UIC at TREC-2003: Robust Track (Draft)

Shuang Liu, Clement Yu

Database and Information System Lab
Computer Science Department, University of Illinois at Chicago
{sliu, yu}@cs.uic.edu

Abstract

In TREC 2003, the Database and Information System Lab (DBIS) at University of Illinois at Chicago (UIC) participate in the robust track, which is a traditional ad hoc retrieval task. The emphasis is based on average effectiveness as well as individual topic effectiveness.

Noun phrases in the query are identified and classified into 4 types: proper names, dictionary phrases, simple phrases and complex phrases. A document has a phrase if all content words in a phrase are within a window of a certain size. The window sizes for different types of phrases are different. We consider phrases to be more important than individual terms. As a consequence, documents in response to a query are ranked with matching phrases given a higher priority. WordNet is used to disambiguate word senses and bring in useful synonyms and hyponyms once the correct senses of the words in a query have been identified. The usual pseudo-feedback process is modified so that the documents are also ranked according to phrase and word similarities with phrase matching having a higher priority. Five runs which use either title or title and description have been submitted.

1. Introduction

We believe that the robust retrieval result can be achieved by: (1) effective use of phrases, (2) a new similarity function capturing the use of phrases, and (3) utilizing suitable synonyms and hyponyms which are properly chosen in a word sense disambiguation process.

Noun phrases, if exist in a query, are recognized and classified into four types: proper names of people or organizations; dictionary phrases which can be found in dictionaries such as WordNet; simple phrases which do not have any embedded phrases; complex phrases which are more complicated phrases.

A document has a phrase if all the content words in the phrase are within a window of a certain size. The window size depends on the type of the phrase. For a proper name, essentially all content words have to be adjacent. That is, the window size for a proper name, after excluding non-content words and words in the name, is close to 0. Content words in a dictionary phrase need not to be adjacent so

its window size can be larger. The window size for a simple phrase is larger than that for a dictionary phrase but smaller than that for a complex phrase.

We consider phrases to be more important than individual content words in retrieving documents. As a consequence, the similarity measure between a query and a document has two components (phrase-sim, term-sim), where phrase-sim is the similarity obtained by matching the phrases of the query against those in the document and term-sim is the usual similarity between the query and the document based on term matches. The latter similarity can be computed by the standard Okapi similarity function [RW00]. Documents are ranked in descending order of (phrase-sim, term-sim). That is, documents with higher phrase-sim will be ranked higher. When documents have the same phrase-sim, they will then be ranked according to term-sim.

In traditional pseudo-feedback, new terms which are highly correlated with the original query terms in the top ranked documents are added to the query [BR99, GF98]. In our framework, we impose an additional constraint, namely a new term is added only if it is highly positively globally correlated with a query term/phrase. This information is used to compute phrase-sim as well as term-sim.

Machine readable dictionaries such as WordNet [Mill90] have been utilized in document retrieval. In our approach, WordNet is used to disambiguate word senses. Once a correct or a dominant sense has been identified, additional synonyms and hyponyms are added into the query.

In the remaining part of this paper, how phrases in a query are identified and how they are classified into the four types are discussed in section 2. Section 3 presents a new similarity function capturing the use of phrases. Section 4 describes how WordNet can be utilized. In section 5 we analysis our submitted 5 runs. Section 6 concludes this paper.

2. Phrase Identification

Noun phrases in a query are classified into proper names, dictionary phrases, simple phrases and complex phrases. Brill's tagger [Brill] is used to assign a part of speech (POS) to each word. The POS information will be used to recognize simple and complex phrases.

We first classify phrases into the following types: named entities which are the names of persons and organizations. Dictionary phrases, they are actual phrases which appear in dictionaries such as WordNet. An example is "prime factor". Simple phrases: they are not found in a dictionary such as WordNet, but they are two word phrases as recognized by a simple grammar. An example is "school uniform". Complex phrases: they are not found in a dictionary and are phrases which are more complicated than simple phrases. A complex phrase may have one or more dictionary or simple phrase embedded in it. For example, a complex phrase is "Canadian building code".

While the words in a named entity are required to be adjacent and be in the same order, the words in the other phrases need not be adjacent. In fact, dictionary phrase, simple phrase and complex phrase may appear in different forms in different documents. For example, the phrase "information retrieval" may appear as "retrieval of information" in a document. As a consequence, we impose different proximity conditions for their recognitions in documents. Specifically, for a dictionary phrase to be recognized in a document, we require its component content words to be within certain number of words, say w1; for a simple phrase, the constraint is that the component words are to be within w2 words, where w2 > w1; for a complex phrase, the component words are within w3 words, with w3 > w2. These constants w1, w2 and w3 will be determined by a learning algorithm to be given below. In addition, we require the component content words in a simple phrase and a complex phrase to be highly correlated in the documents to be considered as having the phrase; otherwise, a phrase is not formed and we match individual content words. Intuitively, a dictionary phrase should have its component words in close proximity with each other, say within 3 words apart. However, this might not be true in practice. For example, for the query "drugs for mental illness" which contains the dictionary phrase "mental illness", an example relevant document (in a TREC collection) has the two words quite far apart as shown in the following paragraph:

"Earlier this week, Rose and the family acknowledged that Joseph Lynch was hospitalized for mental problems more than 2 years ago and had been on lithium for his illness until recently. "

Based on the above observation, we decide that suitable distances between components of different types of phrases should be learned instead of determined rather arbitrarily based on intuition. Specifically, for a set of queries, we identify the types of phrases, the distances of the components of the phrases in the relevant documents and the irrelevant documents having high similarities with the queries (in the TREC collections) and have the information fed to a decision tree (C4.5). The decision tree will then supply for each type of phrases a suitable distance d such that for most relevant documents having components of that type of phrases, the component words are within the distance and for most irrelevant documents having high similarities with the queries, their component words have distances exceed d. This information is then applied to a different set of queries which are disjoint from the set of training queries.

3. Similarity Functions

Our hypothesis is that phrases convey more semantic information than individual words. As a consequence, we design a new similarity function which places more emphasis on phrase matching than word matching. Specifically, the new similarity function produces, for each query and each document, a pair (phrase-sim, term-sim), where phrase-sim is the similarity due to matching of phrases and term-sim is the similarity due to matching of individual words. Term-sim can be

computed using the Okapi formula. Suppose the similarities of two documents D1 and D2 with respect to a query Q are (p1, t1) and (p2, t2) respectively. Then the similarity of D1 is higher than that of D2 if p1 > p2 or if p1 = p2, then t1 > t2. In other words, phrase similarity dominates term similarity and only when the two phrase similarities are equal, then term similarities are used to break the tie.

The phrase similarity of a document with a query can be computed as follows. For a named entity, a dictionary phrase or a simple phrase, if the document has the phrase, then its phrase-sim increases by the idf weight (inverse document frequency weight) of the phrase. For a complex phrase, a document may have the entire complex phrase, a phrase embedded in the complex phrase or none of it. In the first case, the phrase-sim of the document is increased by the sum of the idfs weights of all phrases embedded in the complex phrase, including the complex phrase itself. In the second case, it is increased by the idf weight of the phrase embedded in the complex phrase.

4. Synonyms and Hyponyms Utilization

In Wordnet, synonyms having the same meaning are grouped together to form a synset. For example, the noun baby forms a synset with the word infant in one sense, while in a different sense, baby and sister form a different synset. Associated with each synset of a word, there is a definition of the synset. In addition, there is a frequency value which indicates the extent the word is utilized in this sense. Suppose the noun baby and the word infant form a synset with frequency 611. Suppose the noun baby has other synsets with a total of frequency values 54. Then the noun baby is more likely to be used in the sense of infant than any other sense. In general, if a word with a part of speech has multiple senses and one of its synset, say S, has its frequency value higher than the sum of the frequency values of all other synsets of the same word, then the synset S is called the dominant synset of the word in that part of speech.

If a synonym s of a query word or phrase t is to be added to the query, one of the two following rules will be followed: (i) s and t are identical synonyms i.e. there is an unique synset containing both s and t, and there is no other synset containing s or t. (ii) there is a synset containing both s and t and that synset is dominant for both s and t.

Clearly, if there is a unique synset containing both s and t, the synset is dominant for both terms. We now attempt to disambiguate word senses while at the same time add new words into the query. Our strategy to add new terms and to disambiguate word senses is based on the following principles.

4.1 Utilize the adjacent words in the query for sense disambiguation

A given query is parsed. The POS of each word as well as the phrases in the query are recognized. Suppose there are two adjacent terms t1 and t2 in the query and they form a phrase. Each of t1 and t2 has (a) one or more synsets, (b) for each synset, there is a definition, (c) zero or more synsets containing hyponyms (a hyponym of a term t satisfies the IS-A relationship with respect to t; for example, boy is a hyponym of male). These three items (a), (b), and (c) can be used to disambiguate the senses of the two terms. We now consider determining the sense of t1. The sense of t2 can be determined in a similar manner.

Step (1) Check if term t2 or a synonym of t2 is found in the definition of a synset of t1, say S. In this case, the sense of t1 is determined to be the synset S whose definition contains term t2 or its synonym.

Example 1: Suppose a query contains the phrase "incandescent light". The definition of a synset of incandescent contains the word light. Thus, this synset of incandescent is used.

Whenever the sense of a term t is determined, we examine the possibility of adding the synonyms of t in its synset S to the query. For any term t' in S, if S is a dominant synset of t', then t' is added to the query. The weight of t' is given by:

$$W(t') = f(t', S)/F(t') * f(t, S)/F(t) \qquad (1)$$

where f(t',S) is the frequency value of t' in S; f(t, S) is the frequency value of t in S; F(t) is the sum of frequency values of t in all synsets which contain t and have the same part of speech as t and F(t') is the corresponding value for t'.

The first and the second components of the equation represent the likelihood that t' and t have the same meaning as that conveyed by the synset S respectively. Thus, we may interpret the weight of t' to be the likelihood that t' has the same meaning as t.

Example 2: This is a continuation of Example 1. The synset containing incandescent also contains "candent". It can be verified that the synset is dominant for candent and therefore candent is added to the query.

Special case 1: A synonym of a term t can be a single term or a phrase containing multiple words. Sometimes, the phrase p contains the term t. In that situation, adding the phrase p to the query will not be useful, as a document having the phrase must necessarily have the term t. We therefore examine if the terms in p - t can be added to the query. The criterion to add the words in p - t to the query is that each such word must appear in the definition of the determined synset containing t.

Example 3: A determined synset containing the word induction (see Example 5) is generalization, generalisation, induction, inductive reasoning (reasoning from detailed facts to general principles). This synset is dominant for both the word induction and the phrase inductive reasoning. As a consequence, the phrase is being considered for addition to the query. However, induction and

inductive have the same stem. Thus, we consider adding the word reasoning to the query. It is included in the query, because it occurs in the definition of the synset.

In addition to possibly adding terms from the synset S to the query, we also examine the definition of S and attempt to add some terms which are similar to some query term. Specifically, if a term t' in the definition of t has a prefix which is sufficiently similar to a prefix of a query term t", then t' can be added to the query. The reason is that stemming is not a perfect process and very often two words t' and t" are variants of the same word but stemming [Porter] does not reduce them to the same stem.

When the sense of a query term t is determined, we also want to determine if direct hyponyms of t should be added to a query. If the determined synset of t has a unique child (hyponym synset) U, then for each term t' in the synset U, we check if the synset U is dominant in the synsets containing t'; if so, t' is added to the query, with a weight similar to that given by the formula (1).

Step (2): If the sense of some query term t has yet to be determined, then decide whether there is a dominant synset for t. If there is, the sense of t is assumed to be that dominant synset. For a term t' in the synset, if the synset is also dominant for t', then t' is added to the query with its weight given by formula (1).

4.2 Modification of the query and the similarity function

In the last section, if the senses of query terms are determined, then new terms may be added to the query. Such new terms may make the resulting query a Boolean query as explained in the following paragraph.

Consider a query consisting of two query terms t1 with idf weight w1 and term t2 with idf weight w2. Suppose t1 brings in new terms t1' with weight fi' as given by formula (1) and term t2 brings in term t2' with weight f2' as given by the same formula. The weight fi' can be considered as a relative significance between an occurrence of ti versus an occurrence of ti'. That is, an occurrence of ti' is equivalent to fi occurrence of ti. The idf weight of ti' is assumed to be min. {wi, actual idf weight of ti'}. Thus, having x occurrences of ti' results in an idf weight no higher than that of the actual ti and x * fi occurrences of ti.

If there is no phrase in this query, then a document with a1 occurrences of t1' and a2 occurrences of t2' will get a similarity based on the Okapi formula, in which the idf weight of term i is min. {wi, actual idf of ti'} and the term frequency of ti is ai * fi.

If terms t1 and t2 actually form a phrase, then a document having t1' and t2, or t1 and t2', or t1 and t2, or t1' and t2' will get a phrase-sim (phrase similarity) value of the phrase. In addition, it will get the term similarity due to the terms t1, t1', t2 or t2' using the standard Okapi formula. In effect, in the computation of phrase-simt, the query is equivalent to (t1 AND t2') or (t1' AND t2') or (t1' AND t2).

As an example, if a document has both t1 and t2', then its phrase-sim will be the same as if it has t1 and t2. However, its term similarity is computed based on the occurrences of both t1 and t2'.

4.3 Utilize pseudo-feedback for reinforcement

It is known that pseudo-feedback helps in improving retrieval effectiveness. However, it usually brings in a reasonably large number of extraneous terms. It is also possible that the synonyms and hyponyms which are brought in by the above sense disambiguation process may also consist of both useful and useless terms. Based on this intuitive idea, we suggest the following.

(1) Each of the terms brought in by one of the two processes (pseudo-feedback and sense disambiguation) will be initially be given a weight which is dependent on its correlation with the query terms (in the pseudo-feedback process) or a weight which is dependent on its frequency value in a synset (in the sense disambiguation process). These weights will be adjusted such that they are significantly below that of the original query term.

(2) When a term is brought in by both processes, their weights are added together. Based on the above description, a term which is brought in by both processes is likely to be a useful term and is therefore given a high weight. A term brought in by one but not both of the two processes will get a relatively low weight. As a consequence, even if it is extraneous, it will not adversely affect retrieval effectiveness significantly.

5. Robust Track

In the robust track, we submit 5 runs. Run 1, 3, and 5 use both the title and the description; run 2 and run 4 use the title only. WordNet is used to disambiguate word senses and supply synonyms and hyponyms in each run. Pesudo-feedback is applied to all runs. Table 1 gives the average precisions of the 5 runs over the 50 old topics, the 50 new topics, and the entire set of 100 topics.

	Run1	Run2	Run3	Run4	Run5
50 Old Queries	0.1674	0.1548	0.1622	0.1527	0.1608
50 New Queries	0.3133	0.3037	0.3125	0.3065	0.3172
100 queries	0.2404	0.2293	0.2373	0.2296	0.2390

Table 1. Average Precision for TREC 2003 Robust Track

The average precision gives the overall performance. The individual effectiveness is measured by the (a) number of topics with no relevant document retrieved in the top 10 positions and (b) the area under MAP(X)-vs-X measure where X is the number of topics (queries) having the worst mean precision

and MAP(X) is the mean precision of the Xth worst topic [Robust]. These two measures reflect the robustness of any given retrieval strategy. Table 2 gives the number of topics with no relevant document in the top 10 positions for the new, the old and overall queries sets.

	Run1	Run2	Run3	Run4	Run5
50 Old Queries	8	10	10	11	10
50 New Queries	5	6	6	6	5
100 Queries	13	16	16	17	15

Table 2. Number of topics with no relevant document in the top 10 positions

Table 3 lists the area under MAP(X)-vs-X statistic information. For the entire set of 100 topics, X ranges from 1 to 25 only. For the two sets of 50 topics (50 old and 50 new), X ranges from 1 to 12 only. Worst topics are defined with respect to the individual run.

	Run1	Run2	Run3	Run4	Run5
Old 50 Queries	0.0076	0.0052	0.0065	0.0050	0.0061
New 50 Queries	0.0409	0.0354	0.0402	0.0387	0.0452
Overall	0.0141	0.0114	0.0126	0.0107	0.0124

Table 3. Area under MAP(X)-vs-X evaluation

6. Conclusion

Our TREC-2003 experiment shows that the intuitions regarding the robust retrieval are reasonable. That is the robust retrieval result can be achieved by: (1) effective use of phrases, (2) a new similarity function capturing the use of phrases, and (3) utilizing suitable synonyms and hyponyms which are properly chosen in a word sense disambiguation process. We are experimenting with more complicated techniques of word senses disambiguation in the document retrieval, which hopefully will yield much better effectiveness in the future.

Reference

[BR99] R. Baeza-Yates and B. Ribeiro-Neto, Modern Information Retrieval, ACM Press, Addison-Wesley, 1999.

[BS95] C. Buckley and G. Salton. Optimization of relevance feedback weights. ACM SIGIR95, p351-357

[Brill] Eric Brill. Penn Treebank Tagger, Copyright by M.I.T and the University of Pennsylvania

[GF98] David Grossman and Ophir Frieder, Ad Hoc Information Retrieval: Algorithms and Heuristics, Kluwer Academic Publishers, 1998.
[GF98] David Grossman and Ophir Frieder, Ad Hoc Information Retrieval: Algorithms and Heuristics, Kluwer Academic Publishers, 1998.
[Porter] Martin Porter, Porter Stemmer http://www.tartarus.org/~martin/PorterStemmer/index.html

[RW00] Robertson S E, Walker S. Okapi/Keenbow at TREC-8. *TREC-8*, 2000

[Robust] Robust Track Guidelines. http://trec.nist.gov/act_part/tracks/robust/robust.03.guidelines.html