Recognition and Classification of Noun Phrases in Queries for Effective Retrieval

[1]Wei Zhang, [2]Shuang Liu, [1]Clement Yu, [3]Chaojing Sun, [4]Fang Liu, [5]Weiyi Meng

[1]Department of Computer Science
University of Illinois at Chicago
Chicago, IL 60607, USA
{wzhang,yu}@cs.uic.edu

[2]Ask.com
Edison, NJ 08837, USA
shuang.liu@ask.com

[3]Broadcom Corporation
San Diego, CA 92128, USA
chaojing@gmail.com

[4]Live Search, Microsoft
Redmond, WA 98052, USA
fangliu@microsoft.com

[5]Department of Computer Science
Binghamton University
Binghamton, NY 13902, USA
meng@cs.binghamton.edu

ABSTRACT

It has been shown that using phrases properly in the document retrieval leads to higher retrieval effectiveness. In this paper, we define four types of noun phrases and present an algorithm for recognizing these phrases in queries. The strengths of several existing tools are combined for phrase recognition. Our algorithm is tested using a set of 500 web queries from a query log, and a set of 238 TREC queries. Experimental results show that our algorithm yields high phrase recognition accuracy. We also use a baseline noun phrase recognition algorithm to recognize phrases from the TREC queries. A document retrieval experiment is conducted using the TREC queries (1) without any phrases, (2) with the phrases recognized from a baseline noun phrase recognition algorithm, and (3) with the phrases recognized from our algorithm respectively. The retrieval effectiveness of (3) is better than that of (2), which is better than that of (1). This demonstrates that utilizing phrases in queries does improve the retrieval effectiveness, and better noun phrase recognition yields higher retrieval performance.

Categories and Subject Descriptors

H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing – *dictionaries, linguistics processing*. H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval – *query formulation*.

General Terms

Algorithms, Design, Experimentation.

Keywords

Information Retrieval, noun phrases, proper noun, dictionary phrase, simple phrase, complex phrase, feedback, verification.

1. INTRODUCTION

The objective of this paper is to detect various types of multi-word noun phrases in a query. In this paper, we consider the queries that are short and similar to typical web search queries. The detected noun phrases are used to interpret the original query in order to improve retrieval effectiveness. Noun phrases are classified into four categories: (1) proper noun; (2) dictionary phrase; (3) simple phrase and (4) complex phrase. A proper noun (PN) refers to the name given to a person, place, event, group or organization, etc., for example, "*Tom Smith*". Dictionary phrases (DP) are the noun phrases defined in dictionaries, but not proper nouns, for example, "*computer monitor*". Both the simple noun phrase (SNP) and complex noun phrase (CNP) are the noun phrases that are grammatically correct, used in the daily language, but not formally defined in dictionaries. We require SNP to contain exactly 2 words and CNP to contain three or more words; for example "*small car*" is an SNP and "*local movie theater*" is a CNP. The reason for recognizing phrases in queries, and classifying them into the four types, is that noun phrases are known to be very helpful for document retrieval [1][4][16][28]. A recent paper [20] shows that proper use of these four types of phrases yields significantly higher effectiveness in document retrieval than just using the individual query words. Zhou, et. al [33] reported that the use of concepts, which are equivalent to phrases, in biomedical IR also resulted in higher retrieval effectiveness than just using individual query terms. Our work converts the original query from a set of words to phrases in order to improve document retrieval performance. In this paper, we utilize the following tools:

(i) Wikipedia [32] is a comprehensive online encyclopedia. It is used to recognize proper nouns and dictionary phrases.

(ii) WordNet [12] is an electronic dictionary. It is used to recognize dictionary phrases and certain proper nouns.

(iii) Minipar [18] is used to recognize proper nouns.

(iv) Collins parser [9] is used to recognize SNP and CNP.

(v) Google [14] is used to recognize some proper nouns and to collect documents to provide some statistics. Actually, Google is used as a large document collection. Any search engine, which has the property of ranking documents that have a set of query words in a small window size, ahead of documents that either have a proper subset of the query words, or have the words in a larger window, is sufficient for our purpose.

We apply Wikipedia, WordNet, Minipar and Google to detect PNs and DPs in the queries. We use a document collection to test if these PN/DPs are really used in the real world. They are discarded if not found in the documents. Then the SNPs and CNPs are searched in the queries. These two types of phrases are grammatically correct, used in daily language, but do not have entries in dictionaries. Collins parser and some statistical techniques are used to detect the SNP/CNPs. This includes detecting the shorter noun phrases embedded in longer noun phrases. For example, a query "*Orlando travel agents*" is a CNP. "T*ravel agents*" is a SNP. Both need to be recognized. Our algorithm is tuned using a set of 400 web queries randomly selected from a web query log. It is tested on another set of web queries and a set of multi-word TREC query titles. The contributions of this paper are

(i) The evaluation of various tools and their combinations for their effectiveness on noun phrase detection task.

(ii) The analysis of the errors made by each tool.

(iii) The development of an algorithm that combines different tools for noun phrase recognition in short queries.

(iv) An operational system that yields higher noun phrase recognition accuracy than a baseline system.

(v) Experimental results show that recognizing phrases in queries improves document retrieval effectiveness.

The rest of this paper is organized as follows. Section 2 describes the PN/DP recognition algorithm. The SNP/CNP recognition is explained in Section 3. Experimental results are reported in Section 4. Section 5 reviews the related works. Conclusions are given in Section 6.

2. RELATED WORKS

Phrases have been used in document retrieval [1][4][16][20]. Higher retrieval effectiveness over the individual word method has been reported.

Lima et al [17] studied the proper noun and phrase recognition problem. They used the EM algorithm to estimate the parameters of a probabilistic context-free grammar (PCFG), given a large Web query log and a hand-written context-free grammar. The PCFG was utilized to compute the most probable parse for a query, which was then employed for phrase recognition. They studied on the company names, human names and other short Web queries, where company and human names could fit in our PN category, and other queries could fit in our SNP/CNP categories.

Mihalcea and Moldovan [24] claimed that the implicit phrases were recognized but technical details were not provided.

Florian, et al [13] presented a classifier-combination framework for named entity recognition. Four statistical classifiers are combined. Also, additional dictionary data and two more classifiers are combined. The best f-scores on the training set and the test set are 93% and 88% respectively. Florian's system mostly focuses on named entity (PN) recognition, while ours also finds the ordinary phrases (SNP and CNP) because we aim to provide general phrases to document retrieval systems.

Evans and Zhai [10] extracted meaningful short noun phrases from documents by using both corpus statistics and linguistic heuristics. The performance of their system on the noun phrase recognition was not directly reported. Instead, they compared and reported the document retrieval results of (1) using and (2) not using the recognized noun phrases. The system was tested on TREC-1993 queries 51-100. The setup (1) improves the precisions at all the document levels over (2), 13% improvement at 5-doc level, 6% at 10-doc level and 7% at 15-doc level, etc.

Lin [19] only uses Minipar and WordNet for conference name recognition. We not only add another tool, Wikipedia, but also expand the conference name to various types of proper nouns.

The following aspects of our work appear to be novel: (1) we integrate different strategies into a unified algorithm to recognize different types of noun phrases, (2) Minipar, Wordnet Wikipedia and a large document collection are used jointly to find different types of proper nouns. (3) Phrase verification is carried out at the statistical level, validating whether a sequence of words is commonly used as a whole in the real world. The document collection is also used to determine which of two overlapping phrases is the desired one.

3. PROPER NOUN AND DICTIONARY PHRASE RECOGNITION

The PN/DP recognition algorithm recognizes proper nouns (PN) and dictionary phrases (DP) in a query. The pseudo code is given in Figure 1.

```
1 function recognizePNandDP(t) { //t is an n-word query t (w1 w2 ... wn)
2     for x = n downto 2 {
3         //finding level-x potential PN/DP
4         for each phrase p in the set of x-word sub-phrases of t {
5
6             if (is_wiki_PN(p) or is_wiki_DP(p) or is_wn_PN(p) or is_wn_DP(p)
7                 or is_minipar_PN(p) or is_Name(p) or is_doc_PN(p)) {
8                 if (phrase_verification(p) == TRUE)
9                     save p as a level-x_potential_PN/DP;
10            } // end of if
11        }//end of for p
12        // resolve overlapping at level-x
13        while (overlapped level-x potential_PN/DPs exist) {
14            for each pair of overlapped potential_PN/DP (p1, p2) {
15                p' = pick_one_phrase(p1, p2);
16                p' is a recognized PN/DP;
17            } //end of for each pair
18        } end of while
19        discard lower level phrases, which are contained by level-x potential PN/
DPs, from the candidate set.
20    }// end of for x
21    //resolve overlapping across different levels
22    while (2 overlapped potential_PN/DPs from different levels exist) {
23        for each pair of cross-level overlapped potential_PN/DP (p1, p2) {
24            p' = pick_one_phrase(p1, p2);
25            p' is a recognized PN/DP;
26            set the sub-phrases of the un-picked p as "not checked"
27        } //end of for each pair
28    } end of while
29    go to 2 if cross-level overlapping resolve happens
30 }// end of function
```

Figure 1. Algorithm for PN/DP recognition

From line 2 to line 20, given an n-word query p (w_1 w_2 ... w_n), the PN/DP recognition starts from p itself (n-word candidate, a string of n consecutive words in the original query); if failed, it searches in the (n-1)-word sub-phrases of p; this process repeats until reaching the 2-word candidates. We call the searches among the x-word candidates the "level-x" search. A recognized PN/DP is called a "level-x potential PN/DP", because they still need to be verified. We do not consider the case of two words, which are not in consecutive position, forming a phrase. For example, "*computer price*" is not examined as a candidate given a query "*computer monitor price*".

is_wiki_PN(p). In line 6, function is_wiki_PN(p) uses Wikipedia to check if a phrase candidate *p* is a PN. A PN should be defined in a dictionary if it is well known, and should have all of its content words capitalized. When *p* is submitted to Wikipedia, a definition page should be returned if *p* is defined. We require that the first instance of p in the main body of that page have all its content words capitalized, in order to label *p* as a Wikipedia PN. We emphasize the "main body of the page" because the words in the page title are usually capitalized regardless of their types.

is_wiki_DP(p) is similar to is_wiki_PN(p) but the *p* needs not to be capitalized in the Wikipedia text in is_wiki_DP(p).

is_wn_PN(p). In line 6, this function uses WordNet as another dictionary to check if *p* is a PN. If (1) WordNet recognizes *p* as a defined noun phrase; (2) one of the *p*'s hypernyms in WordNet is a city, province, country, organization, geographical area, person or a syndrome; and (3) the content words of *p* in its WordNet definition are capitalized, *p* is labeled as a WordNet PN.

is_wn_DP(p) recognizes *p* as a DP if it is defined in WordNet but not qualified as a WordNet PN.

is_minipar_PN(p). In line 7, this function uses Minipar to check if *p* is a PN. If Minipar gives *p* a "PN" label, or one of the labels of "person", "country", "corpname", "location", "corpdesig", "fname", "gname", "date", which refer to people, location, organization, family name, given name and dates, respectively, the *p* is labeled as a Minipar PN. Minipar is only used to recognize PN but not for DP, because it is not a pure dictionary. Some of its decisions are made based on grammatical rules.

is_Name(p). In line 7, this function uses a list of first names, a list of last names, and several name-written patterns to detect names of persons. This is necessary because the names of the ordinary people are hardly defined in dictionaries. In our implementation, we use the name lists from the U.S. Census Bureau, because currently we only test our algorithm using documents written in English. The patterns we adopt are "first name initial, last name", "first name initial, middle name initial, last name", "first name, middle name initial, last name", "first name, last name". This function is fired if a whole query itself matches any of the patterns, while other tools can not recognize any PN/DP in this query. During the algorithm tuning, we tried to apply this function to parts of the queries. But it gave us many false positives. So we decided to only apply it to a whole query.

is_doc_PN(p). In line 7, this function detects the less-famous proper names in a query, such as less well-known people, organizations, or locations that are not indexed in dictionaries. This is necessary as is_Name( ) only detects names of persons. A phrase candidate *p* is searched in a document collection. If at least three instances of *p* are found in documents, such that (1) All non-stop words in these instances are capitalized, (2) None of the instances is a sub-string of a longer string, which has all the non-stop words in capitalized form. (3) At least one of the instances is not a sub-string of a longer proper noun. Then we say *p* is a PN recognized in a document collection. In our implementation, we submit *p* to Google. The actual pages of the top 10 returned documents serve as the document set.

We need to emphasize that we use Google as a huge document set and use it to give essential statistics to our system when making a decision. Any document retrieval system, which can return documents that have a set of specified words appeared in the smallest text window ahead of documents having the set of words in larger window sizes, can be used. Using a large text corpus to get statistics is common in IR. The system in [6] use locally stored static corpus to do statistical computation for document retrieval.

phrase_verification (p). In line 8, this function verifies the existence of PN/DPs that are recognized by one of the above functions. It is a simplified is_doc_PN(p). To verify a potential PN *p*, This function looks for at least one instance of *p* in a document collection, such that this instance satisfies the condition (1) and (2) as described in is_doc_PN(p). The criteria is less strict for phrase_verification(p) because *p* has been recognized by one of the above functions. We use the documents returned by Google as the document collection. For example, Minipar labels "*vista window company*" as a PN. An instance of this phrase is found in a document as "*Vista Window Company is proud to* …". A DP does not need to be verified by looking for instances in the documents, because of the definition of DP.

pick_one_phrase(p1, p2) function. When PN/DPs overlap, two resolving processes are conducted from line 12 to 28. First, from line 13 to 18, we resolve the overlapped PN/DPs at the same level. The function pick_one_phrase(p1, p2) at line 15 picks one phrase from two partially overlapping PN/DPs p1 and p2. This function searches all the words of p1 and p2 together in a document collection, and counts the occurrences of p1 and p2 in the documents. The one with the larger number of count is picked. In implementation, we submit all the content words in p1 and p2 to Google. The top 10 returned documents form a document set. For example, in a query "*pocket watch chains*", "*pocket watch*" is a Wikipedia DP and "*watch chains*" is a WordNet DP. The query "*pocket watch chains*" is submitted to Google together to retrieve documents. "*Pocket watch*" has 62 instances while "*watch chains*" has 39 in stances in the returned documents. So "*pocket watch*" is picked.

At line 19, all the sub-phrases that are contained within the recognized PN/DPs are discarded. The intuition is that the words in a PN/DP should not be decomposed.

From line 22 to 28, the overlapping problem between two PN/DPs at different levels is solved. For example, a query is "*starlite drive in movie theatre*", Wikipedia recognizes "*drive in movie theatre*" as a level-4 DP. Is_doc_PN(p) recognizes "*starlite drive in*" as a level-3 PN. The pick_one_phrase(p1,p2) at line 24 solves the overlapping problem. It is as same as the pick_one_phrase(p1, p2) at line 15. "*Starlite drive in*" got 26 instances and "*drive in movie theatre*" got 5. The former one is chosen. Line 26 is necessary because it is possible that a discarded PN/DP candidate may still contain shorter valid PN/DPs. For example, even "*drive in movie theatre*" is discarded; its sub-phrase "*movie theatre*" can still be recognized as a valid DP.

At line 29, the whole procedure (line 2 to 28) will run again as long as there is candidate labeled as "not checked" at line 26.

4. SIMPLE AND COMPLEX PHRASE RECOGNITION

Some noun phrases are not PN or DP, yet are grammatically correct and are used in the English language. If such a noun phrase contains exactly two words, we define it as a "simple noun phrase" (SNP). If it has three or more words, we define it as a "complex noun phrase" (CNP). For example, "*white car*" is a noun phrase, but probably not defined in any dictionary. This type of noun phrases is also useful to improve retrieval effectiveness. We adopt Collins parser [9] to recognize them. Collins parser is a language parser with phrase structure annotation ability. Brill tagger [3] attaches part of speech (POS) tags to the query words, because Collins parser needs them.

4.1 Punctuations and Headwords

Before the parser processes a query, the query is pre-processed. If two words connected by a hyphen, they are either unchanged, merged as a single word or the hyphen is replaced by space. The one that is the most frequent one in a document set is chosen. Other punctuation marks except the apostrophes are removed.

A headword (HW) of a phrase is the element that determines the syntactic function of the whole phrase. In a noun phrase, the head is the noun that refers to the same entity that the whole phrase refers to [11]. It plays the same grammatical role as the whole constituent [28]. For example, "*art*" is the HW of DP "*performing art*". Before actually recognizing SNP/CNP, the PN/DPs in a query are replaced by their HWs. A PN/DP is sent to Collins parser. The parser generates a parse tree. The root of the tree is the corresponding HW. The (PN/DP, HW) mapping information is stored for future PN/DP restoration. The HW replacement is necessary because in some circumstances, the existence of the whole PN/DP may cause parsing error. The HW replacement may help the parser get the correct parse tree.

Example 1. *In a query "download pieces of me", "piece of me" is a PN. The whole query is a valid CNP. Without HW-replacement, Collins parser considers "download pieces" and "of me" as two phrases. With the HW-replacement, "pieces of me" is replaced by its HW "pieces". "download pieces" is parsed as a noun phrase (download/NN pieces/NN). After restoring the "pieces of me", the whole title is correctly recognized as a noun phrase.*

4.2 Implicit Phrase

A coordinate structure in texts involves several components that are connected by "and" or "or". This sometimes indicates that there are implicit phrases in the text. For example, a query "main and contributing factor" has two explicit phrases: the query itself and "contributing factor". But there is also an implicit phrase "main factor". This implicit phrase can not be recognized directly. Given a query with coordinate structure, we use a set of grammatical rules to find the implicit phrases.

4.2.1 Noun Phrase in a Coordinate Structure

Rule1: CONJ: - [Comp_i CC]^n Comp'

Rule2: Comp':- [Mod_i]^m Head

where CONJ is a coordinate phrase; CC designates either "and" or "or"; Compi is an adjective or a noun; Comp' is a noun phrase with at least one modifier, and Head is the headword of Comp'. For the noun phrase "main and contributing factor", its Collins parsing structure is shown in Figure 2. The node "CONJ/factor" refers to a coordinate structure, which includes a noun phrase, an adjective "main", an "and" and a noun phrase "NP/factor", while the node "NP/factor" in turn includes an adjective "contributing" and a noun "factor". So the Comp1 is "main", CC is "and", and Comp' is "contributing factor", Mod is "contributing", Head is "factor", and both m and n are 1.

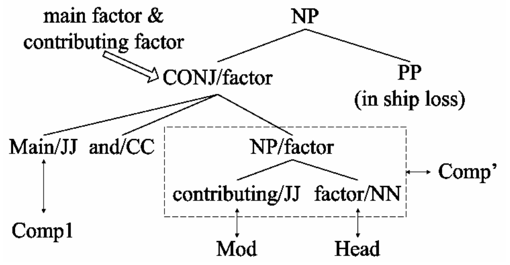

Figure 2. noun phrases in coordinate structure

When this set of rules is fired, the new phrases are generated as:

NP_i:- Comp_i NN (i =1, 2, …, n)

In the example, the new phrase "main factor" is generated.

4.2.2 Coordinate Structure in a Noun Phrase

Rule 1: NP :- CONJ NP'

Rule 2: CONJ :- [Comp_i CC]^n Comp_n

where NP is a noun phrase containing a coordinate noun phrase CONJ, and a noun or noun phrase NP', CC refers to either "or" or "and"; Comp_i represents a component noun phrase or adjective phrase. In the noun phrase "physical or mental impairment", NP' is "impairment", Comp_1 is "physical", CC is "and"; and Comp_2 is "mental". The parse tree is shown in Figure 3.

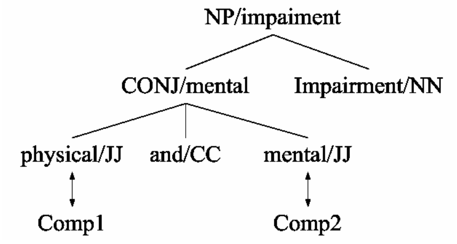

Figure 3: Coordinate structure in noun phrases

When this set of rules is fired, new phrases are generated as:

NP_i :- Comp_i NP' (i = 1, 2, …, n)

In the above example, new simple phrases "physical impairment" and "mental impairment" are generated.

4.2.3 Noun Phrase with Coordinate Structure and Prepositional Phrase

Rule 1: NP :- CONJ PP

Rule 2: CONJ :- [Comp_i CC]^n Comp_n

where NP is a noun phrase which contains a coordinate noun phrase, PP is a prepositional phrase, CC designates "or" or "and"; and Comp_i represents a component noun phrase. For the noun phrase "systematic explorations and scientific investigations of Antarctica", PP is "of Antarctica", Comp_1 is "systematic explorations", CC is "and"; and $Comp_2$ is "scientific investigations". When this set of rules is fired, new phrases are:

$$NP_i \text{ :- } Comp_i \text{ PP } (i = 1, 2, \ldots, n)$$

In the example, phrases "systematic explorations of Antarctica" and "scientific investigations of Antarctica" are generated.

In some cases, the coordinate noun phrase itself may satisfy 2 or more set of rules, for example "[main and contributing factor] in ship loss" satisfies the rule in 6.3, while "main and contributing factor" satisfies rule in 6.1. Thus, the new phrases are "main factor in ship loss" and "contributing factor in ship loss" as shown in Figure 2.

4.3 Recognize SNP and CNP

After the punctuations, headwords and coordinate structure in a query are processed, the Collins parser is used to recognize the SNP and CNP in the query. The pseudo code of the whole algorithm is given in Figure 4.

```
1  function recognizeSNPandCNP(Q) {// Q is a query
2    for each PN/DP p that has been recognized in Q {
3       headWordSet.add(getHeadWord(p), p);
4    } // end of for
5    Q' = HW_replacement(headWordSet, Q);
6    tree = generate_collins_parse_tree(Q');
7    collinsNPs = find_noun_phrases(tree); // a set of collins NPs
8    collinsNPs = restore_PN/DP(collinsNPs, headWordSet);
9    NPs = collinsNPs + get_sub_phrases(collinsNPs);
10   for each item t in the NPs set {
11      if (a_sub_phrase_of_PN/DP(t) or partially_overlap_with_PN/DP(t) ) {
12         discard t;
13      } // end of if
14      else if ( google_verification(t)==true) {t is a verified collins phrase;}
15   } // end of for t
16   for x = 2 to (|Q|-1) { // x-item phrase
17      while ( 2_level-x_verified_collins_phrases_overlap(t1, t2) ) {
18         t' = google_pick_one(t1, t2);
19         t' is a final collins phrase;
20         discard the other collins phrase;
21      } // end of while
22      for y = x+1 to (|Q|-1) {
23         if ( level_y_verified_collins_phrase(y) and
24            partially_overlap(y, level_x_verified_collins_phrase) ) {
25            discard  y;
26         } // end of if
27      } //end of for y
28   } //end of for x
29   for each t in the set of final collins phrases {
30      if (t' has two terms) {label t' as SNP;}
31      else { label t' as CNP; }
32   } // end of for t
33 }// end of function
```

Figure 4. Algorithm for SNP/CNP recognition

Generating Collins noun phrases. At line 6, Collins parser analyzes a modified query and returns a parse tree. The phrases, such as noun phrases, verb phrases and adjective phrases, are annotated in the tree. The noun phrases are picked at line 7. They are labeled as the Collins NPs. In line 9, the sub-phrases of these Collins NPs are also collected as Collins NPs. This is to avoid missing some noun phrases that are not recognized by the parser. For example, in Figure 3, given a query "*best compact sedan*", the parse tree on the left is given by Collins parser. It captures the whole query as a noun phrase, but does not capture the embedded SNP "*compact sedan*". The correct parsing is given on the right hand side of Figure 5. The adjective "*compact*" modifies the noun "*sedan*". The adjective "*best*" modifies the noun phrase "*compact sedan*".

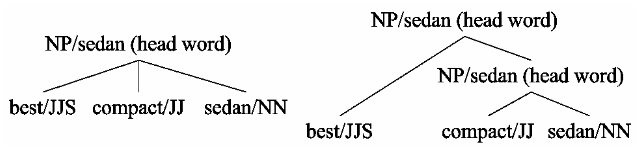

Figure 5: Collins parser fails to generate correct parse tree

The reason why we want to obtain an embedded simple phrase, such as "*compact sedan*" from a complex phrase such as "*best compact sedan*", is that a relevant document may not contain the complex phrase but may contain the simple phrase. Such a document can still have a similarity allocated to the simple phrase, which is part of the similarity allocated to the complex phrase.

Verify Collins phrase. From line 10 to l5, if a Collins phrase *t* is verified by verify_collinsP(t), it becomes a "verified Collins phrase". The idea is that: The noun phrases in the parse tree are grammatically correct. But the parser can not tell if the phrases are meaningful in the real world text. A phrase to be verified must (1) not intersect with a recognized PN/DP, or (2) be a phrase that contains a recognized PN/DP and additional words. For example, "Spider Man tickets" contains a PN "Spider Man" and an additional word "tickets". If a potential noun phrase partially overlaps with a PN/DP, we prefer the PN/DP. For example, the query "*blood pressure leve*l" has two shorter phrases of "*blood pressure*" and "*pressure level*". The former is a DP. So "*pressure level*" is discarded and will not go for verification.

When a two-word phrase *p* is fed to verify_collinsP(t), the function searches *p* in a document collection to examine the existence of *p*. If an instance of *p* is found, such that this instance plus its boundary words is not a sub-phrase of the query, *p* becomes a verified Collins phrase. For example, to verify "*tourist bus*" in "*free tourist bus*", at least one instance of "*tourist bus*" should not have the word "*free*" before it.

If *p* has three or more words, it could be written in various ways. For example, "*colin farrell wallpaper*" can be written as "*wallpaper of colin farrell*". We verify these phrases as follows:

(1) Search the exact *p* in the document set. If *p* is found, and if this instance plus its boundary words is not a sub-phrase of the query, *p* is verified.

(2) Otherwise, we look for a narrow text window in the documents. This window should contain all the content words of *p*. But the ordering of these words in the window can be different from that in *p*. If such a window is found, we label *p* as verified. The more words *p* contains, the wider the text window will be.

In our implementation, a phrase *p* is submitted to Google. The top 20 retrieved documents are used as the document collection.

Solving the phrase-overlapping problem. From 16 to 28, the problem, in which two verified Collins phrases overlap, is resolved. From line 17 to 21, two overlapped phrases having the same number of words are handled. pick_one_phrase(t1, t2) at line 18 counts the occurrences of t1 and t2 respectively as described in Section 2, except that it does not care about the capitalization of the words. The one with a higher count is preferred. From line 22 to 27, the cross-level overlapping problem is solved. We adopt a lower level priority strategy: if two verified

Collins phrases from different levels overlap, the one at the lower level (has fewer words) is preferred, because during the tuning we found this is better than picking the higher level phrase. The cross-level overlapping solving process does not include the original query (|Q|-level) since the original query overlaps with all of its sub-phrases.

From line 29 to 32, a verified Collins phrase is labeled as SNP if it has 2 words. Otherwise it is labeled as a CNP. We use Example 2 as an illustration.

Example 2: *Collins parser labeled the query "sony dvd handycam" as noun phrase. Its sub-phrases "sony dvd" and "dvd handycam" were also added as Collins NPs. All three pass the verification_collinsP(t). "Sony dvd" and "dvd handycam" were examined by pick_one_phrase( ). In the context of "sony dvd handycam", "sony dvd" not followed by "handycam" was found 1 time, while "dvd handycam" not following "sony" was found 12 times. Thus "dvd handycam" was chosen. No cross-level overlap solving was fired. The algorithm stopped.*

5. EXPERIEMENTAL RESULTS

We tune our algorithm using a set of 400 multi-word web queries, randomly selected from a search engine company's web query log, which has more than 170 thousand queries. The algorithm is then tested using another set of web queries from the same query log, and a set of TREC (Text REtrieval Conference) queries. Finally, we apply the recognized phrases from TREC queries to TREC document retrieval tasks, comparing the retrieval effectiveness of the IR system when (1) not utilizing phrases in the queries at all, (2) using phrases recognized by a baseline phrase recognition algorithm, and (3) using the phrases recognized by our algorithm. The results of the noun phrase recognition experiments are reported in recall, precision and f-score [29]. For a phrase type T, the precision (P) is the number of the correctly identified T phrases by our algorithm divided by the total number of the identified T phrases by our algorithm. Recall (R) is the number of the correctly identified T phrases by our algorithm divided by the total number of the T phrases in the golden standard. The F-score is defined as 2PR/(P+R). The results of the document retrieval experiments are reported as Mean Average Precision (MAP) and Geometric Mean Average Precision (GMAP) scores[31].

5.1 Testing Algorithm Using Web Queries

We randomly selected another 500 multi-word queries from the same query log used for selecting the tuning set. These two sets do not overlap. These 500 queries contain 205 2-word queries, 163 3-word queries, 86 4-word queries, 30 5-word queries and 16 6-word queries. Each of three graduate students labeled all of the PNs, DPs, SNPs and CNPs in the queries. Disagreements were solved by majority voting. These labeled phrases were set as the golden standard.

5.1.1 Overall Phrase Recognition Performance

Table 1 shows the noun phrase recognition performance of our algorithm using this 500-web-query set. In the PN row, three major reasons for the errors are: First, the web queries contain many less well-known proper names such as names of people, small companies and organizations. They are not defined in dictionaries. There are also not enough number of instances in the documents for them to be verified. This affects the recall. The second reason is the informal writing of the PNs, such as incomplete name and unofficial names. For example, a query "*zenon z2*" refers to a TV show "*Zenon: the Zequel*". The query used an unofficial name "*z2*". This incorrect title can not be verified. The third reason for the errors in PN is the capitalization of the words in the documents. In the procedure of is_doc_PN(p), we require the content words of a potential PN be capitalized in the documents. In some cases, non-PN phrases also have all of their content words capitalized to emphasize them. They satisify the procedure is_doc_PN(p), becoming the false positive PNs, which affects the precision. For example, "*Return Policy*" is such a phrase because some companies emphasize it as an important issue. The cases involving un-official names and the emphases by capitalizing certain words require further study.

Table 1: Scores of our algorithm on the 500-web-query set

	# in set	Recognized	Correct	Recall	Precision	F-score
PN	263	258	243	0.9240	0.9419	0.9328
DP	102	103	102	1	0.9903	0.9951
SNP	167	183	149	0.8922	0.8142	0.8514
CNP	292	268	252	0.8630	0.9403	0.9000

The DP row has one false positive case. The query is "*young models*" that refers to young persons posing for purpose of art or fashion. Wikipedia has an entry "*young model*" that is about a mathematical model.

In the SNP row, an error type is that is_doc_PN( ) falsely recognizes some SNPs as PNs. This lowers the recall. The "*return policy*" is such an example. Another error type is that some PNs are recognized as SNPs. The third error type is that the pick_one_phrase( ) function made wrong choices. Both the second and the third error types lower the precision.

In the CNP row, the major error type is that some CNPs partially overlap with recognized SNPs, while we adopt a SNP-has-higher-priority strategy, the CNPs are discarded. This affects the recall.

5.1.2 Impact of Individual Tools on the Algorithm

In order to analyze the impact of each individual tool on our algorithm, we test the performances of these tools individually. Tables 2 to 5 show the performance of the individual tools in PN, DP, SNP and CNP recognition tasks respectively. In tables 2 to 5, the "Full" lines refer to the corresponding data in Table 1.

In Table 2, 5 tools are tested in the PN recognition respectively. Minipar got low recall (0.2000). Minipar uses grammatical rules to parse text, which needs context information for correct parsing. But the web queries are too short to provide enough contexts.

Table2. Scores of individual tools on PN recognition task

Tool	Recognized	Correct	Recall	Precision	F-score
Minipar	63	53	0.2000	0.8413	0.3232
WordNet	82	67	0.2548	0.8171	0.3884
Wikipedia	195	180	0.6844	0.9231	0.7860
Doc_set	217	182	0.6868	0.8387	0.7552
Name list	29	26	0.0981	0.8966	0.1769
Full	258	243	0.9240	0.9419	0.9328

The recall of WordNet alone is low (0.2548) because it only recognizes PNs that are defined in its database. Given that many of the PNs do not have entries in WordNet, they are missed.

Wikipedia is an open dictionary. Its open editing architecture makes its data updated with the current affairs of the world.

That's why it got a much higher recall value (0.6844) than the non-open tools of Minipar, WordNet and Name List.

The name list tool got the lowest recall (0.0981) but it is expected, since it only recognizes people names. Its 0.8966 precision demonstrates its effectiveness.

Is_doc_PN(p) alone gets high recall as Wikipedia does, actually the highest (.6868) among the 5 tools. Since we use the document set returned from Google, the documents are also kept up to date.

These 5 tools all got reasonable precisions (0.81 to 0.89). Wikipedia has the highest precision 0.9231 because it has high quality contents. The results show that a single tool usually has a low recall. One single tool is not enough to recognize most of the PNs in the web queries, since the web queries cover very wide topics. Different tools must be used together to achieve desirable result.

Table 3. Scores of individual tools on DP recognition task

Tool	Recognized	Correct	Recall	Precision	F-score
WordNet	62	40	0.3922	0.6452	0.4878
Wikipedia	104	97	0.9510	0.9327	0.9417
Full	103	102	1	0.9903	0.9951

In Table 3, two individual tools are tested in the DP recognition task. The recall of WordNet is low (0.3922) due to the relatively small amount of term definitions it has, comparing to the various topics in the web queries. The precision of WordNet is also low (0.6452). In many cases, WordNet does not recognize a valid PN/DP. But a sub-phrase of this unrecognized PN/DP is still a valid DP. This sub-phrase is recognized, becoming a false positive. For example, "*brookdale community college*" is a PN not recognized by WordNet. But "*community college*" is recognized. Wikipedia alone gets recall of 0.9510 in DP recognition, which is due to its open editing architecture and large data collection. Its precision is 0.9327 because it has the same situation as that of WordNet: a sub-phrase of an unrecognized PN/DP is incorrectly recognized as a DP.

Table 4. Scores of individual tools on SNP recognition task

Tool	Recognized	Correct	Recall	Precision	F-score
Collins0	118	96	0.5749	0.8136	0.6737
Collins1	243	157	0.9401	0.6461	0.7659
Collins2	238	157	0.9401	0.6597	0.7753
Full	183	149	0.8922	0.8142	0.8514

Collins0: Collins phrase (baseline)
Collins1: Collins 0 + sub-phrase
Collins2: Collins 1 + verify_collinsP
Full: Collins 2 + overlap resolving

Table 4 shows the performances of the individual tools on SNP recognition. The Collins0 row uses Collins parser alone as a baseline. The phrases labeled directly by the parser are the results of Collins0. The sub-phrases of these directly labeled Collins phrases are not included (Line 9 Figure 4). The verify_collinsP(p) (Line 14 Figure 4) and the overlapping solving technique (Line 16-28 Figure 4) are not applied. We only remove the phrases that overlap with recognized PN/DPs. The result shows the effectiveness of the Collins parser alone. The 0.8136 precision is acceptable. The 0.5749 recall is low. This shows that the Collins parser alone is not enough for SNP recognition.

The Collins1 is Collins0 plus using the sub-phrases of the Collins phrases in Collins0. The direct output of the parser misses many correct phrases. We add the sub-phrases of these Collins phrases as additional SNP candidates. This will bring in many incorrect phrases so that the precision could be harmed. But we want to find if these additional sub-phrases can improve the recall. The incorrect and redundant phrases will be removed in "Collins2" and "full" rows. We see the recall of "Collins1" increases substantially from 0.5749 to 0.9401. The additional sub-phrases do work. Unfortunately, the precision drops from 0.8136 to 0.6461. But the f-score is still improved from 0.6737 to 0.7659.

Collins2 is Collins1 plus verify_collinsP( ), which tests if a phrase is actually used in the real world. This verification removes 5 (243 to 238) incorrect phrases. The precision increases a little. It gets the same number of correct phrases so the recall is not changed.

"Full" is the full algorithm configuration. It is the Collins2 plus the overlapping problem solving step. This step aims to further remove the incorrect phrases introduced from the parser and the sub-phrases. Comparing to Collins2, the precision increases from 0.6597 to 0.8142, which is the highest precision among the 4 configurations. The recall drops to a still acceptable degree of 0.8922 from 0.9401. The 0.8514 f-score is the highest among the four. Thus the overlapping solving technique is necessary.

Table 5. Scores of individual tools on CNP recognition task

Tool	Recognized	Correct	Recall	Precision	F-score
Collins0	257	238	0.8151	0.9261	0.8670
Collins1	374	286	0.9795	0.7647	0.8589
Collins2	359	273	0.9349	0.7604	0.8387
Full	268	252	0.8630	0.9403	0.9000

Table 5 shows the CNP recognition results of the 4 configurations defined in Table 4. Collins0 is still the baseline. In Collins1, we still see that the additional sub-phrases help increase the recall (0.8151 to 0.9795) but harm the precision (0.9261 down to 0.7647). In Collins2, the verify_collinsP( ) does not help the performance but even decreases it a little. The full configuration greatly improves the precision (0.7604 to 0.9403) at a small cost of the recall (0.9349 down to 0.8630). And the full configuration has the highest f-score among the four. The behaviors of CNP recognition are the same as those in SNP recognition.

5.2 Testing Algorithm Using TREC Queries

We also use a set of TREC queries to test our algorithm. There are 249 queries from the ad-hoc tracks of TREC-6, 7, 8, and the robust tracks of TREC-12 and 13, 238 of which are multi-word queries. These 238 queries contain 70 2-word queries, 143 3-word queries, 23 4-word queries and 2 5-word queries. Three graduate students double-checked and labeled the PNs, DPs, SNPs and CNPs in these 238 queries as the golden standard. These TREC queries are not new. Some TREC related documents, which describe these queries, can be found in the top retrieved documents from Google when searching these queries. These documents should not be used to prove the existence of the phrases. So when we use Google to collect documents, we set the restriction that the returned documents must not contain the terms such as "TREC", "query" and "phrase". This is done by adding "-TREC", "-query" and "-phrase" as query restrictions when searching in Google.

The overall performance of our system on the TREC query set is reported in Table 6. The only error in the PN row is a falsely recognized PN from is_doc_PN(p). The only error in the DPs is due to a query "*food and drug laws*", where the golden standard indicates "*food and drug*" and the entire query is a CNP. Our system recognizes "*drug laws*" as a DP. The errors in the SNP row are mainly caused by the incorrect pick_one_phrase( ) results. In the CNP row, some CNPs are missed because they can not be verified in the phrase_verification(p).

Table 6. Scores of our algorithm on the 238-TREC-query set

	# in set	Recognized	Correct	Recall	Precision	F-score
PN	32	33	32	1	0.9697	0.9846
DP	110	111	110	1	0.9910	0.9955
SNP	99	102	90	0.9091	0.8824	0.8955
CNP	159	146	146	0.9182	1	0.9574

We tested the performance of individual tools as we did in Section 4.1. The results are shown in Table 7 through Table 10.

Table 7. Scores of individual tools on PN recognition task

Tool	Recognized	Correct	Recall	Precision	F-score
Minipar	3	2	0.0625	0.6667	0.1143
WordNet	15	14	0.4375	0.9333	0.5957
Wikipedia	32	32	1	1	1
Doc_set	32	18	0.5625	0.5625	0.5625
Name list	0	0	0	0	0
Full	33	32	1	0.9697	0.9846

In Table 7, the "name list" tool did not recognize anything, because there are 3 people's names in the PNs, all of which are foreign names. None of the "first name, last name" is found by is_name( ). Minipar only finds 2 correct PNs. This again shows that lacking context greatly affects its performance. WordNet got high precision but low recall. This is similar to its performance in Table 2, because of limited number of definitions. Is_doc_PN( )'s performance in the TREC data set is worse than its performance in the web data set as shown in Table 2. A major error type is that a PN is not recognized but its sub-phrase is incorrectly recognized as PN. Wikipedia had a perfect score because the PNs in TREC queries are rather well known.

Table 8. Scores of individual tools on DP recognition task

Tool	Recognized	Correct	Recall	Precision	F-score
WordNet	59	50	0.4545	0.8475	0.5917
Wikipedia	109	108	0.9818	0.9908	0.9863
Full	111	110	1	0.9910	0.9955

In Table 8, two individual components are used to recognize DP respectively. WordNet's low recall value means that its performance is still affected by the limited entries. Wikipedia's performance is good and stable as it does in the PN recognition experiment.

Table 9. Scores of individual tools on SNP recognition task

Tool	Recognized	Correct	Recall	Precision	F-score
Collins0	55	49	0.4949	0.8909	0.6364
Collins1	156	98	0.9899	0.6282	0.7686
Collins2	155	98	0.9899	0.6323	0.7717
Full	102	90	0.9091	0.8824	0.8955

Table 9 shows the performances of the individual tools on SNP recognition in the TREC query set. The four configurations are the same as those defined in Table 4. The pattern of the performances change is similar to that in Table 4. The pure Collins parser has acceptable precision but low recall in Collins0 row. Additional sub-phrases boost the recall but also damage the precision in Collins1. Verify_collinsP( ) improves precision a little in Collins2. At last, the full algorithm increases the precision at a small cost of the recall. The full algorithm still has the best performance.

Table 10. Scores of individual tools on CNP recognition task

Tool	Recognized	Correct	Recall	Precision	F-score
Collins0	155	152	0.9560	0.9806	0.9682
Collins1	164	154	0.9686	0.9390	0.9536
Collins2	156	146	0.9182	0.9359	0.9270
Full	146	146	0.9182	1	0.9574

Table 10 shows the CNP recognition results using the TREC query set. The 0.9560 recall in baseline Collins0 is very high, because the TREC queries have simpler grammar structures. They mainly consist of nouns, while the web queries contain many verbs and prepositions. Other than this, the performances of the four configurations still follow the same pattern as those in Table 5. The full configuration obtains 0.9182 recall and 100% precision.

5.3 Utilizing Noun Phrases for IR

This experiment is to test whether obtaining more correct phrases yields higher information retrieval (IR) effectiveness. We conducted three document retrieval experiments, comparing the retrieval results using phrases recognized by our algorithm in Section 4.2, to those recognized by a baseline system, and to not using phrase at all. We use the IR system by Liu [20]. This system allows both phrases and single terms in the query. The similarity between the query and a document is represented as a pair of (phrase-similarity, term-similarity). The phrase-similarity of a document is defined as the sum of the idf (inverse document frequency) weights of the phrases in common between the document and the query. If a document does not have the recognized phrase, its phrase-sim is 0. The term-similarity is the usual term similarity between the query and the document, which is computed by using Okapi formula [26]. Each query term appeared in the document contributes to the term-similarity, no matter it is in a query phrase or not. The phrase-similarity has high priority than the term-similarity. Given a query, the retrieved documents are ranked in descending order of their phrase similarity values. When documents have the identical phrase similarity value, they are ranked in descending order of their term similarities. So given a query, two documents D1 and D2 have similarities (x1, y1) and (x2, y2), respectively. D1 will be ranked higher than D2 if (1) x1>x2, or (2) x1=x2 and y1>y2.

The 249 TREC queries are from 6 resources, the ad hoc tracks of TREC 6, 7, 8 and the robust tracks of TREC 12, 13, 14. TREC 14 queries are executed on the AQUAINT data collection [31]; other 5 sets are executed on the TREC disks 4 and 5 except the Congressional Records portion [30].

We simplify our phrase recognition algorithm to a weaker "single-tool algorithm". It serves as a baseline phrase recognition algorithm. It utilizes just one tool to recognize one type of phrases, while our full algorithm uses multiple tools for each phrase type. In this single-tool algorithm, Wikipedia alone recognizes the PNs and DPs, because it yields the best results in the PN/DP single-tool experiments. The Collins parser alone recognizes SNPs and CNPs, because it is the fundamental component in the SNP/CNP part of our algorithm. The intuition is that our algorithm has better phrase recognition capability than this baseline. Better phrases should help retrieval system produce higher retrieval effectiveness. From Tables 7, 8, 9 and 10 we can see that this

Table 11. MAP and GMAP scores of IR experiments using different phrase recognition algorithms

Phrase	TREC 6		TREC 7		TREC 8		TREC12		TREC13		TREC14	
	MAP	GMAP	MAP	GMAP	MAP	GMAP	MAP	GMAP	MAP	GMAP	MAP	GMAP
1 No phrase	0.1950	0.1069	0.2246	0.1262	0.2388	0.1577	0.3269	0.2248	0.3341	0.2188	0.2451	0.1712
2 Single-tool	0.3003	0.1564	0.2998	0.1962	0.3180	0.2281	0.4148	0.3401	0.3912	0.2529	0.3286	0.2609
Increase 2 over 1	54%	46.30%	33.48%	55.47%	33.17%	44.64%	26.90%	51.29%	17.09%	15.58%	34.07%	52.39%
3 Our algorithm	0.3293	0.1981	0.3112	0.2160	0.3231	0.2334	0.4291	0.3538	0.4279	0.3036	0.3508	0.2931
Increase 3 over 2	9.66%	26.67%	3.80%	10.09%	1.60%	2.32%	3.45%	4.03%	9.38%	20.05%	6.76%	12.34%

single-tool baseline algorithm has almost the same PN/DP/CNP recognition ability as the full algorithm, and substantially worse SNP recognition ability.

We conduct three experiments.

(1) Feed the queries to the IR system, without recognizing any phrase. The output of the IR system should represent the effectiveness of the system when using only individual terms. Since the phrase similarity is always 0, the documents are ranked in descending order of their term similarities.

(2) Recognize the phrases in the queries by using the "single-tool" baseline algorithm. Then feed the queries and the recognized phrases together to the IR system. The output should show the effect done by the recognized phrases when comparing to the output of (1).

(3) Recognize the phrases in the queries by using our complete phrase recognition algorithm. Then feed the queries and the recognized phrases together to the IR system. The output should show the effect of recognizing phrases with higher qualities, when comparing to the result of (2).

There are 11 single-term queries in the 249 queries. Their retrieval results are also included in the final results. So the difference between (1) and (2), and that between (2) and (3) are just caused by the differences of the phrases.

The retrieval results are presented as mean average precision (MAP) [30] and geometric mean average precision (GMAP) [31] in Table 11. Comparing the scores of line 1 and 2 shows that all of the 6 query sets, when using the phrases from the baseline algorithm, get much higher scores than not using phrases at all (MAP gains from 17% to 54%, GMAP gains from 15% to 55%). This shows that the document retrieval, with the recognition of the phrases, actually improves over just using single terms. Table 11 also shows that our full phrase recognition algorithm helps the retrieval achieve higher scores than the baseline phrase recognition algorithm does. The improvements are from 1.6% to 9.6% in MAP and 2.3% to over 26% in GMAP. This demonstrates that better noun phrase recognition yields better retrieval results.

Table 12. Compare our results to the highest TREC 13 MAP

System	Old topic set	New topic set	Combined
TREC 13	0.317	0.401	0.333
Our algorithm	0.348	0.428	0.364
Improvement	9.78%	6.73%	9.31%

In TREC 13 [30], these 249 queries are used in the robust track. 200 of them from TREC 6, 7, 8 and 12 are called the "old topic set". The other 49 are called the "new topic set". In [30], the best MAP of the "old topic set" is 0.317. The best MAP of the "new topic set" is 0.401. The combined score is 0.333. We calculated the MAP scores for the old, new and the combined set for our algorithm from Table 11. Table 12 shows the comparison between our scores and the TREC 13 scores (Table 12 uses 3 digits because TREC 13 robust track scores were reported in this format [30]). The improvements of our scores over the best scores in these topic sets are 9.78%, 6.73% and 9.31% respectively. Furthermore, the 0.2931 MAP and the 0.3508 GMAP of the TREC 14 query set (Table 11) are 5.7% and 26% higher than the best corresponding scores reported in [31]. So our algorithm helps the IR system achieve higher scores than the best officially reported scores of the same query set and the document collection.

5.4 Comparing to a Related Work

Lima et al [17] studied the proper noun and phrase recognition problem. They reported 0.8786 precision and 0.9010 grammar coverage ([17] used "grammar coverage", which is an upper bound of the recall) on 100 company names; 0.7770 precision and 0.8000 grammar coverage on 100 person names; 0.7983 to 0.8200 precision and 0.9160 to 0.9560 grammar coverage on 200 short queries that have 1.59 words on the average, with an upper-bound of f-score at 0.8827 (denoted by Q1); and 0.8049 to 0.8139 precision and 0.7800 to 0.8520 grammar coverage on 200 queries that have at least 3 words with a 3.59 word average length, with an upper-bound of f-score at 0.8325 (denoted by Q2). To compare our result to theirs, we aggregate their company and person names together as a PN set, and compare it to the PN row of Table 1. We aggregate the 2-word DPs and the SNPs in Table 1 together (0.9302 recall, 0.8727 precision, 0.9006 f-score) to compare to their Q1 set. We aggregate the 3-or-more-word DPs (11 correct) and the CNPs in Table 1 together (0.8680 recall, 0.9427 precision, 0.9038 f-score) to compare to their Q2 set. The results are shown in Table 13..

Table 13. Comparison between Lima et al. and us in F-Score

Phrase Type	Lima et at.	Us
PN	0.8395	0.9328
2-word	0.8827	0.9006
3-or-more-word	0.8325	0.9038

6. CONCLUSIONS

In this paper, noun phrases are classified into four types. We provide an algorithm that recognizes them. The algorithm is tested on a web query set and TREC query titles. High accuracies of recognition are obtained. Utilizing an up-to-date dictionary for recognizing proper names and well-defined phrase recognition seems to be a good method. Looking for instances in a document set is also good for less well-known proper names. Natural language parser and finding phrase instances in documents are good for recognizing SNP and CNP. Our document retrieval experiments also show that recognizing and utilizing phrases in the queries can substantially improve retrieval effectiveness; furthermore, the quality of the phrases has a positive impact on retrieval effectiveness.

7. ACKNOWLEDGMENTS

The authors thank the reviewers for their helpful comments. This work is supported in part by NSF grants IIS-0738727 and IIS-0738652, and by an AOL research grant. The views of this paper are those of the authors, and do not represent those of NSF or AOL.

8. REFERENCES

[1] A Arampatzis, T Tsoris, C Koster, and T van der Weide. Phrase-based Information Retrieval. Information Processing & Management, 34(6):693-707. 1998.

[2] D Bikel, S Miller, R Schwartz and R Weischedel. Nymble: a High-Performance Learning Name-finder. In *proc. of the Conf. on Applied NLP*. 1997.

[3] Eric Brill. Transformation-Based Error-Driven Learning and Natural Language Processing: A Case Study in Part of Speech Tagging. Computational Linguistics. 1995

[4] Bruce Croft, H Turtle, and D Lewis. The use of phrases and structured queries in information retrieval. In *Proc. of SIGIR*. 1991.

[5] J Callan and T Mitamura. Knowledge-based extraction of named entities. In *Proc. of CIKM*. 2002.

[6] Guihong Cao, Jian-Yun Nie, Jing Bai. Integrating Word Relationships into Language Models. In *Proc. of SIGIR*. 2005.

[7] Nancy Chinchor. Overview of MUC-7. In Proc. of MUC. 1998.

[8] Ken Chow, Robert Luk, Kam-Fai Wong and Kui-Lam. Kwok: Hybrid Term Indexing for Weighted Boolean and Vector Space Models. Int. J. Comput. Proc. Oriental Lang. 14(2): 133-151, 2001.

[9] M. Collins, Head-driven statistical models for natural language parsing. PhD thesis, U. of Pennsylvania, 1999.

[10] David Evans and Chengxiang Zhai. Noun-Phrase Analysis in Unrestricted Text for Information Retrieval. In *Proc. of ACL*. 1996

[11] Glossary of linguistic terms, by E Loos, S Anderson, D Day, P Jordan, and D Wingate (editors). SIL International. 2003

[12] C. Fellbaum. WordNet, An electronic Lexical Database. The MIT Press, 1998.

[13] Radu Florian, Abe Ittycheriah, Hongyan Jing, and Tong Zhang. Named Entity Recognition through Classifier Combination. In *Proc. of CoNLL*. 2003.

[14] Google: http://www.Google.com/apis/

[15] David Grossman and Ophir Frieder. Ad Hoc Information Retrieval: Algorithms and Heuristics. Kluwer. 1998.

[16] S Jones and M Staveley. Phrasier: A System for Interactive Document Retrieval Using Keyphrases. In *Proc. of SIGIR*. 1999.

[17] E Lima and J Pedersen. Phrase Recognition and Expansion for Short, Precision-biased Queries based on a Query log. In *Proc. of 22nd ACM SIGIR*. 1999

[18] D Lin. PRINCIPAR - An Efficient, broad-coverage, principle-based parser. In *Proc. of COLING*. 1994.

[19] D Lin. Using collocation statistics in information extraction. In *Proc. of Message Understanding Conference*. 1998.

[20] S Liu, F Liu, C Yu and W Meng. An effective approach to document retrieval via utilizing Wordnet and recognizing phrases. In *Proc. of SIGIR*. 2004.

[21] I Mani and R MacMillan. Identifying Unknown Proper Names in Newswire Text, In Corpus Processing for Lexical Acquisition, MIT Press. 1995.

[22] Christopher Manning and Hinrich Schütze, Foundations of statistical natural language processing, MIT Press. 1999

[23] M Marcus, G Kim, M Marcinkiewicz, R MacIntyre, A Bies, M Ferguson, K Katz and B Schasberger. The Penn Treebank: Annotating Predicate Argument Structure. In *Proc. of the Human Language Technology Workshop*. 1994.

[24] R Mihalcea and D Moldovan. An Automatic Method for Generating Sense Tagged Corpora. In *Proc. of AAAI*. 1999.

[25] G Miller. WordNet: An On-line Lexical Database, Special Issue, International Journal of Lexicography. 1990.

[26] S Robertson and S Walker. Okapi/Keenbow at TREC-8. In *Proc. of TREC*. 1999.

[27] Egidio Terra and Charles Clarke. Frequency Estimates for Statistical Word Similarity Measures. HLT/NAACL. 2003.

[28] University of Glasgow, LILT project, www.arts.gla.ac.uk/SESLL/EngLang/LILT/frameset.htm

[29] C. van Rijsbergen. Information Retrieval. Butterworth, 1979.

[30] E Voorhees. Overview of the TREC 2004 Robust Retrieval Track. In *Proc. of the 13th TREC*. 2004.

[31] E Voorhees. Overview of the TREC 2005 Robust Retrieval Track. In *Proc. of the 14th TREC*. 2005.

[32] Wikipedia: http://en.wikipedia.org

[33] Wei Zhou, Clement Yu, Neil Smalheiser, Vetle Torvik and Hong Jie. Knowledge-intensive Conceptual Retrieval and Passage Extraction of Biomedical Literature. In Proc. of 30th SIGIR. 2007.