# Shallow Morphological Analysis in Monolingual Information Retrieval for Dutch, German and Italian

Christof Monz[⋆] and Maarten de Rijke[⋆⋆]

Language & Inference Technology, University of Amsterdam,
Nieuwe Achtergracht 166, 1018 WV Amsterdam, The Netherlands.
E-mail: {christof,mdr}@science.uva.nl
URL: www.science.uva.nl/∼{christof,mdr}

**Abstract.** This paper describes the experiments of our team for CLEF 2001, which includes both official and post-submission runs. We took part in the monolingual task, for Dutch, German, and Italian. The focus of our experiments was on the effects of morphological analyses such as stemming and compound splitting on retrieval effectiveness. Confirming earlier reports on retrieval in compound splitting languages such as Dutch and German, we found improvements to be around 25% for German and as much as 69% for Dutch. For Italian, lexicon-based stemming resulted in gains of up to 25%.

## 1 Introduction

This is the first year that the University of Amsterdam is participating in the CLEF conference and retrieval comparison. We took part in three monolingual tracks: Dutch, German, and Italian. Each of these languages is morphologically richer than English, and we were particularly interested in the effects of shallow morphological analyses for these languages: stemming or lemmatization, and compound splitting. In languages such as Dutch and German, compound building is a very common issue. For instance, the Dutch noun *zonnecel* (English: *solar cel*) is a compound built from *zon* (English: *sun*) and *cel* (English: *cel*). Previous work has indicated that it may help to enhance retrieval effectiveness for Dutch and German if compounds in queries or documents are split, and their parts added to the query or document. Below, we report on monolingual experiments for Dutch and German that confirm and refine these results using the CLEF data collection. All our experiments were performed using the FlexIR system.

The paper is organized as follows. In Section 2 we describe the FlexIR system as well as our basic retrieval approach. Section 3 is devoted to a detailed description of our techniques for compound splitting, for both Dutch and German. Section 4 describes our official runs for CLEF 2001 and the results we obtained. In Section 5 we discuss the results we have obtained for a small number of post-submission experiments, mainly

[⋆] Supported by the Physical Sciences Council with financial support from the Netherlands Organization for Scientific Research (NWO), project 612-13-001.

[⋆⋆] Supported by the Spinoza project 'Logic in Action' and by grants from the Netherlands Organization for Scientific Research (NWO), under project numbers 612-13-001, 365-20-005, 612.069.006, 612.000.106, and 220-80-001.

concerning the interaction between blind feedback and compound splitting. Finally, in Section 6 we offer some conclusions and outline plans regarding research within our group in the area of document retrieval.

## 2 System Description

All our runs used FlexIR, an information retrieval system developed by the first author. The main goal underlying FlexIR's design is to facilitate flexible experimentation with a wide variety of retrieval components and techniques. FlexIR is implemented in Perl; it is built around the standard UNIX pipeline architecture, and supports many types of preprocessing, scoring, indexing, and retrieval tools.

### 2.1 Approach

The retrieval model underlying FlexIR is the standard vector space model. All our official (and post-hoc) runs for CLEF 2001 used the Lnu.ltc weighting scheme [3] to compute the similarity between a query ($q$) and a document ($d$):

$$sim(q,d) = \sum_{i \in q \cap d} \frac{\frac{1+\log(\mathrm{freq}_{i,d})}{1+\log(\mathrm{avg}_{j\in d}\mathrm{freq}_{j,d})} \cdot \frac{\mathrm{freq}_{i,q}}{\max_{j\in q}\mathrm{freq}_{j,q}} \cdot \log\left(\frac{N}{n_i}\right)}{((1-sl)\cdot pv + sl\cdot \mathrm{uw}_d)\cdot \sqrt{\sum_{i\in q}\left(\frac{\mathrm{freq}_{i,q}}{\max_{j\in q}\mathrm{freq}_{j,q}}\cdot\log\left(\frac{N}{n_i}\right)\right)^2}}$$

For the experiments on which we report in this paper, we fixed *slope* at 0.2; the pivot was set to the average number of unique words per document.

In addition, blind feedback was applied to expand the original query with related terms. Term weights were recomputed by using the standard Rocchio method [15], where we considered the top 10 documents to be relevant and the bottom 250 documents to be non-relevant. We allowed at most 20 terms to be added to the original query. We did not carry out any filtering [11] before applying Rocchio, since some experiments that we carried out on the CLEF 2000 data set indicated a decrease in retrieval effectiveness.

### 2.2 Inflectional Morphology

Previous retrieval experimentation [6] in English did not show consistent significant improvements by applying morphological normalization such as rule-based stemming [14] or lexical stemming [8].

As to the effect of stemming on retrieval performance for languages that are morphologically richer than English, such as Dutch, German, Italian or Spanish, we have a similar mixed picture from CLEF 2000 and other experiments. Kraaij and Pohlmann [9] report that for Dutch the effect of stemming is limited; it tends to help as many queries as it hurts. Likewise, for German and French, reports seem to indicate results similar to those for English [12].

In our participation in this year's edition of CLEF, we focused on Dutch, German and Italian. Although versions of Porter's stemmer are available for each of these languages, we decided to use a lexical-based stemmer, or lemmatizer, where available, because it tends to be less aggressive than rule-based stemmers, and we conjectured that this might benefit further morphological analyses such as compound splitting (see below). For Dutch we used a Porter stemmer developed within the Uplift project [20]. The lemmatizer that we used for German and Italian is part of the TreeTagger part-of-speech tagger [16]. Each word is assigned its syntactic root by lexical look-up. Mainly number, case, and tense information is removed, leaving other morphological processes such as nominalization intact. As an example in German, *Vereinbarung* (English: agreement) and German: *vereinbaren* (English: agree) are not conflated.

## 3 Compound Splitting

Compound splitting (sometimes referred to as decompounding) is not an issue in English since almost all compounds, such as *Computer Science*, *peace agreement*, etc. are separated by a white space, disregarding some exceptions such as *database* or *bookshelf*. In Dutch and German, compounds are not separated and compound building is a very common phenomenon. Kraaij and Pohlmann [10] show that compound splitting leads to significant improvement of retrieval performance for Dutch, and Moulinier et al. [12] obtain similar results for German.

In some of our official runs for Dutch and German we used a compound splitter. Our compound splitter for Dutch was built using the Dutch lexicon provided by Celex [2], while our German compound splitter used the part-of-speech information provided by TreeTagger. There are several forms of compounds, based on different parts-of-speech, including noun-noun (e.g., German: *Straßenbahn*, English: *tram*), verb-noun (e.g., German: *Tankstelle*, English: *gas station*), verb-verb (e.g., German: *spazierengehen*, English: *taking a walk*), noun-adjective (e.g., German: *arbeitslos*, English: *unemployed*), adjective-verb (e.g., German: *sicherstellen*, English: *to secure*); etc., see [4] for a more detailed overview. We decided to limit our compound splitter to noun-noun compounds, since this is the most frequent form of compounding.

To estimate the impact of compound splitting, we first analyzed how frequent the compounding phenomenon is in Dutch and German. We compiled a list of arbitrarily chosen nouns and annotated each noun with its compound parts; Table 1 provides some details on the distribution of compounds.

| *# compound parts* | *# distinct words* | | | |
|---|---|---|---|---|
| | *Dutch* | | *German* | |
| 1 | 410 | (76.6%) | 2156 | (75.5%) |
| 2 | 118 | (22.1%) | 635 | (22.3%) |
| 3 | 7 | (1.3%) | 60 | (2.1%) |
| 4 | 0 | (0.0%) | 2 | (0.1%) |
| Total | 535 | (100%) | 2853 | (100%) |

**Table 1.** Distribution of compounds.

Nouns that cannot be further split up, i.e., having one compound part, form the vast majority of nouns, approximately 75% in both languages. On the other hand, approximately 25% of all nouns are complex and can be decompounded into two or more parts. The differences between Dutch and German are relatively small. The most notable difference is that German contains more compounds of higher complexity, i.e., having more than two compound parts. Of course, the samples we investigated here are far too small to make a strong claim about the distribution of compounds in both languages, nevertheless, we think that they reveal a reasonable approximation of the distribution of compounds.

Since nouns are valuable information carriers, and assuming that, indeed, around 25% of the nouns are more complex, decompounding seems to be an essential component of any information retrieval system for Dutch or German.

## 3.1 Implementation of a Compound Splitter

Our compound splitter works by recursively analyzing each noun to see whether it can be split into a sequence of concatenated nouns. We do allow for a glueing-*s*; e.g., the German compound *Friedensvertrag* (English: *peace agreement*) is split into *Frieden*+s *Vertrag*. Figure 1 shows the pseudo-code for the recursive compound splitting function `split`.

```
string split(string s)
{
  int length = strlen(s);
  string r;
  for(int char_pos=1; char_pos<=length; char_pos++)
  {
    if(substr(1,char_pos,s)∈noun_lex
       && !strcmp(split(substr(char_pos+1,length,s)),''))
    {
      r = split(substr(char_pos+1,length,s));
      return concat(substr(1,char_pos,s),+,r);
    } else if(substr(1,char_pos,s)∈noun_lex
       && strcmp(substr(char_pos+1,char_pos+1,s),'s')
       && !strcmp(split(substr(char_pos+2,length,s)),''))
    {
      r = split(substr(char_pos+2,length,s));
      return concat(substr(1,char_pos,s),+,r);
    };
  };
  if(s∈noun_lex)
    return s;
  else
    return '';
}
```

**Fig. 1.** The algorithm underlying the compound splitter.

The function `split` takes a string, i.e., a potentially complex noun, as argument and it returns a string where the compound boundaries are indicated by a plus sign. For instance, `split(bahnhof)` returns `bahn+hof`. If it cannot split up a string into smaller components it returns the same string, and if it fails to analyze a string at all, it returns the empty string.

A number of things within the pseudo-code in Figure 1 might require further explanation, and we will now discuss the basic components of the function. The `for`-loop in lines 5–19 tries to split the input string at every character position proceeding from left to right. If a noun has been identified as the prefix of the string (line 7), `split` is called again, with the remaining part to the right of the prefix (line 8). If `split` did not fail to analyze this remaining string, the prefix and the analysis of the right part as it was returned by the nested call to `split` is returned as the value of the first call to `split` (line 11). The `else if` block in lines 12–18 is similar to the first `if`-condition, but allows for a glueing-*s* to separate the prefix from the remaining string (line 13). If no further split up was found during the `for`-loop, the input string itself is returned if it could be looked up in a noun lexicon (line 20–21), otherwise, the empty string is returned.

Note that `split` works from left to right, implying that the resulting analysis will be strictly right-branching (or left-branching, depending on how you look at it). For many compounds this is inappropriate from a linguistic point of view. For instance, consider the noun *Autobahnraststätte* (English: *highway restaurant*), among the theoretically possible analyses are (a) and (b) in Figure 2.

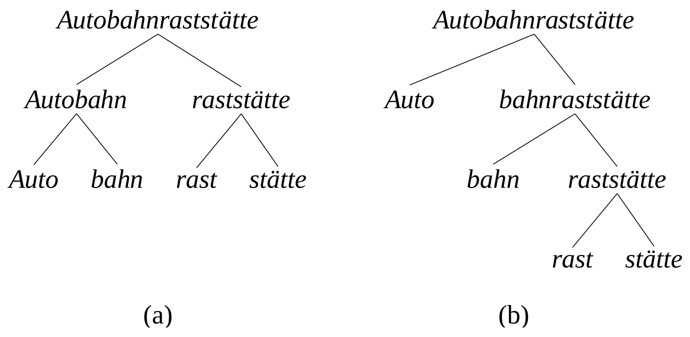

**Fig. 2.** Morphological analyses for the compound *Autobahnraststätte*.

From a linguistic point of view, (a) is clearly preferred over (b), since the intermediate ompound *bahnraststätte* in (b), although syntactically correct, is semantically very awkward. However, since we focus on the leaves of a tree for retrieval purposes, and therefore do not exploit intermediate levels, the difference between (a) and (b) disappears, as both trees have the same set of leaves.

## 3.2 Evaluating the Compound Splitter

Before evaluating the effect of compound splitting in the context of retrieval, in Sections 4 and 5, we consider the quality of the compound splitter itself. To this end we compiled a set of nouns taken from the document collection, and annotated the compound parts of each compound. For each noun, there is a set $S_c$ containing pairs of the form $(p_b, p_e)$, indicating the character position at which a compound part begins and ends. If a noun cannot be further split up, $p_b$ is 1 and $p_e$ is the length of the noun. This set of annotated nouns constitutes our gold standard or reference corpus. In addition to the set of correct compound parts, there is a set $S_g$ of compound parts produced by the compound splitter.

The bracketed match $B$ is the number of pairs that $S_c$ and $S_g$ have in common, i.e., $B = \#(S_c \cap S_g)$. Analogous to the evaluation of syntactic parsers [7] and retrieval systems, the quality of the compound splitter is measured in terms of precision and recall. Both measures are further divided into micro-average precision/recall and macro-average precision/recall, depending on the way the average is computed, see Figure 3, where $N_c$ and $N_g$ are the size of $S_c$ and $S_g$, respectively.

$$\textit{micro-avg. precision} = \frac{\sum_{\textit{Nouns}} B/N_c}{\#\textit{Nouns}} \qquad \textit{micro-avg. recall} = \frac{\sum_{\textit{Nouns}} B/N_g}{\#\textit{Nouns}}$$

$$\textit{macro-avg. precision} = \frac{\sum_{\textit{Nouns}} B}{\sum_{\textit{Nouns}} N_c} \qquad \textit{macro-avg. recall} = \frac{\sum_{\textit{Nouns}} B}{\sum_{\textit{Nouns}} N_g}$$

**Fig. 3.** Evaluation measures for compound splitting.

Precision is the percentage of compound parts returned by the compound splitter that are correct, and recall is the percentage of correct compound parts which have been identified by the compound splitter. For micro averaging, precision and recall are computed for each noun and then they are averaged over the number of nouns. For macro-averaging, precision and recall are computed with respect to the union of all brackets of all nouns. For example, consider two nouns $A$ and $B$ with compound parts $(a_1)$ and $(b_1)(b_2)(b_3)$, where the compound splitter returns $(a_1)$ and $(b_1)(b_2b_3)$. Precision and recall for $A$ are 1, precision for $B$ is 0.5, and recall for $B$ is $0.\overline{3}$. Then, micro-average precision is $(1+0.5)/2 = .75$, and micro-average recall is $(1+0.\overline{3})/2 = 0.\overline{6}$. Macro-average precision is $2/3 = 0.\overline{6}$, since 3 brackets have been assigned in total, of which two were correct, and macro-average recall is $2/4 = 0.5$, because 4 brackets were to be found in total, of which 2 were actually found by the compound splitter.

We decided to use macro-averaging in addition to the more commonly used micro-averaging, because it is more susceptible to small differences in precision and recall.

Table 2 shows the results of the compound splitter for Dutch and German.

| | | *Dutch* | | *German* | |
|---|---|---|---|---|---|
| | | *micro-avg.* | *macro-avg.* | *micro-avg.* | *macro-avg.* |
| *# all nouns* | *precision* | 79.7% | 77.4% | 86.1% | 84.9% |
| | *recall* | 79.7% | 70.2% | 86.0% | 79.1% |
| *# complex nouns* | *precision* | 27.1% | 37.6% | 49.3% | 60.5% |
| | *recall* | 27.3% | 29.3% | 49.0% | 50.6% |

**Table 2.** Evaluation of the compound splitter.

The compound splitter was evaluated with respect to two kinds of sets, first all nouns, i.e., 553 Dutch nouns and 2853 German nouns, cf. Table 1, and second, only nouns which were classified as complex during manual annotation or by the compound splitter. The second set is used to deemphasize the effect of simple nouns and to focus on the complex cases, where compound splitting actually matters. It is worth noting that there is basically no difference between precision and recall when using micro-averaging. This is mainly due to the small number of partially correct decompoundings. In most cases a noun is either correctly or incorrectly split up, which is again due to the small number of nouns with a higher number of compound parts; only 1.3% of the Dutch nouns and 2.2% of the German nouns can be split into more than two components, cf. Table 1. Since macro-averaging averages over the total number of brackets instead of the number of nouns, it is more susceptible to smaller differences, and indeed reveals a better distinction between precision and recall. In both languages, precision is considerably higher than recall, which can be explained by the conservative splitting strategy used. For instance, nouns having *es* as a gluing infix, such as *Landesregierung* (English: *state government*) are not identified by the splitting algorithm as it was described in Figure 1, but it should be trivial to enhance the current algorithm to do so. Of course, there are also non-trivial cases such as *Augapfel* (English: *eyeball*) whoes correct splitting is *Auge*+*Apfel*, where the letter *e* has been removed during compound building and during decompounding has to be added again in order to recognize *Auge* as an existing noun; see [1] for more heuristics in German decompounding. Since these rules for compound splitting are not considered in the algorithm, recall decreases.

Another noteworthy thing in Table 2 is the difference in precision and recall for Dutch and German. The compound splitter performs much worse for Dutch than for German. Although we do not have a clear explanation for this, it could be the different lexicons used for recognizing simple nouns. As mentioned above, the lexicon for German is compiled by tagging the input text and simply storing all lemmas of words which have been tagged as noun. Since we did not have access to a part-of-speech tagger for Dutch, we used Celex as a lexicon, which is somewhat smaller than the tagger-based lexicon used for German, and it is also not based on the terminology used in the actual document collection. The major problem for Dutch decompounding is that the lexicon does not contain plurals, e.g. *klantenservice* (English: *customers service*) is to be split into *klanten*+*service*, where *klanten* is the plural of *klant*. Unfortunately, plurals are not mentioned in Celex, and therefore not recognized as nouns. One solution is to apply stemming to the potential compound parts, but again, stemming is often too aggressive, resulting in non-words. What is needed is either a lexicon containing plurals, or a lemmatizer which returns the morphological stem of a word.

## 3.3 Exploiting Decompounding during Retrieval

For retrieval purposes, each document in the collection is analyzed and if a compound is identified, all of its parts are added to the document. In some cases, compound splitting can give rather awkward results, e.g., German: *Bahnhof* (English: train station) is split into *Bahn* (rail) and *Hof* (court/yard). Whereas 'rail' is semantically related to 'train station,' this is less obvious for 'court' or 'yard.' Hence, it can happen that compound splitting adds some rather unrelated words to a document causing a slight topic drift.

The current versions of our compound splitters are not tuned for retrieval purposes; for instance, we made no attempt to avoid the addition of unrelated compound parts.

Compounds occurring in a query are analyzed in a similar way: the parts are simply added to the query. Since we also expand the documents with compound parts, there is no need for compound formation [13].

Currently, as mentioned above, only the minimal parts of a compound (i.e., the leaves in a tree, as shown in Figure 2) and the compound itself are considered. If a compound is more complex, containing more than two compound parts, intermediate compound parts could also be considered. Although less specific than the compound itself, they are more specific than the minimal compounds. In the current setting we refrained from using intermediate compound parts, because it raises the issue of ambiguity, as explained above, but we think that, given a well-performing compound splitter, it might also be worth adding intermediate compound parts to increase precision.

The approach of adding compound parts to the document itself has some side effects whose impact is not clear yet. For example, what is an appropriate matching strategy for compounds? In our implementation, compounds and their parts are treated independently of each other, i.e., the term weight (tf.idf score) is computed independently for the compound and its parts. If a query contains a compound $A$, split into $(a_1)(a_2)$, and a document also contains compound $A$, computing the similarity score considers all three matches: $A$, $a_1$, and $a_2$. This seems to be inappropriate as the compound parts are conceptually not independent of the compound itself. However, this approach rewards compound matching in contrast to simple term matching, which again seems appropriate since compounds are more specific their compound parts. This issue of compound matching and assigning weights to compounds is very similar to the problem of phrase matching and phrase weighting in English [5, 19].

Another problem with simply adding the compound parts to the document itself concerns document length. Table 3 shows the differences in average document length before and after adding compound parts to the documents in the collection that was used for the CLEF 2001 evaluation exercise.

| *document length/weight* | *Dutch* | | | *German* | | |
|---|---|---|---|---|---|---|
| | *orig.* | *+ comp. parts* | | *orig.* | *+ comp. parts* | |
| *avg. length in byte* | 1587 | 2116 | (+33.3%) | 1567 | 1420 | (+10.4%) |
| *avg. no. unique words* | 171 | 203 | (+18.7%) | 142 | 156 | (+9.9%) |
| *avg. cosine document weight* | 11.01 | 11.74 | (+6.6%) | 10.73 | 11.25 | (+4.8%) |

**Table 3.** Effect of adding compounds on document length and weight.

The three measures listed above are the most commonly used measures for pivoted document normalization, cf. [17]. Adding compound parts has a significant effect on the document length, but it is unclear to what extent this affects retrieval effectiveness.

Finally, some figures about the topics that were used in the CLEF 2001 evaluation exercise. For Dutch 50 topics were used, for German 49. The average number of (non-unique) nouns in the (combined) title and description fields was 6.62 for Dutch and 6.42 for German, with an average of 1.3 (non-unique) compounds for Dutch and 1.46 for German. The average number of (non-unique) parts added per topic by a human compound splitter was 2.66 for Dutch and 3.22 for German.

## 4 Official Runs

At CLEF 2001, the University of Amsterdam participated in the monolingual task only, covering retrieval in Dutch, German, and Italian. For each language we submitted three types of runs:

**Type M (Morphological)** The title and the description field of the topic are used to generate the retrieval query (this was a mandatory requirement to be met by at least one of the runs). Words are morphologically normalized and compounds are split (Dutch and German). Blind feedback is applied to the top 10 documents adding at most 20 terms to the original query. This includes the runs `AmsNlM`, `AmsDeM`, and `AmsItM`.

**Type Nv (Naïve)** The title and the description field of the topic are used to generate the retrieval query. Blind feedback is applied to the top 10 documents adding at most 20 terms to the original query. In contrast to runs of type M, no morphological normalization or compound splitting are applied. This includes the runs `AmsNlNv`, `AmsDeNv`, and `AmsItNv`.

**Type T (Title only)** The same retrieval and document processing techniques are used as for runs of type M, but query formulation is restricted to the title field of the topic. This includes the runs `AmsNlT`, `AmsDeT`, and `AmsItT`.

Our motivations for these runs were as follows. Type M runs were intended to be the most effective runs, using techniques which are considered to improve retrieval effectiveness, such as blind feedback. Type T runs use the same techniques, but queries are much shorter and, therefore, more closely resemble queries posed by non-experts. In contrast to type M runs, type Nv runs apply no language specific techniques such as stemming/lemmatization or compound splitting.

After our submissions had been eveluated and our scores returned, we discovered that the compound splitter malfunctioned for a large number of Dutch nouns due to a bug in the interface between the stemmer and the compound splitter. This affected our type M and type T runs for Dutch submitted to CLEF 2001. Below we report on both the official (submitted) runs and the corrected ones. To start, Figure 4 displays the interpolated precision-recall curves for the three languages, with two plots for Dutch (not corrected and corrected).

Next, considering the non-interpolated avg. precisions for type M and type Nv runs in Table 4, one can see that morphological normalization does result in significant improvements[1] in effectiveness: $\approx$ 25% for German and Italian and even $\approx$ 54% ($\approx$ 69%) for Dutch (Dutch corrected).

The improvements were consistent across all topics, as is witnessed by the histograms in Figure 5, where we have plotted the improvements in average precision of type M runs over type Nv runs for each of the individual topics.

It is not obvious why the improvement for Dutch (whether corrected or not) is so much bigger than for the other two languages. One reason could be that our precision

[1] Note that significant improvement here refers to the definition in [18], where changes of more than 5% are considered significant.

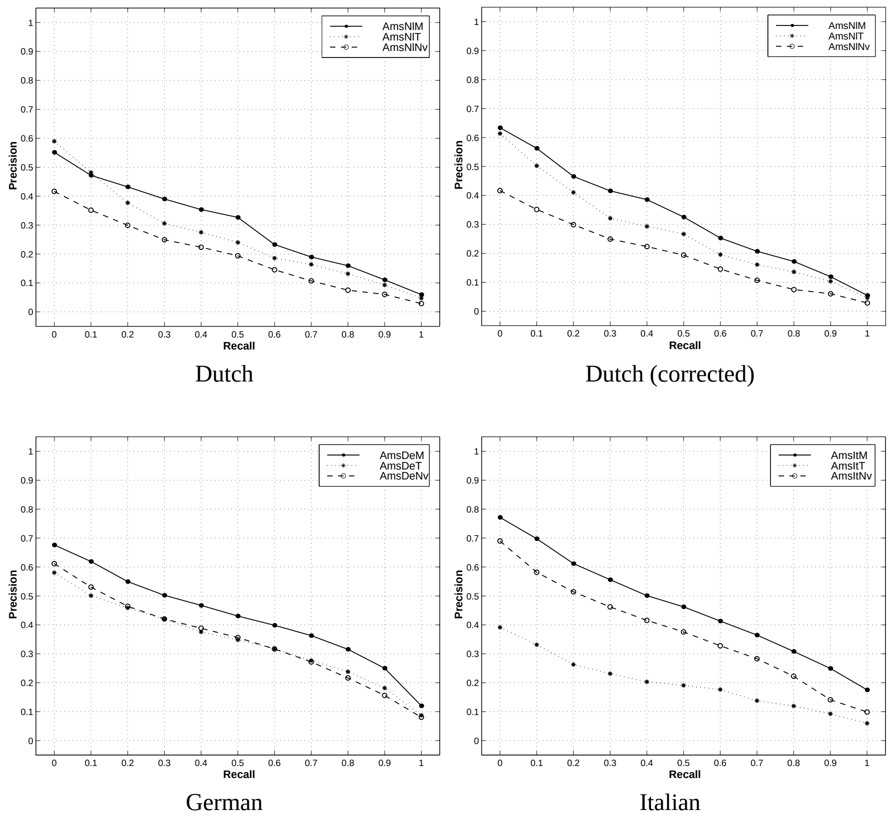

**Fig. 4.** 11pt interpolated avg. precision for all submitted runs.

scores for Dutch are, in general, considerably lower than the precision scores for German and Italian. Our results seem to suggest that the improvements brought about by compound splitting (plus stemming) are independent from the underlying retrieval engine. Observe that there are some topics for which the type M run performs worse than the type Nv run. A closer inspection of the topics showed that this was due to topic drift resulting from blind feedback.

| | *Dutch* | *Dutch (corrected)* | *German* | *Italian* |
|---|---|---|---|---|
| *Naïve (Nv)* | 0.1833 | 0.1833 | 0.3342 | 0.3580 |
| *+ Morph. Anal. (M)* | 0.2833 | 0.3114 | 0.4172 | 0.4485 |
| | +54.6% | +69.9% | +24.8% | +25.3% |

**Table 4.** Non-interpolated avg. precisions of type M runs vs. type Nv runs.

| | *Dutch* | *Dutch (corrected)* | *German* | *Italian* |
|---|---|---|---|---|
| *Morph. Analysis (M)* | 0.2833 | 0.3114 | 0.4172 | 0.4485 |
| *Title only (T)* | 0.2418 | 0.2582 | 0.3342 | 0.1895 |
| | -14.6% | -17.1% | -19.9% | -57.7% |

**Table 5.** Non-interpolated avg. precisions of TD-queries vs. T-queries.

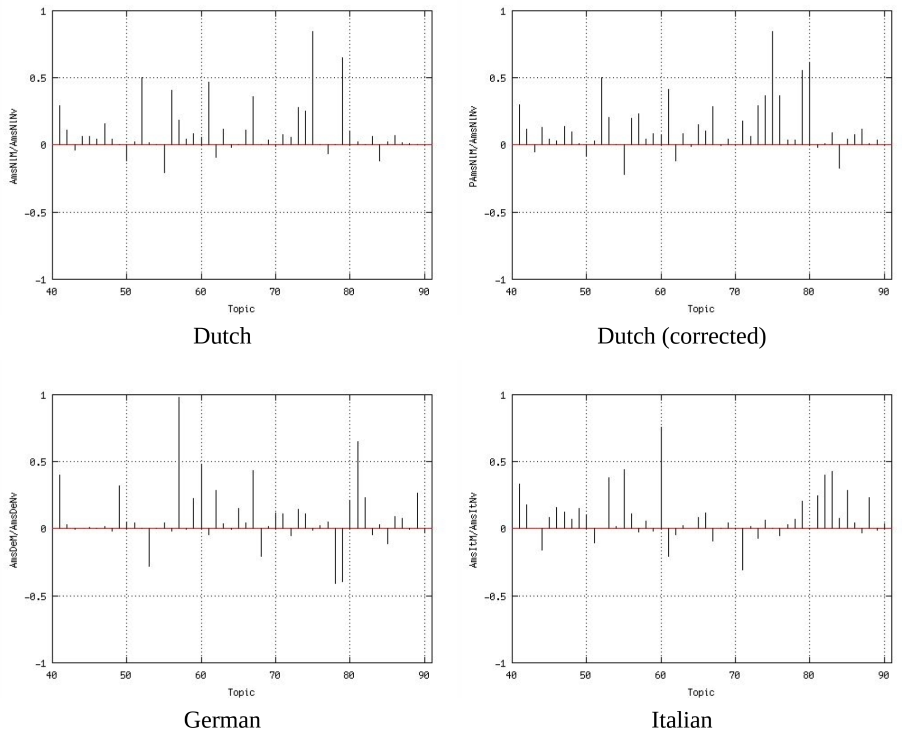

**Fig. 5.** Average precision (individual queries) of type M vs. type Nv runs, Topics 41–90.

Another interesting question is to compare queries that were formulated by using the title and the description field of the topic to queries that were formulated by using the title of the topic only. Queries based on title information only are much shorter and more closely resemble queries a non-expert would formulate. Table 5 shows that for Dutch and German the decrease in effectiveness is certainly significant but not too dramatic. On the other hand, for Italian, using the title field only has a drastic impact on effectiveness, decreasing it by $\approx 57\%$. What causes this dramatic decrease, particularly in comparison to Dutch and German, is not obvious at this stage.

Finally, Table 6 shows the average precisions at a set of fixed ranks. This is again interesting from a regular user's point of view, who will hardly ever consider more than the top 20 documents returned by a retrieval system.

## 5 Post-Submission Experiments

Following the release of the results of our submitted runs and of the evaluation scripts, we conducted a number of post-submission experiments. These were aimed at exploring the interaction between compound splitting and the various other techniques that we used to enhance retrieval effectiveness. In particular, we wanted to understand the

|  | *Dutch (corrected)* |  |  | *German* |  |  | *Italian* |  |  |
|---|---|---|---|---|---|---|---|---|---|
|  | `PAmsNlM` | `AmsNlNv` | `PAmsNlT` | `AmsDeM` | `AmsDeNv` | `AmsDeT` | `AmsItM` | `AmsItNv` | `AmsItT` |
| p@5 | 0.4160 | 0.2760 | 0.3510 | 0.5102 | 0.4490 | 0.4286 | 0.5660 | 0.4255 | 0.2426 |
| p@10 | 0.3480 | 0.2280 | 0.2714 | 0.5102 | 0.4163 | 0.4082 | 0.5170 | 0.4000 | 0.2106 |
| p@15 | 0.3067 | 0.2027 | 0.2408 | 0.4721 | 0.4122 | 0.3878 | 0.4638 | 0.3645 | 0.1957 |
| p@20 | 0.2840 | 0.1840 | 0.2306 | 0.4582 | 0.3878 | 0.3582 | 0.4330 | 0.3340 | 0.1766 |
| p@30 | 0.2520 | 0.1667 | 0.2027 | 0.4102 | 0.3503 | 0.3218 | 0.3695 | 0.3007 | 0.1539 |
| p@100 | 0.1412 | 0.0890 | 0.1231 | 0.2504 | 0.2143 | 0.1980 | 0.1970 | 0.1572 | 0.0898 |
| p@200 | 0.0884 | 0.0552 | 0.0815 | 0.1669 | 0.1441 | 0.1288 | 0.1147 | 0.0951 | 0.0618 |
| p@500 | 0.0402 | 0.0264 | 0.0393 | 0.0793 | 0.0715 | 0.0669 | 0.0502 | 0.0457 | 0.0338 |
| p@1000 | 0.0211 | 0.0140 | 0.0209 | 0.0410 | 0.0380 | 0.0381 | 0.0255 | 0.0240 | 0.0202 |

**Table 6.** Avg. precision at rank *n*.

interaction between blind feedback and compound splitting. As compound splitting is not an issue for Italian, we restricted ourselves to Dutch and German.

For both Dutch and German, we considered four types of runs: with or without blind feedback (F/NoF) and with or without compound splitting (C/NoC); all runs used stemming. Observe that runs of type FC coincide with the earlier type M runs; the difference between type Nv runs and type FNoC runs is that the latter use stemming. We use the suffix ‘Nl’ to indicate Dutch runs, and ‘D’ to indicate German runs.

## 5.1 Dutch

For Dutch, adding compound splitting in the absence of feeedback leads to an improvement in average precision of slightly more than 6%; in the presence of feedback it gives a slightly bigger improvement of 8.5%. As to adding relevance feedback in the presence or absence of decompounding, we couldn’t observe any significant changes: with compound splitting switched off, feedback gave a 1.5% improvement, and otherwise it gave a 3.8% improvement. The combination of feedback and compound splitting resulted in an improvement of 10.1% in average precision over no feedback and no compound splitting; see Table 7 for a summary.

| *Dutch* | *No feedback (NoF)* | *+ Feedback (F)* |  |
|---|---|---|---|
| *No compound splitting (NoC)* | 0.2828 | 0.2871 | +1.5% |
| *+ Compound splitting (C)* | 0.3001 | 0.3114 | +3.8% |
|  | +6.1% | +8.5% | +10.1% |

**Table 7.** Non-interpolated avg. precision for Dutch without/with relevance feedback and without/with compound splitting.

It’s instructive to look at the histograms comparing the various runs on a topic-by-topic basis; see Figure 6. The top row illustrates the impact of adding blind feedback. As was to be expected (based on reports in the literature), in some cases the addition of feedback hurts precision; this is largely independent of compound splitting being switched on, see e.g., Topics 84 and 85.

As to the addition of compound splitting, this leads to some decrease in precision for some topics (e.g., Topic 56), while it leads to more substantial improvements in

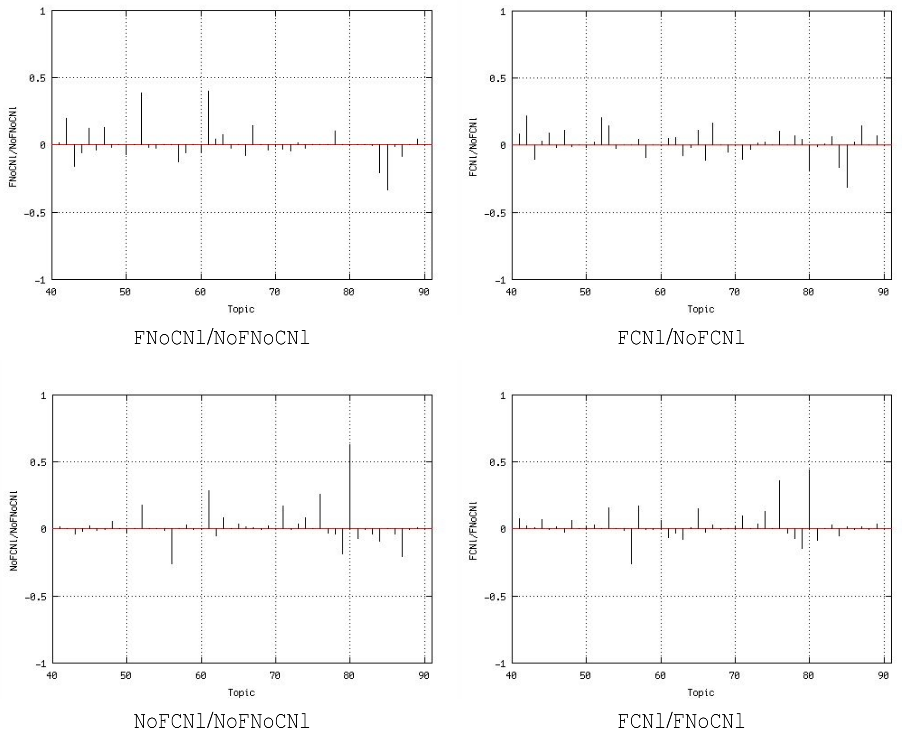

**Fig. 6.** Comparisons of avg. precision (individual queries) for four Dutch runs. (Top): Adding relevance feedback. (Bottom): Adding compound splitting.

others (e.g., Topics 76 and 80), even on top of blind feedback (see `FCNl/FNoCNl`). Note that there is no topic-by-topic correlation between the effects of compound splitting and the effects of blind feedback, but one can observe similar *kinds* of improvements (degredations) across all topics.

## 5.2 German

The phenomena that we observed for German were similar to those observed for Dutch, with the main differences being that the overall improvements — whether caused by blind feedback or by compound splitting — tend to be more significant (as witnessed by Table 8), while some local changes were more dramatic than for Dutch (as witnessed by Figure 7).

To make matters more concrete, let's take a closer look at one of the topics. In particular, feedback caused a serious drop in precision for topic 64:

```
<DE-title>Computermäuse und RSI </DE-title>
<DE-desc>Suche Dokumente, die über Erkrankungen an RSI berichten.
</DE-desc>
```

One of the examples in which splitting worked particularly well on top of blind feedback was topic 57:

| *German* | *No feedback (NoF)* | *+ Feedback (F)* | |
|---|---|---|---|
| *No compound splitting (NoC)* | 0.3551 | 0.3961 | +11.5% |
| *+ Compound splitting (C)* | 0.3892 | 0.4172 | +7.2% |
| | +9.6% | +5.3% | +17.5% |

**Table 8.** Non-interpolated avg. precision without/with relevance feedback and without/with compound splitting.

```
<DE-title>Strafprozess über verseuchte Blutkonserven</DE-title>
<DE-desc>Suche alle Informationen über Gerichtsverfahren zu
verseuchten Blutkonserven in Frankreich, einschließlich der
Gerichtsurteile und der Namen der Verurteilten.</DE-desc>
```

The splitted (and stopped and lemmatized) reformulation of this topic was

```
strafprozess verseucht blutkonserve alle en gerichtsverfahren
gericht verfahren verseucht blutkonserve frankreich
einschliesslich gerichtsurteil gericht urteil name verurteilte
```

Note, by the way, that *Blutkonserven* was not recognized as a compound.

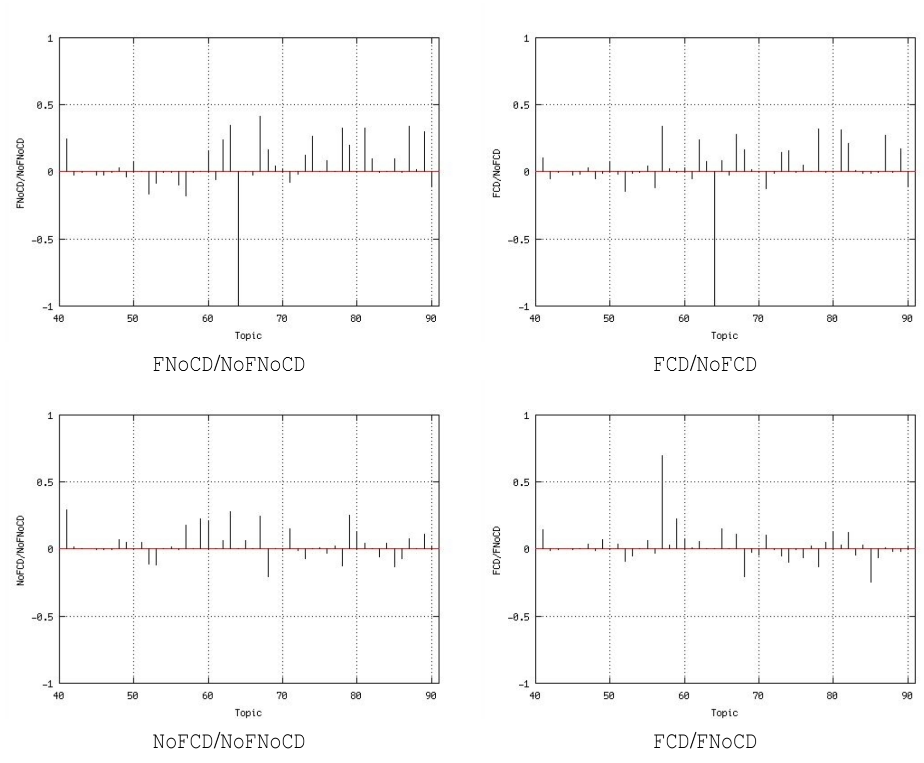

**Fig. 7.** Comparisons of avg. precision (individual queries) for four German runs. (Top): Adding relevance feedback. (Bottom): Adding compound splitting.

## 6 Conclusions

The experiments carried out here strongly confirm the believe that morphological normalization does indeed improve retrieval effectiveness for languages such as Dutch, German and Italian that are morphologically richer than English. In particular, compound splitting was shown to have a positive impact over and above blind feedback for compounding languages such as Dutch and German.

Since the morphological analyses carried out in this paper were still rather restricted, it would be interesting to see what impact additional analyses, e.g., stripping off prefixes and recognizing nominalizations, would have. Another line of interesting questions concerns the relation between the topic drift and the addition of parts of compounds. Finally, we think that a systematic study of the use of compound splitting as mechanism for feedback enhancement (across a variety of compound forming languages) would be very interesting.

## References

1. M. Adda-Decker and G. Adda. Morphological decomposition for ASR in German. In *Workshop on Phonetics and Phonology in Automatic Speech Recognition*, 2000.
2. R. Baayen, R. Piepenbrock, and L. Gulikers. The CELEX lexical database (release 2). Distributed by the Linguistic Data Consortium, University of Pennsylvania, 1995.
3. C. Buckley, A. Singhal, and M. Mitra. New retrieval approaches using SMART: TREC 4. In D. Harman, editor, *Proceedings of the Fourth Text REtrieval Conference (TREC-4)*, pages 25–48. NIST Special Publication 500-236, 1995.
4. G. Drosdowski, editor. *Duden: Grammatik der deutschen Gegenwartssprache*. Dudenverlag, fourth edition, 1984.
5. J. Fagan. *Experiments in Automatic Phrase Indexing for Document Retrieval: A Comparison of Syntactic and Non-Syntactic Methods*. PhD thesis, Department of Computer Science, Cornell University, 1987.
6. W. Frakes. Stemming algorithms. In W. Frakes and R. Baeza-Yates, editors, *Information Retrieval: Data Structures & Algorithms*, pages 131–160. Prentice Hall, 1992.
7. J. Goodman. Parsing algorithms and metrics. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics (ACL'96)*, pages 177–183, 1996.
8. D. Harman. How effective is suffixing? *Journal of the American Society for Information Science*, 42:7–15, 1991.
9. W. Kraaij and R. Pohlmann. Viewing stemming as recall enhancement. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 40–48, 1996.
10. W. Kraaij and R. Pohlmann. Comparing the effect of syntactic vs. statistical phrase index strategies for Dutch. In *Proceedings ECDL'98*, pages 605–617, 1998.
11. M. Mitra, A. Singhal, and C. Buckley. Improving automatic query expansion. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 206–214, 1998.
12. I. Moulinier, J. McCulloh, and E. Lund. West Group at 2001: Non-English monolingual retrieval. In *Proceedings CLEF-2000*, 2000.
13. R. Pohlmann and W. Kraaij. Improving the precision of a text retrieval system with compound analysis. In J. Landsbergen, J. Odijk, K. van Deemter, and G. Veldhuijzen van Zanten, editors, *Proceedings of the 7th Computational Linguistics in the Netherlands Meeting (CLIN 1996)*, pages 115–129, 1996.

14. M. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.
15. J. Rocchio. Relevance feedback in information retrieval. In G. Salton, editor, *The SMART Retrieval System — Experiments in Automatic Document Processing*. Prentice Hall, 1971.
16. H. Schmid. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of International Conference on New Methods in Language Processing*, 1994.
17. A. Singhal, G. Salton, M. Mitra, and C. Buckley. Document length normalization. *Information Processing & Management*, 32(5):619–633, 1996.
18. K. Sparck Jones. Automatic indexing. *Journal of Documentation*, 30(4):393–432, 1974.
19. T. Strzalkowski. Natural language information retrieval. *Information Processing & Management*, 31(3):397–417, 1995.
20. UPLIFT: Utrecht project: Linguistic information for free text retrieval. `http://www-uilots.let.uu.nl/~uplift/`.