LMs are trigram models.

The evaluation is performed on a data set containing 15,657 queries sampled from one year's worth of query logs from a commercial search engine. The spelling of each query is manually judged and corrected by four independent annotators. 2,960 queries are judged as misspelled and are corrected. The average length of queries in the data set is 2.7 words. The spelling correction results are evaluated using the following three metrics.

- **Accuracy:** The number of correct outputs generated by the system divided by the total number of queries in the test set.
- **Precision:** The number of correct spelling corrections for misspelled queries generated by the system divided by the total number of corrections generated by the system.
- **Recall:** The number of correct spelling corrections for misspelled queries generated by the system divided by the total number of misspelled queries in the test set.

Table 4 summarizes the re-ranking results. We see that (1) MWNLM significantly outperform the Google and Gigaword LMs in terms of accuracy. (2) Among MWNLM, the title LM is slightly better than the others because title words are mostly correctly spelled and form a similar vocabulary to that of query words, as discussed in Section 2.1. (3) The Google model achieves the best precision because the model is trained on *n*-grams with very high count cutoff and most misspelled query terms have lower frequency than their correctly spelled counterparts, and thus lead to lower *n*-grams. However, MWNLMs beat the Google LM with a substantial marge in recall, due to the zero count off. (4) The Gigaword LM, which is trained on newswire corpus, can hardly capture any spelling mistakes in search queries because of the language discrepancy between newswire and search queries. The language discrepancy can also be verified by its high OOV rate, i.e., a lot of query terms, either correctly spelled or misspelled, simply do not occur (at least not often enough) in the newswire corpus, and thus are excluded in the dictionary.

4.2 Query Reformulation

Query reformulation can be considered as a generalization of the query spelling correction task. Given an input query, we seek for more effective variants that lead to better web search results via paraphrasing, segmentation, stemming, and query expansion, etc. For example, one possible variant of an input query "heroic acts" is "heroic actions" or "heroic act". We use *Normalized Discounted Cumulative Gain* scores at position 5 (NDCG@5) [11] to measure retrieval effectiveness.

#	System	NDCG@5 (%)	OOV rate (%)
1	MWNLM (query)	62.09	0.02
2	MWNLM (body)	62.16	0.03
3	MWNLM (title)	62.10	0.16
4	MWNLM (anchor)	62.11	0.69
5	Google	61.98	0.74
6	Gigaword	61.38	9.46

Table 5. Query expansion results (NDCG@5). All LMs are trigram models.

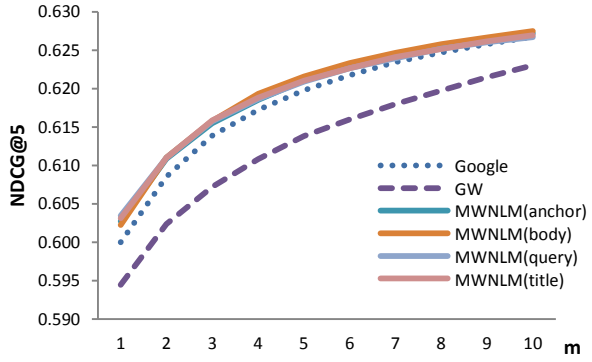

Figure 3. Query expansion results (NDCG@5). All LMs are trigram models

We perform the evaluation using a set of 11,006 queries randomly sampled from one year's worth of query logs from a commercial search engine. For each original query, 30 candidate suggestions are generated by expanding one of its query terms using the OR operator. For example, given an input query "heroic acts", we expand "acts" to "actions" by forming a suggestion "heroic OR(acts actions)". We perform query reformulation via *query expansion* rather than *query substitution*, in which case the suggestion of "heroic acts" would be "heroic actions", is due to the fact that in our experiments query expansion always outperforms query substitution. Our finding is consistent with reported results [e.g., 7].

Then, we use a LM to re-rank the candidate suggestions in their *substitution* form, assuming that a good query substitution often leads to a good query expansion. At query time, for each query q, the top m candidate suggestions in their *expansion* form ranked by the LM (in their *substitution* counterparts) are used to retrieval web documents using a commercial search engine (e.g., Bing in our experiments). The retrieved documents are then manually labeled on a 5-level relevance scale, 0 to 4, with 4 as the most relevant. NDCG scores are finally computed on labeled query-document pairs retrieved by the candidate suggestion. Following Dang and Croft [7], we record the best NDCG@5 obtained by these m candidates as the NDCG@5 of the substitute solution for q. We varied m from 1 to 10 in our experiments.

The main results are shown in Table 5 and Figure 3. We see that the trend is similar to that of the spelling correction results in Table 4. All MWNLM are in a near statistical tie for the first place. The Google model underperforms slightly MWNLM due to its high count cutoff. There is a significant gap between the Gigaword model versus other web scale LMs due to its high OOV rate and the significantly smaller amount of training data.

#	System	News (691)	Web (666)	All (1357)	OOV rate
1	MWNLM (query 4-gram)	27.11	15.18	21.97	0.02
2	MWNLM (body 5-gram)	**28.28**	15.37	**22.74**	0.01
3	MWNLM (title 4-gram)	27.32	15.29	22.15	0.04
4	MWNLM (anchor 4-gram)	27.52	15.38	22.30	0.21
5	Google (5-gram)	27.97	**15.59**	22.64	0.84
6	Gigaword (5-gram)	27.47	15.14	22.16	3.45

Table 6. Summary of machine translation results on the 2008 NIST C2E Open MT Evaluation test set, reported in BLEU scores (%).

4.3 Machine Translation

Given a source-language (e.g., Chinese) sentence, the goal of machine translation is to automatically produce a target-language (e.g., English) translation.

This section evaluates the use of different LMs for machine translation on the Chinese-to-English (C2E) test in the constrained training track of the 2008 NIST Open MT Evaluation [15]. The test set contains 1,357 Chinese sentences, including 691 sentences of newswire and 667 sentences of the web data genre. Each sentence has four reference translations manually generated. In our experiments, we first generated for each Chinese sentence in the test set 100 best candidate translations using a baseline system. Then we used a 5-gram LM to re-rank the 100-best list. The re-ranking results were evaluated using top-1 translation BLEU scores [16].

The baseline system we used to produce 100-best candidate translations is our implementation of the hierarchical phrase-based system, described by Chiang [4], one of the state-of-the-art machine translation systems. It uses a statistical phrase-based translation model that uses hierarchical phrases. The model is a synchronous context-free grammar and it is learned from parallel data without any syntactic information.

The results are summarized in Table 5. Unlike the results in the search query spelling correction and reformulation tasks, difference among different LMs on the MT task is not substantial, although the Microsoft body LM still outperforms the Google and Gigaword models on the News and All corpora with a small but statistically significant margin. The Gigaword model, in spite of its much higher OOV, only slightly underperforms the best models on different portions of the test set.

5. Conclusion

This paper reports the recent development of Microsoft Web N-gram Language Models, as well as the corresponding Microsoft Web N-gram services. The services provide the research communities the access to a set of web scale n-gram language models that are trained on different portions of web data and search queries using a principled manner with zero count cutoffs. We demonstrate the effectiveness of MWNLM on three web search and NLP applications. We sincerely invite our colleagues in the research communities to exploit the web scale LMs for interesting applications using the services we are providing. We will continue the development and upgrade of MWNLM based on what we learned from your feedback.

REFERENCES

[1] Brants, T., and Franz, A. 2006. Web 1T 5-gram corpus version 1.1. Technical report, Google Research.

[2] Brants, T., Popat, A. C., Xu, P., Och, F. J., and Dean, J. 2007. Large language models in machine translation. In *EMNLP-CoNLL*, pp. 858 - 867.

[3] Chaiken, R., Jenkins, B., Larson, P., Ramsey, B., Shakib, D., Weaver, S., and Zhou, J. 2008. SCOPE: easy and efficient parallel processing f massive data sets. In *Proceedings of the VLDB Endowment,* pp. 1265-1276.

[4] Chiang, D. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

[5] Charniak, E. 2001. Immediate-head parsing for language models. In *ACL/EACL*, pp. 124-131.

[6] Chen, S. F., and Goodman, J. 1999. An empirical study of smoothing techniques for language modeling. *Computer Speech and Language*, 13(10):359-394.

[7] Dang, V., and Croft, W. B. 2010. Query reformulation using anchor text. In *Proc. WSDM'10*.

[8] Gao, J., Goodman, J., and Miao, J. 2001. The use of clustering techniques for language modelling -application to Asian languages. *Computational Linguistics and Chinese Language Processing*, 6(1):27–60, 2001.

[9] Gao, J., Li, X., Micol, D., Quirk, C., and Sun, X. 2010. A large scale ranker-based system for search query spelling correction. In *Proc. COLING 2010.*

[10] Huang, J., Gao, J., Miao, J., Li, X., Wang, K., and Behr, F. 2010. Exploring web scale language models for search query processing. In *Proc. WWW 2010.*

[11] Jarvelin, K. and Kekalainen, J. 2000. IR evaluation methods for retrieving highly relevant documents. In *SIGIR*, pp. 41-48.

[12] LDC English Gigaword Fourth Edition. 2009. http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2009T13

[13] Microsoft web n-gram services. 2010. http://research.microsoft.com/web-ngram

[14] Nguyen, P., Gao, J., and Mahajan, M. 2007. MSRLM: a scalable language modeling toolkit. Technical report TR-2007-144, Microsoft Research.

[15] NIST. 2008. The 2008 NIST Open Machine Translation Evaluation. www.nist.gov/speech/tests/mt/2008/doc/

[16] Papineni, K., Roukos, S., Ward, T., and Zhu, W-J. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proc. of ACL*, pp. 311–318.

[17] Wang, K. and Li, X. 2009. Efficacy of a constantly adaptive language model technique for web-scale applications. In *Proc. ICASSP-2009*, Taipei, Taiwan, 4733-4736.