Query Segmentation using Supervised Learning

Asif Makhani
asifm@stanford.edu

Takahiro Aoyama
taoyama@stanford.edu

1 Introduction

Improving both recall and precision to return the specific results that the user originally intended to find is an important aspect of search engine improvements. One component of this problem is related to translating the users' intention to the best query sent to a search engine in order to get the most relevant results. For example, if a user is looking for the cheapest 8 GB flash drive, she may enter the keywords: {cheap 8 GB flash drive} or {sale 8 GB USB drive} into the search box. A search engine attempts to derive the query intent from the keywords as well as any user behavior information such as the previous query and could construct a new structured query in order to retrieve the most relevant results.

There are many techniques to translate the original query to reflect the users' intent such as query expansion and query segmentation. In the case of query expansion, new terms are generated (as additions or replacements) to construct a revised query based on the context of the original query. In the case of query segmentation, the query terms are divided into individual phrases and semantic units from the sequence of user's query terms.

The focus of our research is to develop a novel way using supervised machine learning to segment the original queries into a set of phrases. For example, the query { cheap 8 GB flash drive} can be segmented into 3 distinct segments as { (cheap) (8 GB) (flash drive) }. Generating the segments allows us to provide the necessary hints to a search engine to improve its relevance by ranking documents (eg. web pages) with the segments higher than the documents in which the terms of a segment are not adjacent.

One way to achieve that is to translate the query to specify phrases in order to increase precision. That is, (cheap 8 GB flash drive) becomes (cheap "8 GB" "flash drive"). Alternatively, if the above reduces recall, one can build a search engine where the segmentation is provided as additional hints used only during ranking by boosting documents containing a closer proximity of the segment terms.

There are also other potential applications of query segmentation such as providing search suggestions and spelling correction. By computing and storing a collection of known segments from previous user queries, a search engine can find and suggest the segment "closest" to the current query. For eg., the term harry can be mapped to "harry potter" as a suggestion. Similarly, the query "britany spears" can be mapped to a spelling correction of "britney spears" (using edit distance as a method of defining closeness).

2 Related Work

Interestingly, many researchers have worked on the query segmentation problems in different ways. Risvik et al.(2003) approached the problem by combining the frequency count of a segment and the mutual information (MI) between pairs of words in the segment in a heuristic scoring function. In other NLP-based approaches to query segmentation, Tan and Peng (2008) used a generative query model using EM optimization to estimate n-gram frequencies in order to recover a query's underlying concepts that compose its original segmented form. Bergsma and Wang (2007) went beyond simple N-gram count based features by building upon previous NLP work in noun compound bracketing.

3 Approach

We addressed the query segmentation problem as a binary classification problem. For each query q = (t_1,t_2,…,t_n) where t_i is a query term, we extract all possible segments S = {S_1, S_2 … S_n} where S_i = "t_k ,t_{k+1},…,t_p" is a segment limited to adjacent terms. For this study, we limited the set to segments of size two. For example, for q = (cheap 8 GB flash drive), S = {cheap 8, 8 GB, GB flash, flash drive}. Each input segment S_i is characterized by a set of feature $x^{(i)} \in \Re$ and has a binary output $y^{(i)} \in \{0,1\}$ such that the segment S_i is a relevant segment if and only if $y^{(i)} = 1$. So in the case of the above example, "8 GB" and "flash drive" would be classified as relevant segments, and "cheap 8" and "GB flash" as not.

We used queries sampled from AOL query logs (discussed in detail in section 5) with the relevant segments identified, as training data to produce a model using linear regression to predict which segment of a test query is relevant. For each segment, we computed the input x is a list of features and the output y is the binary classification.

Using logistic regression, we have the following log-likelihood function:

$$l(\theta) = \sum_{i=1}^{m} y^{(i)} \log h(x^{(i)}) + (1 - y^{(i)}) \log(1 - h(x^{(i)})) \quad (1)$$

We want to maximize likelihood function $l(\theta)$, where $h(\theta) = \theta_0 + \theta_1 x^{(1)} + \theta_2 x^{(2)} + \dots . \theta_k x^{(k)}$. We estimated the parameters using the Newton-Raphson algorithm for optimizing the likelihood function $l(\theta)$. The θ 's we learned are applied to find the decision boundary to classify S_i.

$$\theta := \theta - H^{-1} \nabla_\theta l(\theta) \text{, where } H_{ij} = \frac{\partial^2 l(\theta)}{\partial\theta_i \partial\theta_j} \qquad (2)$$

Table 1. Examples of inputs and outputs

Query	PMI web	PMI wiki	PMI AOL	Proximity	…	y
ohio state	0.68323	1.61073	6.02425	1	…	1
ohio movie	-5.1510	0	-0.98073	60	…	0

4 Features

We used multiple input features to characterize a segment *S*S_i. where each feature represented the correlation of the terms in a segment (e.g. Distance between two terms in a document, adjacency of terms, and mutual information between terms). As a primary association features, we used Pointwise Mutual Information (PMI) and found it to be both powerful and simple to compute.

$$\mathrm{PMI}(t_1, t_2) = \log \frac{\mathrm{P}(t_1 t_2)}{\mathrm{P}(t_1) \cdot \mathrm{P}(t_2)} \qquad (3)$$

where $\mathrm{P}(t_i) = \frac{\#\text{of occurence of } t_i}{\#\text{of words in the documents}}$ and

$$\mathrm{P}(t_1 t_2) = \frac{\#\text{of occurence of } t_1 t_2}{\#\text{of all pairs in the documents}}$$

The following distinct features were computed for each segment:

1. PMI based on # results using web search – query order preserved
2. PMI based on # results using web search - order not preserved
3. PMI based on # of pair occurrences in a document corpus – query order preserved
4. PMI based on # of pair occurrences in a document corpus – order not preserved
5. PMI based on # of pair occurrences using a query corpus – query order preserved
6. PMI based on # of pair occurrences using a query corpus –order not preserved
7. Proximity of terms in the document corpus

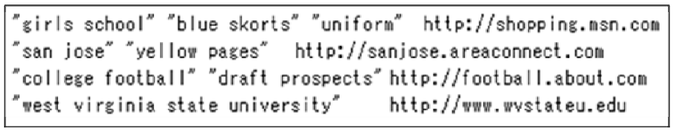
Features (1-6) are PMI using the different data sources (web search engine, document corpus, query corpus) with consideration on the order of the terms in the query. In addition to PMI features, we compute the proximity feature (7) using the distance between terms in a document-based corpus. Since the distance between terms is different in different pages, we assume that distance of terms in the document corpus is given as average of the minimum distance in each document in the corpus:

$$\mathrm{Prox}\ (t_1, t_2) = \frac{\sum \text{Smallest Distance b/w } t_1 \ \& \ t_2 \text{ in i}^{\text{th}} \text{ doc}}{\#\text{of documents containing } t_1 \ \& \ t_2} \quad (4)$$

5 Data

5.1 Training and Testing

Training and test data is a collection of (~1000) queries adapted from the AOL query dataset (Bergsma et al., 2007) with the relevant segments identified to train and test our system. Data contains set of segmented queries and click-URL as seen in Figure 1. For each query with relevant segments identified, we generated both relevant and non-relevant segments. For example, “schools blue” in Figure 1 would be extracted as a non-relevant segment.



```
"girls school" "blue skorts" "uniform"  http://shopping.msn.com
"san jose" "yellow pages"   http://sanjose.areaconnect.com
"college football" "draft prospects" http://football.about.com
"west virginia state university"    http://www.wvstateu.edu
```

Figure 1. Segmented query logs and click-URL

Since our study was limited to segments of size 2, we dropped the data involving larger relevant segments. That is, in the example in Figure 1, a relevant segment “west Virginia state university” does not necessarily imply that all adjacent segments of size 2 are relevant (eg. Virginia state).

5.2 Data Sources

We used the following data sources to compute our features:

a) Web Search Index (using “Alexa Web Search”)

We used web search using the Alexa Web search service to compute PMI based on the number of the search results.

$$\mathrm{P}(t_1 t_2) = \frac{\#\text{of web search results for } "t_1 t_2"}{\text{total } \#\text{ of web pages}} = \frac{N("t_1 t_2")}{T}$$

where N(q) = # of results for the query “q” and T is the total # of web pages (estimated at 5 billion in the case of Alexa’s search index). Thus, we have

$$\mathrm{PMI}(t_1 t_2) = \log \frac{N("t_1 t_2")/T}{N("t_1")/T * N("t_2")/T}$$
$$= \log N("t_1 t_2") + \log T - \log N("t_1") - \log N("t_2")$$

In the cases of features where order is not preserved, we consider both possible orders:

$$\mathrm{P}(t_1 t_2) = \frac{N("t_1 t_2") + N("t_2 t_1")}{T}$$

b) Wikipedia Release Version

In addition to the web search index, Wikipedia Release Version 0.7 [**] is used as a raw document corpus. Wikipedia Release Version is a collection of good quality titles (3152 articles, 10 million words). The motivation of using document-type corpus is to extract the context information, such as words sequence and distance between words, not readily available when using an external search index.

We used the document corpus to compute PMI based on the number of pair occurrences of terms in a segment as well as proximity function based on the distance between terms.

c) AOL query logs

The set of AOL query database (Pass et al., 2006) was also utilized in order to capture the users’ behavior for searching, which is not captured in document based corpus. This data source includes ~20M search queries from 650k real users, containing users’ anonymous ID, query issued by users, time when query was provided, the rank and URL of a website that user clicked. For the purpose of computing PMI based on the number of pair occurrences of terms in a segment, users’ query is only used.

We decided to use both a document-based corpus and a query-based corpus as both has their own tradeoffs. A document corpus such as the entire web provides high coverage but can be costly to compute advance features. A query based corpus, on the other hand, is more manageable but provides less coverage due to its sparseness. However, a query based corpus is much more representative of a users’ intention. In general, we believe that a combination of data sources to get high coverage as well as representation.

6 Evaluation

Our evaluation criteria is based on standard information retrieval metrics of Recall, Precision and F-measure. For each test run, we compare the segments classified as relevant with the actual total relevant segments.

$$\text{Recall} = \frac{|\{\text{relevant segments}\} \cap \{\text{classified relevant segments}\}|}{|\{\text{relevant segments}\}|}$$

$$\text{Precision} = \frac{|\{\text{relevant segments}\} \cap \{\text{classified relevant segments}\}|}{|\{\text{classified relevant segments}\}|}$$

$$\text{F - measure} = 2 \times \frac{(\text{Precision} * \text{Recall})}{(\text{Precision} + \text{Recall})}$$

7 Results

We experimented to build an optimal model by focusing on the computed features.

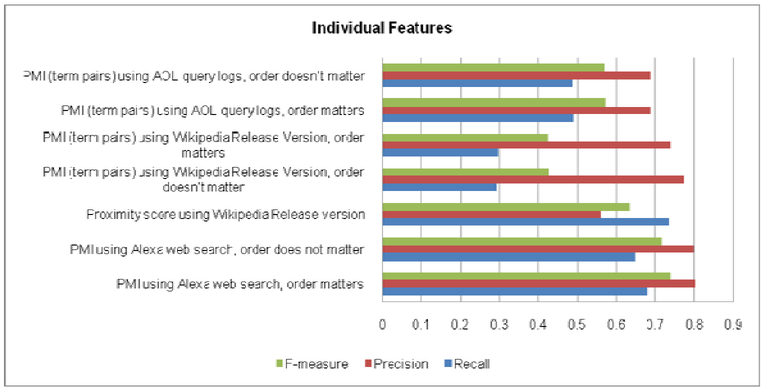


Figure 2: Individual Features

As shown in Figure 2, we first built the model with a single feature and attempted optimizations through various feature combinations as shown in Figure 3 (see Appendix 1 for the results).

In the case of individual features, the feature with both the highest F-measure (73.69%) and highest precision (80.37%) was based on web search with query order preserved. This is mostly related to a high coverage available in a web search index. The feature with the highest recall (73.3%) was based on proximity score using the Wikipedia corpus. The high recall was due to the high coverage achieved when looking at the distance between the terms as a metric and not limiting to adjacent terms only. However, while recall was highest in this feature, precision was the lowest (55.9%) as two terms in the same document may be related but cannot always be considered a relevant segment.

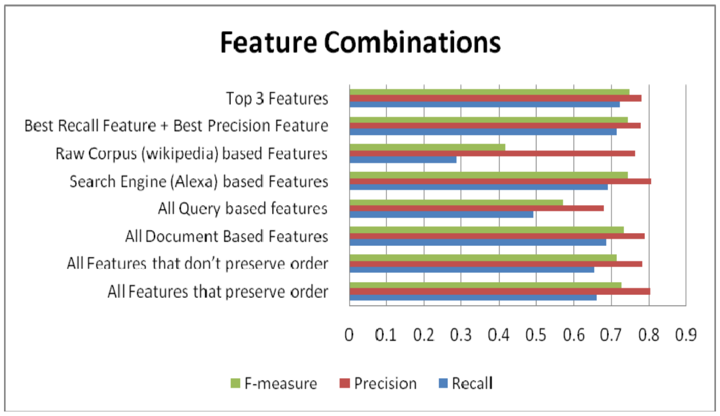


Figure 3: Feature Combinations

Combining features together resulted in improvements in several different ways. Limiting to features that preserve query order resulted in a higher precision (80.52%) and a higher F-measure (72.74%) than features that don't preserve query order (precision = 78.44%, F-measure = 71.46%). This emphasizes the point that order matters in user query segmentation (eg. "new york" is a relevant segment but not "york new").

Due to the sparseness of data in the query logs, document based features were superior to query based features. At the same time, features computed using the Wikipedia document corpus had the lowest recall (28.84%). This is mostly due to limited data but also because wikipedia corpus may not be representative of web search queries which include a significant number of entertainment and shopping related searches that are not captured by Wikipedia.

Web search features had a higher F-measure than other features from other data sources due to the high coverage (5 billion web pages vs 3152 wikipedia articles) as well as additional search engine features such as stopping, stemming, synonyms and punctuation handling that a typical web search engine provides to improve relevance. Since the Wikipedia and query log based features were computed using basic string matching without any sophisticated normalization, this lowered recall as variations of a term present in the corpus were not matched.

The best result (F-measure = 75.13%, Precision = 78.2%, Recall = 72.3%) was achieved by combining the web search features with the proximity score feature based on distance between terms in the Wikipedia corpus. This outperforms query segmentation algorithms using MI (F-measure = 61.6%) as well as other language modeling approaches shown by Tan and Peng (2008) resulting in an F-measure of 67.1% and 71.8% with EM optimization. Our result does underperform when compared with other more sophisticated algorithms such as language modeling with EM optimization augmented with Wikipedia knowledge[1] (F-measure = 80.1%).

8 Future work

The current approach can potentially be improved significantly by using a larger and more representative document corpus for term pair and proximity features. One approach could be to limit the training and feature data to a particular domain or subject (eg. electronics or history) in order to have a representative yet manageable corpus.

Also, future experiments would need to normalize queries and corpus using standard features such as stemming, stopping, punctuation handling in order to increase recall.

Other areas for future work include developing a model for segments of any size (not just 2) as well as considering non-adjacent segments which could help towards query reordering to increase precision. Furthermore, developing a segmentation model that takes into account query context would be beneficial as effective segmentation can be obtained by incorporating the search domain as well as looking at other terms in the current query as well as previous queries.

9 Acknowledgement

We would like to acknowledge Shane Bergsma (University of Alberta) for making available segmented AOL query logs used for training and testing.

10 References

1. Tan and Peng (2008): Unsupervised Query Segmentation Using Generative Language Models and Wikipedia

2. Bergsma and Wang (2007): Learning Noun Phrase Query Segmentation: Query and Feature Data used in Experiments

3. Wikipedia Release Version: http://en.wikipedia.org/wiki/Wikipedia:Release_Version

4. "Alexa Web Search" web service: (http://aws.amazon.com/alexawebsearch/)

11 Appendix

A. *Classification results using Individual Features*

	Feature	Recall	Precision	F-measure
1	PMI using Alexa web search, order matters	0.6804	0.8037	0.7369
2	PMI using Alexa web search, order does not matter	0.6477	0.7972	0.7147
3	Proximity score using Wikipedia corpus	0.733	0.559	0.6343
4	PMI (term pairs) using Wikipedia, order doesn't matter	0.294	0.7724	0.4259
5	PMI (term pairs) using Wikipedia, order matters	0.2983	0.7394	0.4251
6	PMI (term pairs) using AOL query logs, order matters	0.4901	0.6886	0.5726
7	PMI (term pairs) using AOL query logs, order doesn't matter	0.4872	0.6874	0.5702

Table 1: Individual Features

B. *Classification results using Feature combinations*

	Features	Recall	Precision	F-measure
1	All Features that preserve order	0.6634	0.8052	0.7274
2	All Features that don't preserve order	0.6563	0.7844	0.7146
3	All Document Based Features	0.6875	0.7896	0.735
4	All Query based features	0.4929	0.6831	0.5726
5	Search Engine (Alexa) based Features	0.6918	0.8076	0.7452
6	Raw Corpus (wikipedia) based Features	0.2884	0.7632	0.4186
7	Best Recall Feature + Best Precision Feature	0.7159	0.779	0.7461
8	Top 3 Features (1,2,3) from Table 1	0.723	0.7819	0.7513

Table 2: Feature combinations