An Intent Based Grammar for Web-Queries

Microsoft Reasearch India, Bangalore

1 Introduction

The World Wide Web is a large reservoir of information that is still growing at a rapid rate. Web queries can be considered as implicit questions or commands, in that they are performed either to find information on the web or to initiate interaction with web services. Infact, if the web Queries can be parsed at a very top level based on the User’s Intent; it can be very useful for serving them better pages and also showing more appropriate advertisements.

Web users, however, rarely express their intent in full language. For example, to find out "what are the movies of 2010 in which Johnny depp stars", a user may simply query "Johnny depp movies 2010". Today’s search engines, generally speaking, are based on matching such keywords against web documents and ranking relevant results using sophisticated features and algorithms. Many techniques have been devised to parse a web query into some useful chunks. Experiments related to part-of speech tagging on the web-queries suggest that proper-nouns constitute 40% of query terms, and proper nouns and nouns together constitute over 70% of query terms.[1]

2 Earlier work in the field of Query Grammar

Previous works related to Web-Query understanding specifically related to query parsing have evolved to newer levels. Bernard et. al studied a comprehensive classification of user intent for Web searching. They classified it into three hierarchical levels of informational, navigational, and transactional intent. Their findings showed that more than 80% of Web queries are informational in nature, with about 10% each being navigational and transactional.[3]

Allan and Raghavan (2002) and Barr et al. (2008) studied the linguistic structure of queries by performing part-of-speech tagging. Their experiments showed that since most of the words in Queries are Noun words, it’s not very useful to simply parse a Query into POS tags. Mehndi Manshadi and Xiao li (2009) worked on investigating the problem of semantic tagging at the word level, which is to assign a label from a set of pre-defined semantic labels (specific to the domain) to every word in the query. For example, a search query in the "product" domain can be tagged as cheap (sort order) Garmin (brand) street pilot (Model) c340 (Model) gaps (Type). [2]
In this project, we plan to give a more general Grammar for the Web-Queries which are not just domain specific. The grammar will be based on the role of words in a web-query. The query log data suggests that most of the queries have an inherent grammar or intent behind it. The user doesn’t want to merely see the word he has written in his query to just appear many times in his top searches, but actually wants some of them to act as some sort of function (Intention) on the "keyword" of his query. For eg. A query such as Sachin Tendulkar Pictures means that user wants images of sachin tendulkar, But of course he doesn’t want pages that contain the word picture. Thus, here picture is a kind of content specifier. In the next section, we will describe our this new Taxonomy for Web-Queries.

Assuming a bag of words model to parse a web-query doesn’t seem unreasonable. But, again there are certain words which appear together in a particular order with a very high probability and together they form a unit. Hence neglecting that order doesn’t seem a perfect idea either. In section 4, We’ll discuss about the importance of detecting multiword expressions and also introduce a new method for the same.

3 Query Grammar: Introducing the Intent words and the Content words

Looking at large amount of query logs, one can infer that when a user writes a query, all the words are not the "Keywords". Here keyword would mean what the user is looking for. The rest of the words primarily direct that keyword for more suitable results. We can call them 'Intent Words'. Our new grammar suggests that the words in a user query are either Intent words or Content Words. So, in this project, we try to infer some of the properties of the Intent and the Content words from the query log and apply some unsupervised techniques to label a word in a query, as an intent/Content word. In the previous example of the query Sachin Tendulkar Pictures. Clearly, sachin tendulkar is a content word whereas picture is an Intent Word.

3.1 Interesting properties of a Query Grammar

We identified some properties of the Intent and Content words from the Query log data provided by the Bing team, Hyderabad.

Intent Words
a. Occur with various kinds of other words.
b. Typically occur in the beginning /end

Content words
a. Can occur just by them
b. Their co-occurrence with other words is restrictive.
c. A query may not contain an Intent word, but it must have a Content word.

Let’s say $W, W_1, W_2$ and $V, V_1, V_2$ are Intent and content words respectively.

- For queries such as $(W_1 V)$ and $(W_2 V)$. We expect that, $(W_1 V)$ and $(W_2 V)$ should have some similarity in the click distribution of the resulting pages.

- On the other hand, for queries like, $(W V_1)$ and $(W V_2)$; The click through data should not have any overlap.

- In other words, If the content word changes, the context of the query changes and the resulting pages will not be same. If the intent word changes, Some pages may still match. eg. A person may ask for *Sachin Tendulkar Pictures* and *Sachin Tendulkar news*. Here the *news* and *pictures* are the Intent words. Some of the resulting pages in both the cases might be similar.

- This kind of relation in query pair gives evidence for an intent/Content word. But, it does not disqualify the presence of intent/content words.

We expect to see the following properties in the document side wrt to some query:

*There exists some document $D$ relevant for the query: $(W V)$ such that the document $D$ does not contain $W$ but still contains $V$. Then $W$ is an intent word and $V$ is a content word.*

But, it may be difficult to find evidence for all such documents. Since currently the search engines mostly show the result by word matching. Also, this definition may not be true for longer queries which follow (n-1) rule i.e. where the documents satisfying (n-1) words from the queries are also retrieved, but still the missing word is not actually an intent word.

We are looking forward to classify all the words in that query into some 'buckets'. The top level classification of a word is Intent/Content word, and the Intent word itself can be classified in a diverse manner. It should be noted that a word may fall in any of the buckets based on the context of the query. Now, we try to give some examples of such buckets and describe how we come up with the Taxonomy for our Web-Grammar.

Examples:

- **Chinese Restaurant near malleshwaram**
  - Here *Chinese restaurant* is a key word. *Near* is a sort order for *malleshwaram*.
  - *Malleshwaram* may not be present in the pages at all. It is a location word.
  - The pages should be sorted on the basis of "nearness" to *Malleshwaram*. That is the intent of the user.

- **Boats for Sale Australia**
  - Keywords: *Boats.*
- Intent word: *for sale*. *for sale* may not be exactly present in the pages. The pages may contain it in some other form.

- Location word: *Australia*. It is quite unlikely that the page would contain the name of the country on it. But somehow we should be able to determine that it has been written in *Australia*.

- **Learn Guitar**
  - Keyword: *Guitar*
  - Intent word: Learn. The user wants to *learn* or take lessons for *guitar*.

- **Cheap flights to Kolkata**
  - Keyword: *flights*
  - Destination: *to Kolkata*
  - Sort order (Intent Word): *cheap*. It may not be very useful for the user if the page contains *cheap* more number of times than some other page. But what could be more useful is if we are able to tell that the pages should be compared on the basis of prices.

- **Data mining wiki**
  - Keyword: *Data mining*
  - Query source: *wiki*

- **News in India**
  - Keyword: *News*
  - Location word: *India*
Figure 1: Taxonomy of Web-Query Grammar
4 Detecting MultiWord Expressions in Queries

We have assumed that the queries are largely just bag of words. But still there are certain multi-word-expressions which occur together with very high probability. Detecting MWE in a query log is very essential since the combination of words may have an entirely new meaning than the individual components. For eg. 'high', 'school' are different than 'high school' which is much different than 'high school musical', since it a name of a movie. As these words may have a very different and a new meaning compared to their unigrams, we should label this chunk and not just each unigrams.

Statistical techniques are necessary because the lexicons may not always have all the MWEs in it and everyday new MWEs like movies, songs, person’s names and company names are invented.

MWEs can be detected simply by applying the mutual information technique also. But the main disadvantage with this technique is that if any of the word in the gram is very frequent as a unigram itself, then the MWE is not detectable. Also it misses out on the rare MWEs. In this section, We will demonstrate a new technique for detecting MWE.

In the Mutual Information technique, The probability space considered for finding the frequent grams is all the words in the corpus. We argue that it is not necessary to have a probability space of all the words in the corpus. A new probability model for detecting MWE in a corpus containing Web-Queries is described. The model is such that; If we want to detect, if a particular gram say ABC is an MWE then we should only consider the sample space of queries which have A and B and C. And if the presence of ABC together in that sequence is surprisingly high, then that gram is an MWE.

4.1 Test for detecting MWE

Suppose we have a multi-word expression \( M=(a_1 a_2 a_3 \cdots a_t) \) of length \( t \); \( a_1, a_2, a_3 \cdots \) occur in \( K \) Queries but not necessarily together as a gram. Let \( (Z_1, Z_2, Z_3, \cdots Z_k) \) be the indicator elements for each of those Queries.

\[
z_i = \begin{cases} 
1, & \text{With probability } P_i \text{ if } M \in i^{th} \text{ query} \\
0, & \text{Otherwise} 
\end{cases}
\]

Let \( N \) is the number of times the grams occur together in the \( K \) Queries. In order to calculate the probabilities \( P_i \), we make use the fact that the web queries are considered to be bag-of-words,. For a \( l \) length query, a \( l \) length word gram can fit in any of the places. Thus, the probability of all the grams being together(in a particular order) can be given as:

\[
P(Z_i = 1) = \frac{l-t+1}{l!} = \frac{1}{t!} (l) = \frac{l-t+1!}{l!}
\]

(4.1)

\( Z_i \) follows Bernoulli\( (P_i) \). Let \( X \) be a random variable such that

\[
X = \sum_{i=1}^{k} Z_i
\]

(4.2)
\[ E(X) = \sum_{i=1}^{k} P_i \]  
\(4.3\)

In this case, \(X\) follows Binomial distribution say, \(B\) (with mean = \(\sum P_i\) and variance = \(\sum P_i - \sum P_i^2\)).

### 4.1.1 Hoeffding’s Inequality approach

We try to bound the right tail of the pdf of \(X\) using Hoeffding’s Inequality. It provides an upper bound on the probability for the sum of random variables to deviate from its expected value.

Let \(X_1, \ldots, X_n\) be independent random variables. Assume that the \(X_i\) are almost surely bounded; that is, assume for \(1 \leq i \leq n\) that

\[
\Pr(X_i - E[X_i] \in [a_i, b_i]) = 1. \tag{4.4}
\]

Then, for the sum of these variables \(X = X_1 + \cdots + X_n\), we have the inequalities (Hoeffding 1963, Theorem 2):

\[
\Pr(X - E[X] \geq t) \leq \exp \left( -\frac{2t^2}{\sum_{i=1}^{n} (b_i - a_i)^2} \right), \tag{4.5}
\]

which are valid for positive values of \(t\) (where \(E[X]\) is the expected value of \(X\)). Thus, using the above equation for the random variable \(X\) described in \((4.2)\). Here, \(n = k\) and \(b_i = 1, a_i = 0\) for all \(0 \leq i \leq K\).

\[
P(X - E(X) \geq t) \leq e^{-\frac{2t^2}{K}} \tag{4.6}
\]

Now, we bound \(X\) using this equation:

\[
P(X \geq N) < \delta \tag{4.7}
\]

\[
P(X - E(X) \geq N - E(X)) < \delta \tag{4.8}
\]

Hence,

\[
\delta \geq e^{-\frac{2(N - E(X))^2}{K}} \tag{4.9}
\]

The standard way to test where a gram is an MWE or not, would be to fix \(\delta\) and then test if the inequality satisfies equation \((4.9)\). But, we are interested in describing the strength of MWEs w.r.t each other also. This approach can help us chunk the queries into MWEs. Hence we obtain the value of minimum delta for an MWE, which can be obtained by the RHS in equation \((4.9)\). \(\delta\) refers to the surprise factor of an MWE. It means that the occurrence of those grams is much higher than expected. If \(\delta\) is small the surprise factor is more. We should note that the value of \(N - E(X)\) should be positive for a gram to be an MWE. In the experiment section, we will evaluate the value of \(\theta = -\log(\delta)\) for all the grams. We will call the Hoeffding’s score \(S\). The list will be sorted in decreasing order and the values which have a score \(S < 5\) will be removed. It corresponds to the confidence 99.4%. The resulting grams will be considered as multiword expressions.
4.1.2 Normal Approximation approach

We can also make a normal approximation for the Binomial distribution $B$. We call this score as $Z - Score$.

$$B \sim N(\sum P_i, \sum P_i - \sum P_i^2)$$

$$Y = \frac{X}{\sum P_i} - \sum P_i - \sum P_i^2$$

Thus $Y \sim N(0, 1)$, $P(Y > t) = \Phi(t)$ We can obtain $Y$ from equation (4.11) and for a given value of $t$ (for 99.5% confidence, $t = 2.81$). We sort the result based on $Y$ and the values which are less than some particular $t$ are removed. The higher the value of $Y$, signifies higher $Z - Score$ for that MWE.

5 Experiment

5.1 Data Set Characteristics

We used a dataset of 1 million queries each 6 word queries to find significant grams (upto word length 4) in the queries. Since the data set is large we use a Trie to store all the queries.

5.2 Obtaining the List of Word Grams

5.2.1 TRIE

A trie (from retrieval), is a multi-way tree structure useful for storing strings over an alphabet. For general understanding its structure can be understood by the figure 2 of trie. The trie that was actually implemented didn’t have multiple edges from one node. Rather, it had only two edges per node, ie daughter and sibling edge as described by figure 3. The nodes where the words are ending have an EndOfWord marker, the frequency of that word, the scores that we assign (Hoeffdings and the Z-score) and a list of QueryIds attached with it. The grams (MWE) are stored in the same trie using space as the delimiter. It means that the space is treated like a character and it is present as a node in the trie. In order to reduce the disk requirements, this list of Query IDs is stored upto 1st word level only. We do not need the list of queries in which a particular gram is stored, since the EndOfWord marker of that gram will contain the frequency of that gram. But we still need to record how many times the unigrams of a MWE occur in a Query (may be not together or in the same order). Hence, every node ending with end of word marker also contains the information of the number of words in that gram. We use C++ Programming to implement this Trie, find MWE and later apply it to chunk the same query log into MWEs.

5.2.2 Procedure for building the Trie and Calculating Expectation

The Trie that is obtained immediately after feeding all the queries is very vast and contains many nodes whose frequency is $< 5$. In fact many nodes have a frequency 1 also. For a data set of 10 Lack Queries. We first safely remove the queries which are navigational in nature ie queries which contain the word www, http, com au etc. Also the queries which
have no English words are removed. Thus after cleaning the data, we are left with around 8.5 lakh queries.

It is not possible to have all possible grams (up to 4 words) in our Trie because it will be expensive in terms of space. Hence, we use an apriori approach. The Trie is first filled up to 1 word level. That means no bigrams/trigrams/fourgrams are included in the Trie yet. The Trie is pruned after this step. That means, all the words with frequency less than 10 are removed. Then all the bigrams are inserted. The bigrams should be such that both the individual words in that gram should be present in the Trie. Again the Trie is pruned for the same frequency threshold. Thus, following the same procedure we can fill the Trie up to 4 grams. We do not prune after 4 grams. Once the Trie is ready, the expectations are calculated for all the grams.

5.2.3 Algorithm to find the expectations of MWEs in the TRIE

The expectation of a gram is described in equation (4.3). The expectations are calculated for the words with frequency greater than some threshold (in this experiment the threshold is 10). The sample space for calculating $P_i$ in equation (4.3) consists of all the queries which contain all the words of that gram (in any sequence). We need to find the value of $K$, i.e., the number of queries which contain all the words of a gram (in any order). We can find that list of queries say $L$ by intersecting the list of QueryIDs of all the words in a gram at level 1. Then, we calculate the value of $P_i$ for each query in $L$. And summing the $P_i$ over all the queries in $L$ gives us the resultant expectation. The number of queries in that list give us the value of $K$. For each of the queries in that list we can sum up the
probabilities $P_i$. The Hoeffding's score is calculated by equation (??)eq:2) The $Z_{\text{score}}$ is also calculated simultaneously with the Hoeffding's Score.

```
input : Pointer $p$ to the root of the Trie and threshold
output: The Hoeffding's Score and $Z_{\text{score}}$ for all the grams in the Trie
1.1 if $p \neq NULL$ and $p \rightarrow \text{end}_\text{of}_\text{word}$ and $p \rightarrow \text{num}_\text{of}_\text{word} > 1$ then
1.2 new_list = $\cap$ of List_of_Queries of all individual unigrams in word $p$
1.3 $K = \text{Number of Queries in new_list}$
1.4 $t = p \rightarrow \text{num}_\text{of}_\text{word}$
1.5 foreach Query $\in$ new_list do
1.6 $l \leftarrow \text{Number}_\text{of}_\text{words}$ in Query
1.7 Expectation = Expectation + $\frac{l-t+1}{t!}$
1.8 end
1.9 if Expectation > frequency of gram $P$ then
1.10 $P \rightarrow \text{Hoeffding's Score} = \frac{2(P \rightarrow \text{frequency} - \text{Expectation})^2}{K}$
1.11 Expectation($p \leftarrow \text{daughter}$)
1.12 Expectation($p \leftarrow \text{daughter}$)
```

Algorithm 2: Finding Expectation of grams in a Trie

5.2.4 Results

We obtain a list of bigrams, trigrams and fourgrams containing their Hoeffding's and $Z_{\text{score}}$. The data can be sorted based on the Hoeffding's Score. The hoeffding's score may vary from $7.7989 \times 10^5$ to $e^{-4}$. It may be noted that since both $Z_{\text{score}}$ and $\text{Hoeffding's Score}$ give a frequency bound, which is exponential; We do not get very different rankings in both the cases. The word gram 'how to' gets the top rank. We filter out the grams which have the Hoeffding's Score greater than 5 (for a confidence of 99.33).

5.3 Using the Gram-List to chunk the Queries

The same data set of Queries(10 Lac queries 6 word each) is used for chunking. A very simple Algorithm based on the Hoeffding's Score is used to divide the query into chunks. It is assumed that the Unigrams have a score of 0. So, The Queries initially is assumes to have all 1 word, 6 chunks. The total score of a query is the summation score of all chunks. Hence the Initial Score is 0. It is then broken into all possible combinations.(For 6 words Upto 4 grams, we’ll have 30 possible ways in which a Query can be written into chunks). Hence, The Total score of chunks is obtained in each case. and whichever gives the highest is chosen to be the Final chunking of the Query.

5.3.1 Evaluation with respect to Human chunking

A sample of 1000 queries from the data set was generated which is manually chunked. The Gram-List obtained previously is again used to chunk just these queries. The metric
for comparison is very simple. Suppose we have a Query string P Q R S T U. Manually it is chunked as (P Q) (R S T) U and Our system chunks it as (P Q R) (S T) U. We label each word in the query as beginning of a chunk (b), end of a chunk (e), middle of a chunk (m) or a sole word (s). So here the strings are: by manual chunking - bebmes, by machine chunking - bmbmes. We match these two strings and thus get a score of 3/6. Thus after taking an average of these scores for all the queries, we find that the chunking done by our method is around 55% same as the manual chunking.

6 Conclusion and Future Work

Detecting Multiword Expressions is the first step towards classification of a Query into the Taxonomy as suggested by us in this project. In the next step, we can either use an unsupervised or Supervised approach towards query Parsing.

References


**List of Figures**

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Taxonomy of Web-Query Grammar</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>Trie</td>
<td>9</td>
</tr>
<tr>
<td>3</td>
<td>Trie implemented in C++</td>
<td>9</td>
</tr>
</tbody>
</table>