Comparing the Effect of Syntactic vs. Statistical Phrase Indexing Strategies for Dutch

Wessel Kraaij[1] and Renée Pohlmann[2]

[1] Institute of Applied Physics
Netherlands Organisation for Applied Scientific Research (TNO)
Delft
The Netherlands
kraaij@tpd.tno.nl
[2] Utrecht Institute of Linguistics OTS
Utrecht University
Utrecht
The Netherlands
Renee.C.Pohlmann@let.uu.nl
http://www-uilots.let.uu.nl/~uplift

Abstract. In this paper we describe the results of experiments contrasting syntactic phrase indexing with statistical phrase indexing for Dutch texts. Our results showed that we at least need a compound splitting algorithm for good quality retrieval for Dutch texts. If we then add either syntactic or statistical phrases, performance generally improves, but this effect is never statistically significant. If we compare syntactic vs. statistical phrase indexing, syntactic phrases are slightly superior to statistical phrases, particularly at high precision. At higher recall levels syntactic and statistical phrases are equally effective. However, since a compound splitting algorithm requires a dictionary and knowledge about constraints on compound formation, a purely non-linguistic indexing strategy, with or without phrases, does not seem to be very effective for Dutch.

1 Introduction

It is common practice in Information Retrieval (IR) to use phrases as indexing terms in order to enhance precision. 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 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 linguistic processing. So far, both types of strategies have proven to be successful in improving retrieval effectiveness (cf. e.g. [5] and [15]), but comparisons between the two approaches have not been

able to show much difference between them (cf. e.g. [6], [13] and, more recently, [9] and [11]).

In [12] we reported on the results of our experiments with syntactic phrase indexing for Dutch monolingual retrieval. These experiments showed that it is possible to improve retrieval effectiveness for Dutch texts by using syntactic information to create complex index terms. In this paper we will describe the results of a complementary experiment with phrases based on statistical co-occurrence data. The rest of this paper is organized as follows: In section 2 we will summarize our earlier experiment with syntactic phrases. In sections 3 and 4 we will present the results of our experiments with statistical phrases and in section 5 we will formulate our conclusions and give some directions for further research.

2 Syntactic Phrase Experiment

The experiments described in [12] were inspired by similar work for English by Strzalkowski (cf. [14]). Strzalkowski uses a statistical IR system extended with an NLP module which extracts phrases from both queries and documents in addition to single word terms. To create a uniform representation for semantically similar but syntacticly different constructions, e.g. *retrieval of information* vs. *information retrieval*, the phrases are normalized as head-modifier pairs. Strzalkowski's NLP module generates parse trees for entire sentences and extracts the following head-modifier pairs from these structures:

1. a head noun of a noun phrase and its left adjective or noun adjunct
2. a head noun and the head of its right adjunct
3. the main verb of a clause and the head of its object phrase
4. the head of the subject phrase and the main verb

We used the same basic design for our experiments with syntactic phrase indexing. We extended the TRU vector space retrieval engine [1] with an NLP module consisting of the Multext part-of-speech tagger [2], a stemmer based on the CELEX lexical database [3] and the Patz'er NP-parser [18]. We evaluated a number of indexing strategies using different criteria for the extraction of head-modifier pairs from the texts and for the addition of phrase constituents as separate index terms. Our parser, however, does not generate parse trees for entire sentences but only identifies and assigns structure to noun phrases. We therefore only used a subset of the types of head-modifier pairs that Strzalkowski uses (i.e. categories 1 and 2). Like Strzalkowski, we experimented with using corpus statistics to assign head-modifier structure to ambiguous noun phrases. We collected frequency information for unambiguous head-modifier pairs in our corpus and used this information to select the most probable structure in case of ambiguity. We also experimented with assigning a default structure to ambiguous noun phrases. Figure 1 gives an example of head-modifier pair extraction from a complex NP.

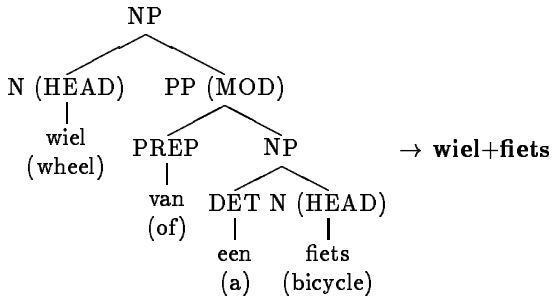

Fig. 1. head-modifier pair extraction from NP with PP modifier

In Dutch, compound nouns are usually written as a single orthographic unit, e.g. *levensverzekeringsmaatschappij* (life insurance company). As a result of this, compound constituents would normally not be treated as separate index terms by our basic retrieval engine. Earlier research in our project on stemming algorithms [10] indicated that splitting up compounds in queries and using the constituents for query expansion improves recall. We therefore incorporated a compound splitter in our NLP module and experimented with adding compound constituents as separate index terms and reducing complex compounds to head-modifier pairs. Figure 2 gives an example of head-modifier pair extraction from a complex compound.

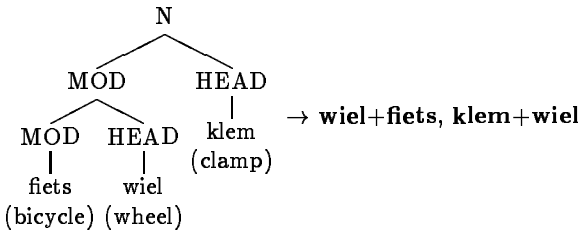

Fig. 2. head-modifier pair extraction from complex compound

The compound splitter was originally developed by Vosse [19] as part of a spelling checker. The splitter will try to split a compound into its components (stems) working from left to right on the basis of a well-formedness table for Dutch compounds and a lexicon. The lexicon used is again CELEX. The well-formedness table used by the splitter is reproduced in Table 1 below (cf. [19] p. 65), 'x' means that a combination is allowed and '?' that a combination is doubtful but not always disallowed.

Apart from the well-formedness table a number of simple heuristics are used to avoid unwanted or unlikely analyses, e.g. compounds analyzed as consisting

Table 1. Well-formedness table for Dutch compounds

WF	1	2	3	4	5	6	7	8	9	10
1	x	x	x	x	x	x	x	x	x	x
2	?	?	?	?	?	?	?	?	?	?
3										
4	x	x	x	x	x	x	x	x	x	x
5										
6	x	x	x	x	x	x	x	x	x	x
7				x	x	x	x	x	x	x
8	x	x	x	x	x	x	x	x	x	x
9										
10										
11	x	x	x	x	x	x	x	x	x	x
12	x	x	x	x	x	x	x	x		
13										

1 = verb (stem), 2 = verb (infinitive), 3 = verb (any other form), 4 = noun (stem), 5 = noun (diminutive singular), 6 = noun (plural), 7 = noun (diminutive plural), 8 = adjective (stem), 9 = adjective (any other form), 10 = adverb, 11 = number, 12 = preposition, 13 = rest (pronouns, interjections ...)

of a large number of two- or three-letter words. If the splitter is unsuccessful, the word is left unchanged. The following results were obtained with the compound splitter using a random sample of approximately 1,000 compounds not included in the CELEX dictionary[1]:

- 5% no analysis
- 3% incorrect analysis
- 92% correct analysis

The test collection used for the experiments described in this paper consists of a document collection of 59,608 articles published in three Dutch regional newspapers (*Het Eindhovens Dagblad*, *Het Brabants Dagblad* and *Het Nieuwsblad*) from January to October 1994 and 66 queries and relevance judgements. 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. Our test collection previously consisted of 36 queries and relevance judgements. We conducted a new user experiment and extended the number of queries to 66. The results quoted in this paper were obtained with the larger test corpus and therefore differ slightly from those published in earlier work (cf. [10] [12]). Some general statistics for the document collection are given in Table 2 below.

We used 4 different measures to evaluate retrieval performance for the experiment. These measures are: average precision, ap5-15 (precision at 5, 10 and

[1] Some frequent compounds are included in the CELEX dictionary.

Table 2. Document collection statistics

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

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). We used the so-called *pooling method*[2] to compute the number of relevant articles per query. 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 [16] and [17]. Details on the statistical tests can be found in the appendix.

The results of our experiment indicated that adding head-modifier pairs to the index can improve retrieval effectiveness. The best overall results (16% improvement in average precision, 15% improvement in ap5-15, 22.7% improvement in R-recall and 16.8% improvement in recall1000, compared to the performance of our basic retrieval engine) were obtained by adding all noun-noun and noun-adjective head-modifier pairs from complex NPs (including complex compounds) to the index while, at the same time, adding all constituents as separate index terms.

We decided that it would be interesting to compare the results obtained in the syntactic phrase indexing experiment with results obtained using statistical phrases for indexing. As discussed in section 1 above, most researchers so far have found little or no difference in performance between linguistic and statistical phrase indexing methods for English test corpora. We wanted to see whether this result would also hold for our Dutch test corpus.

3 Statistical Phrase Experiment I

Our statistical phrase experiment was modeled on similar experiments by the SMART group with the TREC test collection [4]. Their definition of a statistical phrase is "any pair of adjacent non-stopwords which occurs in more than 25 documents of the TREC 1 document set". Because our document collection is significantly smaller than the TREC 1 collection (60.000 vs. 211.000 documents) we experimented with different document frequency thresholds for the statistical phrases. We set the threshold at 1-10, 15, 20 and 25. 3 turned out to be the best option for our corpus. We decided to take a system version with stemming but

[2] See [7] p. 9 *ff.*

without proper name recognition[3] (vBase) as a starting point for our statistical phrase experiments. In Table 3 the results of our experiment are summarized. For reference, the results for vTRU (the TRU vector space engine with no extensions at all) and vBase+prop (a version with proper name recognition) are also shown. vSyn is the best syntactic version of our previous experiments as described in section 2 above[4]. The percentages indicate improvement/decrease compared to the performance of the baseline (vBase).

Table 3. Evaluation measures averaged over queries

version	avp	% change	ap5-15	% change
vTRU	0.282	− 9.0	0.431	− 0.9
vBase	0.310		0.435	
vBase+prop	0.311	+ 0.3	0.437	+ 0.4
vStat	0.323	+ 4.2	0.451	+ 3.7
vStat+weight	0.327	+ 5.6	0.451	+ 3.7
vSyn	0.370	+ 19.2	0.496	+ 14.0
version	R-recall	% change	recall1000	% change
vTRU	0.311	− 6.4	0.785	− 8.2
vBase	0.333		0.855	
vBase+prop	0.336	+ 1.0	0.865	+ 1.2
vStat	0.345	+ 3.6	0.862	+ 0.9
vStat+weight	0.352	+ 5.9	0.863	+ 0.9
vSyn	0.382	+ 14.8	0.917	+ 7.2

It is immediately clear that the performance of the statistical version, although it is slightly better than vBase, is no way near the performance of the syntactic version. Decreasing the weight of the statistical phrases (vStat+weight), a common strategy in experiments with phrase indexing, slightly improves results for average precision and R-recall but the difference with vSyn is still considerable[5].

We decided to do a detailed per query analysis to find out what caused this difference in behaviour between the syntactic and the statistic versions. We identified queries that performed much better or worse than average for the different evaluation measures and system versions and we subsequently looked at these queries in detail. We produced lists which contained the scores and rank for each relevant and non-relevant document retrieved (cut-off at 200), the query terms that matched and their weight and the rank assigned to documents by the

[3] Our proper name recognition algorithm simply glues adjacent words with capitals together. The statistical phrase algorithm should also find the most frequent multi word proper names.

[4] Referred to as vAP1 in [12].

[5] And statistically significant, i.e. they are never in the same equivalence class, cf. the appendix for the results of statistical tests.

reference version. These lists made it easy to inspect the effect of individual query terms and phrases.

This detailed analysis revealed that compound splitting was probably responsible for most of the difference between the statistical and syntactic versions and that phrases only played a secondary role. Recall that, unlike the statistical versions, our syntactic version not only adds phrases but also splits up compounds in both queries and documents (see section 2 above). Compound constituents turned out to be very good search terms for quite a number of queries. Query 55, for example, contains the compound *theatervoorstelling* (theater performance). Of the 128 relevant documents for this query, 40 only contain (one of) the compound constituents *theater* or *voorstelling* but not the compound.

Furthermore, the compound splitting strategy has a secondary effect of emphasizing the compounds in queries and documents. Every compound is effectively doubled (they occur once as a compound but also as separate constituents). Although the results of our previous experiments with syntactic phrase indexing indicated that this effect is not solely responsible for improved performance[6], we decided to run a second experiment with statistical versions, which split up compounds as well as add phrases, so that we could compare the two strategies for phrase indexing and separate out the effect of adding phrases more accurately.

4 Statistical Phrase Experiment II

In this second experiment we tested two new statistical versions, one version (vStat+split) which besides adding statistical phrases also splits up compounds and, in order to separate out the effect of "compound doubling" as described in section 3 above more clearly, we also tested a second variant (vStat+replace) where compounds are *replaced* by their constituents[7]. We also added two new versions without phrases, one with compound doubling (vBase+split) and one with compound replacement (vBase+replace). Table 4 gives the results for this second experiment.

The results of the second experiment confirm our hypothesis that compound splitting is crucial for good quality retrieval for Dutch. Compound splitting alone (with or without doubling) is clearly responsible for most of the improvement in retrieval performance. Emphasizing compounds without adding phrases (vBase+split) always has a positive effect, especially at higher recall levels. Adding syntactic phrases (vSyn) always improves performance compared to vBase+split (although the difference is never statistically significant). The statistical versions with compound splitting (vStat+split and vStat+replace) are much better than the versions without splitting. In most cases, their performance is now comparable to the syntactic version. If we look at high precision

[6] In [12] we compared versions which only split up compounds with versions which also add phrases. The best phrase version (also used in this experiment) clearly outperformed the best split version.

[7] Although we later found that some compounds are still doubled because vStat+replace reunites the constituents in the statistical phrase formation process.

Table 4. Evaluation measures averaged over queries

version	avp	% change	ap5-15	% change
vBase	0.310		0.435	
vStat	0.323	+ 4.2	0.451	+ 3.7
vBase+split	0.364	+ 17.3	0.483	+ 11.0
vStat+split	0.363	+ 17.0	0.466	+ 7.1
vSyn(+split)	0.370	+ 19.2	0.496	+ 14.0
vBase+replace	0.349	+ 12.7	0.482	+ 10.8
vStat+replace	0.365	+ 17.6	0.481	+ 10.5
version	R-recall	% change	recall1000	% change
vBase	0.333		0.855	
vStat	0.345	+ 3.6	0.862	+ 0.9
vBase+split	0.376	+ 13.1	0.905	+ 5.9
vStat+split	0.380	+ 14.4	0.904	+ 5.8
vSyn(+split)	0.382	+ 14.8	0.917	+ 7.2
vBase+replace	0.366	+ 10.0	0.897	+ 4.9
vStat+replace	0.372	+ 11.7	0.902	+ 5.5

(ap5-15), however, the difference between the syntactic version and the two new statistical versions is more prominent (but not statistically significant).

A preliminary conclusion might be that statistical phrases mainly have an effect at higher recall levels, whereas syntactic phrases improve both (high) precision **and** recall. We assume that the higher precision of the syntactic version is caused by the more controlled method of phrase formation. A possible explanation for he difference in performance between vStat+split and vStat+replace at high precision might be a weighting effect. Additional index terms make a document (or query) longer, which results in lower weights for each index term because of our retrieval engine's cosine document length normalization. This has the effect of a deterioration of precision for queries where the additional phrases do not result in extra matches.

5 Conclusions and Further Plans

Our experiments have shown that compound splitting is essential for good quality retrieval of Dutch texts. If we assume a baseline of compound splitting and add either syntactic or statistical phrases, performance generally improves, but this effect is never statistically significant. If we compare syntactic vs. statistical phrase indexing, syntactic phrases are slightly superior to statistical phrases, particularly at high precision. At higher recall levels syntactic and statistical phrases are equally effective. However, since compound splitting requires at least a dictionary and knowledge about constraints on compound formation, a purely non-linguistic indexing strategy, with or without phrases, does not seem to be very effective for Dutch.

In a new experiment we intend to investigate whether an extension of the scope of syntactic analysis to larger units than noun phrases would lead to further improvements. Our results seem to indicate that our method to integrate phrases in the vector space index, i.e. by simply adding them to the document term vectors, might not be ideal. Document length normalization has the effect of a deterioration of precision for queries where the additional phrases do not result in extra matches. Since we have also seen that downweighting phrases can help to improve average precision (see section 3), we also intend to experiment with separate indexes for single words and phrases (cf. [11] for an example of such an approach) so that we can control these effects more precisely.

References

1. IJsbrand Jan Aalbersberg, Ewout Brandsma, and Marc Corthout. Full text document retrieval: from theory to applications. In G.A.M. Kempen and W.A.M. de Vroomen, editors, *Informatiewetenschap 1991, Wetenschappelijke bijdragen aan de eerste STINFON-Conferentie*, 1991.
2. Susan Armstrong, Pierrette Bouillon, and Gilbert Robert. Tools for part-of-speech tagging. Multext project report, ISSCO, Geneva, 1995.
3. R. H. Baayen, R. Piepenbrock, and H. van Rijn, editors. *The CELEX Lexical Database (CD-ROM).* Linguistic Data Consortium, University of Pennsylvania, Philadelphia (PA), 1993.
4. C. Buckley, G. Salton, and J. Allan. Automatic retrieval with locality information using SMART. In Donna Harman, editor, *The First Text REtrieval Conference (TREC-1)*, pages 59–73. National Institute for Standards and Technology, 1993. Special Publication 500-207.
5. C. Buckley, A. Singhal, M. Mitra, and (G. Salton). New retrieval approaches using SMART:TREC4. In Donna Harman, editor, *The Fourth Text REtrieval Conference.* National Institute for Standards and Technology, 1996. Special Publication 500-236.
6. J. L. Fagan. *Experiments in Automatic Phrase Indexing for Document Retrieval: A Comparison of Syntactic and non-Syntactic Methods.* PhD thesis, Cornell University, Ithaca NY, CS Department technical report 87-868, 1987.
7. Donna Harman. Overview of the first Text REtrieval Conference (TREC-1). In Donna Harman, editor, *The First Text REtrieval Conference (TREC-1)*, pages 1–20. National Institute for Standards and Technology, 1993. Special Publication 500-207.
8. William L. Hays. *Statistics for the Social Sciences.* Holt, Rinehart and Winston, London, 1978.
9. David A. Hull, Gregory Grefenstette, B. Maximilian Schulze, Eric Gaussier, Hinrich Schütze, and Jan O. Pedersen. Xerox TREC-5 site report: Routing, filtering, NLP and Spanish tracks. In Donna Harman and Ellen Voorhees, editors, *The Fifth Text REtrieval Conference (TREC-5).* National Institute for Standards and Technology, 1997. Special Publication 500-238.
10. Wessel Kraaij and Renée Pohlmann. Viewing stemming as recall enhancement. In Hans-Peter Frei, Donna Harman, Peter Schauble, and Ross Wilkinson, editors, *Proceedings of the 19th ACM-SIGIR Conference on Research and Development in Information Retrieval (SIGIR96)*, pages 40–48, 1996.

11. Mandar Mitra, Chris Buckley, Amit Singhal, and Claire Cardie. An analysis of statistical and syntactic phrases. In L. Devroye and C. Chrisment, editors, *Proceedings of RIAO'97*, pages 200–214, 1997.
12. Renée Pohlmann and Wessel Kraaij. The effect of syntactic phrase indexing on retrieval performance for Dutch texts. In L. Devroye and C. Chrisment, editors, *Proceedings of RIAO'97*, pages 176–187, 1997.
13. Gerard Salton, Chris Buckley, and Maria Smith. On the application of syntactic methodologies in automatic text analysis. *Information Processing & Management*, 26(1):73–92, 1990.
14. Tomek Strzalkowski. Natural language information retrieval. *Information Processing & Management*, 31(3):397–417, 1995.
15. Tomek Strzalkowski and Jose Perez Carballo. Natural language information retrieval: TREC-4 report. In Donna Harman, editor, *The Fourth Text REtrieval Conference (TREC-4)*. National Institute for Standards and Technology, 1996. Special Publication 500-236.
16. Jean Tague-Sutcliffe. *Measuring Information, An Information Services Perspective*. Academic Press, San Diego (CA), 1995.
17. Jean Tague-Sutcliffe. A statistical analysis of the TREC-3 data. In Donna Harman, editor, *Overview of the Third Text REtrieval Conference (TREC-3)*, pages 385–398. National Institute for Standards and Technology, 1995. Special Publication 500-225.
18. Joost van Surksum and Jan Willem den Besten. Patz'er - een patronenzoeker voor Nederlandstalige teksten. TNO-TPD/Hogeschool Enschede, november 1993.
19. T. G. Vosse. *The Word Connection*. PhD thesis, Rijksuniversiteit Leiden, Neslia Paniculata Uitgeverij, Enschede, 1994.

Appendix A: Statistical Significance Tests

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, [8], 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:

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

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 5, 6, 7 and 8 present the results of the ANOVAs that were run on the data.

Table 5. ANOVA table average precision

Source	DF	Sum of Squares	Mean Square	F val
system	9	0.5311	0.0590	6.3349
queries	65	28.3105	0.4355	46.7565
error	585	5.4494	0.0093	
total	659	34.2910		
s.e.d. (systems): 0.0168				

Table 6. ANOVA table average precision at 5, 10 and 15 documents retrieved

Source	DF	Sum of Squares	Mean Square	F val
system	9	0.3208	0.0356	2.8349
queries	65	50.0273	0.7697	61.2041
error	585	7.3565	0.0126	
total	659	57.7046		
s.e.d. (systems): 0.0195				

Table 7. ANOVA table R-recall

Source	DF	Sum of Squares	Mean Square	F val
system	9	0.3367	0.0374	3.8909
queries	65	24.3800	0.3751	39.0073
error	585	5.6251	0.0096	
total	659	30.3418		
s.e.d. (systems): 0.0171				

Table 8. ANOVA table recall at 1000 documents

Source	DF	Sum of Squares	Mean Square	F val
system	9	0.8992	0.0999	11.3027
queries	65	9.9526	0.1531	17.3223
error	585	5.1710	0.0088	
total	659	16.0227		
s.e.d. (systems): 0.0164				

The most important figures in the ANOVA tables are the F-values in the rightmost 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 versions and queries $-$ 1. Because the F values exceed $F_{.99;9,585}$[8] $= 2.48$, 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 $F_{.99;65,585} = 1.55$. 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:

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

The results of the T-tests are given in Tables 9, 10, 11 and 12 below. In these tables system versions have been divided into equivalence classes indicated by numbers.

[8] The standard value for significance level 1$-$0.01 and the degrees of freedom.

Table 9. Equivalence classes avp

system	avp			
vSyn	0.370	1		
vStat+replace	0.365	1		
vBase+split	0.364	1		
vStat+split	0.363	1		
vBase+replace	0.349	1	2	
vStat+weight	0.327		2	3
vStat	0.323		2	3
vBase+prop	0.311			3 4
vBase	0.310			3 4
vTRU	0.282			4

Table 10. Equivalence classes ap5-15

system	ap5-15			
vSyn	0.496	1		
vBase+split	0.483	1	2	
vBase+replace	0.482	1	2	
vStat+replace	0.481	1	2	
vStat+split	0.466	1	2	3
vStat+weight	0.451		2	3
vStat	0.451		2	3
vBase+prop	0.437			3
vBase	0.435			3
vTRU	0.431			3

Table 11. Equivalence classes R-recall

system	R-r				
vSyn	0.382	1			
vStat+split	0.380	1			
vBase+split	0.376	1	2		
vStat+replace	0.372	1	2		
vBase+replace	0.366	1	2	3	
vStat+weight	0.352	1	2	3	
vStat	0.345		2	3	4
vBase+prop	0.336			3	4
vBase	0.333			3	4
vTRU	0.311				4

Table 12. Equivalence classes recall1000

system	r1000			
vSyn	0.917	1		
vBase+split	0.905	1		
vStat+split	0.904	1		
vStat+replace	0.902	1		
vBase+replace	0.897	1	2	
vBase+prop	0.865		2	3
vStat+weight	0.863			3
vStat	0.862			3
vBase	0.855			3
vTRU	0.785			4