Exploring Web Scale Language Models for Search Query Processing

Jian Huang*
Information Sciences and Technology
Pennsylvania State University
University Park, PA 16802
jhuang@ist.psu.edu

Jianfeng Gao
Microsoft Research
One Microsoft Way
Redmond, WA 98052
jfgao@microsoft.com

Jiangbo Miao
Facebook
704 Hibiscus Place
San Jose, CA, 95117
jbmiao@facebook.com

Xiaolong Li
Microsoft Research
One Microsoft Way
Redmond, WA 98052
Xiaolong.Li@microsoft.com

Kuansan Wang
Microsoft Research
One Microsoft Way
Redmond, WA 98052
Kuansan.Wang@microsoft.com

Fritz Behr
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052
fritzb@microsoft.com

ABSTRACT

It has been widely observed that search queries are composed in a very different style from that of the body or the title of a document. Many techniques explicitly accounting for this language style discrepancy have shown promising results for information retrieval, yet a large scale analysis on the extent of the language differences has been lacking. In this paper, we present an extensive study on this issue by examining the language model properties of search queries and the three text streams associated with each web document: the *body*, the *title*, and the *anchor text*. Our information theoretical analysis shows that queries seem to be composed in a way most similar to how authors summarize documents in anchor texts or titles, offering a quantitative explanation to the observations in past work.

We apply these web scale n-gram language models to three search query processing (SQP) tasks: query spelling correction, query bracketing and long query segmentation. By controlling the size and the order of different language models, we find that the perplexity metric to be a good accuracy indicator for these query processing tasks. We show that using *smoothed* language models yields significant accuracy gains for query bracketing for instance, compared to using web counts as in the literature. We also demonstrate that applying web-scale language models can have marked accuracy advantage over smaller ones.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval; H.4.m [**Information Systems**]: Miscellaneous

Keywords

Web n-gram, language models, very large-scale experiments, search query processing

General Terms

Algorithms, Experimentation

1. INTRODUCTION

The massive data freely available on the web has spurred significant interests to tap on the web as a corpus for all manners of IR and NLP research [2]. Enabled by the power of such web scale data, simple models can yield remarkable results in various real-world applications such as statistical machine translation [17]. Many of these models rely on the web count information, either as web page hits count or as term frequency statistics derived from a large sample of the web (e.g. the Google 1T web corpus [8] — 1T Corpus for short hereafter). *N-gram language models* (LM) represent the probability of the occurrence of a word w accounting for its (limited) history h_w. The advantages of this representation over count statistics are twofold. First, compared to models which are agnostic of word order (e.g. the bag-of-words model), LM accounts for the order of words which is important for many tasks (e.g. 'blue sky' and 'sky blue' are syntactically and semantically different). Second, LM is capable of predicting non-zero probability for a word sequence unseen in training by interpolating or backing off to lower order models, and thus it is more robust to rare n-grams in application[1].

Building web-scale language models can significantly advance many of the IR research and applications, yet the sheer size of the web presents great engineering challenges for applying many of the classical language modeling techniques. For the purpose of developing a web machine translation system, an approximation approach dubbed Stupid Backoff was proposed [9] with a constant backoff coefficient, permitting the model to be efficiently trained for web data

*Work done while first author was visiting Microsoft.

WWW 2010, April 26–30, 2010, Raleigh, North Carolina, USA.
ACM 978-1-60558-799-8/10/04.

[1]Indeed, an order-3 language model (derived from a trillion-token web corpus) with 4 billion n-grams still encountered more than 30% unseen tri-grams in testing. The unseen data problem exacerbates dramatically in higher order models due to the curse of dimensionality.

in the *MapReduce* environment. However, the model is no longer a statistical model for not properly normalizing the probabilities. Enabled by an ingenious data structure to store n-grams in backorder trees, MSRLM [25] allows the smoothing factor to be properly estimated on the web scale. An improved version of MSRLM called Constantly Adaptive Language Modeling technique (CALM) was introduced [30] which allows the language model be built incrementally with the new web data. This work leverages these recent advances in large scale language modeling techniques to build web scale n-gram models.

The n-gram language models in this work are built from different web sources, including the different text fields from the documents such as titles, anchor texts and body texts, as well as web search queries. Recently, IR techniques accounting for these alternative data sources, compared to using only document texts in tradition, have achieved promising results (for instance, [3, 15] demonstrated significant improvement for ranking using search queries). In particular, Svore et al. [27] studied the contributions of these different sources to IR accuracy and significantly improved the state-of-the-art ranking function BM25.

Complementary to these works in document ranking, we showcase the value of these web n-gram language models with a set of *"Search Query Processing"* (SQP) tasks. In the web search setting, SQP tasks distinguish themselves from many traditional IR/NLP tasks in at least two aspects. (1) SQP tasks deal with the query language instead of the formal natural language as that found in the text of a clean corpus such as the *Wall Street Journal*. The query language is found to be dominated by noun phrases that are loosely attached [5]. This sharply contrasts with text written in a formal language that well conforms to the underlying grammar. (2) In the web context, SQP tasks are faced with a living language with an open-domain and dynamic vocabulary, as new words are constantly being created. For instance, the query log of 2009 of a web search engine is drastically different from that of 2007. Even the grammar changes over time: it has been observed that the average query length increases over the years [1] indicating users are becoming more sophisticated. These challenges on the one hand render many conventional or elaborate methods inapplicable. On the other hand, they present opportunities where web n-gram LMs are particularly suitable to apply.

We highlight the contributions of this work as follows:

- We build web-scale language models from web sources including the search queries, as well as the titles, the anchor texts and the bodies of web documents. To the best of our knowledge, this is the first extensive and empirical analysis on the language model properties of these different sources as well as the model sizes and orders on the web scale.

- Unsupervised methods, including a novel query segmentation approach, are applied to efficiently tackle three search query processing tasks from query spelling correction to analyzing sub-query structures. Through these SQP tasks, we explore how these language differences are incarnated in the accuracy of these real world applications in web search engines.

The remainder of the paper is organized as follows. We first introduce the Web N-gram Language Models Collection used in this work in Section 2. This is followed by a large-scale information theoretic study in Section 3 that quantifies the language differences of the language models from different web sources. Motivated by these n-gram property findings, Section 4 illustrates how these different sources as well as the model order and size can impact the accuracy of the query spelling correction task. Section 5 and 6 analyze the syntactic structures of short and long queries respectively and further corroborate the findings in Section 3. Where appropriate, we will discuss the related work of these individual SQP tasks and their distinctions with the traditional counterparts. Section 7 concludes this paper and discusses future work.

2. THE WEB N-GRAM LANGUAGE MODEL COLLECTION

On the outset, we describe the WEB N-GRAM LANGUAGE MODELS COLLECTION (WNLMC) used in this work. The collection is built from the high quality English web documents[2], in the scale of billions of pages and trillions of tokens, served by Microsoft's Bing search engine (`www.bing.com`).

We first highlight several distinctive features of the WNLMC collection. First, during its construction, special attention has been paid to the different **data sources** (see Table 1). The collection consists of several n-gram language models built from two data sources. The *document source* language models are derived from the text streams associated with the documents from the crawl of the web. The *body* language model is built from the segmented sentences from the extracted raw text of web documents. The *anchor* model is built using the extracted anchor text of the hyperlinks and the *title* model using the titles of the web pages. In addition to web documents, we also sampled search queries from half-year worth of search query logs and built the *query* language model. The raw text extracted from these different sources goes through similar preprocessing steps to ensure uniformity: the text is white-space tokenized and lowercased (the trained language models are case insensitive), numbers are retained (concepts such as `windows 7` are included in the models) and no stemming/inflection was performed. Also note that in building the language models, sentence boundaries[3] are preserved by adding special tokens `[S]` and `[/S]`. N-grams appearing less than certain raw frequency thresholds (see Table 1) are discarded from model training. In application, words unseen in training are replaced by the token `[UNK]` (whose probability is derived from the smoothing process). The adoption of very low thresholds [4] permits infrequent n-grams to be included in the models. As we will show in the sequel, language models built from these data sources are very different in various applications.

Second, in addition to the observed raw frequency of n-grams, **smoothing** is performed (made possible with the cloud computing infrastructure) to yield very large scale n-

[2]These pages have the top PageRank scores computed from the web map which is constructed from the web crawl.

[3]A sentence boundary is to be understood broadly here. For *body* text this is the natural sentence boundary. For *anchor*, *title* and *query*, the entire sequence of words constitutes a 'sentence', even if it is not followed by a punctuation.

[4]1T Corpus contains frequency counts with higher cutoff.

Table 1: Statistics of the web n-gram language models collection.

Dataset	*body*	*anchor*	*title*	*query*
#Unigrams	68,704,702	40,281,581	21,281,543	97,539,950
#Bigrams	789,522,728	292,821,791	200,245,295	473,585,620
#Trigrams	2,967,982,818	767,759,313	553,335,418	1,031,936,006
#Four-grams[a]	N/A	1,070,818,529	765,952,682	N/A
Total entries	3,826,210,248	2,171,681,214	1,540,814,938	1,603,061,576
Size on disk[b]	158GB	100GB	70GB	65GB
Count threshold	20	5	1	1

[a] Order 4 *anchor* and *title* LMs are included in the size studies. As we show later, order 4 LMs only yields small gains over order 3 LMs but are much larger.
[b] N-gram entries as well as other statistics and model parameters are stored.

gram language models (LM), the largest LMs we are aware of to date. The main benefit of performing smoothing, compared to using raw frequency in a *maximum likelihood estimation* (MLE) manner, is tiny (but non-zero) probability is given to unseen n-grams. This has been shown to yield significantly better prediction power in various applications [31, 10]. Even though algorithms for building n-gram models are widely known [10], as we mentioned earlier applying them on the web scale (trillion-token corpus) is non-trivial. The smoothing technique implemented in this work is based on recent advances in web language modeling [25, 30]. Specifically, for a trigram model, a general formula in the following form is usually used for smoothing:

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

where $C(\cdot)$ is the raw count of the n-gram in the training corpus and $\alpha(w_{i-2}w_{i-1})$ is a normalization factor ensuring probabilities sum up to 1. $D(C(w_{i-2}w_{i-1}w_i))$ is a discount function for smoothing. We use *modified absolute discounting*[5] as the discount function, whose parameters can be efficiently estimated and performance converges to that of more elaborate state-of-the-art techniques like Kneser-Ney (KN) smoothing [20] in large scale data [25]. This smoothing process is implemented in a large distributed environment to generate web-scale smoothed language models, though the details of which are outside of the scope of this paper.

An up-to-date version of the web n-gram models is available through the Microsoft Web N-gram Service (called Bing It On N-gram Service). The web service delivers n-gram statistics (counts, smoothed probabilities, etc) built from various sources of the vast and evolving web data in an on-demand manner.

3. WEB N-GRAM DATA ANALYSIS

In this section, we conduct an information theoretic analysis of the n-gram models which quantify the differences of the language styles on the web scale as we mentioned earlier. Perplexity is a well known information theory metric for evaluating how well the language model predicts the test data. Formally, the *perplexity* between an empirical distribution of the test data $\tilde{p}$ and a language model q is:

$$\text{Perplexity}(\tilde{p}, q) = 2^{H(\tilde{p},q)} \quad (1)$$

[5]Interested reader can refer to [14] for more details.

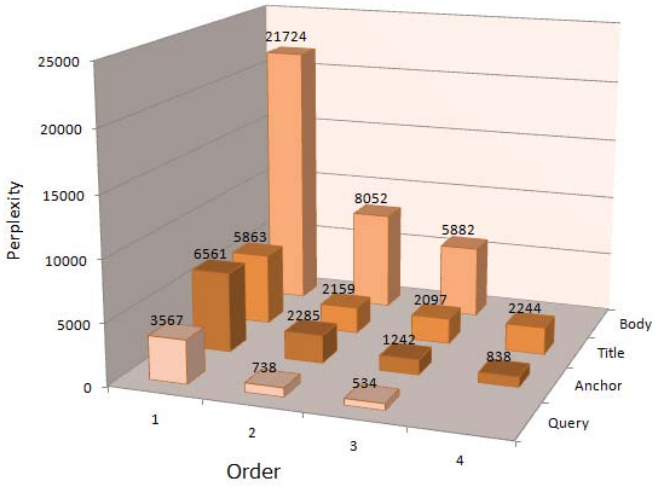

Figure 1: Comparison of query word perplexity with different orders of the language models of similar sizes, derived from different data sources.

where $H(\tilde{p}, q) \stackrel{def}{=} -\sum_s \tilde{p}(s) \log q(s)$ is the cross entropy between the probability distributions $\tilde{p}$ and q, i.e., the average number of bits a test sentence s can be coded by using the model q. The lower the perplexity, the better the prediction power of the language model. As we are interested in query related applications, given a query s, $\tilde{p}$ is the empirical probability of s in the sampled query log and q is the joint probability computed from the language model. Also note that we report the perplexity normalized by the total query length.

Figure 1 illustrates the perplexity of language models from different sources tested on a random sample of 733,147 queries from the search engine's May 2009 query log. A higher order language model in general reduces perplexity, especially when we compare the unigram models with the n-gram models. The query language model is undoubtedly the most predictive for the test queries, though they are from independent query log snapshots. Interestingly, although the *title* model's vocabulary is slightly more similar to that of the test queries than *anchor*, a higher order (3-4) *anchor* language model actually reduces perplexity more (while a fourth order *title* language model results in slightly higher perplexity than order 2 and 3). This suggests that the ordering in the n-gram word structure captured by

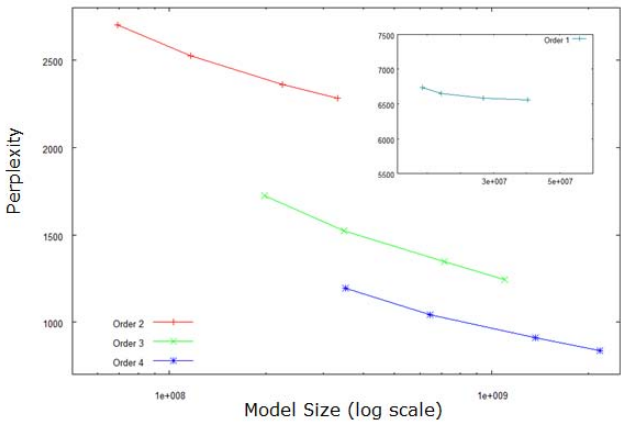

Figure 2: Comparison of query word perplexity with different sizes and orders of the *anchor* language models (inset: perplexity of order 1 *anchor* LMs).

the *anchor* language model is more similar to the queries than that by the *title* language model. Both anchor texts and titles are quantitatively similar to the way one would compose search queries, while the perplexity of the *body* language model appears to be in a different league from these others. Intuitively, this is because body texts are mostly comprised of sentences conformed to the language grammar, completed with functional words and covering the broadest range of vocabulary, whereas queries, anchor texts and titles are to different extents more succinct representations.

We also study the relationship of perplexity and the size of the language model. We adjust the size of the *anchor text* language models by applying different thresholds. In Figure 2, we observe that perplexity reduction is slower than log-linear in terms of the language model sizes (for order 2, 3 and 4 models). Perplexity however does not appear to saturate even with the largest billion-entry language model. In fact, increasing model size allows the language model to more accurately estimate the probability of rare n-grams. This is in sharp contrast with increasing model order, which exacerbates data sparsity due to the curse of dimensionality.

This information theoretic analysis motivates us to further investigate the n-gram LMs through the lens of several query centric applications. First of all, perplexity is a well-known performance indicator of a LM. On the other hand, it is an indirect metric *per se* for real world applications. A natural question is how perplexity reduction as we have shown in this section translates to accuracy improvements in various query processing tasks. We are particularly interested in how the three factors of the language models, i.e. model order, size and data source, correlate with the performance of the applications. For instance, which data source is most suitable for a particular application and why that is the case? Is there a turning point whereby further increasing the order or the size of language models only offers diminishing, if any, gains in accuracy? We will study three search query processing tasks: we first look into the query spelling correction problem using web n-gram LMs and then we zoom into the structure of search queries.

4. QUERY SPELLING CORRECTION

Query spellers are commonly found in search engines that significantly improve the information retrieval process. As a user misspells a query (e.g. 'white zinfandal'), a query speller can suggest the correct spelling ('white zinfandel') in real time. Though lexicon based approaches can handle many of such typographical errors in a traditional word processing setting, it's much more challenging to correct misspelt web search engine queries [12]. First, compiling a high coverage dictionary for the web is near impossible for the tremendous amount of proper nouns and a variety of languages. Also, the web search query language is a living language with new names and concepts constantly being created at a speed far outpacing any static dictionaries can be updated. Indeed, spelling candidates appearing more frequently on the web than the original query could be valid correction by the *wisdom of crowds*.

Another type of spelling error arises when the error results in a valid word and this can only be solved using the context, as illustrated in the following example:

Original query: `woods and water reality`
Candidate 1: `woods and water reality`
Candidate 2: `woods and water realty`
Candidate 3: `words and water reality`

where the first candidate (no correction) is a false negative and the second candidate is the true positive. The inclusion of the search results from the correctly spelt queries improves the results' quality without burdening the user with re-issuing the correct query. N-gram language models are capable of simultaneously handling typographical and context-sensitive spelling correction tasks [16], since they capture the n-gram probability as well as the context.

Context-sensitive spelling correction for **document** text has been studied in NLP and recently applied with much success in word processing products like Office [11]. Unlike that setting where very limited memory necessitates model compression and pruning [11], the distributed environment in commercial web search engines can readily and efficiently store and serve very large language models. This can yield significant accuracy benefits compared to smaller language models. The merits of very large scale text corpora for spelling correction were noted in [4]. A smoothed language model built from a giga-word English corpus was used in [13] to correct errors made by English as second language authors. The 1T Corpus was used in [6] for the spelling correction of document text and observed improved performance compared to using web page hit counts for estimation. The challenges of web search **query** spelling correction was elucidated in [12], which handled this task with extracted unigrams and bigrams from search query logs. We carry out an extensive study of the effects of language models in this task, with different model orders and sizes and extracted from documents and query logs.

Given a set of (at most 20) query spelling candidates found with small edit distance, similar morphology or alternative word breaking[6], the query spelling correction task is recast to rank the candidates such that the correct one is placed in the top rank. We use the languages models to rank the

[6]The details of the candidate selection and decoding process are outside the scope of this paper. For comprehensive coverage, interested readers can refer to [18].

spelling candidates. Specifically, we use the joint probability derived from the language models as a feature for ranking. Given a length k query $\mathbf{q} = w_1 w_2 ... w_k$, the joint probability of the query $\mathbf{q}$ with respect to an order r language model is,

$$P(\mathbf{q}) = P(w_1 w_2 ... w_k) = \prod_{i=1}^{k+1} P(w_i | w_{i-r+1} .. w_{i-1}) \qquad (2)$$

where w_{k+1} is the right sentence boundary `[/S]` and $w_i (i < 1)$ indicates the left sentence boundary `[S]`.

For the context-sensitive query spelling correction task, 15,657 queries were randomly sampled from the query log and human annotators examined each query to generate a speller test set. After annotation, around 15% of all sampled queries were found to require spelling correction (amounts to 2,349 queries in total). Our context-sensitive query speller is evaluated on this set of queries that need correction. To measure the performance of ranking the spelling candidates for spelling correction, we use the *precision@1* metric[7], i.e. the percentage of spelling suggestions at rank 1 being true positives. This is a sensitive metric in practical use.

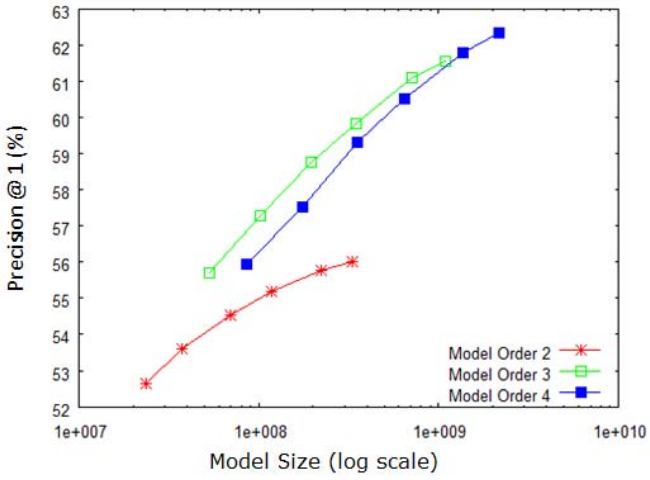

Figure 3: Precision of the query speller using different orders and sizes of *anchor* language models.

We examined the accuracy of the query speller by using the query joint probability as a feature for ranking. Figure 3 illustrates the accuracy curves of the *anchor* language models with different orders and sizes. As indicated by the perplexity findings in Figure 2, increasing the size of order 2 to 4 models does improve precision. For instance, increasing the size of the trigram models by an order of magnitude (from 100M to 1B) improves precision by 5%. Furthermore, we note that with the same model size (measured by the number of n-gram entries), the language models containing trigrams and four-grams significantly outperform the bigram language models. Moreover, the accuracy of a higher order language model saturates much slower than that of a lower order model. Interestingly, we observe that the accuracy

[7]This is also the accuracy for the set of queries needing correction. We focus on this set because considering the data distribution, a baseline on all sampled queries can appear to have high accuracy by not making any correction.

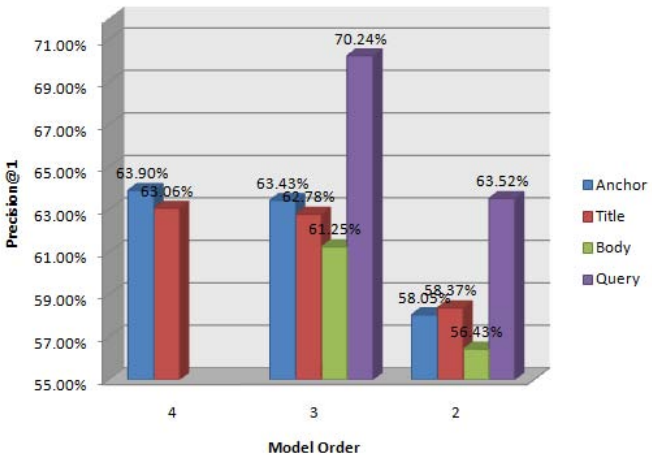

Figure 4: Precision of the query speller using language models of similar sizes from difference sources with different orders.

curve of the four-gram language model crosses over that of the trigram model at around 1 billion n-gram entries. This means that it is advantageous in this case to use a trigram language model unless we can afford to use a huge language model in practice[8] (note the log-scale in the number of entries). Also, the largest four-gram language model does not appear to hit a performance ceiling, though further improvement would require orders of magnitude more data.

Figure 4 showcases the precision of spelling correction with language models of different orders and from different sources. Note that here we set the language models to be of similar sizes by adjusting the cutoff thresholds. Using trigrams in the language models significantly improves precision by 5% to 7% than using only unigrams and bigrams. On the other hand, adding 4-gram entries only slightly improves precision by around 0.5%. From the perspective of data sources, the *query* language model significantly outperforms the language models built from document sources by 5% to 9%. This is not surprising because although the test queries and the queries for building the language models are drawn independently, they are essentially 'sentences' from the same 'query language'. It is highly likely that the correctly spelt query components appear more often in the query logs (some even in the form of query rewrites for the misspelt one). Among the document source language models, the *anchor* language models perform better than *title* (order 3 and 4), and *title* in turn outperforms *body*. This corroborates our perplexity findings in the previous section, as anchor texts and queries are oftentimes succinct representation (e.g. in the form of noun phrase compounds) not necessarily adhering to the grammar. Titles are more complete, while body texts are most grammatically coherent. As indicated by the perplexity, the joint probability computed from the *body* language model is the least useful for ranking the spelling candidates. These latter two sources are weaker for query spelling correction as they are more dissimilar to the query language model.

[8]Another reason that trigram models are sufficient for this SQP task is the average length of queries is lower than 3.

5. QUERY BRACKETING

In the following two applications, we drill down and study the structure of search queries. Automatically analyzing and parsing search queries is an important step for developing the proximity dynamic ranking features in search engines. Proximity dynamic ranking can be regarded as a post processing step of retrieval to promote or demote the ranking of a web page based on the appearances of proximal query terms. Naively favoring proximal query terms can be misleading, however. Take the query *'big blue sky'* as an example, boosting the adjacent terms 'big blue' can lead to many irrelevant pages about IBM (nicknamed 'Big Blue'). Thus the mere frequent co-occurrence of query terms is insufficient to accurately yield salient proximal term ranking features. As n-grams can properly account for the context, we use language models to study query structures. We begin with the analysis of length-3 queries in this section, which is the minimal query length to contain certain syntactic substructures useful for developing proximity features. It is well known that web search queries are short[9], in particular, length 1, 2 and 3 queries each accounts for more than 20% of the total query occurrences [1]. Hence the study of three-word query is of much practical importance.

The query bracketing task is related to, though not the same as[10], the *noun compound bracketing* task [23] in NLP. The noun compound bracketing task can be summarized as: given a three-word noun phrase (NP)[11] $n_1n_2n_3$, determine the sub-NP structure either as left or right bracketing. Consider the following example:

Left bracketing:	`[sore gum] treatment`
Right bracketing:	`sore [gum treatment]`

Since the treatment is for sore gum, left bracketing should be chosen in this case. Previous works typically compare the associations in the two types of bracketing and opt for the one with the stronger association. Specifically, the *adjacency model* compares the association of n_1n_2 to n_2n_3 whereas the *dependency model* compares n_1n_2 to n_1n_3. Unsupervised methods with different corpus statistics have been studied in the literature. Lauer et al. [23] derived the corpus statistics from a small 8 million word corpus, and used a thesaurus to esimate concept probability instead of word probability due to the data sparsity problem. Later, the web page hit counts from bigram queries have been used to achieve nearly as good results [22] without recourse to the taxonomy as in the previous more elaborate approach. More recently, the experiments in [29] using the web counts derived from the 1T Corpus have shown additional improvement. However, these methods used raw counts (some with un-smoothed probabilities esimated in an MLE manner) and were applied on the noun compounds from very clean corpora (e.g. Grolier's encyclopedia [23] and MEDLINE abstracts [24]). Our work, on the other hand, uses smoothed web language models and shows significant improvement for web queries which are much more diverse.

[9]E.g. [26] showed that Excite's average query length was 2.3, though users become more sophisticated over time [1].

[10]In a study of English queries, around 70% are found to be noun phrases [5]. For instance, the query `download adobe flash` is a verb phrase (VP).

[11]Three-word proper nouns such as "yellowstone national park" are excluded from consideration in this problem.

In either of the bracketing models we mentioned, different measures of word association can be applied. We adopt some of the best known metrics as follows using the n-gram language model:

- *Conditional probability (CP).* CP is the earliest association measure used for bracketing [23]. Given a word pair w_i and w_j, the strength of the head-modifier relation is $Pr(w_i \rightarrow w_j|w_j)$. Using the language model, this is the bigram conditional probability,

$$\mathrm{CP}(w_i, w_j) = P(w_i|w_j) \tag{3}$$

- *Pointwise mutual information (PMI).* $\mathrm{PMI}(w_i, w_j)$ is a widely used information theory metric that measures the amount of information for the occurrence of w_j given the occurrence of word w_i, formally defined as:

$$\mathrm{PMI}(w_i, w_j) = \log \frac{P(w_iw_j)}{P(w_i)P(w_j)} \tag{4}$$

  To demonstrate the difference between smoothed language models and raw web counts as applied in pervious work, we also experiment with the MLE version of the PMI measure. Let $C(w)$ denote the raw frequency count and N the corpus size,

$$\begin{aligned} \mathrm{PMI}^{[MLE]}(w_i, w_j) &= \log \frac{C(w_iw_j)/N}{C(w_i)/N \cdot C(w_j)/N} \\ &\propto \log \frac{C(w_iw_j)}{C(w_i)C(w_j)} \end{aligned} \tag{5}$$

- *Chi-square test statistic (χ^2).* Treating the occurrences of w_i and w_j as random variables, χ^2 can be used to measure their (in)dependence. Let O_l denote the 'observed' count[12] and E_l the expected count in the 2×2 contigency table (of cooccurrence). χ^2 is defined as,

$$\chi^2 = \sum_{l=1}^{4} \frac{(O_l - E_l)^2}{E_l} \tag{6}$$

Table 2: Query bracketing annotation examples. Square brackets are added to the original queries to indicate the choice of bracketing.

Query	Label
`[sore gum] treatment`	Left(Strong)
`solar [security lights]`	Right(Strong)
`[healthcare management] internships`	Left(Weak)
`free [invoice template]`	Right(Weak)

For this query bracketing task, 1,028 three-word queries with either right or left bracketing (excluding three-word proper noun queries) were sampled from the query log and annotated. In addition to the annotation of bracketing, we also differentiate between *strong bracketing* and *weak bracketing*. Three-word queries with strong bracketing structure are considered more important for the purpose of developing proximity features. In all, 725 or 70.5% of these three-word queries were identified as strong bracketing.

[12]The modified discounted count, computed as part of the smoothing process as mentioned in Section 2, is used here.

Table 3: Accuracy of query bracketing using the adjacency model with different word association measures. Baseline predicts the majority label (i.e. left bracketing). MLE denotes using raw web count.

Dataset	Association	*anchor*	*title*	*body*	*query*
Strong	[Baseline]	66.3%			
	PMI	92.0%	90.9%	93.3%	91.0%
	$PMI^{[MLE]}$	86.3%	84.0%	86.2%	84.7%
	χ^2	90.6%	90.0%	91.1%	90.7%
	Cond. Prob.	88.1%	86.9%	86.9%	85.9%
Strong + Weak	[Baseline]	60.8%			
	PMI	90.2%	89.5%	90.9%	88.9%
	$PMI^{[MLE]}$	84.5%	83.1%	84.7%	83.0%
	χ^2	88.6%	88.3%	89.6%	89.3%
	Cond. Prob.	86.3%	85.2%	84.5%	84.1%

Table 2 shows the sample queries from the test set with different types of bracketing. After annotation, we found that slightly over 60% of these queries were left bracketing and this is chosen as the baseline as in previous work [22].

Table 3 demonstrates the performance of the adjacency model using different word association measures and on two different query datasets. For the query bracketing task, we see that the adjacency model can very accurately determine the bracketing of three-word queries. The best feature achieves more than 93% accuracy and is thus very close to human annotation. We found that the adjacency model is more suitable for query bracketing than the dependency model which lags a few percentage points behind in accuracy, as not all queries are noun compounds and the relative merits of the two models are data dependent [29].

We summarize the bracketing results in Table 3 as follows:

First, as aforementioned, using the smoothed language models has significant performance advantage over using raw web counts as in previous work [22]. For instance, PMI with a smoothed language model reduces bracketing error by 42% than that estimated in a maximum likelihood fashion using web counts (6% to 7% accuracy gain).

Second, as expected, bracketing queries marked as weak is more difficult than those marked as strong (overall, accuracy deteriorates by around 2%). This implies that we can choose to apply the bracketing result as a proximity feature only when the model is confident enough (e.g. requiring the ratio of the word association features to be above some threshold); otherwise refrain from supplying such a feature.

Among the language models from different sources, we again find that *anchor* achieves better accuracy than *title*. Interestingly, *body* yields quite competitive results and sometimes outperforms *anchor*. We hypothesize that because the bracketing task is essentially analyzing the syntactic structure of a short three-word query (which typically appears in the form of a noun phrase), *body* captures such grammatical information to the best extent than the others in its language model. We also find that the *query* language model underperforms the *body* language model for analyzing the sub-query bracketing structure.

Finally, we find that pointwise mutual information to be the best word association feature for this task. Conditional probability, on the other hand, is the weakest as was found in previous work.

6. LONG QUERY SEGMENTATION

For the final SQP task, we turn our attention to analyze long queries (defined in this work as those with length greater than or equal to 4). Long queries account for more than 20% of the query mass and are often used by the user to convey more sophisticated information requests. Unfortunately, in practice the performance of many commercial search engines deteriorates with such complicated queries. Query segmentation aims to break the long query into semantic concepts which are conducive to the improvement of search results.

Despite the importance of this task, related research in this area is relatively little. For the goal of generating query substitutions, pointwise mutual information based on query terms was used in [19] to segment queries. Similar term based PMI was used in [21] to find high quality sub-queries. [28] suggests that term-based PMI is not appropriate to capture the correlation in long entities and proposed a segment based segmentation approach. Raw web frequency and EM were used to estimate the model parameters. We adopt a similar rationale for segmentation, but using web n-gram language models which significantly simplify the segmentation model. [7] proposed a supervised approach for segmenting noun phrase queries, yet this requires developing a large gold standard and well designed feature sets from queries that are widely different. In contrast to the flat query segments produced from these previous works, the segmentation technique proposed in this section generates a parsing tree structure, which is more useful both for human interpretation and for developing proximity ranking features. We also note that unlike many of these previous works or the NLP literature on parsing, the ultimate goal of this SQP task is not to correctly parse the long query but to improve the retrieval performance of long queries. Therefore, the query segmentation task is a means (but not an end) to either spawn meaningful segments for dynamical ranking as aforementioned, or generate short query rewrites.

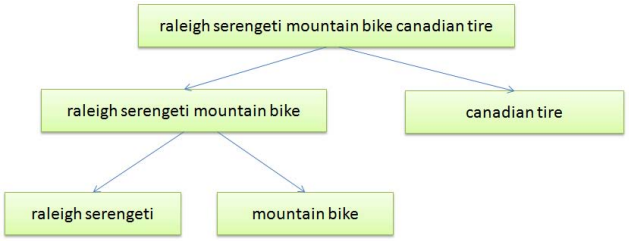

Figure 5: An example of segmenting a long query and representing it in a segmentation tree structure.

We begin with an example to motivate our proposed method for long query segmentation. As we observed from query logs (also found by previous work [5]), long queries are usually constructed as a concatenation of keywords/concepts (primarily noun phrases), each qualifying a certain aspect of the information request. Figure 5 illustrates an example long query which is constituted by an interested item (`raleigh serengeti mountain bike`) and a shopping site (`canadian tire`). Hence segments, instead of terms, are good units for analysis. Moreover, each segment may have its own substructure that can be best represented in a parsing tree structure (termed *segmentation tree*). In this example, the

segment `raleigh serengeti` qualifies the brand of the item rather than the shopping site. A segmentation tree also offers the flexibility for the practitioner to choose the desired granularity of segmentation (e.g., whether to segment `raleigh serengeti mountain bike` into two parts).

Similar to using the different measures of word association for query bracketing in the previous section, we can use the degree of independence of the segments as a criterion for segmenting a long query. Take the PMI metric as an example. Given a long query $\mathbf{q} = w_1 w_2 ... w_k$ and a segmentation boundary $t (2 \leq t \leq k)$ which segments $\mathbf{q}$ into $\mathbf{q}_l = w_1 w_2 ... w_{t-1}$ and $\mathbf{q}_r = w_t w_{t+1} ... w_k$, we define the segment-based pointwise mutual information (SPMI) as,

$$\text{SPMI}(\mathbf{q}, t) = \log \frac{P(\mathbf{q}_l \mathbf{q}_r)}{P(\mathbf{q}_l) P(\mathbf{q}_r)} \tag{7}$$

With the Markov property and using an order r language model, we can derive the SPMI metric as the following (see detailed derivation in Appendix):

$$\text{SPMI}(\mathbf{q}, t) = \sum_{i=0}^{\min\{r-2, k-(r-2)\}} \log \frac{p(w_{t+i} | w_{t+i-r+1} .. w_{t+i-1})}{p(w_{t+i} | w_t .. w_{t+i-1})} \tag{8}$$

Hence SPMI measures the degree of independence of the segments by evaluating the cumulative discrepancy between the probability of the n-grams spanning across the boundary and those separated by the boundary.

We opt to use a best first approach, similar to those used for growing decision trees, to generate the segmentation tree for a long query. Formally, let $\mathcal{C}$ denote the set of tree nodes considered for splitting in a segmentation step. The chosen tree node $\mathbf{n}^*$ by the best first approach is,

$$\mathbf{n}^* = \arg\min_{\mathbf{n} \in \mathcal{C}} \min_{2 \leq t \leq |\mathbf{q}(\mathbf{n})|} \text{SPMI}(\mathbf{q}(\mathbf{n}), t) \tag{9}$$

where $\mathbf{q}(\mathbf{n})$ denotes the query segment of node $\mathbf{n}$. The chosen node $\mathbf{n}^*$ is then segmented to a non-empty left and right node. The algorithm runs in iterations until the minimum SPMI reaches a termination threshold. In practice, a good way to determine the termination threshold is to observe the ROC curve of the algorithm on a held out set and choose the one satisfying certain criteria (for instance, optimal tradeoff between true positives and false positives). We will return to this point in the experiment.

For this SQP task, we randomly sampled 2,086 queries from the search query log. As our goal is to improve the search results of long queries, independent human judges were asked to discover and annotate important phrasal segments from these queries. 1,462 queries with at least one annotated segment agreed upon by the human judges were retained for evaluation. In total, we obtained 1,863 valid annotations which our evaluation is based upon.

Since the proposed segmentation algorithm produces a tree structure, we identify the different cases where the human annotated segment matches those in the segmentation tree:

- *Exact match*: the annotated segment exactly matches a node of the segmentation tree. A special case is *exact leaf match*, where the match occurs in a leaf node.
- *Cover*: a leaf node covers the annotated segment.

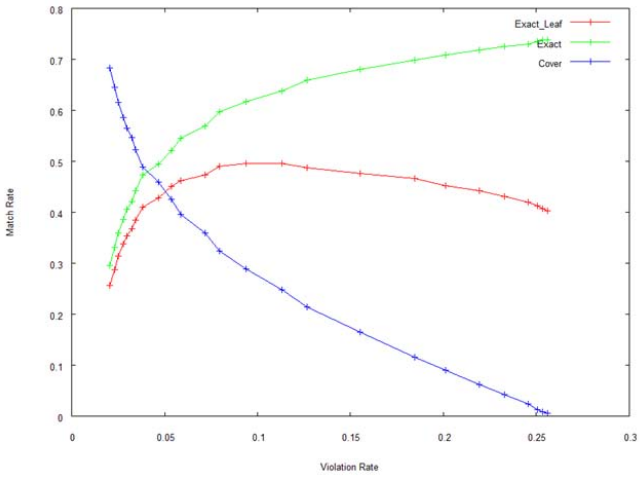

Figure 6: Segmentation performance using trees of different granularity (*anchor* language model).

- *Violation*: the annotated segment spans across different nodes (and not matching any particular node) in the tree.

We note that exact leaf match is the most preferred case. Eyeballing the results, exact match in the internal nodes usually identifies the detailed structure of the human annotated segment (since human annotation focuses on the precision rather than the recall of the segments). Hence both the matching node and its offsprings are appropriate as proximity features. For instance, in our example the segment `raleigh serengeti mountain bike` is tagged by human judges. However, the two sub-segments `raleigh serengeti` and `mountain bike` can also be useful proximity features . On the other hand, proximity features from the violation cases are detrimental for dynamic ranking while those from the cover cases are neutral to the results.

We vary the different thresholds of SPMI to generate segmentation trees with different levels of granularity using the *anchor* language model. As shown in Figure 6, the coarsest segmentation trees have the most cover cases. As we let the trees to be finer grained, the segmentation trees start to reveal the important segments human judges annotated. Most of the cover cases convert to exact matches as shown by the sharp growth of exact (leaf) match rates. A turning point appears at around 0.12 violation rate, where the exact leaf match rate peaks and starts to deteriorate. Exact match rate also grows slower than the violation rate after this point. Setting the violation rate at this point, the algorithm can discover about 70% of the human annotated segments deemed important (50% found as leaf nodes exactly matching human judgments).

We further investigate the segmentation performance using language models from different document sources[13]. Figure 7 illustrates the tradeoff between violation rate and exact match rate. Similar to a ROC curve, the curve covers the most area underneath it is the best. Among the language models, *anchor* clearly yields the best performance. This is because anchor texts are usually single or a combination

[13]The query language model yields slightly inferior results as in the previous task for similar reasons.

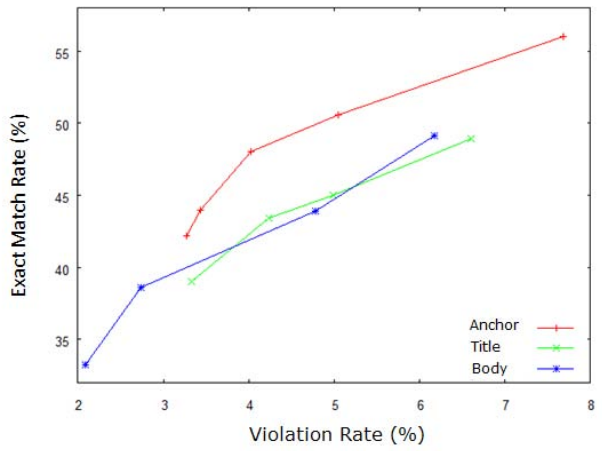

Figure 7: Tradeoff between the exact match rate and the violation rate for long query segmentation using different n-gram language models.

of keywords, most similar to the way long queries are composed. The performance difference of the *body* and *title* language models are not particularly clear cut in this case.

7. CONCLUSION

The vast world wide web offers a tremendous amount of treasure freely available *in the wild*. This work chooses the n-gram language model, a representation that has found many successful applications in IR and NLP, to capture the web language information hidden in the massive unlabeled and unfiltered web data. Enabled by the recent advances in web-scale language modeling techniques, we built language models from web document sources including the *anchor texts*, the *titles* and the document *bodies*, as well as from search queries. We conduct a large-scale information theoretical analysis on these language models to quantitatively examine the language differences in these data sources with different orders and sizes. We find that web search queries are composed in a way most similar to how authors summarize documents in anchor texts or titles.

In practice, web language models can be applied in a variety of document and query processing tasks in a web search engine. This work focuses on three Search Query Processing (SQP) tasks. Through the lens of context-sensitive query spelling correction, we observe how the difference of language translates to that of the accuracy in a real-world application. This suggests that the alternative data sources can be even more effective than the document bodies that have been used in past work. We further analyze the sub-query bracketing structure for three-word queries and propose a novel hierarchical long query segmentation method using web language models. The methods investigated in these SQP tasks are all unsupervised methods driven by the unlabeled web data that are abundantly available. As such, they can be readily and efficiently applied in the real world.

Our exploration yields several guidances for practice. We show that a theoretical measure – perplexity – to be a good accuracy indicator for these tasks and hence can be used to intelligently allocate engineering resources in practice. We demonstrate that smoothed n-gram models can significantly improve accuracies compared to web counts. With very large scale empirical studies, we also find that trigram models can significantly outperform unigram/bigram models, yet increasing the order of the language model beyond three generally yields diminishing returns in practice. Another important dimension – size – is explored in this work, which is often overlooked in past work. We find that increasing the sizes of the language models can both reduce the model perplexity and improve the accuracy in query-related tasks such as spelling correction. Unlike the order of the model, improvement from the power of the size of data does not appear to be easily capped even at the current web scale.

We identify several avenues as future directions. First, we are investigating how these applications of web language models, in conjunction with other modules such as dynamic ranking, can improve the relevance of search results, which is probably the utmost important indicators of user search experience. Besides these search query processing tasks, we will also leverage these different web-scale language models in many more document and query processing applications, such as document classification, OCR and machine translation. Finally, we advocate with Lapata [22] that the web language models can be a baseline for various IR/NLP research and the research on the modeling of the living web language *per se* can be a fruitful direction.

Acknowledgments

The authors would like to thank Zijian Zheng for the very helpful suggestions and collaboration.

8. ADDITIONAL AUTHORS

C. Lee Giles (Information Sciences and Technology, Pennsylvania State University, USA, email: `giles@ist.psu.edu`).

9. REFERENCES

[1] Hitwise 2009 press releases, 2009.

[2] Special issue on web as corpus. *Computational Linguistics*, 29(3), September 2003.

[3] E. Agichtein, E. Brill, and S. Dumais. Improving web search ranking by incorporating user behavior information. In *Proceedings of 29th international ACM conference on Research and development in information retrieval (SIGIR)*, pages 19–26, 2006.

[4] M. Banko and E. Brill. Scaling to very very large corpora for natural language disambiguation. In *Proceedings of 39th Annual Meeting on Association for Computational Linguistics (ACL)*, pages 26–33, 2001.

[5] C. Barr, R. Jones, and M. Regelson. The linguistic structure of english web-search queries. In *Proc. of Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1021–1030, 2008.

[6] S. Bergsma, D. Lin, and R. Goebel. Web-scale n-gram models for lexical disambiguation. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1507–1512, 2009.

[7] S. Bergsma and Q. I. Wang. Learning noun phrase query segmentation. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing (EMNLP) and Computational Natural Language Learning (CoNLL)*, pages 819–826, 2007.

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

[9] T. Brants, A. C. Popat, P. Xu, F. J. Och, and J. Dean. Large language models in machine translation. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing (EMNLP) and Computational Natural Language Learning (CoNLL)*, pages 858–867, 2007.

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

[11] K. Church, T. Hard, and J. Gao. Compressing trigram language models with Golomb coding. In *Proceedings of EMNLP and CoNLL*, pages 199–207, 2007.

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

[13] M. Gamon, J. Gao, C. Brockett, A. Klementiev, W. Dolan, D. Belenko, and L. Vanderwende. Using contextual speller techniques and language modeling for ESL error correction. In *Proc. of IJCNLP*, 2008.

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

[15] J. Gao, W. Yuan, X. Li, K. Deng, and J.-Y. Nie. Smoothing clickthrough data for web search ranking. In *Proceedings of the 32nd international SIGIR conference on Research and development in information retrieval (SIGIR)*, pages 355–362, 2009.

[16] A. R. Golding and Y. Schabes. Combining trigram-based and feature-based methods for context-sensitive spelling correction. In *Proceedings of the 34th ACL*, pages 71–78, 1996.

[17] A. Halevy, P. Norvig, and F. Pereira. The unreasonable effectiveness of data. *IEEE Intelligent Systems*, 24(2):8–12, 2009.

[18] X. Huang, A. Acero, and H.-W. Hon. *Spoken Language Processing: A Guide to Theory, Algorithm and System Development*. Prentice Hall PTR, 2001.

[19] R. Jones, B. Rey, O. Madani, and W. Greiner. Generating query substitutions. In *Proc. of 15th World Wide Web (WWW)*, pages 387–396, 2006.

[20] R. Kneser and H. Ney. Improved backing-off for M-gram language modeling. *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 1:181–184, 1995.

[21] G. Kumaran and V. R. Carvalho. Reducing long queries using query quality predictors. In *Proc. of 32nd international conf. on Research and development in information retrieval (SIGIR)*, pages 564–571, 2009.

[22] M. Lapata and F. Keller. The web as a baseline: Evaluating the performance of unsupervised web-based models for a range of nlp tasks. In *Proc. of Human Language Technologies - North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*, pages 121–128, 2004.

[23] M. Lauer. Corpus statistics meet the noun compound: some empirical results. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics (ACL)*, pages 47–54, 1995.

[24] P. Nakov and M. Hearst. Search engine statistics beyond the n-gram: Application to noun compound bracketing. In *Proc. of 9th Conf. on Computational Natural Language Learning*, pages 17–24, 2005.

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

[26] A. Spink, D. Wolfram, M. B. J. Jansen, and T. Saracevic. Searching the web: the public and their queries. *Journal of American Society for Information Science and Technology*, 52(3):226–234, 2001.

[27] K. Svore and C. Burges. A machine learning approach for improved bm25 retrieval. In *Proceedings of 18th ACM Conference on Information and Knowledge Management (CIKM)*, pages 1811–1814, 2009.

[28] 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)*, pages 347–356, 2008.

[29] D. Vadas and J. R. Curran. Corpus statistics meet the noun compound: some empirical results. In *Proceedings of 10th Conference of the Pacific Association for Computational Linguistics (PACLING)*, pages 104–112, 2007.

[30] K. Wang and X. Li. Efficacy of a constantly adaptive language modeling technique for web-scale applications. In *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4733–4736, 2009.

[31] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to information retrieval. *ACM Transactions on Information Systems*, 22(2):179–214, 2004.

APPENDIX

A. DERIVATION OF THE SPMI METRIC

Using an order r n-gram LM with the Markov assumption,

$$P(\mathbf{q}) = P(w_1 w_2 ... w_k) = \prod_{i=1}^{k} P(w_i | w_{i-r+1}..w_{i-1})$$

With a segmentation boundary t, the probability of the left and right segments can be written similarly. Given the following SPMI definition:

$$\text{SPMI}(\mathbf{q}, t) = \log \frac{P(\mathbf{q}_l \mathbf{q}_r)}{P(\mathbf{q}_l) P(\mathbf{q}_r)} \tag{10}$$

we can write out individual terms as below,

$$\log \frac{\prod_{i=1}^{k} P(w_i | w_{i-r+1}..w_{i-1})}{\prod_{i=1}^{t-1} P(w_i | w_{i-r+1}..w_{i-1}) \prod_{i=t}^{k} P(w_i | w_{i-r+1}..w_{i-1})}$$

Note that most terms cancel out and only at most $r-2$ terms behind the segmentation boundary remain, i.e.

$$\log \frac{P(w_t | w_{t-r+1}..w_{t-1}) P(w_{t+1} | w_{t-r+2}..w_t) ... P(w_{t+r-2} | w_{t-1}..w_{t+r-3})}{P(w_t) P(w_{t+1} | w_t) ... P(w_{t+r-2} | w_t..w_{t+r-3})}$$

Simplifying and noting the query's right boundary, we get the desired result:

$$\text{SPMI}(\mathbf{q}, t) = \sum_{i=0}^{\min\{r-2, k-(r-2)\}} \log \frac{p(w_{t+i} | w_{t+i-r+1}..w_{t+i-1})}{p(w_{t+i} | w_t..w_{t+i-1})}$$

□