Weighting Common Syntactic Structures for Natural Language Based Information Retrieval

Chang Liu  
School of Computing and Mathematics  
University of Ulster, Newtownabbey  
Northern Ireland, UK, BT37 0QB  
c.liu@ulster.ac.uk

Hui Wang  
School of Computing and Mathematics  
University of Ulster, Newtownabbey  
Northern Ireland, UK, BT37 0QB  
h.wang@ulster.ac.uk

Sally McClean  
School of Computing and Information Engineering  
University of Ulster, Coleraine  
Northern Ireland, UK, BT52 1SA  
si.mcclean@ulster.ac.uk

Epaminondas Kapetanios  
School of Electronics and Computer Science  
University of Westminster  
London, HA1 3TP  
E.Kapetanios@westminster.ac.uk

Denis Carroll  
KTP Unit  
University of Westminster  
London, W1W 6XH  
D.C.Carroll@westminster.ac.uk

ABSTRACT

Natural Language Processing (NLP) techniques are believed to hold the potential to assist "bag-of-words" Information Retrieval (IR) in terms of retrieval accuracy. In this paper, we report a natural language based IR approach where the common syntactic structures between documents and the query is regarded to as a query-dependent feature for documents. Specifically, a "structural weight" is proposed for query terms, which can be seen as a weight to model the degree of term's involvement in the common syntactic structures. This structural weight is used together with the TF-IDF weighting scheme, which results in a new ranking function. The accumulation of this structural weight of all the query terms in the new ranking function will be seen as a measure of how much a document and a query share the common syntactic structures. The experimental results show that by using this ranking function, significant improvements in the retrieval performance are achieved.

Categories and Subject Descriptors

H.3.3 [Information Search and Retrieval]: Search Process; H.3.1 [Content Analysis and Indexing]: Linguistic processing

General Terms

Design

Keywords

Information Retrieval, Ranking, Dependency Tree

1. OUR APPROACH, OVERVIEW

In the line of research of utilizing syntactic structural information of natural language to assist the purely statistical IR ("bag of words" IR), much research work has been done at the syntactic level, to use more accurate and meaningful syntactic phrases or patterns to supplement the single terms based document representation. Our approach [1], "Syntactic Information Retrieval" (SIR), aims to maximally capture common syntactic structures in the context of dependency trees (D-trees), and model the common syntactic structure as a document feature in the ranking stage. Figure 1 shows the SIR architecture. Serving as a benchmark, SIR remains all the structures and features where a "bag of words" IR system has. Moreover, syntactic structures of natural language appear and are manipulated in almost every procedure in a standard retrieval pipeline in SIR. In parsing, all the documents will experience a full text syntactic analysis where syntactic structures are represented by D-trees. D-trees are "pruned" by removing functional words that also appear as leaves. In indexing, D-trees will be retained in a separate file (called T-sequence file) and the inverted index is revised to link to the T-sequence file. In ranking, the query will be also parsed into D-trees. Syntactic structures will be modeled in this stage to assist the "bag of words" IR which is the main contribution of SIR.

2. DEPENDENCY TREE BASED DOCUMENT REPRESENTATION

In SIR D-tree is chosen as the syntactic representation of the natural language text. A dependency relation holds between a head word X and a dependent word Y in a D-tree. A dependency link is an arrow pointing from a head to a dependent: X->Y. A D-tree is a set of links connecting heads to dependents which can easily form a tree with the
3. SIR INDEX

First, a sequence data structure which stores a tree with all the structural information retained is designed. We call it a T-sequence. This is the basic for storing a D-tree in the index. As we can see in Figure 3, a T-sequence is actually based on an array of paths/edges in the tree, with each edge starting from the child to the parent. Taking the right tree in Figure 2 as an example, we have edges/dependency relations <man catch>, <butterfly catch>, <with catch>, and <net with> which form the base of its T-sequence. In an edge such as <man catch>, "man" is the child (dependent) and "catch" is the parent (head). All the edges are stored in its child’s order in the preorder traversal sequence of the tree. Additionally, two offsets are added, the number of tree nodes, and the root node, in the beginning of the T-sequence. In the example, it is "5" and "catch". Given the T-sequence, a unique tree can be generated by linking each child to its parent based on the information in edges.

4. RANKING WITH STRUCTURAL WEIGHT

In the ranking stage, a document is scored by adding the weights of the query terms it contains. The process of adding in contributions to the ranking score of a document of one query term at a time is known as term-at-a-time scoring or accumulation [2]. The term frequencies, document frequencies, etc. which serve as document weights, are the most common features for characterizing the documents. The ranking function is demanded to be able to accumulate evidence of a document’s relevance from multiple sources [2]. The novel features of SIR is to regard the common syntactic structures as a query-dependent document feature. This feature appears in the ranking function as the proposed Structural Weight, short for SW.

4.1 Structural Weight

Taking the idea of both considering the document and query in the using of syntactic information in IR, the proposed SW will be computed only for the terms which occur in both the documents and query. The SW of the query terms that do not occur in the documents is actually zero since this kind of terms do not involve in any of the common syntactic structures. The SW of a query term can be seen as a weight to measure the degree of this term’s involvement in the common syntactic structures in a document-query matching. The accumulation of the SW of all the query terms in the ranking process will be seen as a measure of how much a document and a query share the common syntactic structures.

In a matching between a document and a query, we will have a set of dependency trees from a document by reading
from the T-sequence file in the SIR index, denoted as $DT()$. Also the query is parsed, resulting in a set of D-tree from the query, $QT()$. For one of the query term $t$, in $DT()$ and $QT()$, we will have the D-trees or subtree of the $DT_1, DT_2, ..., DT_m$ and $QT_1, QT_2, ..., QT_m$, which are all rooted at $t$. Each $DT_i$ and $QT_j$ will group a pair of D-trees. In the context of each individual pair, we will consider an Individual Structural Weight (isw) for term $t$, which represents $t$’s involvement in the common syntactic structures in this particular $DT_i$ and $QT_j$ match. The SW of $t$ is the sum of all the isw of $t$ computed from all $n \times m$ pairs. We can see SW is actually is a sum number. It can be obtained by summing up of all the isw.

The definition of isw is inspired by the kernel concept of summing up of all the common substructures as a measure of the similarity between discrete structures [3]. A tree kernel function is able to measure the similarity between trees by counting the number of their common subtrees. In order to find a term’s involvement in all the common syntactic structures between a document and a query, we think it is reasonable to count all the common subtrees/syntactic structures which are all rooted at a query term as a measure.

The isw of a query term $t$ is defined in a pair of document $DT_i$ and a query D-tree $QT_j$ as the number of all the common subtrees rooted at $t$. Figure 4 gives an example of a pair of $DT_i$ and $QT_j$ rooted at a query term “catch”. Since tree kernel methods count all the common subtrees as a measure of common syntactic structures, we can use the isw count by extracting the results set computed from tree kernel methods which are all the common subtrees rooted at the root $t$. Partial Tree (PT) Kernel [4] which allows most general subtree is used to define the isw of a query term.

We revised PT kernel algorithm so it only counts the number of common subtrees for the root node and is used to form the base of isw. Figure 5 shows the isw of “catch” in the context of the two trees in Figure 4. isw of “catch” is 6 and counts the common subtrees: [catch [butterfly]], [catch [with] ], [catch [with [net]]], [catch [butterfly with ]], [catch [butterfly with net]]], and [catch]. For isw, we also count the node term itself as a common subtree. This comes empirically from our ad hoc experiments, the combined use of idf and isw that counts the node itself in the ranking function (Equation (2)) leads to the best retrieval performance (better than isw which does not count itself).

By using the same principle of computing isw, the isw for other terms in pair of D-trees are "catch": 6; "butterfly": 1; "with": 2; "net": 1. We can see that "man" and "woman" has no isw or their isw are 0 as neither of them occurs in both the document D-tree and query D-tree.

4.2 Ranking Function with Structural Weight

The ranking score of a document in SIR has two contributions, one is from the traditional statistical retrieval in terms of statistical term weights, the other is from the common syntactic structural measure in terms of the proposed structural weights for query terms.

We use a instantiation of BM25 weighting function, known as QACW (query adjusted combined weight), as the representative of the statistical based retrieval model [5]. We refer to the score computed for each query term $t_i$ with regard to a document "d" as "vsm($d,t_i$)"

\[
\text{vsm}(d, t_i) = qtf_{t_i,d} \cdot idf_{t_i} \cdot \frac{k_1 \cdot tf_{t_i,d}}{tf_{t_i,d} + k_1 \cdot (1 - b + b \cdot \frac{L_d}{ave_L})}
\]

where $qtf_{t_i,d}$ is the term frequency of the query term in the query; $tf_{t_i,d}$ is the term frequency of the query term in the document d; $idf_{t_i}$ is the inverse document frequency of the query term $t_i$; $k_1$ and $b$ are two parameters where $k_1=1.2$ and $b=0.75$; $L$ is the length of the document and $ave_L$ is the average length of all the documents in the documents collection.

For a query with the query terms $t_1, t_2, ..., t_n$, we have the ranking function in the Equation (1) from below, which serves as a benchmark of statistical IR model:

\[
VSM(d, q) = \sum_{i=0}^{n} vsm(d, t_i)
\]

On the basis of the ranking function (1), we bring in the Structural Weight of the query terms. The Structural Weight is accumulated on the basis of vsm($d,t_i$), which results in the ranking function $SIR(d, q)$ (2) in our approach:

\[
SIR(d, q) = \sum_{i=0}^{n} (vsm(d, t_i) + idf_{t_i} \cdot sw(d, t_i))
\]

where $vsm(d, t_i)$ is the score computed under the vector space model for the query term $t_i$ and the document d; $idf_{t_i}$ is the inverted document frequency of $t_i$, $sw(d, t_i)$ is the structural weight of $t_i$ with respect to the document d and is computed on the basis of the partial tree kernel. Since SW of a query term is the number of all the common subtrees

Figure 4: a pair of $DT_i$ and $QT_j$, context for calculating isw

Figure 5: isw of "catch": 6
Table 1: Topics in different length querying dataset-1

<table>
<thead>
<tr>
<th>Topic</th>
<th>VSM</th>
<th>SIR</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;title&gt;</td>
<td>0.3074</td>
<td>0.3061</td>
</tr>
<tr>
<td>&lt;desc&gt;</td>
<td>0.4145</td>
<td>0.4134</td>
</tr>
<tr>
<td>&lt;narr&gt;</td>
<td>0.3765</td>
<td>0.4204</td>
</tr>
</tbody>
</table>

rooted at the term including the term itself, it has the same nature as the term frequency. Therefore, it is put into the ranking function in combination with $idf$. However, the performance improvements made by ranking function (2) is all contributed by the adds-on factor $idf_{t_i} \cdot sw(d, t_i)$, since the replacement of $idf_{t_i}$, $tf_{d_i t_i}$ does not make any improvements at all in our IR experiments. There may have more than one ranking function under our approach, the one in Equation (2) is the one leading to the optimal retrieval performance in our experiments.

5. AD HOC IR EVALUATION

We run the ad hoc retrieval experiments in a prototype system that is developed to implement all the concepts and processes under SIR that described in the previous sections. Due to the fact that SIR requires extensive computation as a result of full syntactic analysis for each of the documents, we therefore have designed an ad hoc retrieval experiment on the scale of 1000 documents with a set of 50 queries. The documents are from the Wall Street Journal and Associated Press newswire sections of TREC disk 2. The topics are in the range of 251-300. We have two dataset, dataset-1 and dataset-2, each of both contains 1000 documents. We refer the retrieval results produced by the benchmark ranking function (1) in the SIR prototype as "VSM", while the results under SIR approach is denoted as "SIR".

We first conduct an ad hoc experiment on the dataset-1 by querying the TREC topics in different lengths. A TREC topic has three fields <title> (shortest, two or three words), <desc>, and <narr> (longest, a couple of sentence). From Table 2, we can see SIR works better than the benchmark with respect to longer queries (<narr>, which is believed containing more syntactic structures) in terms of Mean Average Precision (MAP) (0.4204 > 0.3765). On the other hand, SIR does not make much difference in processing short queries (<title>, <desc>).

Then we focus on TREC topics in their <narr> fields in the rest of experiments. We give a comprehensive evaluation of the proposed SIR approach in the ad hoc IR experiments by a range of evaluation measures. P(5), P(10), and P(100) is the precision of the top 5, 10, 100 documents in the ranked list respectively; $P@R(=100\%)$ is the precision at the point where all the relevant documents are retrieved (100%). We first report the ad hoc retrieval results produced on the dataset-1. Then we compare it with the results gained on the dataset-2 to verify the consistency and effectiveness of the ranking function (2) in SIR.

From Table 3 and Table 4, in terms of all the evaluation measures, SIR achieves better retrieval results than the benchmark on both datasets. In order words, the use of the ranking function (2) where the syntactic structures are measured in the form of SW, outperforms the use of ranking function (1) which represents a “bag-of-words” IR approach. SIR significantly improve the retrieval performance in terms of MAP by 11.6% and 9.8% on the two datasets respectively. Moreover, SIR makes improvements over all precision levels consistently and particularly makes more contribution in the high ranking documents. It also yields a high increase of improvement on the high recall (a 15.1% increase and a 14.8% increase in terms of the precision at the level when recall achieves 100% on the two datasets).

6. CONCLUSION

Ad hoc IR experiments prove that the SIR approach works better with respect to longer natural language queries. SIR does help to improve the retrieval results compared with the statistical "bag of words" retrieval alone. It makes improvements over all precision levels consistently and in particular makes a great contribution on improving the precision in the high ranking documents. However, although SIR yields a significant improvement in the range of 9%-12% over the statistical IR model in regard to queries with meaningful and standard syntactic structures. It does not help much if the query contains no or less syntactic structures.

7. ACKNOWLEDGMENT

The author would like to thank Professor Alessandro Moschitti, Department of Information Engineering and Computer Science at University of Trento, for allowing access to and use of his Partial Tree Kernel source code.

8. REFERENCES