Unsupervised Query Segmentation Using Generative Language Models and Wikipedia

Bin Tan*
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801
bintan@uiuc.edu

Fuchun Peng
Yahoo! Inc.
701 First Avenue
Sunnyvale, CA 94089
fuchun@yahoo-inc.com

ABSTRACT

In this paper, we propose a novel unsupervised approach to query segmentation, an important task in Web search. We use a generative query model to recover a query's underlying concepts that compose its original segmented form. The model's parameters are estimated using an expectation-maximization (EM) algorithm, optimizing the minimum description length objective function on a partial corpus that is specific to the query. To augment this unsupervised learning, we incorporate evidence from Wikipedia.

Experiments show that our approach dramatically improves performance over the traditional approach that is based on mutual information, and produces comparable results with a supervised method. In particular, the basic generative language model contributes a 7.4% improvement over the mutual information based method (measured by segment F1 on the *Intersection* test set). EM optimization further improves the performance by 14.3%. Additional knowledge from Wikipedia provides another improvement of 24.3%, adding up to a total of 46% improvement (from 0.530 to 0.774).

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Retrieval models

General Terms

Algorithms

Keywords

Query segmentation, concept discovery

1. INTRODUCTION

A typical search engine of today usually allows a user to submit a bunch of keywords as a query. This simple user interface greatly eases use of the search engine, and is sufficient for the traditional bag-of-words retrieval model, where query and document are assumed to be composed of individual words neither related nor ordered. In the quest for higher retrieval accuracy, however, it is necessary to understand the user's search intent beyond the simple bag-of-words query model. At a minimum, we would like to know whether some words comprise an entity like an organization name, which makes it possible to enforce word proximity and ordering constraints on document matching, among other things.

A partial solution provided by many search engines is that a user can put double quotes around some query words to mandate they appear together as an inseparable group in retrieved documents. Aside from sometimes being overly limiting, this double-quote proximity operator requires additional efforts from the user. However, a typical user may be unaware of the syntax, or reluctant to make this clarification unless really dissatisfied with the results. It would be desirable if the structural relationship underlying the query words could be *automatically* identified.

Query segmentation is one of the first steps in this direction. It aims to separate query words into segments so that each segment maps to a semantic component, which we refer as a "concept". For example, for the query "new york times subscription", [new york times] [subscription] is a good segmentation, while [new] [york times] [subscription] and [new york] [times subscription] are not, as segments like [new york] and [york times] greatly deviate from the intended meaning of the query.

Ideally, each segment should map to exactly one "concept". For segments like [new york times subscription], the answer of whether it should be left intact as a compound concept or further segmented into multiple atomic concepts depends on the connection strength of the components (i.e. how strong / often are "new york times" and "subscription" associated) and the application (e.g., whether query segmentation is used for query understanding or document retrieval). This leaves some ambiguity in query segmentation, as we will discuss later.

Query segmentation could help a retrieval system to improve its accuracy, as segments carry implicit word proximity / ordering constraints, which may be used to filter documents. For example, we would expect "new" to be near (and probably just before) "york" in matching documents. Query segmentation should also be useful in other query processing tasks such as query rewriting and query log analysis, where one would be able to work on semantic concepts instead of individual words [2, 8, 21].

Interestingly, there is little investigation done of query segmentation, in spite of its importance. To our knowledge, two approaches have been studied in previous works. The first is based on (pointwise) mutual information (MI) between pairs of query words. In [23, 14], if the MI value between two adjacent words is below a predefined threshold, a segment boundary is inserted at that position. The problem with this approach is that MI, by definition, cannot capture correlations among more than two words, thus it cannot handle long entities like song names, where MI may be low in certain positions (e.g., between "heart" and "will" in "my heart will go on"). Another problem with this approach, which is also true with most unsupervised learning methods, is its heavy reliance on cor-

*work done while the author was an intern at Yahoo! Inc.

WWW 2008, April 21–25, 2008, Beijing, China.
ACM 978-1-60558-085-2/08/04.

pus statistics. In some cases highly frequent patterns with incomplete semantic meaning may be produced. For example, "leader of the" is a fairly frequent pattern, but it is certainly not a self-contained semantic concept.

The second approach [4] uses supervised learning. At each word position, a binary decision is made whether to create a segment boundary or not. The input features to the decision function are defined on words within a small window at that position. The features include part-of-speech tags, position, web counts, etc., with their weights learned from labeled training data. Although the window size is set to 4 to capture a certain degree of long-range dependencies, the segmentation decisions are still local in essence. Moreover, their features are specifically designed for noun-phrase queries (queries comprised of multiple noun phrases). For example, one feature is the web count of "X and Y" for two adjacent words X and Y, which is not helpful for general queries. Another problem with this approach, and with all supervised learning methods, is that it requires a significant number of labeled training samples and well designed features to achieve good performance. This makes it hard to adapt to other languages, domains, or tasks.

In this paper, we propose an *unsupervised* method for query segmentation that alleviates the shortcomings of the current approaches. Three characteristics make our approach unique. First, we use a statistical language modeling approach that captures the high-level sequence generation process rather than focusing on local between-word dependencies as mutual information does. Second, in order to avoid running expectation-maximization (EM) algorithm on the whole dataset, which is quite expensive, we propose an EM algorithm that performs optimization on the fly for incoming queries. Third, instead of limiting to use query logs, we combine resources from the Web. In particular, we use Wikipedia to provide additional evidence for concept discovery, which proves to be quite successful. In our experiments, we test our method on human annotated test sets, and find language modeling, EM optimization and Wikipedia knowledge each brings improvement to query segmentation performance.

In the rest of the paper, we will first describe other related work in Section 2. We then in Section 3 describe the generative language model for query segmentation. Section 4 discusses the techniques for parameter optimization, including a method for estimating long n-gram's frequency and an EM learning algorithm. In Section 5, we describe how to incorporate additional knowledge such as Wikipedia into our segmentation model. Practical system implementation is discussed in in Section 6. We present experiment results in Section 7 and its discussions in Section 8. Finally, we conclude the paper in Section 9.

2. RELATED WORK

In natural language processing, there has been a significant amount of research on text segmentation, such as noun phrase chunking [22], where the task is to recognize the chunks that consist of noun phrases, and Chinese word segmentation [5, 18, 25], where the task is to delimit words by putting boundaries between Chinese characters. Query segmentation is similar to these problems in the sense that they all try to identify meaningful semantic units from the input. However, one may not be able to apply these techniques directly to query segmentation, because Web search query language is very different (queries tend to be short, composed of keywords), and some essential techniques to noun phrase chunking, such as part-of-speech tagging [6], can not achieve high performance when applied to queries. Thus, detecting noun phrase for information retrieval has been mainly studied in document indexing [3, 11, 26, 24] rather than query processing.

In terms of unsupervised methods for text segmentation, the expectation maximization (EM) algorithm has been used for Chinese word segmentation [19] and phoneme discovery [16], where a standard EM algorithm is applied to the whole corpus. As pointed out in [19, 16], running EM algorithm over the whole corpus is very expensive. To avoid this costly procedure, for each incoming query, we run an EM algorithm on the fly over the affected sub-corpus only. Since a query is normally short and the sub-corpus relevant to it is also quite small, this online procedure is very fast. A second difference is that we augment unsupervised learning with Wikipedia as external knowledge, which dramatically improves query segmentation performance. We combine multiple evidence using the minimum description length principle (MDL), which has been applied widely to problems such as grammar induction [13] and word clustering [15]. To some extent, utilizing external knowledge is similar to having some seed labeled data for unsupervised learning as used in [1], only that Wikipedia is readily available.

Wikipedia has been used in many applications in NLP, including named entity disambiguation [7, 9], question answering [10], text categorization [12], and conference resolution [20]. To our knowledge, this is the first time that it is used in query segmentation.

3. A GENERATIVE MODEL FOR QUERY SEGMENTATION

When a query is being formulated in people's mind, it is natural to believe that the building blocks in the thinking process are "concepts" (possibly multi-word), rather than single words. For example, with the query "new york", the two words must have popped into one's brain together; if they were come up separately, the intended meaning would be vastly different. It is when the query is uttered (e.g., typed into a search box) that the concepts are "serialized" into a sequence of words, with their boundaries dissolved. The task of query segmentation is to recover the boundaries which separate the concepts.

Given that the basic units in query generation are concepts, we make the further assumption that they are independent and identically-distributed (I.I.D.). In other words, there is a probability distribution P_C of concepts, which is sampled repeatedly, to produce mutually-independent concepts that construct a query. This is essentially a unigram language model, with a "gram" being not a word, but a concept / segment.

The above I.I.D. assumption carries several limitations. First, concepts are not really independent of each other. For example, we are more likely to observe "travel guide" after "new york" than "new york times". Second, the probability of a concept may vary by its position in the text. For example, we expect to see "travel guide" more often at the end of a query than at the beginning. We could tackle the above problems by using a higher-order model (e.g., the bigram model) and adding a position variable, but this will dramatically increase the number of parameters that are needed to describe the model. Thus for simplicity the unigram model is used, and it proves to work reasonably well for the query segmentation task.

Let $T = w_1 w_2 \cdots w_n$ be a piece of text of n words, and $S^T = s_1 s_2 \cdots s_m$ be a possible segmentation consisting of m segments, where $s_i = w_{k_i} w_{k_i+1} \cdots w_{k_{i+1}-1},\ 1{=}k_1{<}k_2{<}\cdots{<}k_{m+1}{=}n{+}1$

For a given query Q, if it is produced by the above generative language model, with concepts repeatedly sampled from distribution P_C until the desired query is obtained, then the probability of it being generated according to an underlying sequence of concepts (i.e., a segmentation of the query) S^Q is

$$P(S^Q) = P(s_1)P(s_2|s_1)\cdots P(s_m|s_1 s_2 \cdots s_{m-1})$$

The unigram model says

$$P(s_i|s_1 s_2 \cdots s_{i-1}) = P_C(s_i)$$

Thus we have

$$P(S^Q) = \prod_{s_i \in S^Q} P_C(s_i)$$

And the cumulative probability of generating Q is

$$P(Q) = \sum_{S^Q} P(S^Q)$$

where S^Q is one of 2^{n-1} different segmentations, with n being the number of query words.

For two segmentations S_1^T and S_2^T of the same piece of text T, suppose they differ at only one segment boundary, i.e.,

$$S_1^T = s_1 s_2 \cdots s_{k-1} \boldsymbol{s_k s_{k+1}} s_{k+2} \cdots s_m$$
$$S_2^T = s_1 s_2 \cdots s_{k-1} \boldsymbol{s'_k} s_{k+2} \cdots s_m$$

where $s'_k = (s_k s_{k+1})$ is the concatenation of s_k and s_{k+1}.

Naturally, we will favor segmentations with higher probability of generating the query. In the above case, $P(S_1^T) > P(S_2^T)$ (thus is preferred) if and only if $P_c(s_k)P_c(s_{k+1}) > P_c(s'_k)$, i.e., when s_k and s_{k+1} are negatively correlated. In other words, a segment boundary is justified if and only if the pointwise mutual information between the two segments resulting from the split is negative:

$$\mathrm{MI}(s_k, s_{k+1}) = \log \frac{P_c(s'_k)}{P_c(s_k)P_c(s_{k+1})} < 0$$

Note that this is fundamentally different from the MI-based approach in [14]. MI as computed above is between adjacent segments, rather than words. More importantly, the segmentation decision is non-local (i.e., involving a context beyond the words near the segment boundary of concern): whether s_k and s_{k+1} should be joined or split depends on the positions of s_k's left boundary and s_{k+1}'s right boundary, which in turn involve other segment decisions.

If we enumerate all possible segmentations, the "best" segmentation will be the one with the highest likelihood to generate the query. We can also rank them by likelihood and output the top k. Having multiple possibly correct segmentations is desirable if the correct segmentation cannot be easily determined at query preprocessing time: with a small number of candidates one can afford to ask the user for feedback, or re-score them by inspecting retrieved documents.

In practice, segmentation enumeration is infeasible except for short queries, as the number of possible segmentations grows exponentially with query length. However, the I.I.D. nature of the unigram model makes it possible to use dynamic programming (Algorithm 1) for computing top k best segmentations. The complexity is $\mathrm{O}\big(n\,k\,m\,\log(k\,m)\big)$, where n is query length, and m is maximum allowed segment length.

Input: query $w_1 w_2 \cdots w_n$, concept probability distribution P_c
Output: top k segmentations with highest likelihood

$B[i]$: top k segmentations for sub-text $w_1 w_2 \cdots w_i$
For each segmentation $b \in B[i]$, $segs$ denotes the segments and $prob$ denotes the likelihood of the sub-text given this segmentation

for i **in** $[1..n]$
  $s \leftarrow w_1 w_2 \cdots w_i$
  if $P_C(s) > 0$
    $a \leftarrow$ **new** segmentation
    $a.segs \leftarrow \{s\}$
    $a.prob \leftarrow P_C(s)$
    $B[i] \leftarrow \{a\}$
  for j **in** $[1..i-1]$
    for b **in** $B[j]$
      $s \leftarrow w_j w_{j+1} \cdots w_i$
      if $P_C(s) > 0$
        $a \leftarrow$ **new** segmentation
        $a.segs \leftarrow b.segs \cup \{s\}$
        $a.prob \leftarrow b.prob \times P_C(s)$
        $B[i] \leftarrow B[i] \cup \{a\}$
  sort $B[i]$ by $prob$
  truncate $B[i]$ to size k
return $B[n]$

Algorithm 1: Dynamic programming algorithm to compute top k segmentations

4. PARAMETER ESTIMATION

The central question that needs to be addressed is: how to determine the parameters of our unigram language model, i.e., the probability of the concepts, which take the form of variable-length n-grams. As this work focuses on unsupervised learning, we would like the parameters to be estimated automatically from provided textual data.

The primary source of data we use is a text corpus consisting of a 1% sample of the web pages crawled by the Yahoo! search engine. We count the frequency of all possible n-grams up to a certain length ($n = 1, 2, \cdots, 5$) that occur at least once in the corpus. It is usually impractical to do this for longer n-grams, as their number grows exponentially with n, posing difficulties for storage space and access time. However, for long n-grams ($n > 5$) that are also frequent in the corpus, it is often possible to approximate their counts using those of shorter n-grams.

4.1 Frequency lower bounds for long n-grams

We compute lower bounds of long n-gram counts using set inequalities, and take them as approximation to the real counts. For example, the frequency for "harry potter and the goblet of fire" can be determined to lie in the reasonably narrow range of [5783, 6399], and we just use 5783 as an estimate for its true frequency. A dynamic programming algorithm is given next.

If we have frequencies of occurrence in a text corpus for all n-grams up to a given length, then we can infer lower bounds of frequencies for longer n-grams, whose real frequencies are unknown. The lower bound is in the sense that any smaller number would cause contradictions with known frequencies.

Let $\#(x)$ denote n-gram x's frequency. Let A, B, C be arbitrary n-grams, and AB, BC, ABC be their concatenations. Let $\#(AB \vee BC)$ denote the number of times B follows A or is followed by C in the corpus. We have

$$\#(ABC) = \#(AB) + \#(BC) - \#(AB \vee BC) \quad (1)$$
$$\geq \#(AB) + \#(BC) - \#(B) \quad (2)$$

(1) follows directly from a basic equation on set cardinality:

$$|\mathcal{X} \cap \mathcal{Y}| = |\mathcal{X}| + |\mathcal{Y}| - |\mathcal{X} \cup \mathcal{Y}|$$

where $\mathcal{X}$ is the set of occurrences of B where B follows A and $\mathcal{Y}$ is the set of occurrences of B where B is followed by C.

Since $\#(B) \geq \#(AB \vee BC)$, (2) holds.

Therefore, for any n-gram $x = w_1 w_2 \cdots w_n$ ($n \geq 3$), if we define

$$f_{i,j}(x) \stackrel{\text{def}}{=} \#(w_1 \cdots w_j) + \#(w_i \cdots w_n) - \#(w_i \cdots w_j)$$

we have

$$\#(x) \geq \max_{1<i<j<n} f_{i,j}(x) \tag{3}$$

(3) allows us to compute the frequency lower bound for x using frequencies for sub-n-grams of x, i.e., compute a lower bound for all possible pairs of (i, j), and choose their maximum. In case $\#(w_1 \cdots w_j)$ or $\#(w_i \cdots w_n)$ is unknown, their lower bounds, which are obtained in an recursive manner, can be used instead. Note that what we obtain are not necessarily greatest lower bounds, if all possible frequency constraints are to be taken into account. Rather, they are best-effort estimates using the above set inequalities.

In reality, not all (i, j) pairs need to be enumerated: if $i \leq i' < j' \leq j$, then

$$f_{i,j}(x) \geq f_{i',j'}(x) \tag{4}$$

because: $\Big(\#(i,j) \stackrel{\text{def}}{=} \#(w_i w_{i+1} \cdots w_j) \Big)$

$$\begin{aligned}
& f_{i,j}(x) \\
= \; & \#(1,j) + \#(i,n) - \#(i,j) \\
\geq \; & \Big(\#(1,j') + \#(i,j) - \#(i,j')\Big) + \#(i,n) - \#(i,j) \\
= \; & \#(1,j') + \#(i,n) - \#(i,j') \\
\geq \; & \#(1,j') + \Big(\#(i',n) + \#(i,j') - \#(i',j')\Big) - \#(i,j') \\
= \; & \#(1,j') + \#(i',n) - \#(i',j') \\
= \; & f_{i',j'}(x)
\end{aligned}$$

where the inequalities are obtained using (2).

(4) means that there is no need to consider $f_{i',j'}(x)$ in the computation of (3) if there is a sub-ngram $w_i \cdots w_j$ longer than $w_{i'} \cdots w_{j'}$ with known frequency. This can save a lot of computation.

Algorithm 2 gives the frequency lower bounds for all n-grams in a given query, with complexity $\mathrm{O}(n^2 m)$, where m is the maximum length of n-grams whose frequencies we have counted (5, in our case).

Next we explore several possible ways to estimate the n-gram probabilities given that their frequencies are known to us.

4.2 Estimation with raw frequencies

With the bag-of-words assumption that word occurrences are I.I.D., word frequencies in a piece of text follow a multinomial distribution. Given a set of word frequencies, the Maximum Likelihood Estimate (MLE) for the words' probabilities (parameterizing the multinomial distribution) is

$$P(w) = \frac{\#(w)}{\sum_{w' \in V} \#(w')}$$

where $\#(w)$ is word w's frequency, and V is the vocabulary of all words.

It is easy to adapt the above equation to compute n-gram probabilities in the concept distribution:

$$P_C(x) = \frac{\#(x)}{\sum_{x' \in V} \#(x')} \tag{5}$$

where $\#(x)$ is the "raw" frequency for n-gram x in the text corpus.

Input: query $w_1 w_2 \cdots w_n$, frequencies for all n-grams not longer than m
Output: frequencies (or their lower bounds) for all n-grams in the query

$C[i, j]$: frequency (or its lower bound) for n-gram $w_i \cdots w_j$

```
for l in [1..n]
    for i in [1..n − l + 1]
        j ← i + l − 1
        if #(w_i ··· w_j) is known
            C[i, j] ← #(w_i ··· w_j)
        else
            C[i, j] ← 0
            for k in [i + 1..j − m]
                C[i, j] ← max(C[i, j], C[i, k + m − 1]
                + C[k, j] − C[k, k + m − 1])
return C
```

Algorithm 2: Dynamic programming algorithm to compute lower bounds for n-gram frequencies

There are two serious problems with this approach, however. First, it is unclear what should be included in V, the set of n-grams whose frequency sum is used for normalization (so that probabilities sum up to 1). Obviously, we cannot include arbitrarily long n-grams, as there is an astronomical number of them. One way is to limit to n-grams up to a certain length, for example, all n-grams whose exact frequencies have been counted (in our case, $n \leq 5$). But it is hard to justify why the other n-grams should be excluded from the normalization sum.

More importantly, the probability of an n-gram in the concept distribution should describe how likely the n-gram is to appear in a piece of text *as an independent concept*. However, this cannot be captured by the raw frequencies. For example, since it is always true

$$\#(\text{york times}) \geq \#(\text{new york times})$$

MLE will infer

$$P_C(\text{york times}) \geq P_C(\text{new york times})$$

but obviously "york times" is unlikely to appear alone; $P_C(\text{york times})$ should be very small. This inherent drawback of using raw n-gram frequencies with MLE leads us to look for alternative methods that preserve some form of concept intactness.

4.3 Estimation with EM Optimization

Suppose we have already segmented the *entire* text corpus into concepts in a preprocessing step. We can then use (5) without any problem: the frequency of an n-gram will be the number of times it appears in the corpus *as a whole segment*. For example, in a correctly segmented corpus, there will be very few "york times" segments (most "york times" occurrences will be in the "new york times" segments), resulting in a small value of $P_C(\text{york times})$, which makes sense. However, having people manually segment the documents is only feasible on small datasets; on a large corpus it will be too costly.

An alternative is unsupervised learning, which does not need human-labeled segmented data, but uses large amount of unsegmented data instead to learn a segmentation model. Expectation

maximization (EM) is an optimization method that is commonly used in unsupervised learning, and it has already been applied to text segmentation [16, 19]. The EM algorithm goes like this: In the E-step, the unsegmented data is automatically segmented using the current set of estimated parameter values, and in the M-step, a new set of parameter values are calculated to maximize the complete likelihood of the data which is augmented with segmentation information. The two steps alternate until a termination condition is reached (e.g. convergence).

The major difficulty is that, when the corpus size is very large (in our case, 1% of crawled web), it will still be too expensive to run these algorithms, which usually require many passes over the corpus and very large data storage to remember all extracted patterns.

To avoid running the EM algorithm over the whole corpus, we propose an alternative: running EM algorithm on the fly, only on a partial corpus that is specific to a query. More specifically, when a new query arrives, we extract parts of the corpus that overlap with it (we call this the *query-relevant partial corpus*), which are then segmented into concepts, so that probabilities for n-grams in the query can be computed. All non-relevant parts unrelated to the query of concern are disregarded, thus the computation cost is dramatically reduced.

We can construct the query-relevant partial corpus in a procedure as follows. First we locate all words in the corpus that appear in the query. We then join these words into longer n-grams if the words are adjacent to each other in the corpus, so that the resulting n-grams become longest matches with the query. For example, for the query "new york times subscription", if the corpus contains "new york times" somewhere, then the longest match at that position is "new york times", not "new york" or "york times". This longest match requirement is effective against incomplete concepts, which is a problem for the raw frequency approach as previously mentioned. Note that there is no segmentation information associated with the longest matches; the algorithm has no obligation to keep the longest matches as complete segments. For example, it can split "new york times" in the above case to "new york" and "times" if corpus statistics make it more reasonable to do so. However, there are still two artificial segment boundaries created at each end of a longest match (which means, e.g., "times" cannot associate with the word "square" following it but not included in the query). This is a drawback of our query-specific partial-corpus approach.

Because all non-query-words are disregarded, there is no need to keep track of the matching positions in the corpus. Therefore, the query-relevant partial corpus can be represented as a list of n-grams from the query, associated with their longest match counts:

$$\mathcal{D} = \{\big(x, c(x)\big) | x \in Q\}$$

where x is an n-gram in query Q, and $c(x)$ is its longest match count.

The partial corpus represents frequency information that is most directly related to the current query. We can think of it as a distilled version of the original corpus, in the form of a concatenation of all n-grams from the query, each repeated for the number of times equal to their longest match counts, with other words in the corpus all substituted by a wildcard:

$$\underbrace{x_1 x_1 \cdots x_1}_{c(x_1)} \underbrace{x_2 x_2 \cdots x_2}_{c(x_2)} \cdots \underbrace{x_k x_k \cdots x_k}_{c(x_k)} \underbrace{w w \cdots w}_{N - \sum_i c(x_i)|x_i|} \quad (6)$$

where $x_1, x_2, \cdots, x_k$ are all n-grams in the query, w is a wildcard word representing words not present in the query, and N is the corpus length. We denote n-gram x's size by $|x|$, so $N - \sum_i c(x_i)|x_i|$ is the length of the non-overlapping part of the corpus.

Practically, the longest match counts can be computed from raw frequencies efficiently, which we have either counted or approximated using lower bounds. The procedure is described below:

Given query Q, let x be an n-gram in Q, $\mathcal{L}(x)$ be the set of words that precede x in Q, and $\mathcal{R}(x)$ be the set of words that follow x in Q. For example, if Q is "new york times new subscription", and x is "new", then $\mathcal{L}(x) = \{\text{times}\}$ and $\mathcal{R}(x) = \{\text{york, subscription}\}$.

The longest match count for x is essentially the number of occurrences of x in the corpus not preceded by any word from $\mathcal{L}(x)$ and not followed by any word from $\mathcal{R}(x)$, which we denote as a.

Let b be the total number of occurrences of x, i.e., $\#(x)$.

Let c be the number of occurrences of x preceded by any word from $\mathcal{L}(x)$.

Let d be the number of occurrences of x followed by any word from $\mathcal{R}(x)$.

Let e be the number of occurrences of x preceded by any word from $\mathcal{L}(x)$ and at the same time followed by any word from $\mathcal{R}(x)$.

Then it is easy to see $a = b - c - d + e$

Algorithm 3 computes the longest match count. Its complexity is $O(l^2)$, where l is the query length.

Input: query Q, n-gram x, frequencies for all n-grams in Q
Output: longest match count for x

$c(x) \leftarrow \#(x)$
for $l \in \mathcal{L}(x)$
    $c(x) \leftarrow c(x) - \#(lx)$
for $r \in \mathcal{R}(x)$
    $c(x) \leftarrow c(x) - \#(xr)$
for $l \in \mathcal{L}(x)$
    for $r \in \mathcal{R}(x)$
        $c(x) \leftarrow c(x) + \#(lxr)$
return $c(x)$

Algorithm 3: Algorithm to compute n-gram longest match counts

If we treat the query-relevant partial corpus $\mathcal{D}$ as a source of textual evidence, we can use maximum a posteriori estimation (MAP), choosing parameters θ (the set of concept probabilities) to maximize the posterior likelihood given the observed evidence:

$$\theta = \arg\max P(\mathcal{D}|\theta) P(\theta)$$

where $P(\theta)$ is the prior likelihood of θ.

The above equation can also be written as

$$\theta = \arg\min \Big(-\log P(\mathcal{D}|\theta) - \log P(\theta) \Big)$$

where $-\log P(\mathcal{D}|\theta)$ is the description length of the corpus, and $-\log P(\theta)$ is the description length of the parameters. The first part prefers parameters that are more likely to generate the evidence, while the second part disfavors parameters that are complex to be described. The goal is to reach a balance between the two by minimizing the *combined* description length.

For the corpus description length, we have the following according to the distilled corpus representation in (6):

$$\begin{aligned} \log P(\mathcal{D}|\theta) \quad = \quad & \sum_{x \in Q} \log P(x|\theta) \cdot c(x) + \\ & \log \big(1 - \sum_{x \in Q} P(x|\theta)\big) \cdot \big(N - \sum_{x \in Q} c(x)|x|\big) \end{aligned}$$

where x is an n-gram in query Q, $c(x)$ is its longest match count, $|x|$ is the n-gram length, N is the corpus length, and $P(x|\theta)$ is the probability of the parameterized concept distribution generating x as a piece of text. The second part of the equation is necessary, as it keeps the probability sum for n-grams in the query in proportion to the partial corpus size.

The probability of text x being generated can be summed over all of its possible segmentations:

$$P(x|\theta) = \sum_{S^x} P(S^x|\theta)$$

where S^x is a segmentation of n-gram x. Note that S^x are hidden variables in our optimization problem.

For the description length of prior parameters θ, we compute it as

$$\log P(\theta) = \alpha \sum_{x \in \theta} \log P(x|\theta)$$

where α is a predefined weight, $x \in \theta$ means the concept distribution has a non-zero probability for x, and $P(x|\theta)$ is computed as above. This is equivalent to adding α to the longest match counts for all n-grams in the lexicon θ. Thus, the inclusion of long yet infrequent n-grams in the lexicon is penalized for the resulting increase in parameter description length.

To estimate the n-gram probabilities with the above minimum description length set-up, we use the variant Baum-Welch algorithm from [16]. We also follow [16] to delete from the lexicon all n-grams that reduce the total description length when deleted. The complexity of the algorithm is $O(kl)$, where k is the number of different n-grams in the partial corpus, and l is the number of deletion phases (usually quite small). In practice, the above EM algorithm converges quickly and can be done without user's awareness.

5. USE OF WIKIPEDIA

The main problem with the above minimum description length approach is that it only tries to optimize the statistical aspects of the concepts; there is no linguistic consideration involved to guarantee that the output concepts are well-formed ones. For example, for the query "history of the web search engine", the best segmentation is [history of the][web search engine]. This is because "history of the" is a relatively frequent pattern in the corpus, helping to reduce the corpus description length quite much when it is assigned a high probability. Clearly we need to resort to some external knowledge to make sure that the output segments are well-formed concepts, not just frequent patterns. Many such knowledge resources are available from the NLP works, for example, named entity recognizer and noun phrase models, each shedding unique insight on what a well-formed concept should look like. In this paper, we explore a new approach, using Wikipedia to guide query segmentation.

Wikipedia, the largest encyclopedia on the Internet, is known for its high-quality collaboratively edited page contents. We find the Wikipedia *article titles* particularly suitable to our needs: First, the articles span a huge number of topics with extremely wide coverage, ranging from persons to locations, from movies to scientific theorems, able to match the great diversity of web search queries. Second, most articles are about well-established topics that are known to a reasonably-sized community, thus we can avoid dealing with large numbers of infrequently used concepts (e.g. the name of a local business, which is more easily handled by a specialized name identifier). Third, the articles are updated very fast, thus one can keep the concept dictionary up with the latest trend.

There are two difficulties with Wikipedia titles. First, the titles are mostly canonicalized. For example, plural form of nouns are rarely seen in titles. To allow other forms to be used, we include anchor texts / aliases pointing to a page as alternatives to its title, if the frequency is not low. For example, our vocabulary includes both "U.S.A" and "United States", despite that only the latter is an article title. Second, some pages are about topic involving multiple entities, e.g., "magical objects in harry potter". In this case, we set a threshold on the number of in-links (eliminating those pages whose counts are beneath it, which tend to be discussing rare topics) and allow long titles to be segmented into shorter concepts. For example, we can get two concepts, "magical objects" and "harry potter" from the previous title. In August 2007, we extract a total of 2.3 million "concepts" from 1.95 million articles in the English Wikipedia.

To use Wikipedia titles, we introduce a third term in our description length minimization scheme:

$$\beta \sum_{x \in \mathcal{W}} \log P(x|\theta) \#_{\mathcal{W}}(x)$$

where β is a predefined weight, $\mathcal{W}$ is the collection of Wikipedia concepts, and $\#_{\mathcal{W}}(x)$ the count of x in titles and links. This way, we will prefer to have large probabilities for concepts that occur frequently as Wikipedia titles (or anchor texts).

6. SYSTEM ARCHITECTURE

To use the parameter estimation algorithms for query segmentation, which is a task usually demanding real-time response, it is critical to have fast access to arbitrary n-grams' frequencies. According to Algorithm 2, long n-grams' frequencies can often be approximated using those of shorter ones, but there are still a lot of n-grams whose frequencies need to be maintained. In our case, we count frequencies for all n-grams with length up to 5 and non-zero occurrences in the web corpus. Due to the huge corpus size (33 billion words in length), we choose to accumulate counts using a simple map-reduce procedure that is run on a Hadoop[1] cluster. The resulting ngram number is 464 million, which makes it almost impossible to store the entire ngram-to-frequency dictionary in memory. We tackle this using Berkeley DB[2], which is a high-performance storage engine for key/value pairs. Berkeley DB enables us to maintain an in-memory cache, from which the most frequent n-grams can be looked up quickly (without reading from disk). We use BTree as the access method. We sort the ngram-frequency pairs before inserting them into DB, so that the resulting database file is highly compact. The cache size is set to 1GB.

Figure 1 illustrates the architecture of our system. With such an implementation, our system can segment 500 queries per second on a single machine.

7. EXPERIMENTATION

7.1 Data sets

We experiment on the data sets previously used by [4] for supervised query segmentation, so that it is easy to make performance comparison between our work and theirs. According to [4], the queries were sampled from the AOL search logs [17], each of length four or greater, with only determiners, adjectives and nouns as query words, and having at least one clicked results.

There are three sets, each containing 500 queries, one for training, one for validation, and another for testing. One annotator (we

[1] http://lucene.apache.org/hadoop

[2] http://www.oracle.com/database/berkeley-db

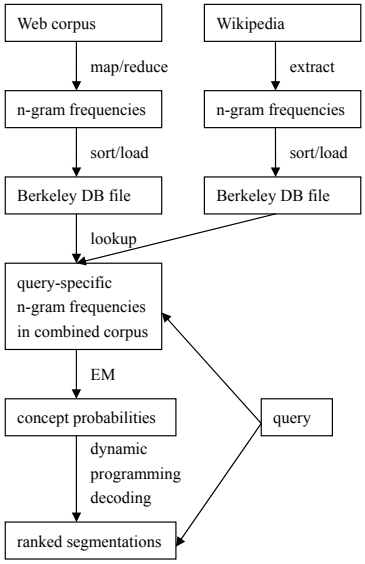

Figure 1: System architecture

call him/her *A*) manually segmented all three sets, and two additional annotators (*B* and *C*) segmented the testing set to provide alternative judgments. Since our work is unsupervised query segmentation, we have no need for the training set, and we are able to test against all three annotators. We use the combined training and validation sets for parameter tuning, and evaluate on the testing set.

As will be discussed later, the inter-annotator agreement is found to be quite low. For example, A gives an identical segmentation only to 58% of all queries when checking with B, and 61% with C. This vast inconsistency arises largely from the difficulty in appropriately defining a segment. For example, one might think "bank manager" should be kept in a single segment, but when retrieving documents, it is advantageous to separate them to two segments, to match such text as "manager of ABC bank". We evaluate our results against all three annotators as a result of this segmentation ambiguity. As in [4], we pick out all queries for which the three annotators make a unanimous segmentation decision. This set of 219 queries is referred as *Intersection*. We make another derived test set called *Conjunction*, on which it is considered correct when a segmentation matches what is given by *any* of the three annotators.

7.2 Evaluation Metrics

We adopt three different evaluation metrics:

1. **Classification accuracy**

   Bergsma and Wang [4] view query segmentation as a classification problem: at each position between two words, one makes a binary decision on whether to insert a segment boundary or not. Thus one can measure the accuracy of the classification decisions. The metric is known as Seg-Acc in [4].

2. **Segment accuracy**

   If we treat the human-annotated segmentation as a set of "relevant" segments, we can measure how well the predicted segmentation recovers these segments. Using Information Retrieval terminology, we compute precision (percentage of segments in the predicted segmentation that are relevant), recall (percentage of relevant segments that are found in the predicted segmentation) and F measure (2 × precision × recall / (precision + recall)).

3. **Query accuracy**

   This is the percentage of queries for which the predicted segmentation matches the human-annotated one completely. It is the same as Qry-Acc in [4].

We see that from classification accuracy to segment accuracy, to query accuracy, the metrics become more strict in judging how well the predicted segmentation matches the human-annotated one.

7.3 Results

We use mutual information based segmentation as our baseline (referred as *MI*), whose parameters are tuned on the training + validation query set. Our methods include the language modelling approach using MLE on raw n-gram frequencies (referred as *LM*), LM approach optimized by the EM algorithm (*EM*), LM approach augmented by Wikipedia knowledge (*LM+Wiki*), and EM approach augmented by Wikipedia knowledge (*EM+Wiki*).

We set α (weight on parameter description length) to 10, and β (weight on Wikipedia knowledge) to 100000. Both are tuned on the training + validation set.

Table 1 presents the evaluation results of different query segmentation algorithms on the test sets, with the best performance of segment F on each dataset in bold. There are some interesting observations from Table 1. The first thing to note is that all three types of segmentation accuracy are highly consistent: the Pearson's linear correlation coefficients is 0.91 between segment F and classification accuracy, and 0.89 between segment F and query accuracy.

For web search, it is particularly important to be able to identify query concepts, which is directly measured by segment accuracy. Thus we place special emphasis on this metric in the following discussion. We also draw a bar chart (Figure 2) comparing segment F of different algorithms across the test sets.

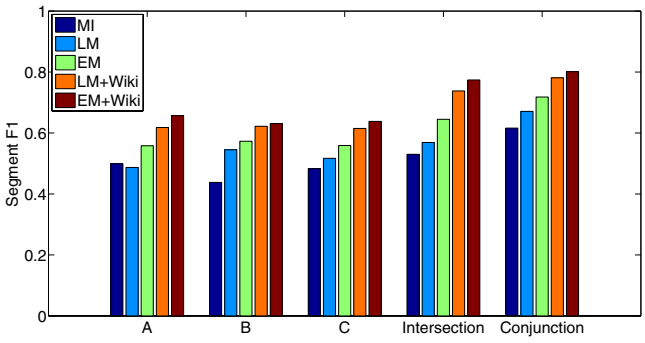

Figure 2: Segment F of different query segmentation algorithms

From Table 1 (or Figure 2), we observe a consistent trend of segment F increasing from left to right, with the only exception on dataset A, where there is a slight decrease from *MI* (0.499) to *LM* (0.487) (statistically insignificant with p-value of t-test being 0.47).

Comparing *LM* with *MI*, *LM* has much higher query accuracy (increases range from 30% (on A) to 78% (on B)) and classification accuracy (increases range from 7% (on A) to 22% (on B)), with all increases being statistically significant (p-value of t-test is close to 0). For segment accuracy, *LM* tends to have balanced precision and recall, while *MI* tends to have higher recall than precision. This is because *MI*, being a local model that does not capture

Table 1: Performance of different query segmentation algorithms

		MI	LM	EM	LM+Wiki	EM+Wiki
Annotator A	query accuracy	0.274	0.356	0.414	0.500	0.526
	classification accuracy	0.693	0.741	0.762	0.801	0.810
	segment precision	0.469	0.502	0.562	0.631	0.657
	segment recall	0.534	0.473	0.555	0.608	0.657
	segment F	0.499	0.487	0.558	0.619	**0.657**
Annotator B	query accuracy	0.244	0.436	0.440	0.508	0.494
	classification accuracy	0.634	0.776	0.774	0.807	0.802
	segment precision	0.408	0.552	0.568	0.625	0.623
	segment recall	0.472	0.539	0.578	0.619	0.640
	segment F	0.438	0.545	0.573	0.622	**0.631**
Annotator C	query accuracy	0.264	0.394	0.416	0.492	0.494
	classification accuracy	0.666	0.754	0.759	0.797	0.796
	segment precision	0.451	0.528	0.558	0.622	0.634
	segment recall	0.519	0.507	0.561	0.608	0.642
	segment F	0.483	0.517	0.559	0.615	**0.638**
Intersection	query accuracy	0.343	0.468	0.528	0.652	0.671
	classification accuracy	0.728	0.794	0.815	0.867	0.871
	segment precision	0.510	0.580	0.640	0.744	0.767
	segment recall	0.550	0.560	0.650	0.733	0.782
	segment F	0.530	0.569	0.645	0.738	**0.774**
Conjunction	query accuracy	0.381	0.570	0.597	0.687	0.692
	classification accuracy	0.758	0.846	0.856	0.890	0.891
	segment precision	0.582	0.680	0.715	0.787	0.797
	segment recall	0.654	0.663	0.721	0.775	0.807
	segment F	0.616	0.671	0.718	0.781	**0.801**

long dependencies, tends to create more segments, thus producing high recall yet low precision. Overall (with the F measure), the *LM* model achieves higher segment accuracy, up to a 24% increase (on B). The fact that the language modeling approach performs significantly better than the mutual information approach shows the advantage of modeling query structure with the concept generation process.

The performance of *LM* is further improved with EM optimization. Comparing to *LM*, segment F has an additional improvement of 5% - 15% on different datasets. This confirms the benefit of the iterative optimization algorithm to produce more accurate concept probabilities.

With the addition of Wikipedia knowledge, segmentation performance is dramatically improved. Comparing *LM+Wiki* with *LM*, segment F increases significantly from 14% to 30% on different datasets. Comparing *EM+Wiki* with *EM*, there is also a segment F increase from 10% to 20% on different datasets. The merit of Wikipedia being a high-quality repository of concepts is also seen in the fact that its weight β is tuned to a large number 100000 on the training + validation sets (which means that any n-gram found as a concept in Wikipedia would receive 100000 "bonus" frequency in corpus). The effect of Wikipedia even overshadows that of the EM algorithm: the segment F improvement from *LM+Wiki* to *EM+Wiki* is in the range of only 1.4% to 6%.

The effectiveness of Wikipedia data makes us to wonder if such a good resource will render our language model unnecessary. We have tried a simple longest string match segmentation algorithm using Wikipedia as the dictionary. The result is summarized in Table 2 for the *Intersection* dataset, which turns out to be rather poor. Therefore, Wikipedia knowledge is not the only contributor to the good performance we have obtained; language modeling and the EM optimization algorithm are also very important.

Table 2: Segmentation performance from longest string match with Wikipedia title dictionary

query accuracy	0.259
classification accuracy	0.667
segment precision	0.444
segment recall	0.525
segment F	0.481

The supervised method [4] reported 0.892 of classification accuracy and 0.717 of query accuracy on the *Intersection* dataset. Segment accuracy was not measured. In comparison, we obtain a 0.871 classification accuracy and a 0.671 query accuracy on this dataset using *EM+Wiki*. Since we do not have the segmentation output of the supervised method, we can not perform significance test on the performance difference. On the other hand, it is difficult to have a head to head comparison between the supervised and unsupervised methods because different resources and methods are used. Their method relies on a good POS tagger and the features are specifically designed for noun phrase queries (many of those features would be of little use to other types of queries). On the other hand, our method makes no such assumptions. Also, their classifier was specifically tuned to the peculiarities of annotator A; it was mentioned that "results are lower" for the other two annotators, while our unsupervised method produces similar performance with different annotators.

8. DISCUSSIONS AND ERROR ANALYSIS

One of the major reasons why query segmentation is difficult is its inherent ambiguity. Among the 500 queries, there are only 219 queries for which the 3 annotators give identical segmentation. Ta-

ble 3 reports the inter-agreement among the annotators, evaluating the segmentations by one annotator against those of another. A and C has the highest agreement, yet still it is poor: using A as the golden truth, C has only query accuracy of 0.606, classification accuracy of 0.847, and segment accuracy of 0.718. Considering this intrinsic difficulty, our segmentation accuracy is quite encouraging: our performance on *Conjunction* is as high as 0.801 for segment F, which means we can correctly identify more than 80% of the concepts from the queries. Comparing to the baseline *MI* with segment F of 0.616, our approach achieves 30% improvement.

Table 3: Segmentation annotation inter-agreement

		A	B	C
Annotator A	query accuracy		0.580	0.606
	classification accuracy		0.842	0.847
	segment precision		0.698	0.723
	segment recall		0.676	0.713
	segment F		0.687	0.718
Annotator B	query accuracy	0.580		0.604
	classification accuracy	0.842		0.842
	segment precision	0.677		0.691
	segment recall	0.699		0.701
	segment F	0.688		0.696
Annotator C	query accuracy	0.610	0.604	
	classification accuracy	0.849	0.842	
	segment precision	0.716	0.701	
	segment recall	0.725	0.691	
	segment F	0.720	0.696	

The biggest limitation to the current dataset provided by [4] is that the queries only contain noun phrases. But general web search queries has many other types, and a method specifically tailored to noun phrase queries would not work for an arbitrary query. To test our unsupervised method, we internally sampled a small random set of 160 queries and manually segmented them. Experiments (details not reported) confirm the high performance of our approach compared to the mutual information based one.

We have performed an error analysis of the segmentations from the EM + Wiki method that do not agree with *Intersection*, and categorize them as follows:

1. Ambiguous (segmentation-wise) noun phrase

   There are many phrases composed of nouns, which can be either treated as composite concepts, or segmented into smaller ones. Examples include "NCAA bracket", "salt deficiency" and "dirt bike parts". Both ways of doing segmentation are reasonable when a phrase is the combination of its components without much altering in meaning. A composite concept may be more useful for query understanding (as component words are grouped together), while separate smaller concepts are more flexible in retrieval (as component words do not have to appear adjacent to each other in a matching document). These constitute the largest proportion of disagreements. A good solution to this problem, which we are considering as a future extension to this work, is to build a concept hierarchy or graph that models the dependency between concepts.

2. Incorrect frequent patterns

   These are patterns that occur so frequently in the corpus that our method choose to keep them in segments, even if they are not good concepts. For example, for "lighted outdoor palm tree", our method puts the first two words in a segment, since they are highly correlated in the corpus. Some form of linguistic analysis is needed to identify and correct these cases.

3. Interference from Wikipedia

   When a part of the query happens to match in Wikipedia, the result can sometimes be undesirable. For example, for "the bang bang gang" (a movie title), we mistakenly make "bang bang" a segment, which is a title of a Wikipedia article. The mistake wouldn't have happened, however, if the correct segment were frequent enough in the corpus.

4. High-order dependence

   For the query "hardy county virginia genealogy", our incorrect segmentation is [hardy county][virginia genealogy]. [virginia genealogy] may be an acceptable segmentation for a query containing just these two words, but certainly not for the current query, where "genealogy" is about "hardy county". To solve this, we need a higher-order model which can capture the dependence between the non-adjacent [hardy county] and [genealogy].

9. CONCLUSIONS AND FUTURE RESEARCH

We have proposed a novel unsupervised approach to query segmentation based on a generative language model. To estimate the parameters (concept probabilities) efficiently, we use a novel method to estimate frequencies of long n-grams, and an EM algorithm that does optimization on the fly. We also explore additional evidence from Wikipedia to augment unsupervised learning. Experiments demonstrate the effectiveness and efficiency of our approach, with segment accuracy increasing to 0.774 from 0.530 of the traditional mutual information based approach.

For the future work, the first thing to explore is higher order language models. Currently we only use the unigram language model, and we expect a higher order language model would capture dependencies in long queries better.

Second, since our minimum description length framework allows additional knowledge to be incorporated easily, we would like to use more resources, especially noun phrase and named entity models. We are not going to directly apply noun phrase chunking to queries, as this would not work well for general queries. Instead, we can run a noun phrase chunker on the Web documents, and collect all extracted noun phrases, which we can then use to estimate the probability of a segment being a noun phrase.

Finally, we want to evaluate our methods in the context of Web search, to see how well they can be used to improve proximity matching and query expansion. After all, improving Web search is our major motivation of developing better query segmentation techniques and thus the ultimate evaluation standard.

10. ACKNOWLEDGMENT

We would like to thank Nawaaz Ahmed for his insightful discussions during our study and his comments on the draft, Benoit Dumoulin, Yumao Lu and Chengxiang Zhai for their comments on the draft.

11. REFERENCES

[1] R. K. Ando and L. Lee. Mostly-Unsupervised Statistical Segmentation of Japanese Kanji Sequences. *Journal of Natural Language Engineering*, 9:127–149, 2003.

[2] P. Anick. Exploiting clustering and phrases for context-based information retrieval. *ACM SIGIR Forum*, 31:314–323, 1997.

[3] A. Arampatzis, T. van der Weide, C. Koster, and P. van Bommel. Phrase-based Information Retrieval. *Information Processing and Management*, 34(6):693 – 707, 1998.

[4] S. Bergsma and Q. I. Wang. Learning Noun Phrase Query Segmentation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 819–826, 2007.

[5] M. R. Brent and T. A. Cartwright. Distributional regularity and phonotactic constraints are useful for segmentation. *Cognition*, 61:93 –125, 1996.

[6] E. Brill and G. Ngai. Man vs. Machine: A Case Study in Base Noun Phrase Learning. In *Proceedings of 37th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 65–72, 1999.

[7] R. Bunescu and M. Pasca. Using Encyclopedic Knowledge for Named Entity Disambiguation. In *Proceedings of 11th Conference of European Chapter of the Association for Computational Linguistics (EACL)*, pages 9–16, 2006.

[8] Y. Chang, I. Ounis, and M. Kim. Query Reformulation Using Automatically Generated Query Concepts from a Document Space. *Information Processing and Management*, 42 (2):453 – 468, 2006.

[9] S. Cucerzan. Large-Scale Named Entity Disambiguation Based on Wikipedia Data. In *Proceedings of EMNLP-CoNLL 2007*, pages 708–716, 2007.

[10] D. Ahn and V. Jijkoun and G. Mishne and K. Muller and M. de. Rijke. Using Wikipedia at the TREC QA Track. In *The Thirteenth Text Retrieval Conference (TREC 2004)*, 2005.

[11] D. Evans and C. Zhai. Noun-phrase Analysis in Unrestricted Text for Information Retrieval. In *34th Annual Meeting of the Association Computational Linguistics (ACL)*, pages 17–24, 1996.

[12] E. Gabrilovich and S. Markovitch. Overcoming the Brittleness Bottleneck using Wikipedia: Enhancing Text Categorization with Encyclopedic Knowledge. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI)*, pages 1301–1306, 2006.

[13] P. Grunwald. A Minimum Description Length Approach to Grammar Inference. In G. S. S. Wermter, E. Riloff, editor, *Symbolic, Connectionist and Statistical Approaches to Learning for Natural Language Processing*, pages 203–216, 1996.

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

[15] H. Li and N. Abe. Clustering Words with the MDL Principle. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING)*, pages 5–9, 1996.

[16] C. G. D. Marcken. *Unsupervised language acquisition*. PhD thesis, MIT, 1996. Supervisor: Robert C. Berwick.

[17] G. Pass, A. Chowdhury, and C. Torgeson. A Picture of Search. In *The First International Conference on Scalable Information Systems*, 2006.

[18] F. Peng, F. Feng, and A. McCallum. Chinese Segmentation and New Word Detection using Conditional Random Fields. In *Proceedings of The 20th International Conference on Computational Linguistics (COLING)*, pages 562–568, 2004.

[19] F. Peng and D. Schuurmans. Self-Supervised Chinese Word Segmentation. In *IDA '01: Proceedings of the 4th International Conference on Advances in Intelligent Data Analysis*, pages 238–247, 2001.

[20] S. P. Ponzetto and M. Strube. Exploiting Semantic Role Labeling, WordNet and Wikipedia for Coreference Resolution. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 192 – 199, 2006.

[21] Y. Qiu and H.-P. Frei. Concept based Query Expansion. In *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR)*, pages 160–169, 1993.

[22] L. Ramshaw and M. Marcus. Text Chunking Using Transformation-Based Learning. In *Proceedings of the Third Workshop on Very Large Corpora*, pages 82–94, 1995.

[23] K. M. Risvik, T. Mikolajewski, and P. Boros. Query Segmentation for Web Search. In *The Twelfth International World Wide Web Conference (WWW)*, 2003.

[24] J. I. Serrano and L. Araujo. Statistical Recognition of Noun Phrases in Unrestricted Text. In *IDA '05: Proceedings of the 6th International Conference on Advances in Intelligent Data Analysis*, pages 397–408, 2005.

[25] R. Sproat, W. Gale, C. Shih, and N. Chang. A Stochastic Finite-state Word Segmentation Algorithm for Chinese. *Computational Linguistics*, 22(3):377–404, 1996.

[26] C. Zhai. Fast Statistical Parsing of Noun Phrases for Document Indexing. In *Proceedings of the fifth Conference on Applied Natural Language Processing (ANLP)*, pages 312 – 319, 1997.