Improving the Precision of a Text Retrieval System with Compound Analysis

Renée Pohlmann[*]
Wessel Kraaij[†]

Abstract

In this paper we describe research on compound analysis in the UPLIFT information retrieval project. Results of earlier experiments indicated that splitting up compounds in the query and forming new compounds by combining query terms improves recall while precision does not deteriorate. We investigated whether adding syntactic constraints to the compound splitting and formation processes would improve our initial results. We compared different strategies for compound formation and we also investigated the effect of adding compound constituents as separate index terms. The results of our experiments show that using information about head-modifier relationships to create complex index terms can improve both recall and precision significantly but only if all constituents are also added separately. We found that using both noun-adjective and noun-noun head-modifier pairs produced the best results.

1 Introduction

The work described in this paper is part of the UPLIFT project[1]. UPLIFT investigates whether linguistic tools can improve and extend the functionality of vector space text retrieval systems (cf. Salton (1989), p. 312 *ff*). Earlier experiments in the UPLIFT project focussed on improving recall by using stemming algorithms[2]. This paper describes an experiment with syntactic phrase indexing techniques for Dutch texts, aimed at improving precision as well as recall. The basic idea behind phrase indexing is that phrases characterize document content more effectively than single word terms. When a single word index is used, a query containing the phrase *information retrieval* will also match with documents containing only *information* or *retrieval*. If *information retrieval* is recognized as a unit, however, these matches may be avoided or given a much lower score (depending on the matching

[*]Utrecht Institute of Linguistics OTS
[†]Netherlands Organization for Applied Scientific Research (TNO), Institute of Applied Physics
[1]UPLIFT: Utrecht Project: Using Linguistic Information for Free Text retrieval. UPLIFT Home Page: *http://www-uilots.let.ruu.nl/ ~uplift*
[2]See Kraaij and Pohlmann (1996) for details.

strategy). Different strategies have been used to identify suitable phrases for indexing, the most important distinction being between strategies based on statistical co-occurrence data and strategies based on syntactic processing. So far, both types of strategies have proven to be equally successful (cf. e.g. Fagan (1987), Salton et al. (1990) and, more recently, Hull et al. (1997)). Results of earlier experiments in the UPLIFT project motivated us to take compounds as a starting point for our experimentation with phrase indexing. Our approach was further inspired by the work of Strzalkowski, as described in Strzalkowski (1995) and Strzalkowski and Perez Carballo (1996). Strzalkowski uses syntactic information to identify phrases in queries and documents. These phrases are subsequently normalized (i.e. semantically similar but syntactically different constructions, e.g. *retrieval of information* vs. *information retrieval*, are represented identically) as head-modifier pairs. Other recent work on syntactic phrase indexing includes Evans and Zhai (1996) and Smeaton et al. (1995). In sections 2 to 6 we describe our approach and discuss the set-up and the results of the experiments. In section 7 we present the conclusions and give some possibilities for further research.

2 Compounds and related constructions

Earlier research in the UPLIFT project showed that when a query is expanded with the constituents of compounds already occurring in it[3] and new compounds are added to the query by combining query terms, recall improves while precision does not deteriorate. The following example illustrates this approach.

> Query: *Ik zoek documenten over computers en natuurlijke taalverwerking* ("I am looking for documents on computers and natural language processing")

This query would result in the following index terms (after removal of stop words):

document	
computer	
natuurlijk	
taalverwerking	
taal	compound splitting
verwerking	"
computertaal	compound formation
taalcomputer	"

In the example, the compounds *computertaal* (computer language) and *taalcomputer* (language computer) are added to the query by combining *computer* and *taal*. Both are valid compounds[4] but, although the second compound may retrieve relevant articles for this query, the first (a synonym for programming language) will probably retrieve many unrelated documents.

[3]In Dutch, compounds are usually written as a single orthographic unit, e.g. *levensverzekeringsmaatschappij* (life insurance company). As a result of this, compound constituents are normally not considered as separate index terms.

[4]New compounds are validated using a list of all the compounds found in the document collection.

We decided to investigate whether it would be possible to improve precision as well by using syntactic information to constrain the compound splitting and compound formation processes.

We restricted compound splitting by creating system variants which only add the heads or both heads and modifiers as separate index terms. To split up compounds into their constituents we used the dictionary-based compound splitter developed by Theo Vosse for the CORRie project (cf. Vosse (1994)). The compound splitter does not assign structure to the compound but simply yields a list of constituents. Identifying head-modifier relationships in compounds is not trivial because of possible structural ambiguities. In Dutch, compounds existing of two parts are usually right-headed (a *fietswiel* is a type of *wiel*) but compound construction is recursive and both the head and the modifier can be compounds themselves, resulting in structural ambiguities, e.g. [[X1 X2] X3] or [X1 [X2 X3]]. We have not attempted to implement a strategy to solve all structural ambiguities in compounds but we have applied two different heuristics to assign probable structures. In a recent study, ter Stal (1996) found that simply assuming that all compounds have a left-branching structure produced ± 70% correct results. Although his results are for English, we decided to try this strategy. As an alternative, we also implemented a strategy where we use unambiguous cases collected from the corpus to confirm a certain choice. If we find independent evidence for a left-branching structure (X1 modifies X2 in unambiguous contexts) or a right-branching structure (X1 modifies X3) we select the appropriate structure. If we do not find independent evidence for either structure we choose a left-branching structure by default[5].

The formation of new compounds was restricted by using only terms which occur in a certain syntactic context to generate new complex terms. We restricted compound generation to term pairs originating from complex Noun Phrases (NPs) containing a specific type of Prepositional Phrase (PP) (with the preposition *van*, *voor* or *door*) as a noun post-modifier. The choice for this construction was motivated by the fact that many compounds in Dutch can be paraphrased using a specific type of PP, e.g. *fietswiel* (bicycle wheel) ↔ *wiel van een fiets* (wheel of a bicycle), see, for instance, Geerts et al. (1984) p. 103. The term pairs were created by combining the head noun of the main NP with the head noun of the NP contained in the PP. Figure 1 illustrates this process. PP-modification structures exhibit similar ambiguities to the ones in complex compounds, e.g. in *the man with the dog with the spots* it is not clear whether the PP *with the spots* modifies *the man* or *the dog*. We decided to treat these structures analogously to the compound structures. The default strategy we adopted was to assume that PP modification structures are right-branching (i.e. each PP modifies the noun immediately preceding it). We again also implemented a second strategy, using corpus data for disambiguation. We later extended compound generation by using all PP post-modifiers and adjective pre-modifiers as well.

To ensure matching, both original and new complex terms were normalized as head-modifier pairs. Complex constructions consisting of more than 2 constituents are represented as several head-modifier pairs. See figure 2 for an example. Fur-

[5]A third option, where the structure is simply left ambiguous and all interpretations are selected, was not implemented for lack of time.

thermore, both queries and documents were treated analogously. This was not possible in our earlier approach (combining all the terms in a document to create compounds is clearly not feasible). In this way we ensured that matches between compounds and equivalent constructions would be given the same score as literal matches.

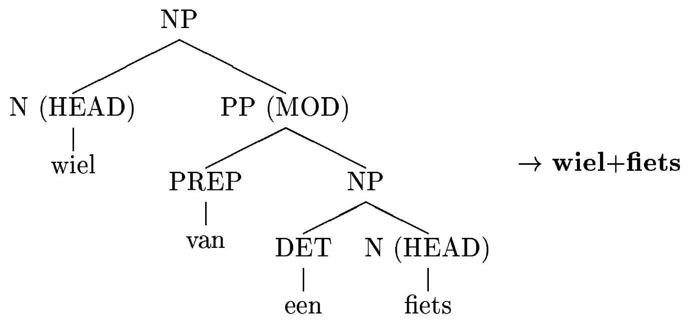

Figure 1: Term pair extraction from NP with PP modifier

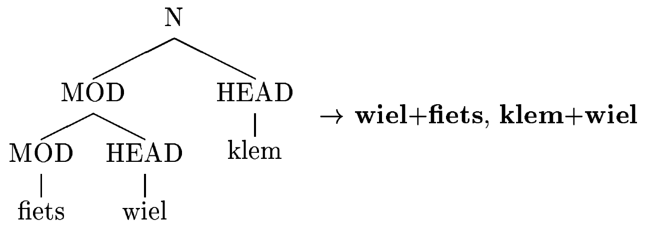

Figure 2: Term pair extraction from complex compound

3 Indexing module

Based on the options described in section 2 above we developed several different versions of an indexing module and integrated each of these with our retrieval engine[6] to create different system variants. The indexing modules consist of the following basic sub-routines.

A string segmentation algorithm (**tokenizer**) is used to identify sentence and word boundaries.

A **lexical look-up** algorithm, based on the CELEX lexical database for Dutch (Baayen et al. (1993)) assigns part-of-speech tags to the words.

A **tagger** is used to resolve ambiguities in tag assignment. We used the Multext tagger, (cf. Armstrong et al. (1995)), a Hidden Markov Model tagger, which has

[6]The retrieval engine used in the UPLIFT project is the TRU vector space engine developed by Philips Research (cf. Aalbersberg et al. (1991)).

$$\begin{tikzcd}
\text{text} \arrow[r] & \boxed{\text{tokenizer}} \arrow[r] & \boxed{\text{lexical look-up}} \arrow[r] & \boxed{\text{tagger}} \arrow[r] & \boxed{\text{proper names}} \arrow[r] & \boxed{\text{NP-parser}} \arrow[r] & {} \\
\boxed{\text{pair extraction}} \arrow[r] & \boxed{\text{stop words}} \arrow[r] & \boxed{\text{stemmer}} \arrow[r] & \text{simple and complex index terms}
\end{tikzcd}$$

Figure 3: Indexing process

the advantage that it requires only a partially disambiguated corpus for training. After training, the tagger produced 91.5% correct results.

A very simple heuristic based on the distinction upper case–lower case is used to glue sequences of **proper names** together (e.g. *Verenigde_Staten_van_Amerika* (United States of America)). In this way we ensure that proper names are treated as a unit and term pairs are not extracted from them.

An **NP-parser** is used to identify NPs in the texts. The parser we use was developed by TNO-TPD, (cf. van Surksum and den Besten (1993)). This parser is deterministic and requires fully disambiguated input from the tagger. It is also robust and fast (244 sentences per second on a Sun-Sparc 10/40). The coverage of the NP-grammar is not complete[7], but for the purpose of our experiment it was considered to be sufficient.

Since the parser is deterministic and only generates one analysis for ambiguous structures, separate **pair extraction** modules extract the appropriate word pairs and single words from the output of the parser.

A **stop word list** is used to identify and eliminate so-called stop words (mostly function words).

Finally, all remaining words (and compound constituents) are replaced by their stem using a dictionary-based (CELEX) **stemming** algorithm. We used the best variant of all the stemming algorithms tested in previous UPLIFT experiments (cf. Kraaij and Pohlmann (1996)). This variant handles inflection only. Figure 3 shows how the different sub-routines work together.

4 System variants

We developed and tested a large number of system variants (23). These variants are summarized below. The names are abbreviations of the type vABC which must be interpreted as follows: A refers to the syntactic context from which of the head-modifier pairs are generated, B to the strategy used for the disambiguation of complex structures and C to the treatment of constituents of complex structures.

vXXX No compound analysis but tagging, proper name identification and stemming are included. We added this version to see whether tagging, stemming and proper name recognition alone would already be sufficient to improve precision.

vM.. Head-modifier pairs are generated from compounds.

[7]Relative clauses, for instance, are not included.

vS.. Head-modifier pairs from complex NPs with specific PP post-modifiers (see section 2 above) are added.

vP.. Head-modifier pairs from all PP post-modifiers are added.

vA.. Head-modifier pairs from adjective pre-modifiers are added.

vAP.. A combination of the two previous versions.

v.a. Complex terms are analyzed using the default strategies.

v.c. Complex terms are analyzed using corpus data.

v..1 All constituents of complex terms are also added separately to the index.

v..2 Only heads (including heads of complex modifiers) are added to the index separately.

v..3 Only the head of the entire complex construction is added as a separate index term.

v..4 Constituents are not added separately.

We compared these variants with the following two versions:

vn Baseline. TRU retrieval engine, no extensions.

c4fow Best version from previous experiments, all constituents of compounds are added to the query and new compounds are generated by arbitrarily combining query terms.

5 Test procedures

The test collection used for the experiments was compiled during previous research in the UPLIFT project on stemming algorithms. It consists of a document collection of 59,608 articles published in *Het Eindhovens Dagblad*, *Het Brabants Dagblad* and *Het Nieuwsblad* from January to October 1994 and 36 queries and relevance judgements. Some general statistics for the document collection are given in table 1 below.

Total number of documents	59,608
Total number of words (tokens)	26,585,168
Total number of terms (types)	434,552
Max number of words per document	5,979
Av. number of words per document	446
Max number of terms per document	2,291
Av. number of terms per document	176

Table 1: Document collection statistics

The queries were formulated by test subjects recruited among staff and students of Utrecht University. Test subjects also performed the relevance judgements for their queries.

Retrieval performance is usually evaluated using measures derived from the following two main parameters:

$$Recall = \frac{number\ of\ relevant\ items\ retrieved}{total\ number\ of\ relevant\ items\ in\ collection}$$

$$Precision = \frac{number\ of\ relevant\ items\ retrieved}{total\ number\ of\ items\ retrieved}$$

The values for recall and precision range from 0 (low) to 1 (high). When precision is high, recall is usually low and vice versa.

The computation of recall is a traditional problem in IR evaluation. It is impossible to estimate the total number of relevant items in a document collection for a certain query without doing relevance assessments for nearly the complete collection. The common solution to this problem is to use the so-called *pooling method*[8]. This method is based on the assumption that if one uses a variety of different retrieval systems to create a document *pool* for each query, the probability that most relevant documents will be contained in the pool is high. The list of relevant documents for a each query is then compiled by judging only those documents contained in the pool.

We used 4 different derived measures to evaluate retrieval performance for this experiment. These measures are: average precision, ap5-15 (precision at 5, 10 and 15 documents retrieved, averaged), R-recall (recall at R, where R is the number of relevant articles for a particular query) and recall1000 (recall at 1000 documents retrieved). Average precision and R-recall measure general performance for precision and recall respectively. The ap5-15 measure should give an idea of the performance of the system variants for shallow searches where only the first few documents will be considered and the recall1000 measure is aimed at more in-depth searches. We also performed statistical significance tests to establish whether the differences between values are significant or should be attributed to chance. The design chosen for these statistical tests is based on Tague-Sutcliffe (1995a) and Tague-Sutcliffe (1995b). Details on the statistical tests can be found in the appendix.

6 Results

The results of the experiment are summarized in table 2. The percentages indicate improvement/decrease compared to the performance of the baseline (vn)[9]. The results of the statistical significance tests are summarized in tables 3, 4, 5 and 6. In these tables system versions have been divided into equivalence classes indicated by numbers.

The results show that tagging, proper name recognition and stemming alone are not sufficient to improve average precision significantly (vXXX is assigned to the

[8]See Harman (1993), p. 9 *ff*.

[9]Note that figures have been rounded. This accounts for small differences between seemingly equivalent versions.

version	avp	% change	ap5-15	% change
vXXX	0.330 (0.218)	+ 5.4	0.420 (0.285)	+ 8.4
vMa1	0.350 (0.215)	+ 11.9	0.443 (0.284)	+ 14.3
vMc1	0.350 (0.215)	+ 11.8	0.444 (0.284)	+ 14.4
vMa2	0.340 (0.225)	+ 8.7	0.448 (0.309)	+ 15.6
vMc2	0.341 (0.225)	+ 9.0	0.446 (0.308)	+ 15.0
vMa3	0.344 (0.227)	+ 9.9	0.451 (0.311)	+ 16.3
vMc3	0.344 (0.227)	+ 9.9	0.450 (0.311)	+ 16.2
vMa4	0.330 (0.212)	+ 5.4	0.427 (0.274)	+ 10.0
vMc4	0.330 (0.212)	+ 5.6	0.426 (0.274)	+ 9.9
vSa1	0.346 (0.214)	+ 10.7	0.441 (0.283)	+ 13.9
vSc1	0.348 (0.213)	+ 11.3	0.443 (0.282)	+ 14.3
vSa2	0.284 (0.240)	− 9.3	0.366 (0.334)	− 5.7
vSc2	0.286 (0.239)	− 8.6	0.365 (0.331)	− 5.8
vSa3	0.285 (0.240)	− 8.9	0.361 (0.335)	− 6.9
vSc3	0.286 (0.238)	− 8.6	0.363 (0.331)	− 6.3
vSa4	0.277 (0.236)	− 11.3	0.349 (0.329)	− 10.0
vSc4	0.279 (0.235)	− 10.8	0.349 (0.329)	− 10.0
vPa1	0.354 (0.221)	+ 13.2	0.451 (0.282)	+ 16.2
vAa1	0.347 (0.210)	+ 11.1	0.443 (0.277)	+ 14.3
vAa4	0.309 (0.216)	− 1.0	0.383 (0.277)	− 1.3
vAPa1	0.358 (0.217)	+ 14.6	0.448 (0.276)	+ 15.7
c4fow	0.319 (0.200)	+ 2.2	0.427 (0.277)	+ 10.0
vn	0.313 (0.214)		0.388 (0.291)	
version	R-recall	% change	recall1000	% change
vXXX	0.316 (0.206)	+ 12.1	0.855 (0.166)	+ 11.8
vMa1	0.344 (0.186)	+ 22.1	0.914 (0.102)	+ 19.5
vMc1	0.343 (0.186)	+ 22.0	0.914 (0.102)	+ 19.5
vMa2	0.313 (0.206)	+ 11.2	0.879 (0.136)	+ 14.8
vMc2	0.312 (0.207)	+ 10.8	0.873 (0.145)	+ 14.1
vMa3	0.312 (0.207)	+ 11.0	0.875 (0.141)	+ 14.4
vMc3	0.312 (0.207)	+ 11.0	0.875 (0.141)	+ 14.3
vMa4	0.311 (0.201)	+ 10.3	0.850 (0.165)	+ 11.1
vMc4	0.310 (0.201)	+ 10.3	0.849 (0.167)	+ 10.9
vSa1	0.343 (0.184)	+ 21.7	0.918 (0.104)	+ 19.9
vSc1	0.343 (0.184)	+ 21.8	0.918 (0.104)	+ 19.9
vSa2	0.260 (0.212)	− 7.7	0.783 (0.230)	+ 2.4
vSc2	0.260 (0.210)	− 7.6	0.778 (0.232)	+ 1.7
vSa3	0.254 (0.216)	− 9.6	0.758 (0.246)	− 0.9
vSc3	0.256 (0.214)	− 9.2	0.760 (0.241)	− 0.7
vSa4	0.246 (0.212)	− 12.5	0.746 (0.249)	− 2.5
vSc4	0.248 (0.210)	− 11.8	0.746 (0.247)	− 2.6
vPa1	0.348 (0.189)	+ 23.6	0.919 (0.100)	+ 20.2
vAa1	0.343 (0.188)	+ 21.8	0.920 (0.100)	+ 20.2
vAa4	0.279 (0.204)	− 1.0	0.818 (0.461)	+ 6.9
vAPa1	0.354 (0.195)	+ 25.8	0.918 (0.105)	+ 19.9
c4fow	0.317 (0.191)	+ 12.7	0.881 (0.148)	+ 15.1
vn	0.281 (0.195)		0.765 (0.216)	

Table 2: Evaluation measures averaged over queries (including variance)

system	avp					
vAPa1	0.358	1				
vPa1	0.354	1				
vMa1	0.350	1	2			
vMc1	0.350	1	2			
vSc1	0.348	1	2	3		
vAa1	0.347	1	2	3		
vSa1	0.346	1	2	3		
vMa3	0.344	1	2	3		
vMc3	0.344	1	2	3		
vMc2	0.341	1	2	3		
vMa2	0.340	1	2	3		
vMc4	0.330	1	2	3		
vXXX	0.330	1	2	3		
vMa4	0.330	1	2	3		
c4fow	0.319	1	2	3	4	
vn	0.313		2	3	4	5
vAa4	0.309			3	4	5
vSc2	0.286				4	5
vSc3	0.286				4	5
vSa3	0.285				4	5
vSa2	0.284				4	5
vSc4	0.279					5
vSa4	0.277					5

Table 3: Equivalence classes avp

system	ap5-15			
vMa3	0.451	1		
vPa1	0.451	1		
vMc3	0.450	1		
vAPa1	0.448	1		
vMa2	0.448	1		
vMc2	0.446	1		
vMc1	0.444	1		
vSc1	0.443	1		
vMa1	0.443	1		
vAa1	0.443	1		
vSa1	0.441	1		
c4fow	0.427	1	2	
vMa4	0.427	1	2	
vMc4	0.426	1	2	
vXXX	0.420	1	2	
vn	0.388		2	3
vAa4	0.383		2	3
vSa2	0.366			3
vSc2	0.365			3
vSc3	0.363			3
vSa3	0.361			3
vSa4	0.349			3
vSc4	0.349			3

Table 4: Equivalence classes ap5-15

system	R-Recall				
vAPa1	0.354	1			
vPa1	0.348	1	2		
vMa1	0.344	1	2		
vMc1	0.343	1	2		
vAa1	0.343	1	2		
vSc1	0.343	1	2		
vSa1	0.343	1	2		
c4fow	0.317	1	2	3	
vXXX	0.316	1	2	3	
vMa2	0.313	1	2	3	
vMa3	0.312		2	3	
vMc3	0.312		2	3	
vMc2	0.312		2	3	
vMa4	0.311		2	3	
vMc4	0.310		2	3	
vn	0.281			3	4
vAa4	0.279			3	4
vSc2	0.260				4
vSa2	0.260				4
vSc3	0.256				4
vSa3	0.254				4
vSc4	0.248				4
vSa4	0.246				4

Table 5: Equivalence classes R-recall

system	r1000							
vAa1	0.920	1						
vPa1	0.919	1						
vAPa1	0.918	1						
vSc1	0.918	1						
vSa1	0.918	1	2					
vMc1	0.914	1	2	3				
vMa1	0.914	1	2	3				
c4fow	0.881	1	2	3	4			
vMa2	0.879	1	2	3	4			
vMa3	0.875	1	2	3	4			
vMc3	0.875	1	2	3	4			
vMc2	0.873	1	2	3	4			
vXXX	0.855	1	2	3	4			
vMa4	0.850		2	3	4	5		
vMc4	0.849			3	4	5		
vAa4	0.818				4	5	6	
vSa2	0.783					5	6	7
vSc2	0.778						6	7
vn	0.765						6	7
vSc3	0.760						6	7
vSa3	0.758						6	7
vSa4	0.746							7
vSc4	0.746							7

Table 6: Equivalence classes recall1000

same equivalence class (2) as vn). Results also show that our initial attempts to improve precision by using a subset of PP post-modifiers to create new compounds (vS.. versions) were not successful. These versions are all in equivalence classes which include vn. Compared to the vM.. versions which only normalize original compounds, average precision even decreases, although in most cases the difference is not significant.

If we look at the results in more detail we see that the distinction head-modifier is not relevant for compound splitting. Versions v..1 which add all subparts of a complex term to the index usually outperform the other versions (versions v..2/3/4). If we only consider the first 15 documents retrieved (ap5-15) then versions vM.2 and vM.3 show a slight advantage over vM.1. However, this difference is not statistically significant. We also see that the two strategies for handling ambiguous structures (versions v.a. and v.c.) are equivalent. It may be that our corpus is too small to render sufficient data for the corpus-based approach. It may also be that the default strategy simply works well for our data.

In table 7 some statistics for versions c4fow and vSa1 are given. The figures show that although the number of compounds found by c4fow in greatly exceeds the number of compounds found by the syntactic version, the percentage of relevant combinations (actually found in relevant articles) is higher for the syntactic version. We concluded that the compound generation strategy employed by the vS.. versions was too restricted and should be extended to include other head-modifier pairs. We experimented with several extensions. We implemented a version which instead of a subset of PP-modifiers uses all PP-modifiers for term pair generation (version vPa1). Besides this version we also developed a version which adds noun-adjective head-modifier pairs to the index (vAa1). Version vAPa1 combines these two strategies.

version	number of compounds	relevant compounds	% relevant
c4fow	147	35	20.5
vSa1	46	18	39.1

Table 7: Relevant compounds found in queries by c4fow vs. vSa1

If we look at the results for these versions we see that version vPa1, the version which adds noun-noun pairs from all PP post-modifiers, improves precision compared to the baseline (vn). In fact, there is a statistically significant difference between this version and vn for all 4 evaluation measures. Version vAa1, the version which adds noun-adjective pairs is slightly worse than vPa1 if we look at precision but the noun-adjective pairs seem to have a positive effect on recall (see recall1000). If we combine the two types of head-modifier pairs (version vAPa1) we get the best overall results.

We may conclude that adding head-modifier pairs to the index can improve retrieval performance, but only if all constituents are also added as separate index terms. Although c4fow, the version which does not use any syntactic information, performs fairly well, especially when we look at recall, we are able to improve results even further by adding syntactic information.

7 Conclusions and future work

The results of our experiments have shown that it is possible to improve retrieval quality for Dutch texts significantly by using syntactic information to create complex index terms. Without using syntactic information we were already able to improve recall by up to 15%, but by adding syntactic information we are not only able to improve recall even further (up to 25%) but we are also able to improve precision as well (up to 16%), provided that all subparts of the complex terms are also added to the index separately. For the experiments described above we used a standard *tf.idf* term weighting scheme which does not differentiate between simple and complex index terms. Since term re-weighting schemes have proven to be successful in previous UPLIFT experiments, we intend to investigate the effect of alternative weighting strategies in the future. We also plan to adapt our strategy to English texts and investigate cross-language retrieval.

Acknowledgements

The research described in this paper was funded by Philips Research, TNO-TPD, the Utrecht Institute of Linguistics OTS, the NBBI, the Dutch Ministry of Education and Science and the Dutch Ministry of Economics. The authors would like to thank Herbert Ruessink and Dirk Heylen for their contribution to this work and the reviewers for their helpful comments on a draft of this paper.

References

Aalbersberg, I. J., E. Brandsma, and M. Corthout (1991). Full text document retrieval: from theory to applications. In G. Kempen and W. de Vroomen (Eds.), *Informatiewetenschap 1991, Wetenschappelijke bijdragen aan de eerste STINFON-Conferentie.*

Armstrong, S., P. Bouillon, and G. Robert (1995). Tools for part-of-speech tagging. Multext project report, ISSCO, Geneva.

Baayen, R. H., R. Piepenbrock, and H. van Rijn (Eds.) (1993). *The CELEX Lexical Database (CD-ROM).* University of Pennsylvania, Philadelphia (PA): Linguistic Data Consortium.

Evans, D. A. and C. Zhai (1996). Noun-phrase analysis in unrestricted text for information retrieval. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics (ACL96)*, pp. 17–24.

Fagan, J. L. (1987). *Experiments in Automatic Phrase Indexing for Document Retrieval: A Comparison of Syntactic and non-Syntactic Methods.* Ph. D. thesis, Cornell University, Ithaca NY, CS Department technical report 87-868.

Geerts, G., W. Haeseryn, J. de Rooij, and M. C. van der Toorn (Eds.) (1984). *Algemene Nederlandse Spraakkunst.* Groningen: Wolters Noordhoff.

Harman, D. (1993). Overview of the first text retrieval conference (TREC-1). In D. Harman (Ed.), *The First Text REtrieval Conference (TREC-1)*, pp. 1–20. National Institute for Standards and Technology. Special Publication 500-207.

Hays, W. L. (1978). *Statistics for the Social Sciences.* London: Holt, Rinehart and Winston.

Hull, D. A., G. Grefenstette, B. M. Schulze, E. Gaussier, H. Schütze, and J. O. Pedersen (1997). Xerox TREC-5 site report: Routing, filtering, NLP and Spanish tracks. In D. Harman (Ed.), *The Fifth Text REtrieval Conference (TREC-5).* to appear (see http://www-nlpir.nist.gov/TREC/).

Kraaij, W. and R. Pohlmann (1996). Viewing stemming as recall enhancement. In H.-P. Frei, D. Harman, P. Schauble, and R. Wilkinson (Eds.), *Proceedings of ACM-SIGIR96*, pp. 40–48.

Salton, G. (1989). *Automatic Text Processing - The Transformation, Analysis, and Retrieval of Information by Computer.* Reading (MA): Addison-Wesley Publishing Company.

Salton, G., C. Buckley, and M. Smith (1990). On the application of syntactic methodologies in automatic text analysis. *Information Processing & Management 26*(1), 73–92.

Smeaton, A. F., R. O'Donnell, and F. Kelledy (1995). Indexing structures derived from syntax in TREC-3: System description. In D. Harman (Ed.), *Overview of The Third Text REtrieval Conference (TREC-3)*, pp. 55–63. National Institute for Standards and Technology. Special Publication 500-225.

Strzalkowski, T. (1995). Natural language information retrieval. *Information Processing & Management 31*(3), 397–417.

Strzalkowski, T. and J. Perez Carballo (1996). Natural language information retrieval: TREC-4 report. In D. Harman (Ed.), *The Fourth Text REtrieval Conference (TREC-4).* National Institute for Standards and Technology. Special Publication 500-236.

Tague-Sutcliffe, J. (1995a). *Measuring Information, An Information Services Perspective.* San Diego (CA): Academic Press.

Tague-Sutcliffe, J. (1995b). A statistical analysis of the TREC-3 data. In D. Harman (Ed.), *Overview of the Third Text REtrieval Conference (TREC-3)*, pp. 385–398. National Institute for Standards and Technology. Special Publication 500-225.

ter Stal, W. (1996). *Automated Interpretation of Nominal Compounds in a Technical Domain.* Ph. D. thesis, Technische Universiteit Twente, UT Repro, Enschede.

van Surksum, J. and J. W. den Besten (1993, november). Patz'er - een patronenzoeker voor Nederlandstalige teksten. TNO-TPD/Hogeschool Enschede.

Vosse, T. G. (1994). *The Word Connection.* Ph. D. thesis, Rijksuniversiteit Leiden, Neslia Paniculata Uitgeverij, Enschede.

Appendix: results of the statistical analysis

The design chosen for the statistical analysis is a repeated measures single factor design, sometimes also referred to as randomized block design (see, for instance, Hays (1978), chapter 13). This design has the advantage that the query (or subject) effect is separated from the system effect. We know that different queries will render different results so if we separate this effect from the system effect we are able to single out the factor we are interested in. The statistical model for the randomized block design can be summarized as follows:

(1) $$Y_{ij} = \mu + \alpha_i + \beta_j + \epsilon_{ij}$$

Y_{ij} represents the score (e.g. average precision) for system variant i and query j, μ is the overall average score, α_i is the effect of the ith system, β_j is the effect of the jth query and ϵ is the random variation about the average.

The H_0 hypothesis which is tested by an analysis of variance (ANOVA) is:

The averages of the observed statistic are equal for all system versions, i.e. the system effect (α) is zero.

If this hypothesis is falsified, we can conclude that at least one pair of averages differs significantly. T-tests are subsequently applied to determine which pairs of system versions really show a significant difference. Tables 8, 9, 10 and 11 present the results of the ANOVAs that were run on the data.

Source	DF	Sum of Squares	Mean Square	F val
system	22	0.6146	0.0279	3.9557
queries	35	35.5577	1.0159	143.8590
error	770	5.4378	0.0071	
total	827	41.6101		
s.e.d. (systems): 0.0198				

Table 8: ANOVA table average precision

Source	DF	Sum of Squares	Mean Square	F val
system	22	1.1607	0.0528	4.6266
queries	35	65.6220	1.8749	164.4138
error	770	8.7808	0.0114	
total	827	75.5635		
s.e.d. (systems): 0.0252				

Table 9: ANOVA table average precision at 5, 10 and 15 documents retrieved

The most important figures in the ANOVA tables are the F-values in the right-most column, which represent the quotient of the variance in measurements which can be attributed to the effect we are interested in (Mean Square system or query) and the variance due to chance (Mean Square error). This quotient is dependent on the degrees of freedom (DF) of the variables in the model, i.e. number of system

Source	DF	Sum of Squares	Mean Square	F val
system	22	1.0440	0.0475	6.1685
queries	35	27.4588	0.7845	101.9800
error	770	5.9237	0.0077	
total	827	34.4265		
s.e.d. (systems): 0.0207				

Table 10: ANOVA table R-recall

Source	DF	Sum of Squares	Mean Square	F val
system	22	3.2825	0.1492	7.3180
queries	35	15.4449	0.4413	21.6433
error	770	15.6995	0.0204	
total	827	34.4269		
s.e.d. (systems): 0.0337				

Table 11: ANOVA table recall at 1000 documents

versions and queries − 1. Because the F values exceed $F_{.99;22,770}$[10] $= 1.85$, we may conclude that the system effect is significant at the 0.99 level for all ANOVAS, This means that we can reject the hypotheses that the system effects of the corresponding measures are equal to zero with a certainty of 99%. The query effect is also clearly significant for all evaluation measures. The F-values exceed $F_{.99;35,770} = 1.55$. This justifies the choice for a randomized block design where the query effect is separated from the system effect.

Because the ANOVA shows that there are significant differences between system versions, it is necessary to do multiple pairwise comparisons to detect which specific versions are concerned. We have used T-tests to identify significant differences between specific versions. The standard error of difference (s.e.d.) values rendered by the ANOVA are used to discriminate significantly different versions in the following way:

$$\mid \bar{x}_1 - \bar{x}_2 \mid > 2 \times s.e.d. \qquad (2)$$

The results of the T-tests are given in tables 3, 4, 5 and 6 in section 6 above.

[10]The standard value for significance level 1−0.01 and the degrees of freedom.