

Short Query Linguistic Expansion Techniques: Palliating One-Word Queries by Providing Intermediate Structure to Text

Gregory Grefenstette

Rank Xerox Research Centre, 38240 Meylan, France
Gregory.Grefenstette@grenoble.rsrc.xerox.com

Abstract. The usual approach to finding information on the WWW via existing Web browsers is to use a one or two word query. Browsers return a number of documents containing these words, and the user examines those documents, or their abstracts, sees how the word or words in their query are being used and alters their initial query accordingly. This contrasts markedly with the Information Retrieval models explored by researchers over the past thirty-five years. These models were designed for longer queries and do not provide an adequate response to the user needs. On the other hand, recent advances in natural language processing permit the extraction of typed information that is axed on one or two words. We review a selection of this typed information and describe how it could be used to present an intermediate structure for the user fitting between their short queries and the documents found in a heterogeneous text collection such as the WWW.

1 The Gap Between Practice and Research

There is a well-known gap between the characteristics of the typical user studied by Information Retrieval (IR) researchers and the typical information seeker on the Web. Most real searchers use only a few words[33], but the bulk of Information Retrieval research since the 1960's has been based on longer queries.

There are two primary reasons for this gap between practice and research. The first is that most of the research in information retrieval was funded by government agencies, and it is standard practice in government intelligence agencies to express information needs as long, detailed descriptions of the subjects that interest them. Government agents will willingly research and list all the ramifications touching a topic. These descriptions generate long queries that can be used for filtering documents.

Secondly, in order to prove that one information retrieval technique works better than another, also a necessary task to secure research funding, one must have objective bases for judging when a system has gotten the right answer. Cleverdon *at al.* [12] helped devised the first information retrieval testbeds, called

the Cranfield collections¹, which are composed of a set of documents, a set of natural language queries on these documents, and a relevance mapping, listing which documents are relevant to which queries.

In order that the relevance judgments represent some objective standard against which to measure, the queries must be relatively explicit about what is being searched for, which means they must be verbose. A short one or two word query could not be used in such experiments without the relevance judgments either becoming subjective (the judge reinterpreting what the searcher was looking for), or being reduced to a simple judgment of when a pattern was found in a text (something any file system search tool provides). These two reasons, then, that government funding agencies used long queries, and that long queries are necessary in order to objectively measure retrieval accuracy, led to the creation of a number of information retrieval testbeds, many of which can be found on the ftp server at Cornell University, and to the structure of queries tested in the current series of Text Retrieval Conferences (TREC) sponsored by the National Institute of Standards since 1992[26].

1.1 The Vector Space Model and Query Length

As a result of the above, the vector space model of information retrieval has been favored as an underlying IR implementation in research. This model[37] considers each word in the language as an axis in a highly dimensional space. Each dimension, each axis, in this space corresponds to a unique word found in the language. Each document and query are plotted in this space, with the distance along each axis being dependent upon the number of times the word appears in the document or the query. This distance is normalized using the document length[6]. To find the closest documents to a query, one calculates the cosine of the angle between the query point and all the document points, taking the points found closest as the relevant documents. This geometrically based theory is very attractive, theoretically. But this model works best when many axes (many query words) are being used to circumscribe the space of relevant documents, and when the model can exploit the repetition of certain important words within the query in order to give greater weight, and thus focus the query more accurately.

For short queries, the vector space model is inadequate. This inadequacy is recognized by traditional Information Retrieval researchers. Kwok[32] remarks that short queries lack two elements that make vector space retrieval over long queries successful: the variety of terms used to match the query to documents, and the indication of the importance of the words in the query. This latter point, which most affects precision according to Kwok, is automatically estimated in traditional information retrieval systems from the frequency with which the word

¹ One of these Cranfield collections is still publically available, at <ftp://ftp.cs.cornell.edu/pub/smart/cran>. This ftp server also contains the Information retrieval testbeds: *adi*, *cacm*, *cisi*, *med*, and *npl*; as well as a research version of the information retrieval system SMART.

appears in the query. Of course, in short queries, each word will usually only appear once and frequency can not give us any clues of the terms importance.

In order to show the interest of giving more weight to the important words in a short query, Kwok first performed a series of experiments in which a human user manually picked the important words in a query, presumably something that one might ask of Web Browser user, and then these words were repeated twice in the query submitted to the IR system. This simple technique improved the average precision of the short queries by 10%, a significant gain in precision in information retrieval circles.

Kwok then devised an automatic method of weighting query words by replacing their initial weight of 1 by a weight representing the average number of times a word appears in a document, when it appears, divided by a function of the absolute frequency of the word in the whole document database. This weighting gives rare words which appear fairly often in the documents they appear in more weight than common words or rare words that appear sporadically. He showed[32] that this automatic weighting performed as well as having a user manually choose the most important words in a query.

1.2 Actual Web Use

Most queries produced by users on WWW browsers are only a few words long. The user is essentially casting out a fishing line into an enormous ocean of pages to see what bites. The words that are used in the query are weighted based on their relative frequency² in the pages indexed and returned by the browser. No one can say how good a browser is. The user is happy to get any reaction from the system, ignorant of what they might be missing, a common situation with real use of information retrieval systems[2]. It is difficult to judge goodness because, not only are there no objective testbeds such as those created in the information retrieval community for long queries, but given the typical short query, one can not objectively say whether a given Web page containing that word or those words is relevant to a given user, since it is not clear what the user is really looking for. The users may not know themselves. This is why the term “browser” fits these information retrieval tools so well.

These browsers return more precise results when either more words are given or some positional restrictions are specified on the query words. Some search engines allow the user to specify a positional arrangement so that the query words appear “near” each other, or in a certain order. The burden of predicting how the words are used is left nonetheless on the user, the browser considering the whole space of the Web as a flat index structure. As one response to this flat structure, a few search engines, such as Yahoo!³, provide a hierarchical structure

² This method of ranking pages has led to the “word spamming problem” in which a malicious HTML programmer fills in a WWW page with a large number of occurrences of a hidden key word (such as a competitor’s company name) so that the programmer’s page is shown first to the unsuspecting user.

³ <http://www.yahoo.com>

into which pages are inserted, but this insertion is done manually which limits the number of pages that can be indexed.

1.3 Feedback from Short Query Results

A method for improving precision of a query by including frequently occurring terms from an initial retrieved document set can be implemented in the DIALOG information retrieval system using the RANK command⁴, first introduced in 1994. Once a user performs a search, RANK will sort all the indexed words appearing in the result set and display them to the user, who can then use them to make their query more precise. This feedback step can happen, after performing a search, but before the user examines the first set of documents returned.

Recently, François Bourdoncle of the Ecole des Mines de Paris has developed a somewhat similar version of this intermediate approach between query and documents, called LiveTopics, for the Altavista⁵ browser. LiveTopics shows, in either a graph or a list, the words which appear in the neighborhood of the original query words in the retrieved documents⁶. One can click on these words which are then added to the query to improve the precision of the original query.

The above methods are a first response to the fact that a short query can often pick up many different aspects of meaning in the document collection containing the words in the query. A more sophisticated response is given in Hearst *et al.*, a paper that describes a system in which the documents returned from an initial query are clustered along these strands of meaning. This clustering is automatically performed on the documents that contain the short query terms by a technique developed in a collection browsing system called Scatter-Gather[14]. This linear-time technique works by splitting the returned documents into a fixed set of clusters, each cluster is then represented by its most representative terms, which are displayed to the user along with the first few titles in each cluster. The user can then easily see which of the groupings correspond most closely to the information that they were looking for and use the whole cluster as a new request via a traditional relevance feedback technique[25].

For example, Hearst *et al.* [27] show that, given the one-word query *star* over the text of a general encyclopedia, the 400 documents returned are automatically clustered by the Scatter-Gather technique into groups characterized by words such as **Cluster 2:** *flag, rug, weave, carpet, pattern, stripe, force*; **Cluster 3:** *game, player, team, league, ball, football, professional*; **Cluster 4:** *energy, hydrogen, radiation, planet, temperature, gas*, etc. These clusters show different aspects of the meaning of the polysemous word *star*, as a pattern, as a person, as a physical phenomena, and so on. This technique, which retrieves and then

⁴ <http://www.dialog.com/dialog/publications/rank-qrc.html>

⁵ <http://altavista.digital.com>

⁶ The same idea was described by Doyle [17] in the early 1960's as a realization of her Semantic Road Maps.

clusters documents, gives a general view of the words used in the same documents as short query words, and provides an alternative dynamic approach to using the more fine-grained pre-stored structures proposed below.

2 Information Extraction and Information Structuring

It is a very simple matter to say whether or not a certain string appears in text stored on a computer. In essence, this is what Web browsers do. They crawl[10] through the Web, retrieve pages, tokenize the text, and store each token with a pointer to its Web page. Given a one-word query, the browser returns the pages pointed to by that string. The pages are usually ranked, so that if the string appears in the <title> field of the HTML page, the page is ranked higher. The page is also ranked higher in function of the frequency of the word in the page. Most browsers will match on a string without regard to upper or lower case, maybe giving a slightly higher weight to the pages in which the case of the search word and the token in the page match.

But the browsers could be finding out much more about a word than its simple presence in the page. The word appears in a context that determines the word meaning. Current natural language processing allows much of this structure to be recognized, and stored, as a page is being retrieved and indexed by the Web crawler. If this structure is stored at index time, then an alternative route from a one or two-word query could pass through this static previously-recognized structure, before Web pages are presented to the user. Rather than delving straight into the documents as is the current case, the user would be able to see what abstract linguistic structures a word appears in. These more precise structures can then be clicked to reach the final Web-based documents.

Given the structures in which a word is found, its meaning can often be deduced. For example, “dog” can mean many things. But one easily sees its nuances of meaning when given such phrases as “dog biscuit” or “rabid dog” or “walk a dog” or even “hot dog.”

These and other types of linguistically-motivated structures can be automatically extracted from text using present-day natural language technology. Not only for English, but for many other languages, there exists robust natural language technology[19] for performing basic processing of raw text. In the remaining sections we will overview these natural language processing technologies, covering how they work, and what they can extract. Then in section 3 we will show how this information might be succinctly presented to a user in response to their short, one word query.

2.1 Preliminary Natural Language Processing

Language Identification In the context of documents brought back from a Web crawler, one of the first tasks after removing HTML markup and before performing any language-specific processing is, of course, determining what is the original language of the text to be treated, if this is not mentioned in the

Danish	Dutch	English	French	German	Italian	Norweg.	Portug.	Spanish	Swedish
i	de	the	de	der	di	og	de	de	och
af	van	and	la	die	e	det	a	la	i
og	het	to	le	und	il	han	que	que	att
at	een	of	à	den	che	i	o	el	som
til	en	a	et	in	la	er	e	en	en
for	in	in	des	von	a	på	do	y	är

Fig. 1. Most frequent short tokens per language derived from the ECI Multilingual Corpus.

HTML metalanguage which it rarely is. As an example of this problem, the one-word query “parole” on Altavista in the Spring of 1997 gave back six French, three Italian, eleven English and one Japanese page among the first twenty document returned. An automatic Web crawler that brings back these pages for indexing must be able to identify which language processing tools to apply to which pages.

Fortunately, language identification[23] is a rather simple task. One could attempt to rely on Internet domain names, but many domains, such as **.ca** for Canada or **.be** for Belgium, cover pages written in many languages. It is better to rely on characteristic short words of the common languages (Figure 1) or characteristic character sequences (Figure 2), which can be calculated from any large body of text in a given language. Automatic language identifiers have been available on the Web since 1996⁷.

Danish	Dutch	English	French	German	Italian	Norweg.	Portug.	Spanish	Swedish
er_	en_	_th	_de	en_	_di	et_	_de	_de	en_
en_	de_	he_	es_	er_	to_	en_	de_	de_	er_
for	_de	the	de_	_de	_de	er_	os_	os_	et_
et_	et_	nd_	ent	der	di_	_de	do_	_la	tt_
ing	an_	ed_	nt_	ie_	_co	_ha	que	el_	_de
_fo	n_d	_an	_le	ich	la_	an_	_qu	la_	ar_

Fig. 2. Most frequent trigrams per language derived from the ECI Multilingual Corpus.

Tokenization Once the language is identified, language specific tokenizers can be applied. A tokenizer is a device that segments an input stream into an ordered sequence of *tokens*, each token corresponding to an inflected word form, a number, a punctuation mark, or other kind of unit to be passed on to subsequent natural language processing.

Most sequences of uninterrupted alphabetic characters compose a token in most languages, but the use of separators inside words varies from language to

⁷ <http://www.rsrc.xerox.com/research/mltt/Tools/identifier.html>

language. For example, the sequence `l'amour` might split into two tokens in French, the article `l'` and the noun `amour`; while `won't` might be considered as a single token in English, as it is in the Brown Corpus, a 1 million hand-tagged sample corpus heavily used in natural language processing. On the other hand, in certain language-dependent instances, a sequence of words (e.g., `hot dog`, `ein bisschen`, `a priori`, e. g., `parce que`, `a fuera de`, `in order to`) may be considered as a single token for further linguistic treatment.

Though proper tokenization is not necessary when words are taken as isolated indexing units, for natural language processing of text it is important to do it right. In order to recognize structure in text, it is necessary to recognize sentence boundaries[20], since parser tools work on entire sentences as input. It is also necessary to correctly recognize each word, since information used by the parser, such as whether the word is a noun, a verb, an adjective, a number, etc., has to be correctly associated with each token during morphological analysis.

One approach[38] to tokenization is to provide a cascade of language-dependent finite-state transducing tokenizers. These tokenizers segment text by introducing a token boundary (usually a new-line) into the output stream. The cascade is composed of a *basic tokenizer* which segments any sequence of input characters into simple tokens (i.e. no multiword units) and one or several *multiword staplers* which identify multiwords and group them together as single units. The development and implementation of a finite-state longest match operator [31] has made this development both practical and possible.

Morphological Analysis Once the text of the WWW page is tokenized, each token has to be identified for subsequent language processing. In order to recognize the syntactic structures that word participates in, it is necessary to know what grammatical roles a word can play. For example, the token “watches” can be a plural noun or the present singular tense of a verb. On the other hand, the token “witches” can only be a plural noun. Information about what parts-of-speech a word can play in a language is contained in a lexicon⁸ for that language[7,8]. A program which uses this lexicon to classify tokens is called a morphological analyzer⁹[35].

Morphological analysis usually returns, for each analyzed token, the parts of speech that token can play, as well as the normalized dictionary entry form, the *lemma* of the word. The proper lemma for each word is chosen after the part-

⁸ See samples of lexicons for English, French, Spanish and German at <http://www.lpl.univ-aix.fr/projects/multext/MUL5.html>. The MULTEX project is working to make a wide variety of lexicons and natural language processing tools available for the research community. EAGLES (<http://www.ilc.pi.cnr.it/EAGLES/home.html>), a partner project, is concerned with normalizing lexicon formats.

⁹ Interactive morphological analyzers for English, Dutch, French, German, Italian, Portuguese, and Spanish, as well as part-of-speech taggers for these languages, are available at <http://www.rsrc.xerox.com/research/mltt/toolhome.html>.

of-speech the word plays is disambiguated, see the next section. Morphological analysis can be contrasted with the traditional information retrieval of *stemming*.

A stemmer, such as the Porter¹⁰ stemmer[34], contains a set of suffixes to be stripped from words. The suffix is removed under certain conditions, such as being preceded by a consonant. When the suffix is removed, another suffix may be added, for example, removing “ies” and adding “y.” The purpose of a stemmer is, using the most productive derivational mechanisms of a given language, to reduce all the words that derive from a same source to a unique stem, so that if a user in an information retrieval setting uses one of the words, they will get answers concerning any of them. Most stemmers contain a set of exceptions so that words like “better” are not stemmed to “bet.” And as the list of exception and the number of transformation rules grow, stemmers approximate linguistically motivated morphological analyzers better and better[28] at recognizing variants of words, but stemmers do not provide the part-of-speech information needed for subsequent natural language processing.

Input Phrase: *pending antitrust investigations involving an alleged violation*

Stemmed Version: *pend antitrust investig involut an alleg violat*

Lemmatized version: *pending antitrust investigation involve an allege violation*

Fig. 3. Difference between normalization of words from an inflectional lemmatization and from Porter stemming of an English phrase.

Part of Speech Tagging In order to recognize linguistic structure, a parser must know or decide what role each word is playing in a given sentence. It is common to use a part-of-speech tagger to eliminate most or all of these ambiguities before parsing. For example, “watches” is ambiguous as we said above, but “the watches” is not, since an article is much more likely to be followed by a noun than a verb. A standard practice in natural language processing is to use these probabilities to eliminate unlikely readings for words.

The most common method of developing a probabilistic part-of-speech tagger is to first create a manually tagged corpus, i.e. for a certain amount of text, a human decides what part-of-speech tag should be assigned to each word. From this manually tagged text, probabilities are calculated about the frequency with which tag sequences are found[15]. If enough text is tagged, then the frequency with which a certain tag appears with a certain word can also be stored[5]. Then, from this statistical information, a Hidden Markov Model is built and used to disambiguate the parts-of-speech by calculating the most probable path through all the possible tags[9][Chap 3.3].

¹⁰ <http://www.cs.jhu.edu/~weiss/stem.c> lists source code for the English version of the Porter stemmer.

<http://www.wots.let.ruu.nl/UPLIFT/stinfon.ps> describes a Dutch version of the Porter stemmer.

Alternative techniques are to train a tagger over untagged text so that the entropy of tag transitions is minimized[13]¹¹, or to use a manually tagged corpus to generate automated correction rules[4]¹², or to create a set of rules describing which transitions (such as article followed by verb) are impossible[41]¹³

Once the part-of-speech that each word plays in a sentence is determined, the words can be lemmatized, i.e. reduced to their normalized forms. This reduction allows different variants of similar structures to be recognized as being identical, which will allow the subsequent counting of the event to be more accurate.

Noun Phrase Recognition One of the most important multiword structures that one can recognize easily after part-of-speech tagging is the simple noun phrase. Much of the terminology found in a document collection is composed of noun phrases, and when a word is included in one of these phrases, one has a good indication of the word's meaning. For example, alone, the noun "star" is ambiguous, but not when it is embedded in the noun phrase "one star hotel."

In order to recognize noun phrases, one can define a regular pattern[39] of part of speech tags describing the contour that such phrases can take. For example, Figure 4 shows the output of a part-of-speech tagger and a definition of a noun phrase filter. When the filter is applied to the tagged text, the following noun phrases are matched: *industrial products*, *red warning lights*, *computerized design of integrated radio circuits*.

Let's reconsider the example *star*. Running a noun phrase filter over a large British corpus yields noun phrases containing *star* such as: *favourite pop star*, *orange K-type star*, *former England star*, *central star cluster*, *bright new star*, *white dwarf star*, *western film star*, *unorthodox ballroom star*, *red giant star*, *poor star vehicle*, *major league baseball star*, *international film star*, *very heavy star*, *very bright star*, *very average star*, *near naked-eye star*, *huge red star*, . . . In nearly every case, except maybe for *very heavy star*, the use of *star* in these phrases is no longer ambiguous. In the current situation, though, even though these phrases are automatically extractable, the Web user has no access to them except by guessing their existence and posing a positional query to a Web browser.

Light Parsing Extending the techniques for noun phrase recognition, one can define regular patterns which recognize verb groups, and which identify the heads of the noun groups and verb groups; low-level syntactic relations can then be drawn between these group heads[21,18]. Finite-state compilers[30,40] are very powerful tools for describing the patterns and the constraints that patterns must respect in order to be recognized as a syntactic group. Just as one technique for tokenizing text is to feed the input to a cascade of transducers that insert token

¹¹ The Common Lisp source code for such a tagger, version 1.2 of the Xerox part-of-speech tagger, is available at <ftp://parcftp.xerox.com/tagger-1-2.tar.Z>.

¹² This tagger created by Eric Brill for English can be found via <ftp://ftp.cs.jhu.edu/pub/brill/>.

¹³ The ENGCG tagger can be tested by sending an e-mail to engcg@ling.helsinki.fi.

Input Phrase: *industrial products, such as red warning lights, are now produced by computerized design of integrated radio circuits*

<i>original</i>	<i>lemma</i>	<i>tag</i>
industrial	industrial	+JJ
products	product	+NNS
,	,	+CM
such as	such as	+IN
red	red	+JJ
warning	warning	+NN
lights	light	+NNS
,	,	+CM
are	be	+BER
now	now	+RB
produced	produce	+VBN
by	by	+RB
computerized	computerize	+VBN
design	design	+NN
of	of	+IN
integrated	integrate	+VBN
radio	radio	+NN
circuits	circuit	+NNS

<i>Tags</i>	<i>meaning</i>
+BER	form of verb 'to be'
+CM	comma
+IN	preposition
+JJ	adjective
+NN	noun
+NNS	plural noun
+RB	adverb
+VBN	past participle

Noun Phrase Filter:

define **PRE** as one of the following tags: +JJ, +VBN, +NN, +NNS

define **NOUN** as one of: +NN or +NNS

define Filter as: **(PRE)* NOUN ((of (PRE)* NOUN))***

Phrases matched by Filter:

industrial products,

red warning lights,

computerized design of integrated radio circuits

Fig. 4. Defining a noun phrase filter as a regular expression over tags and words. The star(*) stands for any number of repetitions, including zero. The filter matches any number of members of the PRE class followed by a noun, followed by any number of the sequence “of” followed by possibly modified nouns.

boundary markers such as newlines in the input, higher level syntactic markings can be inserted into tagged text[1].

Figure 5 shows how additional markings can be added into tagged text using more elaborate regular expressions[29] compiled into transducers that insert or remove symbols. These additional markings allow one to easily define other regular expressions for recognizing specific syntactic configurations, for example, a configuration corresponding to subjects of passive verbs. These filtering expressions can include transducing elements that delete all the unimportant elements in the configuration, and which insert a label specifying the configuration. A passive direct object filter will extract a relation between *correlation* and *obtain*: PDOBJ(correlation,obtain). This relation can be made to correspond to relations extracted by the direct object filter which would like DOBJ(obtain,correlation). In this way, surface positional variation can be abstracted away from relations between words, just as lemmatization remove surface variation from different inflectional forms of a single word.

Elaborate patterns for around 100 nominalizations of communication verbs in English and French were extracted in the framework of the European project DECIDE [24]. This project, involving RXRC, the University of Liege (Belgium), and the University of Stuttgart (Germany), used automatic means of exploring corpora in order to extract collocations, in particular, support verbs for nominalizations. An online multilingual lexicon¹⁴ was produced of French, German, and English equivalents of these corpus-derived collocations.

3 Short Query Linguistic Expansion Techniques

It was argued earlier that in the current state of affairs, the burden is one the Web user to know or to discover how words are used in the pages indexed by the Web Browser. The previous section sought to demonstrate that it is possible, with current linguistic technology, for Web Browsers to extract much more meaning-identifying structure from the text that they index. The Web user should have access to this structure to browse, just as they have access to the original Web pages. A person who uses an indexed book can see how a word is used before reading the pages on which the word is found.

In this section, we will examine a more extended example of the information can be derived from a large heterogeneous corpus around a given word. For this example, we extracted from the British National Corpus¹⁵ all the sentences that contain the string “watch.” There are 15881 such sentences. This 2 M-byte corpus corresponds in a sense to gathering a large number of WWW pages containing any form of that string.

Running the BNC *watch* corpus through a noun phrase extractor produces a list of noun phrases, such as “neighbourhood watch,” “watch tower,” “digital watch,” “watch dog,” “Rolex Show,” etc. 1066 different simple noun phrases can

¹⁴ Consultable at <http://engdep1.philo.ulg.ac.be/decide/lexicon>.

¹⁵ The BNC is a 100 million word corpus of British written and spoken English. See <http://info.ox.ac.uk/bnc/index.html> for information. We parsed files *a* to *g*.

Original Phrase: *Significant correlations were obtained between the maternal and fetal glucose levels and the maternal and fetal ffa levels.*

Group	Heads	original	lemma	tag
<NG		Significant	significant	+JJ
	*HeadN	correlations	correlation	+NNS
NG>				
<VG		were	be	+BED
	*PasV	obtained	obtain	+VBN
VG>				
<NG		between	between	+IN
		the	the	+AT
		maternal	maternal	+JJ
		and	and	+CC
		fetal	fetal	+JJ
		glucose	glucose	+NN
	*PrepN	levels	level	+NNS
		and	and	+CC
		the	the	+AT
		maternal	maternal	+JJ
		and	and	+CC
		fetal	fetal	+JJ
		ffa	ffa	+JJ
	*PrepN	levels	level	+NNS
NG>				

Tags	meaning
+BED	form 'to be'
+CC	conjunction
+AT	article
+JJ	adjective
+NN	noun
+NNS	plural noun
+VBN	past participle
*HeadN	head of noun phrase
*PrepN	tied to preposition
*PasV	passive verb head
<NG	begin Noun group
NG>	end Noun group
<VG	begin Verb group
VG>	end Verb group

Fig. 5. Additional marking inserted by low-level parser. Just as one can describe the contour of noun phrases using regular expressions, it is possible to define the contour of Noun Groups, and Verb Groups. Once these marks are inserted, other regular expression insert head markings.

be found, 63 of which appear more than twice. Each of these phrases can be linked back to the pages from which they were extracted, in the WWW setting.

But how should this linguistically structured information be presented to the user? In some cases, *watch* is the head of the noun phrase, and in other cases it modifies some other noun. We cannot expect the naive user of the WWW to know or care about these linguistic distinctions. But we can rephrase these relations in a way that may be more understandable to the lay user. We can label those phrases in which “watch” appears at the end as “types of watches.” And those in which it appears in the beginning as “things involved in watches.” Though the actual relations that a word can play in a two word phrase are multifarious [36,42], these labels are vague enough to cover many of them. Given these labels, we can present the noun phrases structures in order of decreasing frequency to the user in a way such as in Figure 6. Due to tagging errors, some of words “watch” are actually being used a verb in the things listed as *watch things*. Still, the association of “watch” with another word often makes its meaning clearer.

<i>types of watch:</i>	gold watch, close watch, good watch, night watch, digital watch, careful watch, just watch, fob watch, wrist watch, stop watch, pocket watch, own watch, neighbourhood watch, ...
<i>watch things:</i>	watch television, watch TV, watch tower, watch people, watch video, watch glass, watch film, watch face, watch child, watch strap, watch spring, watch shop, watch salesman, watch press, watch mark, watch duty, watch cricket, watch chain, watch case, watch alarm

Fig. 6. Explaining the uses of “watch” in noun phrases.

For languages such as English which use capitalization as a proper name indicator, it easy to use the patterns of upper and lower case letters characteristic of names to recognize a large number of proper names[21, p. 150]. Alternative methods uses lists of proper name markers[3]. Recognized names can be semantically typed [16] by decoding their subcomponents. Once noun phrases have been recognized, one can isolate those noun phrases containing non-sentence-initial upper case from lower-case noun phrases, shown in Figure 7, providing a further automatic structuring to the WWW user.

In addition to noun phrases, verbal phrases involving “watch” used either as a verb or as a noun can be extracted using Light Parsing technology mentioned in the previous section. When our “watch” corpus is parsed in this way, we find the typical direct objects of the verb “watch” and the typical verbs associated with the noun watch. For example, the most frequently found direct object of “watch” (other than a personal pronoun or “it”) was “television” (34 times in 5134 sentences). The naive user of the WWW will not want to know about direct objects and subjects. In fact, most naive users of the WWW will hardly remember what a noun or a verb is. The information may be conveyed in a more

<i>Named watch:</i>	Rolex watch, Swiss watch, Cartier watch, Seiko watch, Rab watch, Dali watch, British watch, American watch, Abberley watch, Z114 watch, Tudor watch, Thomas Pride watch, Taiwanese watch, Tag Heuer watch...
<i>Names</i> watch:	<i>involving</i> Neighbourhood Watch, Black Watch, Watch Committee, Night Watch,, Americas Watch, International Dolphin Watch, Green Leaf Watch, Bank Watch, Whitehall Watch, Watchorn Alfreton, Watches of Switzerland,..., Rolex Watch Co.

Fig. 7. Noun phrases in which “watch” appears with a proper name.

simple way by using circumlocutions such as *Things one can watch* instead of the more academic *direct objects of watch*. In a similar the information concerning the use of *watch* as a verb can be presented as in Figure 8.

Above we saw how noun phrases involving the word *watch* might be presented to the linguistically naive Web user, the uses of the noun *watch* with verbs might also be explained by listing things that a *watch* can do, or things that are done to a *watch*, as in Figure 9, which cover the uses of the word as a direct object and as a subject.

The linguistically derived structures mentioned in the previous sections show the words that are syntactically tied to “watch.” If the WWW users pose a one-word query “watch” then presenting them with these lists would provide a useful overview for distinguishing the meaning the searcher was initially interested in. In every document in which “watch” appears, there are additional words, not directly entering into syntactic relations with “watch” but sharing a first order affinity [22], that is, often found in the same vicinity, though not syntactically attached to “watch.” These words give another view of how the word is used, and like with the RANK function of Dialog and LiveTopics of Altavista, can be added to a boolean query to make the results more precise. In our “watch” corpus, the words most often used in the same sentences with “watch” outside of the words syntactically tied to it, and outside of stopwords such as article, prepositions, etc., are “time”, “look,” “stand,” “way,” “face,” “night,” “small,” “years,” etc. If a reference corpus exists, then these words can be further classified by applying a statistic such as “mutual recall” as has become popular in lexicography [11].

4 Conclusion

We have presented a number of natural language processing tools that could be used to provide an intermediate structure between short queries and the WWW. The tools provide for each word, a list of ways in which that word is used with other words over the indexed documents. The presence of the other words allows the user to see the possible uses of the word, determine the meaning and to decide if they want to access the documents with those uses. The user has access

<i>Things one can watch:</i>	watch somebody, watch it, watch television, watch man, watch film, watch face, watch TV, watch video, watch news, watch game, watch people, watch play, watch match, watch programme, watch woman, watch one, watch show, watch movie, watch world, watch child, watch this, watch girl, watch walk, watch programmes, watch bird, watch football, watch space, watch scene, watch move, watch light, watch performance, ...
<i>Who can watch:</i>	somebody watch, people watch, man watch, it watch, eye watch, child watch, those watch, one watch, worth watch, stand watch, woman watch, anyone watch, everyone watch, sit watch, crowd watch, mother watch, time watch, girl watch, boy watch, hour watch, all watch, day watch, bird watch, million watch, face watch, audience watch, there watch, other watch, window watch, friend watch, ...
<i>How can one watch:</i>	watch out, watch just, watch back, watch down, watch on television, watch still, watch also, watch even, watch closely, watch somebody, watch carefully, watch now, watch too, watch home, watch over somebody, watch away, watch on TV, watch by, watch always, watch able, watch with interest, watch often, watch again, watch from window, watch never, watch in silence, watch together, watch with somebody, watch for somebody

Fig. 8. Verb phrases involving the verb “watch.”

<i>Things that one does to a watch:</i>	keep watch, have watch, check watch, take watch, stand watch, consult watch, wear watch, sit watch, get watch, go watch, could watch, make watch, say watch, come watch, see watch, set watch, can watch, do watch, leave watch, receive watch, find watch, stop watch, wait watch, put watch, steal watch, buy watch, sell watch, need watch, watch watch, close watch, lose watch, ...
<i>What can a watch can do:</i>	watch stop, watch say, watch tell, watch happen, watch get, watch go, watch lie, watch move, watch leave, watch know, watch see, watch pull, watch make, watch let, watch hold, watch take, watch ask, watch like, watch agree, watch cost, watch cause, watch demand, watch learn, watch work, watch find, watch think, watch miss, ...

Fig. 9. verb phrases in which “watch” appears as a noun.

to the way a word is used without having to access Web pages, The burden of the work is placed upon the computer during initial indexing, but existing natural language processing tools are already capable of providing the robust text analysis necessary, and the question of additional storage costs that such indexes would incur becomes less meaningful daily.

References

1. Salah Ait-Mokhtar and Jean-Pierre Chanod. Incremental finite-state parsing. In *ANLP'97*, pages 72–79, Washington, 1997.
2. D.C. Blair and M.E. Maron. An evaluation of retrieval effectiveness. *Communications of the ACM*, 28:289–299, 1985.
3. C. Borkowski. An experimental system for the automatic identification of personal names and personal titles in newspaper texts. *American Documentation*, 18:131, July 1967.
4. Eric Brill. A simple Rule-Based part of speech tagger. In *Proceedings of the Third conference on Applied Natural Language Processing*, Trento, Italy, 1992. ACL.
5. Ted Briscoe, Greg Grefenstette, Lluís Padró, and Iskander Serai. Hybrid techniques for training hmm part-of-speech tagger. Technical Report MLTT-007, Rank Xerox Research Centre, 1994.
6. Chris Buckley, Amit Singhal, and Mindhar Mitra. New retrieval approaches using smart: Trec4. In D.K. Harman, editor, *The Fourth Text REtrieval Conference (TREC-4)*, pages 25–48. U.S. Department of Commerce, 1996. NIST Special Publication 500–236.
7. John Carroll and Ted Briscoe. The derivation of a large computational lexicon for english from ldoce. In B. Boguraev and T. Briscoe, editors, *Computational Lexicography for Natural Language Processing*, London, 1989. Longman.
8. J.P. Chanod and P. Tapanainen. Creating a tagset, lexicon and guesser for a french tagger. In *Proceedings of the ACL SIGDAT Workshop*, Dublin, Ireland, 1995.
9. Eugene Charniak. *Statistical Language Learning*. MIT Press, Cambridge, Mass, 1993.
10. Fah-Chun Cheong. *Internet Agents: Spiders, Wanderers, Brokers and 'Bots*. New Riders Publishing, Indianapolis, 1996.
11. Kenneth Ward Church and Patrick Hanks. Word association norms, mutual information, and lexicography. *Computational Linguistics*, 16(1):22–29, March 1990.
12. Cyril W. Cleverdon. The significance of the cranfield tests on index languages. In A. Bookstein, Y. Chiaramella, G. Salton, and V. V. Raghavan, editors, *Proceedings of the 14th Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval*, pages 3–131, New York, Oct 13-16 1991. SIGIR'91, Association for Computing Machinery. Special issue of the SIGIR Forum.
13. Doug Cutting, Julian Kupiec, Jan Pedersen, and Penelope Sibun. A practical part-of-speech tagger. *Proceedings of the Third Conference on Applied Natural Language Processing*, April 1992.
14. Douglas Cutting, Jan O. Pedersen, David Karger, and John W. Tukey. Scatter/Gather: A cluster-based approach to browsing large document collections. In *Proceedings of SIGIR'92*, pages 318–329, Copenhagen, Denmark, June 21-24 1992. ACM.

15. Steven J. DeRose. Grammatical category disambiguation by statistical optimization. *Computational Linguistics*, 14(1):31–39, Winter 1988.
16. David D. Donaldson. Internal and external evidence in the identification and semantic categorization of proper names. In B. Boguraev and J. Pustejovsky, editors, *Proceedings of the SIGLEX Workshop on Acquisition of Lexical Knowledge from Text*, pages 32–43, Columbus, OH, 1993.
17. Lauren B. Doyle. Semantic road maps for literature searchers. *Journal of the ACM*, 8(4):553–578, October 1961.
18. G. Grefenstette. Light parsing as finite state filtering. In *Workshop on Extended finite state models of language*, Budapest, Hungary, Aug 11–12 1996. ECAI'96.
19. G. Grefenstette and F. Segond. Multilingual natural language processing. *International Corpus of Corpus Linguistics*, 2(1), 1997.
20. G. Grefenstette and P. Tapanainen. What is a word, what is a sentence? Problems of tokenization. In *3rd Conference on Computational Lexicography and Text Research*, Budapest, Hungary, 7–10 July 1994. COMPLEX'94. <http://www.rsrc.xerox.com/publis/mltt/mltt-004.ps>.
21. Gregory Grefenstette. *Explorations in Automatic Thesaurus Discovery*. Kluwer Academic Press, Boston, 1994.
22. Gregory Grefenstette. Corpus-derived first, second and third-order word affinities. In *Sixth Euralex International Congress*, Amsterdam, Aug 3—Sept 3, 1994.
23. Gregory Grefenstette. Comparing two language identification schemes. In *Proceedings of the 3rd International Conference on the Statistical Analysis of Textual Data, JADT'95*, Rome, Dec 11–13, 1995.
24. Gregory Grefenstette, Ulrich Heid, and Thierry Fontenelle. The DECIDE project: Multilingual collocation extraction. In *Seventh Euralex International Congress*, University of Gothenburg, Sweden, Aug 13–18, 1996.
25. Donna Harman. Relevance feedback revisited. In *Proceedings of SIGIR'92*, Copenhagen, Denmark, June 21–24 1992. ACM.
26. Donna Harman, editor. *The First Text REtrieval Conference (TREC-1)*. U.S. Government Printing Office, Washington, 1993. NIST Special Publication 500–207.
27. Marti A. Hearst, David Karger, and Jan O. Pedersen. Scatter/gather as a tool for the navigation of retrieval results. In Robin Burke, editor, *Working Notes of the AAAI Fall Symposium on AI Applications in Knowledge Navigation and Retrieval*, Cambridge, MA, November 1995. AAAI.
28. David A. Hull. Stemming algorithms: A case study for detailed evaluation. *JASIS*, 47(1), January 1996. Special Issue on the Evaluation of Information Retrieval Systems.
29. L. Karttunen, J.P. Chanod, G. Grefenstette, and A. Schiller. Regular expression for language engineering. *Journal of Natural Language Engineering*, 1997.
30. Lauri Karttunen. Finite-state lexicon compiler. Technical Report ISTL-NLTT–1993-04-02, Xerox, Palo Alto Research Center, April 1993.
31. Lauri Karttunen. Directed replacement. In *Proceedings of the 34th Annual Meeting of the ACL*, Santa Cruz, CA, 1996.
32. K. L. Kwok. A new method for weighting query terms for ad-hoc retrieval. In *Proc. of the 19th ACM/SIGIR Conference*, pages 187–196, 1996.
33. X. A. Lu and R. B. Keefer. Query expansion/reduction and its impact on information retrieval effectiveness. In Donna Harman, editor, *The Third Text REtrieval Conference (TREC-3)*, pages 231–239, Washington, 1995. U.S. Government Printing Office. NIST Special Publication 500–225.

34. M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.
35. G. Russell, S. Pulman, G. Ritchie, and A. Black. A dictionary and morphological analyser for english. In *11th International Conference on Computational Linguistics*, pages 277–279, Bonn, Germany, 1987.
36. Gerard Salton. A note on information retrieval models. In *RIAO'85*, pages 2–27, Grenoble, France, March 18–20 1985. CID, Paris, and IMAG.
37. Gerard Salton and M. McGill. *An Introduction to Modern Information Retrieval*. McGraw-Hill, New York, 1983.
38. Anne Schiller. Multilingual finite-state noun phrase extraction. In *Workshop on Extended finite state models of language*, Budapest, Hungary, Aug 11–12 1996. ECAI'96.
39. Anne Schiller. Multilingual part-of-speech tagging and noun phrase mark-up. In *15th European Conference on Grammar and Lexicon of Romance Languages*, University of Munich, Sept 19–21 1996.
40. Pasi Tapanainen. RXRC finite-state compiler. Technical Report MLTT-020, Rank Xerox Research Centre, Grenoble, April 1995.
41. Atro Voutilainen, Julia Heikkilä, and Arto Anttila. A lexicon and constraint grammar of english. In *Proceedings of the Fourteenth International Conference on Computational Linguistics*, Nantes, France, July 1992. COLING'92.
42. Beatrice Warren, editor. *Semantic Patterns of Noun-Noun Compounds*. Acta Universitatis Gothoburgensis, Goteborg, Sweden, 1978. Gothenburg Studies in English, 41.