Query Segmentation Revisited

Matthias Hagen Martin Potthast Benno Stein Christof Bräutigam

Bauhaus-Universität Weimar
99421 Weimar, Germany
<first name>.<last name>@uni-weimar.de

ABSTRACT

We address the problem of query segmentation: given a keyword query, the task is to group the keywords into phrases, if possible. Previous approaches to the problem achieve reasonable segmentation performance but are tested only against a small corpus of manually segmented queries. In addition, many of the previous approaches are fairly intricate as they use expensive features and are difficult to be reimplemented.

The main contribution of this paper is a new method for query segmentation that is easy to implement, fast, and that comes with a segmentation accuracy comparable to current state-of-the-art techniques. Our method uses only raw web n-gram frequencies and Wikipedia titles that are stored in a hash table. At the same time, we introduce a new evaluation corpus for query segmentation. With about 50 000 human-annotated queries, it is two orders of magnitude larger than the corpus being used up to now.

Categories and Subject Descriptors: H.3.3 [Information Search and Retrieval]: Query Formulation

General Terms: Algorithms, Experimentation

Keywords: Query Segmentation, Web N-Grams, Corpus

1. INTRODUCTION

The interfaces of today's web search engines are mainly keyword-based: users submit queries by typing several keywords into a search box. It is not uncommon that queries comprise phrases and compound concepts; take `times square` as an example. A search engine that is informed about such phrases and concepts by means of proper quotation may consider them as indivisible units and use them to improve retrieval precision (e.g., by excluding documents that do not contain words in the exact same order of the phrases). Other server-side algorithms that benefit from quotation include query reformulation, which could be done on the level of phrases instead of keywords, and query disambiguation, which has to cope with tricky queries like `times square dance`. Without quotes, it is difficult to know whether the user intends to find newspaper articles on `square dance` in the London `times` or rather `dance` events at `times square`, New York (locating the user might also help, of course). Skilled web searchers surround phrases with quotes, but experience shows that most searchers are not even aware of this option. Hence, search engines should apply pre-retrieval algorithms that automatically divide queries into segments in order to second-guess the user's intended phrases and to improve the overall user satisfaction.

Our algorithmic contribution in this respect is a new and robust approach to the task of query segmentation; it relies only on web n-gram frequencies and Wikipedia titles. We achieve a segmentation accuracy that is competitive with state-of-the-art algorithms on a widely used evaluation corpus comprising 500 queries. As part of our evaluation, we compare our algorithm with 7 others proposed in the literature. Our second contribution relates to evaluation and verifiability; it is a new query segmentation corpus comprising 50 000 queries. Our corpus subsumes the current standard corpus and, unlike that, meets the requirements of representative large-scale evaluations, which was one of the main points raised at the SIGIR 2010 workshop on "Query Representation and Understanding" [8].

The new query segmentation approach is inspired by our recently proposed naïve query segmentation technique, which involves an exponential weighting function to normalize web n-gram frequencies [10]. However, until now, no convincing explanation could be given as to why naïve query segmentation performs so well. We will close this gap by giving an empirical justification for the naïve query segmentation approach.

The paper is organized as follows. Section 2 reviews existing approaches to query segmentation. Section 3 presents the basic notation of our query segmentation framework. In Section 4, we revisit the naïve query segmentation method and provide justification for its remarkable performance. Based on these findings, we develop our new method in Section 5. An empirical evaluation in Section 6 shows the segmentation accuracy of our method on the current gold standard, compared to existing approaches. Furthermore, this section introduces our new corpus, outlines its construction with the aid of crowdsourcing, and compares the segmentations obtained this way to expert segmentations. Section 7 provides an outlook on future work.

2. RELATED WORK

Recent research suggests a variety of approaches to query segmentation. For instance, Guo et al. [9], Yu and Shi [20], and Kiseleva et al. [13] use methods based on conditional random fields (CRF). Guo et al. evaluate their method on a proprietary query corpus and tackle the broader problem of query refinement that simultaneously involves spelling correction, stemming, etc. Hence, their approach is not entirely comparable to ours, since we assume that spelling correction is done prior to query segmentation—an assumption shared by most other query segmentation studies. The other CRF-based methods by Yu and Shi and Kiseleva et al. also address query segmentation in settings different from ours. Yu and Shi focus on query segmentation in the context of text stored in relational databases and use database-specific features not available in web search. Kiseleva et al. focus on product queries and aim to improve the user experience in web shops.

One of the earliest approaches to *web* query segmentation is by Risvik et al. [18]. They segment queries by computing so-called connexity scores that measure mutual information within a segment and the segment's frequency in a query log. Jones et al. [12] also use a mutual information-based scoring that finds segments in which adjacent terms have high mutual information. However, neither Risvik et al. nor Jones et al. evaluate the segmentation accuracy of their approaches. In a very recent paper, Huang et al. [11] also use segment-based pointwise mutual information scores obtained from web-scale language models. For a given query, they derive a tree of concepts. The tree is then used to obtain a final query segmentation. However, Huang et al. evaluate their method only on a proprietary query corpus without comparing it to other approaches. Note that in many query segmentation studies, mutual information-based segmentation is used as a baseline, often performing worse than the more involved methods.

One of the earliest methods that does not rely only on mutual information is the supervised learning approach by Bergsma and Wang [4]. Bergsma and Wang incorporate many features: statistical ones like phrase frequencies on the web and in query logs, as well as dependency features that focus on noun phrases. They also established the first gold standard corpus of 500 queries, each segmented by three human annotators. Subsequent work [7, 10, 19, 21] has adopted this gold standard; as do we in our evaluations in order to ensure comparability. However, Bergsma and Wang's evaluation corpus is rather small, so that we decided to introduce a larger and more representative corpus in this paper. As for the query segmentation, Bergsma and Wang's supervised learning method is trained on queries segmented by a single annotator who also segmented the gold standard. This leaves some doubts with regard to the generalizability of the results. Nevertheless, Bendersky et al. [2] successfully use a version of Bergsma and Wang's method as a sub-procedure in their two-stage query segmentation approach.

Instead of the supervised approach that requires training data, Tan and Peng [19] and Zhang et al. [21] suggest unsupervised methods. Zhang et al. compute segment scores from the eigenvalues of a correlation matrix corresponding to a given query. Tan and Peng's method, like ours, uses only n-gram frequencies from a large web corpus as well as Wikipedia. However, Tan and Peng state that raw n-gram frequencies by themselves cannot be used for query segmentation. Hence, they build a language model from the n-gram frequencies via expectation maximization. In a second step, Tan and Peng boost a segment's score derived from the language model if it is used prominently in Wikipedia. Our new method uses the same features as Tan and Peng's but with superior segmentation accuracy (cf. Section 6). Moreover, in contrast to Tan and Peng's assumption, our naïve query segmentation method [10] shows how raw n-gram frequencies can be exploited for query segmentation using an appropriate normalization scheme (cf. Section 4).

The recent snippet-based method by Brenes et al. [7] is quite simple compared to the aforementioned approaches: it segments a web query based on search result snippets for the unquoted query. Brenes et al. evaluate different techniques of deriving a query's segmentation from snippets. Obviously, the main concern with this approach is runtime. Most queries have to pass the retrieval pipeline twice before any results are returned: once unquoted to obtain the snippets used for segmentation, and once more with quoted segments.

Bendersky et al. [3] also suggest a method that involves time consuming double-retrieval for most queries. Their method introduces a segment break between two words whenever a likelihood ratio is below some threshold. The likelihood ratios are obtained by combining web n-gram probabilities and pseudo-relevance feedback (PRF) from the top-ranked documents for the original, unquoted query. The PRF-based method achieves a promising experimental segmentation accuracy. However, Bendersky et al.'s evaluation corpus is rather small (250 queries) and it has been segmented by only one annotator, which suggests a bias in the segmentations. Furthermore, the PRF-based segmentation method is not compared with state-of-the-art approaches.

Our previous naïve query segmentation method [10] scores all segmentations for a given query by the weighted sum of the frequencies of contained n-grams, obtained from a large web corpus. Besides raw n-gram frequencies, no other features are involved, making this approach easy to explain and straightforward to implement. The weighting scheme of naïve query segmentation aims at "normalizing" the n-gram frequencies, so that a longer segment like "`toronto blue jays`" has a chance to achieve a higher score than shorter sub-segments like "`blue jays`". With respect to segmentation accuracy, the naïve approach performs comparable to the other approaches that use, supposedly, more sophisticated features. Furthermore, storing the n-gram frequencies in a large hash table ensures very competitive runtime on machines with sufficient main memory. However, until now, no explanation was given why the exponential normalization scheme of naïve query segmentation performs so well. We close this gap with an in-depth analysis in Section 4. Our new segmentation method, which is inspired by naïve query segmentation, is introduced in Section 5.

Finally, the very recent approach of Mishra et al. [14] compares with our method in terms of feature complexity. But instead of web n-gram frequencies, Mishra et al. exploit n-gram frequencies from a large query log. Their experiments on a proprietary query corpus indicate an improvement over a mutual information baseline.

3. BASIC NOTATION AND FRAMEWORK

We regard a query q as a sequence $(w_1, w_2, \ldots, w_k)$ of k keywords. A valid segmentation S for q is a sequence of disjunct segments s, each a contiguous subsequence of q, whose concatenation equals q. There are 2^{k-1} valid segmentations for q, and $(k^2 - k)/2$ potential segments that contain at least two keywords from q.

The basic and major assumption of both our approaches is that phrases contained in queries must exist on the web—otherwise they cannot increase web retrieval performance. The idea then is to use the web as a corpus of potential query phrases. The largest obtainable collection of web phrases, besides the web itself, is the Google n-gram corpus [5]. It contains n-grams of length 1 to 5 from the 2006 Google index along with their occurrence frequencies. For n-grams up to $n = 5$, the frequencies can be directly retrieved from the corpus; for longer n-grams up to $n = 9$, estimations can be made analogously to the set-based method described in [19]. For example, the frequency of a 6-gram $ABCDEF$ can be lower-bounded by the sum of the Google n-gram frequencies of the 5-grams $ABCDE$ and $BCDEF$ decreased by the frequency of their 4-gram intersection $BCDE$. In practice, however, despite being set-theoretically sound, these lower bound estimations often evaluate to 0 (i.e., the intersection's frequency is too large to be compensated by the frequencies of the two longer n-grams). That said, there are still situations where non-zero lower bound estimations appear and do some good.

Using the web occurrence frequencies from the Google n-gram corpus and the set-based estimations, both our approaches score and rank all possible segmentations of a query. They apply normalization schemes to the raw n-gram frequencies and allow long segments to achieve scores comparable to their shorter sub-segments.

For example, `blue jays` will always have a larger raw n-gram frequency than `toronto blue jays`, but the latter should be the preferred segmentation in queries concerning the baseball team. In the following section, we detail our previously proposed naïve normalization scheme and provide some additional insights, as it inspired our new technique explained in the section thereafter.

4. NAIVE NORMALIZATION

Naïve query segmentation derives a score for each valid segmentation S of a query q as follows. The n-gram frequencies $\mathit{freq}(s)$ of all potential segments s with at least two words are retrieved. Having the frequencies at hand, all valid segmentations are enumerated systematically, and each segmentation S is scored according to the following function:

$$\mathit{score}(S) = \begin{cases} \displaystyle\sum_{s \in S, |s| \geq 2} |s|^{|s|} \cdot \mathit{freq}(s) & \text{if } \mathit{freq}(s) > 0 \text{ for} \\ & \text{all } s \in S, |s| \geq 2 \\ -1 & \text{else.} \end{cases}$$

The weight factor $|s|^{|s|}$ is a means of normalizing $\mathit{freq}(s)$ to compensate the power law distribution of web n-gram frequencies. This way, a long segment s has a chance of being selected compared to its shorter sub-segments. For a query q, the valid segmentations are ranked according to their scores and naïve query segmentation selects the one that maximizes $\mathit{score}(S)$.

For example, `toronto blue jays` has an n-gram frequency of 0.8 million, which is smaller than the 1.4 million of `blue jays`; simply using the length $|s|$ of s as weight factor does not help to prefer the three-word segment (*score* of 2.4 million vs. 2.8 million). However, using the exponential weight factor $|s|^{|s|}$, the segmentation "`toronto`" "`blue jays`" with a *score* of 5.2 million is discarded in favor of "`toronto blue jays`" with a *score* of 21.6 million.

In the above scoring function, the n-gram frequencies of single word segments are implicitly set to 0, so that the "null"-segmentation that consists of only single word segments—the unquoted query—gets a *score* of 0 (the else-case does not match and the summation is empty). A segmentation gets a negative score of -1 if it contains a segment (with at least two words) that does not exist in the web n-gram corpus. Such a segmentation cannot help to improve retrieval performance since the non-existent segment is not found on the web at all, that is, in our n-gram representation of the web. Scoring such segmentations with a negative score ensures that the unquoted query will be chosen as a fallback solution in case all other valid segmentations contain non-existent phrases.

Compared to other methods, naïve query segmentation is the most basic approach as it uses only raw n-gram frequencies instead of many different features. Also, the scoring can be explained (and implemented) within a few lines (of code). What remains to be illustrated is why the exponential scoring performs well in practice, and not only for the `toronto blue jays`.

Empirical Justification

In what follows, we provide empirical evidence that the exponential scoring scheme of naïve query segmentation reproduces human preferences of segmenting queries. This sheds light on why the method achieves its convincing segmentation accuracy reported in [10] and in Section 6. To this end, we take a closer look at the underlying data used in the naïve scoring scheme and its evaluation: the Google n-gram frequencies and the 500 human-annotated queries from the Bergsma-Wang-Corpus.

The Google n-grams contain occurrence frequencies for all 1- to 5-grams that appeared more than 40 times within Google's web index as of 2006. Some overview figures are given in Table 1, including the average and the median frequencies for all 2- to 5-grams in the original corpus (rows "all") as well as for a cleaned version of the corpus (rows "cleaned") from which all n-grams were removed that contain non-alphanumeric characters. We assume that n-grams occurring in real web queries are better represented by the cleaned corpus. Note that the resulting average and median frequencies are quite low and thus illustrate the presumed long tail distribution of n-gram frequencies (most n-grams have very low frequencies).

Table 1: Overview of the Google n-gram corpus.

Corpus Subset		Entries	Frequency	
			Average	Median
2-grams	all	314 843 401	2 893	110
	clean	215 114 473	2 065	108
3-grams	all	977 069 902	756	91
	clean	492 457 391	608	89
4-grams	all	1 313 818 354	387	80
	clean	556 957 524	330	79
5-grams	all	1 176 470 663	300	75
	clean	421 372 587	258	73

To experimentally justify the naïve scoring scheme, we address the situation of a query in which a longer n-gram should be favored over its shorter prefixes in a segmentation. The longer n-gram then has to obtain at least a better score than any of its shorter prefixes. To simulate this scenario, we sampled 10 million 5-grams from the cleaned 5-gram corpus. The sampling probability is chosen proportional to a 5-gram's frequency, so that frequent 5-grams are favored. We speculate that favoring frequent 5-grams for our sample makes it more representative of the most frequent segments observed in web queries. From all of the sampled 5-grams, we computed all prefixes of length 2 to 4 words.

Table 2 gives an overview of the 5-gram sample and their prefixes. Observe that the median frequencies of the sample show an exponential decrease. In order to further compare this decrease to the $|s|^{|s|}$-scoring, we also provide the ratio of the median 2-gram frequency to the median s-gram frequency. Note that the last two columns of Table 2 show that the $|s|^{|s|}$-compensation for the exponential frequency decrease is always in the correct order of magnitude. In this connection, one may wonder why not to choose the observed ratios instead of $|s|^{|s|}$-scoring (i.e., multiplying $\mathit{freq}(s)$ of a 3-gram s with 44 instead of 3^3, etc.). To test this possibility, we performed a pilot study where the ratios from Table 2 were used as "hard-wired" weight factors instead of $|s|^{|s|}$: the achieved segmentation accuracy dropped significantly.

An explanation for this behavior can be found when comparing the expected median *score*-values of our sample with the query corpus used in our evaluation. As can be seen in Table 3, the hard-wired scoring achieves a very good normalization of the expected *score*-values for the 5-gram sample in the sense that all median

Table 2: Overview of the sample of 10 million 5-grams.

$\|s\|$-grams	Unique Entries	Median Freq.	$\frac{\text{2-gram Freq.}}{\|s\|\text{-gram Freq.}}$	$\|s\|^{\|s\|}$
2-grams	2 409 063	3 461 030	1	4
3-grams	5 431 544	78 733	44	27
4-grams	8 073 863	7 356	470	256
5-grams	10 000 000	1 129	3 065	3 125

Table 3: Expected $score$ in the 5-gram sampling experiment.

$\|s\|$-grams	Median Freq.	$\|s\|^{\|s\|}$-scoring	"Hard-Wired"
2-grams	3 461 030	13 844 120	3 461 030
3-grams	78 733	2 125 791	3 464 252
4-grams	7 356	1 883 136	3 457 320
5-grams	1 129	3 528 125	3 460 385

$score$-values have the same order of magnitude. Then again, $|s|^{|s|}$-scoring clearly favors 2-grams, whereas the longer n-grams are somehow "on par." To compare this with human segments, Table 4 compiles the segment length distribution of the 3 human annotators of the widely used Bergsma-Wang-Corpus. The last row of Table 4 contains only queries that all three annotators segmented in the same way. Observe that human annotators also favor 2-grams instead of longer segments, especially in queries they all agree upon. The $|s|^{|s|}$-scoring of naïve query segmentation reproduces this behavior on our 5-gram sample, which explains, to some extent, why it performs so well.

5. WIKIPEDIA-BASED NORMALIZATION

In pilot experiments, the aforementioned approach that normalizes each segment's frequency with hard-wired average frequencies shows an inferior segmentation accuracy compared to the naïve approach that involves a similar, but less even normalization scheme. Nevertheless, normalizing segment frequencies with average frequencies also bears the idea of normalizing in a segment-specific way, which is exactly what we propose for our Wikipedia-based normalization scheme. As the name suggests, the new scheme involves another feature besides the raw n-gram frequencies, namely a Wikipedia title dictionary obtained from a dump of the English Wikipedia. Note that Tan and Peng [19] also use this feature in their segmentation approach. Our dictionary contains all Wikipedia titles and their respective disambiguations, but no words from within articles. Otherwise, the new normalization scheme is just as simple as naïve query segmentation.

Again, the n-gram frequencies $freq(s)$ of all potential segments s with at least two words are retrieved. For each segment s, we check whether it is present in the Wikipedia title dictionary. If a segment s appears in the dictionary, we replace its frequency $freq(s)$ with the maximal Google n-gram frequency found among the sub-2-grams $s' \sqsubseteq s$, given by the following $weight$:

$$weight(s) = \begin{cases} |s| + \max\limits_{\substack{s' \sqsubseteq s \\ |s'|=2}} freq(s') & \text{if } s \text{ is a title in Wikipedia} \\ freq(s) & \text{else.} \end{cases}$$

This way, the normalization of a segment's frequency is segment-specific in that every potential segment's $freq$-value is treated separately rather than normalizing it with some average frequency. Note that $|s|$ is added to the maximal sub-2-gram frequency for convenience reasons, as it allows us to prove that queries that consist of a single Wikipedia title will not be split into sub-segments (shown in the next subsection). Otherwise, adding $|s|$ had no measurable effect in our experiments so that a query segmentation system "in the field" could safely omit it. Based on the Wikipedia-normalized $weight$-values, a valid segmentation S of q is scored as follows:

$$score(S) = \begin{cases} \sum\limits_{s \in S, |s| \geq 2} |s| \cdot weight(s) & \text{if } weight(s) > 0 \text{ for all } s \in S, |s| \geq 2 \\ -1 & \text{else.} \end{cases}$$

Table 4: Segment length distribution of human segmentations.

Annotator	Segment Length					
	1	2	3	4	5	6
A	451	699	74	14	2	
B	351	541	113	77	9	2
C	426	588	100	51	5	1
Agree	151	318	31	9		

Again, for a query q we choose from all valid segmentations the one that maximizes $score(S)$.

Remarks. The Wikipedia-based normalization can run into the special case of encountering a "new" concept s that is present in the Wikipedia titles but not in the n-gram corpus (e.g., some new product or brand that did not exist back in 2006). If all the sub-2-grams of s exist in the n-gram corpus, this is not an issue since the Wikipedia-based normalization still works. Otherwise, however, $weight(s)$ would be set to $|s|$, although s is prominently placed in Wikipedia, which is not satisfactory. We tackle this problem by a simple additional rule for the more general case that only some sub-2-grams of s are not yet present in the n-gram corpus. In that case, we set the missing 2-gram frequencies to the median 2-gram frequency 3 461 030 from Table 2. As before, $weight(s)$ is set to the maximal frequency found among the sub-2-grams of s (which now also could be the median frequency).

Another straightforward method of circumventing the situation of Wikipedia titles being out of sync with the n-gram corpus exists at search engine site: to update the n-gram corpus in real-time. However, we expect that n-gram updates will still be done less frequently compared to updating the Wikipedia title dictionary. And since Wikipedia typically contains pages on new, important "concepts" very quickly, setting the corresponding frequencies to the median 2-gram frequency is a reasonable heuristic. As a side note, in our experiments we did not find any "new" Wikipedia concepts since the Bergsma-Wang-Corpus and our newly developed corpus both stem from the AOL query log of 2006 and thus fit the 2006 Google n-gram corpus very well.

Theoretical Justification

The idea of Wikipedia-based normalization is similar to that of the naïve normalization approach, namely giving longer segments a chance to be preferred over their shorter sub-segments. But now this goal is achieved by using Wikipedia for segment re-weighting instead of some "magic," yet powerful exponential factor. Wikipedia serves as a database of named entities that helps to identify widely known concepts like names, places, etc. These concepts should not be split up when segmenting a query. In case a concept forms an entire query, it can be proven that the above scoring function will prefer the complete concept as one segment.

LEMMA 1. *Let s be a sequence of words found in the Wikipedia title dictionary. Whenever a query $q = s$ shall be segmented, the Wikipedia-based scoring function will choose the segmentation "s" without any intermediate quotes.*

PROOF. Let ℓ be the largest $freq$-value of any sub-2-gram of s. As s is found in the Wikipedia title dictionary, the segmentation "s" gets $score(\text{"}s\text{"}) = |s| \cdot (\ell + |s|)$.

Now consider any other valid segmentation S of s that splits s into sub-segments and gets a positive score by the Wikipedia-based normalization scoring function. The worst possible case is that all sub-segments of s also are Wikipedia concepts and that all sub-2-grams of s have ℓ as their *freq*-value. Note that this worst case scenario is independent of whether all sub-2-grams of s are in the n-gram corpus or not: it does not matter whether ℓ is the median 2-gram frequency from Table 2 or not.

As S divides s into non-overlapping sub-segments (otherwise S is not valid), the sum of the length-weight-factors of the sub-segments cannot be larger than $|s|$. Furthermore, the Wikipedia-normalized $weight$-value of each of these sub-segments cannot be larger than $\ell + |s| - 1$ as the largest sub-segments of s have size $|s| - 1$. Basic calculus yields $score(S) \leq |s| \cdot (\ell + |s| - 1) < score(\text{"}s\text{"})$ so that the segmentation "s" will be chosen. □

As an example, consider the query `new york yankees`—to stay within the baseball domain. The relevant Google n-gram frequencies for this query are 165.4 million for `new york` and 1.8 million for `new york yankees`. Note that naïve query segmentation would segment the concept as `"new york" "yankees"` since this achieves a naïve scoring of 661.6 million compared to 28.8 million for `"new york yankees"`. However, with Wikipedia-based normalization, `new york yankees` gets as *weight* the 165.4 million frequency of `new york`. Hence, `"new york yankees"` achieves a 496.2 million *score* compared to 330.8 million for `"new york" "yankees"`.

The only way that a Wikipedia title s is split up during segmentation is when a word in front of or after s co-occurs very often with a prefix or a suffix of s, respectively. This is especially the case when two Wikipedia titles are interleaved within a query. However, s is only split up if the $score$-value of some segmentation containing a sub-segmented s is superior to all the segmentations containing s as a complete segment. This is in line with the rationale that larger $score$-values represent better segmentations.

An example where a Wikipedia concept is split, is the query `times square dance` from the introduction. The relevant segments that appear in the Wikipedia title dictionary are `times square` and `square dance`. Based on the Google n-gram corpus, `times square` gets a 1.3 million *weight* and `square dance` achieves 0.2 million. Our new segmentation scoring function then assigns `"times square" dance` the highest *score* of 2.6 million while `times "square dance"` achieves 0.4 million—either way, a Wikipedia concept is split.

It is of course debatable whether the segmentation `"times square" dance` is a good choice, since we have pointed out the ambiguity of this query. This situation can only be resolved by incorporating information about the user's context. For instance, if the user stems from London, reads "The Times" and is a passionate folk-dancer, this might make the alternative segmentation `times "square dance"` preferable. If no such context information is at hand, there is still another option: the search engine may present the results of the best scoring segmentation to the user and offer the second best segmentation in a "Did you mean" manner.

Besides the theoretical justification for Wikipedia-based normalization, our new method also shows very promising experimental performance with respect to segmentation accuracy against human-segmented queries, as the next section shows.

6. EVALUATION

In this section we report on an evaluation that compares our approach to 7 others proposed in the literature. We employ the performance measures proposed in [4, 19], use a mutual information-based method as baseline, and evaluate against three corpora: the Bergsma-Wang-Corpus 2007 [4], an enriched version of that corpus, and a new, significantly larger corpus, called Webis Query Segmentation Corpus 2010. The latter two have been compiled with the aid of Amazon's Mechanical Turk. In this connection, we also compare segmentations from experts with those of laymen.

6.1 Performance Measures

Performance evaluation of query segmentation algorithms is twofold. On the one hand, *runtime* is crucial since query segmentation must be done on-the-fly during retrieval. Runtime is typically given as throughput (i.e., the number of queries segmentable per second). On the other hand, the computed segmentations should be as accurate as possible. There are three levels on which to measure segmentation accuracy in a supervised manner:

Query Level. As a whole, a computed segmentation of a query is correct iff it contains exactly the same segments as a human reference segmentation. Hence, the *query accuracy* is the ratio of correctly segmented queries for a given corpus.

Segment Level. Let S denote the set of segments of a human segmentation of a query q. A computed segmentation S' can be evaluated using the well-known measures precision and recall. The *segment precision* and the *segment recall* are defined as follows:

$$\text{seg prec} = \frac{|S \cap S'|}{|S'|} \qquad \text{and} \qquad \text{seg rec} = \frac{|S \cap S'|}{|S|}.$$

Both values can be combined into a single score by computing their harmonic mean, called *segment F-Measure* :

$$\text{seg F} = \frac{2 \cdot \text{seg prec} \cdot \text{seg rec}}{\text{seg prec} + \text{seg rec}}.$$

Break Level. Note that query segmentation can also be considered a classification task in which, between each pair of consecutive words in a query, a decision has to be made whether or not to insert a segment break. This allows for $k - 1$ potential break positions in a query with k keywords. For every break position, the computed segmentation may either decide correctly or not, according to a reference segmentation. The *break accuracy* measures the ratio of correct decisions.

As an illustration, consider the query `san jose yellow pages` with the reference segmentation `"san jose" "yellow pages"`. A computed segmentation `"san jose" yellow pages` is not correct on the query level, resulting in a query accuracy of 0. However, on the segment-level, `"san jose" yellow pages` at least contains one of the two reference segments, yielding a segment recall of 0.5. But since the other two single word segments are not part of the reference segmentation, precision is 0.333, yielding a segment F-Measure of 0.4. The break accuracy is 0.666, since `"san jose" yellow pages` decides incorrectly only for one of the three break positions.

6.2 Baseline: Mutual Information

As a baseline for query segmentation we adopt the mutual information method (MI) used throughout the literature. A segmentation S for a query q is obtained by first computing the pointwise mutual information score for each pair of consecutive words (w_i, w_{i+1}) in q, with $i \in \{1, \ldots, k-1\}$ and $k = |q|$:

$$\text{PMI}(w_i, w_{i+1}) = \log \frac{p(w_i, w_{i+1})}{p(w_i) \cdot p(w_{i+1})},$$

where $p(w_i, w_{i+1})$ is the joint probability of occurrence of the 2-gram (w_i, w_{i+1}), and $p(w_i)$ and $p(w_{i+1})$ are the individual occurrence probabilities of the two words w_i and w_{i+1} in a large text corpus. Second, segment breaks are introduced into q whenever the pointwise-mutual information score of two consecutive words is below a pre-specified threshold τ (i.e., when $\mathrm{PMI}(w_i, w_{i+1}) < \tau$).

In our evaluation, the probabilities for all words and 2-grams have been computed using the Microsoft Web N-Gram Services [11].[1] More specifically, the language model of web page bodies from April 2010 has been used. We recorded all probabilities for all our corpora in order to ensure replicability. For our experiments, we chose $\tau = 0.894775$, which maximizes the MI method's break accuracy on the Bergsma-Wang-Corpus.

6.3 The Bergsma-Wang-Corpus 2007

The Bergsma-Wang-Corpus 2007 (BWC07) consists of 500 queries which have been sampled from the AOL query log dataset [15]. The sample was chosen at random from the subset of queries that satisfy all of the following constraints:

- A query consists of only determiners, adjectives, and nouns.
- A query is of length 4 words or greater.
- A query has been successful (i.e., the searcher clicked on one of the search result URLs returned by the search engine).

The query sample was then segmented independently by 3 annotators, who were instructed to first guess the user intent based on the query in question and the URL the user clicked on, and then segment the query so that it describes the user intent more clearly. In 44% of the queries, all three annotators agree on the segmentation, and in about 60% of the cases, at least two annotators agree.

Remarks. The BWC07 has a number of shortcomings that render evaluations based on this corpus less insightful: the query sample of the corpus is not representative, partly because of the small number of queries and partly because of the sampling constraints. The first constraint particularly raises concerns since its motivation was to accommodate design limitations of the proposed query segmentation algorithm; the authors stated that "*as our approach was designed particularly for noun phrase queries, we selected for our final experiments those AOL queries containing only determiners, adjectives, and nouns*" [4]. The other two constraints are less of a problem, though one might argue that queries of length 3 should also be subject to segmentation, and that, unless query quality predictors are employed, a search engine cannot know in advance whether a query will be successful. Moreover, the number of annotators per query appears to be too small, since in 40% of the queries no agreement was achieved. This, in turn, tells something about the difficulties involved in manually segmenting queries. On a minor point, there are also a few duplicate queries, spelling errors, and character encoding errors. The former have a measurable effect on segmentation performance, given the small size of the corpus, while the latter two should not be part of a query segmentation corpus. Query segmentation does not necessarily involve spell checking and encoding normalization, as those may be treated as separate problems. However, none of the above should be held to the disadvantage of the corpus' authors, since their intentions were first and foremost to evaluate their approach on a new retrieval problem, and not to construct a reference corpus, which it became nonetheless.

Experiments. Several previous studies evaluate their query segmentation algorithms against the BWC07 [4, 7, 10, 19, 21]. We follow suit, to ensure comparability to the existing evaluations. However, prior to that we chose to correct the aforementioned minor errors and remove duplicate queries; the cleaned corpus consists of 496 queries. Since the existing studies do not mention any problems with the BWC07, we have asked the respective authors for their segmentations of the BWC07 queries. Our rationale for this is to avoid an error-prone reimplementation of the existing algorithms, since their output for the BWC07 suffices to recalculate the accuracy measures. The authors of [4, 7, 10, 21] kindly provided their segmentations. Based on this data, we have been able to verify the performances reported in the respective papers, and to reevaluate the algorithms on the cleaned version of the BWC07.

Table 5 shows the results of our evaluation. Each column presents the segmentation accuracies of one algorithm. The rightmost two columns show the results of our naïve query segmentation [10] and those of Wikipedia-based normalization (entitled "Our"). The table should be read as follows: three annotators—A, B, and C—independently segmented the 496 queries of the BWC07 and they agreed on 218 of them, denoted in the rows "Agree." The rows "Best of A, B, C" evaluate simultaneously against the up to three reference segmentations of A, B, and C, choosing the one that maximizes a computed segmentation's break accuracy. Note that, compared to the originally published results, the segmentation accuracies of most systems slightly decrease in our evaluation. This is due to the removed duplicate queries that are rather "easy" and correctly segmented by most systems. An exception is the remarkable performance of our mutual information baseline with a significant improvement over previously reported values. One reason for this is that the language model underlying the Microsoft Web N-Gram Services presumably gives more accurate probabilities than those used in previous evaluations. However, more importantly, note that the baseline's threshold τ was derived on the BWC07 queries with the explicit aim of maximizing the resulting break accuracy. Hence, it represents the best case MI approach for the BWC07 queries and yields a challenging baseline.

Since we did not get the query segmentations of Tan and Peng [19], we include the values they published in their paper (in gray). However, the removal of duplicate queries would most likely decrease their performances as well. Since some previous studies did not report all measures for their algorithms, Table 5 contrasts for the first time all performance values for all algorithms that have been evaluated against the BWC07.

The results show that the approach of Bergsma and Wang [4] performs very well on annotator A as well as on the queries all annotators agree upon. However, this is not too surprising as their approach is based on a supervised learning algorithm that was explicitly trained on queries segmented by annotator A (the agreement queries also match A's segmentation). This leaves some doubts with regard to generalizability, underpinned by the inferior performance of their approach on the two other annotators B and C. As for the other systems, note that our naïve query segmentation algorithm with n-gram frequency normalization achieves a performance comparable to the best approaches of Brenes et al. [7], Tan and Peng [19], and Zhang et al. [21]. Furthermore, note that our new Wikipedia-based frequency normalization method (column "Our") outperforms all other methods with respect to query accuracy and segment precision, while being on par with the best performing system at break accuracy.

An Excursus on Retrieval Performance. Besides measuring segmentation accuracy against a gold standard, one might as well evaluate whether a segmentation actually yields an improvement in retrieval performance (i.e., retrieving results that are more relevant). To this end, we have conducted the following small-scale experiment in order to compare the two best performing approaches: Bergsma and Wang's supervised learning approach and our Wikipedia-based normalization scheme. For each of the

[1] `http://web-ngram.research.microsoft.com`

Table 5: Segmentation performance on the Bergsma-Wang-Corpus.

Annotator	Performance Measure	MI	[4]	[19]	[21]	[7]	[14]	[10]	Our
A	query	0.407	**0.609**	0.526	0.518	0.540	0.256	0.597	0.573
	seg prec	0.553	**0.748**	0.657	0.673	0.686	0.476	0.724	0.706
	seg rec	0.550	**0.761**	0.657	0.650	0.672	0.566	0.711	0.679
	seg F	0.552	**0.754**	0.657	0.662	0.679	0.517	0.717	0.692
	break	0.761	**0.859**	0.810	0.810	0.797	0.681	0.826	0.826
B	query	0.413	0.435	0.494	0.504	0.383	0.185	0.438	**0.508**
	seg prec	0.539	0.582	0.623	**0.637**	0.527	0.369	0.577	0.635
	seg rec	0.548	0.608	**0.640**	0.632	0.533	0.450	0.583	0.627
	seg F	0.544	0.595	0.631	0.634	0.530	0.405	0.580	**0.631**
	break	0.765	0.783	0.802	**0.811**	0.741	0.598	0.777	0.803
C	query	0.417	0.472	0.494	0.484	0.440	0.224	0.480	**0.508**
	seg prec	0.553	0.623	0.634	0.627	0.580	0.424	0.618	**0.647**
	seg rec	0.557	**0.643**	0.642	0.613	0.575	0.515	0.613	0.632
	seg F	0.555	0.633	0.638	0.620	0.578	0.465	0.615	**0.640**
	break	0.764	0.795	**0.796**	0.789	0.755	0.639	0.777	0.795
Agree	query	0.555	0.688	0.671	0.670	0.615	0.294	0.693	**0.720**
	seg prec	0.659	0.792	0.767	0.787	0.731	0.491	0.789	**0.810**
	seg rec	0.665	**0.809**	0.782	0.770	0.724	0.575	0.782	0.792
	seg F	0.662	0.800	0.774	0.779	0.727	0.529	0.785	**0.801**
	break	0.828	**0.889**	0.871	0.883	0.834	0.699	0.868	0.885
Best of A, B, C	query	0.583	0.702	0.692	0.694	0.629	0.333	0.700	**0.726**
	seg prec	0.693	0.812	0.797	0.811	0.749	0.558	0.800	**0.820**
	seg rec	0.697	**0.831**	0.807	0.801	0.746	0.649	0.796	0.807
	seg F	0.695	**0.821**	0.801	0.806	0.747	0.600	0.798	0.814
	break	0.849	0.899	0.891	0.897	0.857	0.736	0.889	**0.900**

218 BWC07 queries on which all three annotators agree, we submit 4 queries to the Bing web search engine, each time storing the top-50 results: the BWC07 segmentation, the computed segmentation of Bergsma and Wang, the computed segmentation of our method, and the unquoted query. We consider the top-50 results obtained by the BWC07 segmentation as "relevant," which allows us to measure the recall of the other three queries. Averaged over all 218 trials, the supervised learning approach achieves a recall of 0.844, our Wikipedia-based normalization achieves 0.836, whereas unquoted queries achieve a recall of 0.553.

Presuming the user intent is well captured in the segmented queries of the BWC07 (three annotators agreed upon these segmentations), the results might be interpreted as indication that segmenting queries improves recall over unquoted queries. Comparing the average recall of Bergsma and Wang's segmentations and ours also suggests that their worse query accuracy is compensated by their better segment recall so that the overall retrieval recall is comparable. It is important to note that this small experiment is not meant to replace a large-scale TREC-style evaluation, since the results have not been judged manually as relevant or not. Instead, this experiment is meant as a first step toward not just comparing segmentation accuracy. After all, the ultimate goal of query segmentation is to improve retrieval performance.

6.4 The Enriched Bergsma-Wang-Corpus

One point of criticism about the BWC07 is the non-agreement of its annotators in 40% of the queries. We attribute this problem to the fact that not all queries are the same, and that some are more ambiguous than others. A search engine user who poses an ambiguous query would know of course, if being asked, how exactly to segment it, whereas an annotator has a hard time figuring this out afterwards. One way to overcome this problem is to collect more opinions from different annotators and then make a majority decision. Recently, paid crowdsourcing has become an important tool in this respect, enabling researchers to significantly scale up corpus construction.[2]

Amazon's Mechanical Turk (AMT) is a well-known platform for paid crowdsourcing. In short, it acts as a broker between workers and so-called requesters, who offer tasks and payment for their successful completion. Since real money is involved and since workers remain anonymous, the platform attracts scammers who try to get paid without actually working. Hence, requesters get the opportunity to check submitted results and reject those that are unsatisfactory. Besides saving money, rigorous result checking is of course a necessity to ensure quality.

In an effort to enrich the BWC07, we have offered a query segmentation task on AMT in which we asked to segment the queries of this corpus. At least 10 valid segmentations per query were collected, while manually checking and rejecting the invalid ones. To measure the success of our initiative and to learn about how much laymen agree with experts, we compute all of the aforementioned performance measures, treating the segmentations obtained from AMT as if they were segmentations returned by an algorithm. For each query, the segmentation that was submitted most often by the workers was used. The results of this comparison are shown in Table 6. Again, "Best of A, B, C" means that from the three reference segmentations of A, B, and C the one is chosen for comparison that maximizes the AMT workers' break accuracy. Note that the workers of AMT outperform the algorithmic query segmentation approaches (cf. Table 5) on all accounts. However, the AMT workers cannot achieve perfect segmentation accuracy when compared to the expert segmentations of BWC07, since it must be admitted that even experts make errors.

Having established that the segmentations obtained via AMT are indeed valid, we reevaluated all of the segmentation algorithms against the AMT segmentations as well as against the combination

[2]See [1], and our previous works [16, 17].

Table 6: Segmentation performance of Amazon's Mechanical Turk (AMT) workers on the Bergsma-Wang-Corpus.

Annotator	Performance Measure	"Algorithm" AMT
Best of A, B, C	query	0.821
	seg prec	0.892
	seg rec	0.883
	seg F	0.887
	break	0.941

of the BWC07 segmentations and the AMT segmentations. The results of this evaluation are shown in Table 7. As can be observed, our naïve query segmentation algorithm [10] performs best on the enriched BWC07. Our Wikipedia-based normalization comes in second. A speculative explanation for the remarkable performance of the naïve $|s|^{|s|}$-scoring might be that web n-gram frequencies are correlated with the n-grams an average AMT worker knows.

6.5 The Webis Query Segmentation Corpus

Given the encouraging results achieved with enriching the Bergsma-Wang-Corpus by means of crowdsourcing, the logical next step is to build a larger corpus, thus addressing the remaining points of criticism about the BWC07, namely its small size and the sampling constraints. Following a new sampling strategy (detailed below), we have sampled 50 000 queries from the AOL query log. These queries were checked for correctness of spelling and encoding, and then segmented by workers recruited on Amazon's Mechanical Turk. We followed the same strategy as with enriching the BWC07 corpus, however, this time using the enriched BWC07 as check queries to identify workers who perform poorly. As a result, we present the new Webis Query Segmentation Corpus 2010 (Webis-QSeC-10).[3] Finally, we have evaluated our query segmentation algorithms as well as the baseline algorithm against this corpus, the results of which are shown in Table 8. Unsurprisingly, the absolute performance values are far below those on the BWC07, since our new corpus contains the whole spectrum of queries and not only noun phrase queries. While the mutual information baseline allows for some comparability to the earlier results, a complete comparison of all algorithms against our new corpus is still missing. So far, our Wikipedia-based normalization scheme performs best on this corpus, but once again, mutual information serves as a reasonable and challenging baseline.

Corpus Construction. The 50 000 queries for our corpus were chosen in three steps from the AOL query log: first, the raw query log was filtered in order to remove ill-formed queries; second, from the remainder, queries were sampled at random; and third, the sampled queries were corrected.

In the filtering step, queries were discarded according to the following exclusion criteria:

- Queries comprising remnants of URLs (navigational queries) or URL character encodings.
- Queries from searchers having more than 10 000 queries.
- Queries from searchers whose average time between consecutive queries is less than 1 second.
- Queries from searchers whose median number of letters per query is greater than 100.
- Queries that contain non-alphanumeric characters except for dashes and apostrophes in-between characters.
- Queries that are shorter than 3 words or longer than 10.
- Queries from searchers that duplicate preceding queries of themselves (result page interaction).

Of the 36 389 567 queries in the raw AOL query log, 6 027 600 queries remained after filtering. The majority of the filtered queries (22.8 million) were removed by the criteria pertaining to special characters and query length (e.g., queries shorter than 3 words).

In the sampling step, 50 000 queries were chosen at random from the filtered query log, while maintaining the original distributions of query frequency and query length. To accomplish this, the log was divided into query length classes, where the i-th class contains all queries of length $i \in \{3, \ldots, 10\}$, keeping duplicate queries from different searchers. Then, the query length distribution was computed and the amount of queries to be expected for each length class in a 50 000 query sample was determined (see Table 9). Based on these expectations, for each length class, queries were sampled without replacement until the expected amount of distinct queries was reached. Hence, our sample represents the query length distribution of the filtered AOL log. And since each length class in the

[3] http://www.webis.de/research/corpora

Table 8: Segmentation performance on the Webis Query Segmentation Corpus 2010.

Annotator	Performance Measure	Algorithm MI	[10]	Our
Best of AMT	query	0.598	0.599	**0.616**
	seg prec	0.727	0.736	**0.744**
	seg rec	0.738	0.733	**0.739**
	seg F	0.732	0.734	**0.742**
	break	0.844	0.842	**0.850**

Table 7: Segmentation performance on the *enriched* Bergsma-Wang-Corpus.

Annotator	Performance Measure	Algorithm MI	[4]	[19]	[21]	[7]	[14]	[10]	Our
Best of AMT	query	0.738	0.812	n/a	0.831	0.794	0.546	**0.859**	0.857
	seg prec	0.802	0.890	n/a	0.891	0.869	0.758	**0.909**	0.908
	seg rec	0.806	0.904	n/a	0.889	0.868	0.830	**0.909**	0.908
	seg F	0.804	0.897	n/a	0.890	0.868	0.792	**0.909**	0.908
	break	0.916	0.944	n/a	0.945	0.924	0.850	0.950	**0.952**
Best of A, B, C and AMT	query	0.754	0.825	n/a	0.849	0.802	0.546	**0.873**	0.867
	seg prec	0.809	0.897	n/a	0.910	0.876	0.753	**0.921**	0.917
	seg rec	0.813	0.910	n/a	0.910	0.874	0.825	**0.921**	0.914
	seg F	0.811	0.903	n/a	0.910	0.875	0.787	**0.921**	0.916
	break	0.920	0.947	n/a	0.954	0.928	0.847	**0.957**	0.956

Table 9: Overview of the Webis-QSeC-10 query sample.

Length	AOL Queries	Distribution	Sample
3	2 750 697	45.64%	22 820
4	1 620 818	26.89%	13 445
5	846 449	14.04%	7 020
6	418 621	6.95%	3 475
7	202 275	3.36%	1 680
8	102 792	1.70%	850
9	55 525	0.92%	460
10	30 423	0.50%	250
Σ	6 027 600	100.00%	50 000

filtered log contained duplicate entries of queries according to their frequency, our sample also represents the query frequency distribution in the filtered query log.

In the final correction step, we attempted to correct spelling errors present in the sampled queries by means of semi-automatic spell checking. We collected a 1 million word dictionary of words by combining various dictionaries and other sources for often-checked words available online, such as aspell, WordNet, Wiktionary, and Wikipedia titles, to name only a few. Using this dictionary as well as their Google n-gram counts, we applied the statistical spell checker proposed by Peter Norvig[4] to the query sample. However, we did not follow the spell checker blindly, but reviewed each replacement manually. This way, about 14% of the queries in our sample have been corrected. It must be mentioned, though, that not all errors can be identified this way, and that correcting the queries will remain an ongoing task.

Corpus Anonymization. The AOL query log has been released without proper anonymization, other than replacing the searchers' IP addresses with numerical IDs. This raised a lot of concerns among researchers as well as in the media, since some AOL users could be personally identified by analyzing their queries. We address this problem in our corpus by removing the searcher IDs entirely. This way, only queries from our sample that are unique in the raw log may be mapped back onto their original searcher IDs, if someone would choose to do so.

6.6 Runtime

As mentioned at the outset, it is important to achieve a low runtime per query in order for a query segmentation algorithm to be practical. Since our method heavily relies on looking up potential segments in the Google n-gram corpus, an efficient implementation of an external hash table is required. We have used the approach described in [6], which employs a minimal perfect hash function, to implement a hash table that maps n-grams to their frequencies. This hash table fits in 13 GB of main memory. The implementation of our query segmentation method is capable of segmenting about 3 000 queries per second on a standard PC. Unfortunately, for lack of implementations of the other query segmentation algorithms, we cannot yet report their runtime performances. However, given the sometimes high number of features proposed in the literature and their complexities, we doubt that any of these approaches can beat ours in a fair comparison.

7. CONCLUSION AND OUTLOOK

We introduced a new approach to query segmentation that is competitive with state-of-the-art algorithms in terms of segmentation accuracy, while simultaneously being more robust and less complicated. The approach can be understood as a normalization scheme for n-gram frequencies based on Wikipedia background knowledge. All relevant feature values are pre-processed and stored in a hash table, rendering the approach very fast and efficient. We provided theoretical and empirical arguments to better understand the rationale of our approach. For this purpose, a much larger evaluation corpus also became necessary. We developed a new query segmentation corpus with 50 000 queries, which is two orders of magnitude larger than the reference corpus used in earlier publications. The paper introduced this new corpus and its construction via Amazon's Mechanical Turk. We are planning to release the corpus in the course of an open query segmentation competition.

Furthermore, we pointed out several future directions for research and development. The current evaluation of query segmentation algorithms relies solely on comparing segmentation accuracy against corpora of human-segmented queries, which is fine in itself, but which cannot tell whether query segmentation is actually useful in practice. Hence, future evaluations should also focus on the question whether query segmentation leads to a significant improvement of retrieval performance. In this paper, first steps in this direction were taken.

Another promising future research task is to analyze more elaborate performance measures for query segmentation algorithms. Since many queries are ambiguous, measures that quantify rankings of alternative segmentations for a given query should be investigated. In practice, it is an interesting option for a search engine to deal with ambiguous segmentations by presenting the results of the top-ranked segmentation, but also to offer the second-ranked segmentation in a "Did you mean" manner. Within our experimental evaluation, we observed that whenever our approaches did not match the BWC07 segmentation for the queries on which all three BWC07 annotators agreed, the systems' second-ranked segmentation very often did.

Acknowledgments

We would like to thank Shane Bergsma and Qin Iris Wang (the authors of [4]); David J. Brenes, Daniel Gayo-Avello, and Rodrigo Garcia (the authors of [7]); Chao Zhang, Nan Sun, Xia Hu, Tingzhu Huang, and Tat-Seng Chua (the authors of [21]); and Nikita Mishra, Rishiraj Saha Roy, Niloy Ganguly, Srivatsan Laxman, and Monojit Choudhury (the authors of [14]) for providing us with their experimental data, especially for doing so on a very short notice.

8. REFERENCES

[1] O. Alonso and S. Mizzaro. Can We Get Rid of TREC Assessors? Using Mechanical Turk for Relevance Assessment. In *Proceedings of the SIGIR 2009 Workshop on The Future of IR Evaluation*.

[2] M. Bendersky, W. B. Croft, and D. Smith. Two-stage Query Segmentation for Information Retrieval. In J. Allan, J. A. Aslam, M. Sanderson, C. Zhai, and J. Zobel, editors, *Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2009, Boston, USA, July 20-24, 2009*, pages 810–811.

[3] M. Bendersky, W. B. Croft, and D. Smith. Structural Annotation of Search Queries Using Pseudo-Relevance Feedback. In J. Huang, N. Koudas, G. J. F. Jones, X. Wu, K. Collins-Thompson, and A. An, editors, *Proceedings of the 19th ACM Conference on Information and Knowledge*

[4] `http://norvig.com/spell-correct.html`

Management, CIKM 2010, Toronto, Ontario, Canada, October 26-30, 2010, pages 1537–1540.

[4] S. Bergsma and Q. Wang. Learning Noun Phrase Query Segmentation. In *Proceedings of Conference on Empirical Methods in Natural Language Processing and Conference on Computational Natural Language Learning, EMNLP-CoNLL 2007, June 28-30, 2007, Prague, Czech Republic*, pages 819–826.

[5] T. Brants and A. Franz. Web 1T 5-gram Version 1. Linguistic Data Consortium LDC2006T13, Philadelphia, 2006.

[6] T. Brants, A. C. Popat, P. Xu, F. J. Och, and J. Dean. Large Language Models in Machine Translation. In *Proceedings of Conference on Empirical Methods in Natural Language Processing and Conference on Computational Natural Language Learning, EMNLP-CoNLL 2007, June 28-30, 2007, Prague, Czech Republic*, pages 858–867.

[7] D. Brenes, D. Gayo-Avello, and R. Garcia. On the Fly Query Entity Decomposition Using Snippets. In *Proceedings of the First Spanish Conference on Information Retrieval, CERI 2010, June 15-16, 2010, Madrid, Spain*.

[8] W. B. Croft, M. Bendersky, H. Li, G. Xu. Query Representation and Understanding Workshop. *SIGIR Forum*, 44(2):48–53, 2010.

[9] J. Guo, G. Xu, H. Li, and X. Cheng. A Unified and Discriminative Model for Query Refinement. In S. Myaeng, D. Oard, F. Sebastiani, T. Chua, and M. Leong, editors, *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2008, Singapore, July 20-24, 2008*, pages 379–386.

[10] M. Hagen, M. Potthast, B. Stein, and C. Bräutigam. The Power of Naïve Query Segmentation. In F. Crestani, S. Marchand-Maillet, H.-H. Chen, E. N. Efthimiadis, and J. Savoy, editors, *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2010, Geneva, Switzerland, July 19-23, 2010*, pages 797–798.

[11] J. Huang, J. Gao, J. Miao, X. Li, K. Wang, and F. Behr. Exploring Web Scale Language Models for Search Query Processing. In M. Rappa, P. Jones, J. Freire, and S. Chakrabarti, editors, *Proceedings of the 19th International Conference on World Wide Web, WWW 2010, Raleigh, North Carolina, USA, April 26-30, 2010*, pages 451–460.

[12] R. Jones, B. Rey, O. Madani, and W. Greiner. Generating Query Substitutions. In L. Carr, D. D. Roure, A. Iyengar, C. A. Goble, and M. Dahlin, editors, *Proceedings of the 15th International Conference on World Wide Web, WWW 2006, Edinburgh, Scotland, UK, May 23-26, 2006*, pages 387–396.

[13] J. Kiseleva, Q. Guo, E. Agichtein, D. Billsus, and W. Chai. Unsupervised Query Segmentation Using Click Data: Preliminary Results. In M. Rappa, P. Jones, J. Freire, and S. Chakrabarti, editors, *Proceedings of the 19th International Conference on World Wide Web, WWW 2010, Raleigh, North Carolina, USA, April 26-30, 2010*, pages 1131–1132.

[14] N. Mishra, R. Roy, N. Ganguly, S. Laxman, and M. Choudhury. Unsupervised Query Segmentation Using Only Query Logs. In *Proceedings of the 20th International Conference on World Wide Web, WWW 2011, Hyderabad, India, March 28-April 1, 2011*.

[15] G. Pass, A. Chowdhury, and C. Torgeson. A Picture of Search. In X. Jia, editor, *Proceedings of the First International Conference on Scalable Information Systems, Infoscale 2006, Hong Kong, May 30-June 1, 2006*, article 1.

[16] M. Potthast. Crowdsourcing a Wikipedia Vandalism Corpus. In F. Crestani, S. Marchand-Maillet, H.-H. Chen, E. N. Efthimiadis, and J. Savoy, editors, *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2010, Geneva, Switzerland, July 19-23, 2010*, pages 789–790.

[17] M. Potthast, B. Stein, A. Barrón-Cedeño, and P. Rosso. An Evaluation Framework for Plagiarism Detection. In C.-R. Huang and D. Jurafsky, editors, *Proceedings of the 23rd International Conference on Computational Linguistics, COLING 2010, Beijing, China, August 23-27, 2010*, pages 997–1005.

[18] K. M. Risvik, T. Mikolajewski, and P. Boros. Query Segmentation for Web Search. In *Proceedings of the Twelfth International World Wide Web Conference, WWW 2003, Budapest, Hungary, May 20-24, 2003. Posters*.

[19] B. Tan and F. Peng. Unsupervised Query Segmentation Using Generative Language Models and Wikipedia. In J. Huai, R. Chen, H. Hon, Y. Liu, W. Ma, A. Tomkins, and X. Zhang, editors, *Proceedings of the 17th International Conference on World Wide Web, WWW 2008, Beijing, China, April 21-25, 2008*, pages 347–356.

[20] X. Yu and H. Shi. Query Segmentation Using Conditional Random Fields. In M. T. Özsu, Y. Chen, and L. Chen, editors, *Proceedings of the First International Workshop on Keyword Search on Structured Data, KEYS 2009, Providence, Rhode Island, USA, June 28, 2009*, pages 21–26.

[21] C. Zhang, N. Sun, X. Hu, T. Huang, and T.-S. Chua. Query Segmentation Based on Eigenspace Similarity. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics and the Fourth International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL-IJCNLP 2009, August 2-7, 2009, Singapore. Short papers*, pages 185–188.