Wordsyoudontknow: Evaluation of lexicon-based decompounding with unknown handling

Karolina Owczarzak **Ferdinand de Haan** **George Krupka** **Don Hindle**

Oracle Language Technology
1111 19th Street NW #600, Washington, DC 20036, USA
{karolina.owczarzak,ferdinand.de.haan,george.krupka,
don.hindle}@oracle.com

Abstract

In this paper we present a cross-linguistic evaluation of a lexicon-based decomposition method for decompounding, augmented with a "guesser" for unknown components. Using a gold standard test set, for which the correct decompositions are known, we optimize the method's parameters and show correlations between each parameter and the resulting scores. The results show that even with optimal parameter settings, the performance on compounds with unknown elements is low in terms of matching the expected lemma components, but much higher in terms of correct string segmentation.

1 Introduction

Compounding is a productive process that creates new words by combining existing words together in a single string. It is predominant in Germanic and Scandinavian languages, but is also present in other languages, e.g. Finnish, Korean, or Farsi. Many languages that are not usually thought of as "compounding" nevertheless display marginal presence of compounds, restricted, for instance, to numerical expressions (e.g. Polish *czterogodzinny* 'four-hour'). Depending on a language, compounding can be a very frequent and productive process, in effect making it impossible to list all the compound words in the dictionary. This creates serious challenges for Natural Language Processing in many areas, including search, Machine Translation, information retrieval and related disciplines that rely on matching multiple occurrences of words to the same underlying representation.

In this paper, we present a cross-linguistic evaluation of a lexicon-based decomposition method augmented with a "guesser" for handling unknown components. We use existing lexicons developed at Oracle Language Technology in combination with a string scanner parametrized with language-specific input/output settings. Our focus is on the evaluation that tries to tease apart string segmentation (i.e. finding boundaries between components) and morphological analysis (i.e. matching component parts to known lemmas).

The paper is organized as follows: Section 2 gives an overview of related research; Section 3 describes the compound analyzer used in our experiments; Section 4 presents experimental results; Section 5 contains error analysis and discussion. Section 6 concludes and suggests future research.

2 Related research

Current research on compound splitting is predominantly lexicon-based, with a range of selection methods to choose the most likely decomposition. The lexicons used to identify components are usually collected from large monolingual corpora (Larson et al., 2000; Monz and de Rijke, 2001; Alfonseca et al, 2008; Holz and Biemann, 2008; von Huyssteen and von Zaanen, 2004).

The problem with pure lexicon-based approach without any constraints is that it will produce many spurious decompositions, matching small substrings that happen to be legitimate words in the language. Therefore, some approaches introduce maximizing component length (or, conversely, minimizing the number of components) as one of the selection factors (von Huyssteen and von Zaanen, 2004; Holz and Biemann, 2008; Macherey et al., 2011; Larson et al., 2000); others use part of speech to eliminate short components which tend to be function words (Koehn and Knight, 2003; Monz and de Rijke, 2001). In other cases, Named Entity Recognition is used to filter out proper names that should not be decomposed but that can contain frequent short components like "-berg" or "-dorf" (Alfonseca et al., 2008).

Even after removing unlikely small component candidates, there is enough ambiguity in decomposition to warrant further filtering methods. And so, approaches related to Machine Translation use bilingual parallel corpora to find the most likely components by checking whether their translations match elements of the whole compound translation (Koehn and Knight, 2003; Macherey et al., 2011). Other filtering methods are based on combined frequency of the components (Koehn and Knight, 2003; Holz and Biemann, 2008), point-wise mutual information of components, or occurrence of components in related locations, such as anchor text (Alfonseca et al., 2008). A very interesting lexicon-free approach is presented in Aussems et al. (2013), which uses point-wise mutual information to detect likely boundaries between characters that would identify a compound.

A major issue with the current research is the absence of common training and testing data, particularly across multiple languages, which then translates into limited evaluations of presented methods. Using pre-annotated frequency lists, we create gold standard test sets for 10 languages: Norwegian, Danish, Dutch, Estonian, Finnish, German, Hungarian, Korean, Farsi, Swedish, which range from around 600 to 15,000 compounds. This allows a more thorough comparison of the analyser performance across different languages.

3 Lexicon-based analyzer

Our approach follows the main line of research in that it uses lexicons to identify potential components in a compound; however, our lexicons contain lemmas rather than word forms, in contrast to lexicons harvested from monolingual corpora. However, the lexicons we use contain as well as partial lemmas whose occurrences are restricted to compounds (e.g. German verb forms without the final *–en*; for example *schließ-*). In addition, we use morphological rules to map recognized inflected forms to base (lexicon) lemmas. Both the lexicons and the morphological rules have been previously created by computational linguists and native speakers for use in a variety of NLP applications at Oracle Language Technology.

On the most basic level, a compound can be explicitly added to the lexicon, with a specific decomposition and appropriate part of speech and grammatical features; this option is used when the decomposition is irregular or non-obvious, for instance when the component appears in a form that is not directly analyzable to its lemma, as in the example below, which shows irregular plurals and deletion of consonant:

(1) a. Danish: *barn* 'child' plural: *børn*

børnebog	barn+e+bog	[child-*connector*-book]	'children's book'

b. Norwegian Bokmål: deletion of repeated consonant

musikkorps	musikk+korps	[music-band]	'music band'

Lexicalized compounds are treated like any other words, and their inflected forms will be recognized. Explicitly adding the compound to the lexicon is also useful when the compound can have multiple decompositions, and we want to restrict the output only to the semantically correct analysis. In Dutch, for instance, the compound part *stem* can refer to the noun *stem* 'voice' or to the root of the verb *stemmen* 'vote'. These readings are distinguished in the lexicon by listing explicit decompositions for compounds that contain the part:

(2) Dutch *stem* N vs. V

a.	stemband	stem#band	[voice-cord]	'vocal cord' (N-N)
b.	stembureau	stemmen#bureau	[vote-station]	'polling station (V-N)

However, adding all compounds to the lexicon is simply unfeasible for many languages where the compounding process is highly productive. For this reason, we also use a compound analyser to identify components in a dynamic manner, based on available component lemmas in the lexicon. Components are found by removing any recognizable inflections from the candidate string, scanning it left-to-right, and looking for all matching lemmas, subject to constraints based on part of speech, length, number of components, and available features. For speed reasons, we apply greedy matching, and prefer decompositions with the longest prefix and the smallest number of components.

Since our goal is developing language processing systems that are as universal as possible, leaving context-dependent decisions to higher-level applications, we are not particularly concerned with always selecting the single best decomposition for a compound, since in many cases is will be dependent on the domain and application. However, it is useful to filter out decompositions that would be highly unlikely in any context, for instance those containing small function words mentioned in previous section. For this purpose, we apply constraints described below.

3.1 Rules for compound sequences

For each language, we list the possible part of speech sequences that can appear in compounds. These rules serve not only to prevent the decompositions that would not appear in the language (for instance, *noun-verb-particle*), but also to restrict sequences that are fairly infrequent, but that would lead to considerable over-generation if they were added. For example, in German, there are relatively few compounds that end with a verb, unless it is a combination of movable prefix particle (*aus*, *an*, *ab*, *ein*, etc.) and the verb (*aus+gehen*, *auf+stehen*, *um+steigen*, etc.). These verbs are functionally analyzed as compounds, i.e. a concatenation of two lemmas. However, since sequences noun/adjective/verb + verb are much less productive (*spazieren+gehen*, *auto+fahren*), it is more efficient to restrict the verb-final compounds to *particle-verb* only, and add the exceptions to the lexicon. A few examples of compound part of speech sequences for different languages are shown in (3).

(3)	a.	Dutch:	
		cardinal_number + verb	e.g., *vier+en+delen* ‘quarter’
	b.	Estonian:	
		noun+adjective	e.g. *silmi+pimestav* ‘eye-dazzling’
	c.	German:	
		ordinal_number + **adjective**	e.g. *zweit+größt* ‘second largest’
	d.	Swedish:	
		noun + noun	e.g *citron+saft* ‘lemon juice’

Another issue is compounds of cardinal or ordinal numbers, which can also occur in some languages like Italian (*cinquecento+sessanta+nove* ‘five hundred sixty nine’) or Greek (*οκτακόσιοι*, *οκτώ + ακόσιοι* ‘eight hundred’). These number compounds can be very productive and are also included in the lists of allowed compound sequences.

3.2 Connectors

In many compounding languages, the subparts of a compound can be connected with extra material, a connector (or linking element). These are semantically empty elements that have a mainly phonological role in connecting the compound parts (Bauer, 2009). In many Germanic languages connectors are derived from plural or genitive morphemes (such as *–er* or *–s* in German), but do not have this role any more, as evidenced, among others, by the fact that in certain cases the connector is optional and compounds with and without a connector co-exist (4a) or by the fact that there are cases where two different connectors co-occur (4b) (Krott et al., 2007):

(4) a. Norwegian Bokmål:			
	rettssak	rett + s + sak	‘court case’
	rettsak	rett + Ø + sak	
b. Dutch:			
	paddestoel	pad + e + stoel	‘toadstool’
	paddenstoel	pad + en + stoel	

For each language, we create a set of allowed connectors, a few examples of which can be seen in (5).[1] Note that it might be useful to restrict certain connectors to appear only in certain sequences (e.g. between noun and noun, but not adjective and verb); we plan to implement this restriction in future work.

(5) Connector examples

a.	Dutch	**s**	e.g. *water+s+nood* 'flood'
b.	German	**zu**	e.g. to match *auf+stehen* and *auf+zu+stehen* 'stand up'
c.	Swedish	**o**	e.g. *veck+o+slut* 'weekend'

3.3 Decompounding settings

Another factor in successful dynamic decompounding is restrictions on possible number of components, and on length of candidate strings and candidate components. Choosing to allow fewer components of longer length helps to prevent spurious over-analysis, where several short words can accidentally match the string which is being analyzed. However, setting the limits too high might also prevent legitimate decomposition, so this trade-off needs to be carefully balanced. There are four basic length settings, as shown in Table 1 below; the values are dependent on language.

Maximum number of elements: Limits the number of components in a compound. Low values help prevent spurious decompositions into many small elements.

Minimum length of compound: The minimum length of string that should be subject to decompounding; short strings are unlikely to be compounds, so for efficiency reasons, they are not decompounded.

Minimum length of component: Specifies the minimum length of potential compound elements; shorter substrings are excluded to avoid accidental matching of very short words.

Minimum length of component with connector: A version of the above setting, it specifies the minimum length of potential element when this element is next to a connector; to avoid spurious matches of the short word + connector combination (e.g. Dutch *paspoort* should be decomposed as *pas+poort*, not *pa+s+poort*).

setting	**value**
maximum number of elements	2-4
minimum length of compound	4-11
minimum length of component	2-4
minimum length of component with connector	2-4

Table 1. Length settings for dynamic decompounding.

The values for these settings are established manually and separately for each language, based on review of top N most frequent compounds in the lexicon and the general knowledge of that language's grammar and patterns.

4 Experimental results

Despite all the constraints and settings described above, decompounding is still an imperfect process: there can be multiple competing (i.e. overlapping) decompositions, and many decompositions that are technically possible are incorrect due to semantic reasons. This problem becomes even more challenging when some of the components are not present in the lexicon. Since lexicons are limited, and real world text can contain misspellings, proper names, or obscure words, we need to address the issue of decompounding with unknown elements. Therefore, we set out to evaluate the performance of our lexicon-based method on a gold standard set of known compounds, and compare it to an augmented version that also tries to construct potential components from unknown substrings.

[1] Note that for our purposes, particle *zu* in German is also treated as a connector, to match the movable particle verbs that can appear with and without *zu*: *auf* + *zu* + *stehen* and *auf* + *stehen* 'get up'.

4.1 Test set

For our experiments, we collected compounds from the top 90% frequency lists based on large news and Wikipedia corpora. Each compound was annotated with the correct decomposition(s) by a linguist who was also a native speaker of the target language according to simple instructions: if the meaning of the word is compositional (i.e. can be fully described by the component elements), treat it as a compound and provide component lemmas.

Approximate sizes of source corpora per language are given in Table 2; column "compounds" shows the count of compounds; column "lexical" shows how many of these are lexicalized compounds (i.e. compounds that have been added to the lexicon for reasons of irregularity). While two-part compounds are by far the most frequent in all the languages we examined, there is also some percentage of compounds with more than two parts; the distribution is shown in the last four columns.

language	news corpus MB	wiki corpus MB	compounds	lexical	2-part	3-part	4-part	5-part
Danish	335	154	1,982	1,326	1,856	122	4	0
Dutch	512	103	3,439	1,909	3,186	245	8	0
Estonian	204	41	2,343	562	2,166	169	8	0
Farsi	512	244	648	340	635	13	0	0
Finnish	512	78	1,868	1,665	1,703	154	11	0
German	520	227	15,490	5,087	14,544	915	31	0
Hungarian	512	257	1,841	1,537	1,794	45	2	0
Korean	826	190	11,398	4,774	10,919	425	39	5
Norwegian	512	88	3,582	1,106	3,405	175	2	0
Swedish	512	204	9,677	5,608	8,901	744	31	5

Table 2. Size of corpora per language, count of compounds, distribution of parts.

4.2 Dynamic decompounding with available lemmas

As mentioned before, it is not feasible to add all (or even the majority) of possible compounds, so we need to examine our performance using only dynamic decompounding. For this purpose, we removed all lexicalized compounds from the lexicon, and then ran the analyzer on the compound test set described above. This means that all the compound analysis was done dynamically, using only the available simple lemmas and compound rules and length restrictions. Table 3 shows the results. The scores for lexicalized + dynamic decompounding are given only for reference; they are high but less interesting, since they reflect the fact that the lexicalized compounds were largely collected from the same corpora (among other sources). Our focus is on the dynamic scores, which show performance on unknown compounds assuming a nearly "perfect" lexicon that contains almost all the component lemmas. As such, these scores will serve as the upper bound for our next experiment, in which we remove at least one of the component lemmas from the lexicon and test the resulting performance.

As can be seen in Table 3, for most languages recall decreases considerably – this suggests that lexicalized compounds are of the kind that are not covered by the decompounding rules or whose correct analysis is blocked by another decomposition.

4.3 Dynamic decompounding with missing lemmas

While dynamic decompounding can handle the productive nature of compounds, it is still limited to finding components that are already present in the lexicon. However, in the real world compounds will contain elements unknown to a lexicon-based analyzer, whether it is because they are domain-specific vocabulary, proper names, foreign borrowings, or misspellings. In those cases, it is still useful to attempt analysis and return the known parts, with the option of returning the unknown substring as the missing lemma.

	lexicalized + dynamic			dynamic only		
	prec	rec	f-score	prec	rec	f-score
Danish	98.18	99.6	98.88	87.99	66.9	76.01
Dutch	98.84	100	99.42	84.46	80.49	82.43
Estonian	98.25	99.83	99.03	95.69	90.27	92.9
Farsi	92.9	100	96.32	65.75	72.84	69.11
Finnish	98.74	100	99.37	84.55	68.63	75.76
German	96.11	99.98	98.01	88.01	89.03	88.52
Hungarian	90.44	99.84	94.91	77.42	72.19	74.71
Korean	99.72	100	99.86	95.23	59.49	73.23
Norwegian	99.6	100	99.8	93.25	86.32	89.65
Swedish	96.35	99.88	98.08	86.67	75.75	80.84

Table 3. Precision, recall, and f-measure for dynamic decompounding.

To evaluate the performance of our analyzer in case where some component lemmas are unknown, we applied a “compound guesser” function that tries to find known elements of unknown compounds, even if a complete decomposition to only known elements is impossible. The guesser has its own constraints, independent of the main compound analyzer, which are shown in Table 4.

setting	value
maximum number of elements	2-20
minimum length of compound	3-20
minimum length of component	2-5
minimum length of unknown element	1-5
minimum percent of string covered	0-100%

Table 4. Settings for dynamic decompounding with unknown elements.

The first three settings are parallel to the settings for regular dynamic decompounding; however, we also add restrictions on length for unknown elements (*minimum length of unknown element*) and total string coverage (*minimum percent of string covered*). Restriction on length of unknown element mean that any unknown string shorter than the minimum length will be treated as a potential connector/suffix/prefix and will not be returned as a lemma:

(6) German: assuming *freundlicher* ‘friendlier’ is unknown:
umweltfreundlicher -> umwelt + freundlich (! + er) [environment + friendly]

The last setting allows a more fine-grained control over the proportion of known to unknown parts; however, since any value less than 100% will restrict the number of produced candidate decompositions, resulting in no output if the unknown substring is too long, we do not test the impact of this setting.

For this experiment, we collected all component lemmas from the test compounds, and removed from lexicon at least one component lemma per compound. This renders the whole string unanalyzable by regular means. Then we ran the compound guesser with each combination of settings from Table 4, to find the optimal set of values.

Table 5 shows results obtained with the optimal guesser settings per language, compared to scores from Table 3: a fully functional decomposition that has access to both dynamic decomposition and lexicalized compounds, and dynamic decomposition with near-perfect component lexicon. It is clear that even with optimal settings, the guesser performance falls well below the level of full functionality, even when we compare to a system that has no access to lexicalized compounds. The highest score achieved by the guesser is 34 for the Hungarian test set, which includes mostly simple two-part compounds, and where the lexicon does not provide too many spurious sub-matches.

language	lexical + dynamic	dynamic only	dynamic guesser	dynamic guesser - string segmentation
Danish	98.88	76.01	25.93	**51.25**
Dutch	99.42	82.43	27.13	**64.01**
Estonian	99.03	92.9	9.56	**53.89**
Farsi	96.32	69.11	27.16	**78.68**
Finnish	99.37	75.76	19.49	**51.6**
German	98.01	88.52	25.1	**52.29**
Hungarian	94.91	74.71	34	**53.5**
Korean	99.86	73.23	16.81	**76.54**
Norwegian	99.8	89.65	22.56	**49.74**
Swedish	98.08	80.84	25.56	**54.18**

Table 5. Dynamic decomposition with missing lemmas, optimal settings; string segmentation shows accuracy score; remaining values are harmonic f-score of precision and recall.

However, a major problem with this evaluation is that output of the regular decompounding process produces lemmas in their dictionary form, without inflection, whereas the guesser can only return surface strings for the unknown elements which might carry grammatical inflection or stem alternations. Therefore, it would be more fair to compare the guesser to dynamic decompounding in terms of pure string segmentation – whether it finds the same boundaries between components, without concern for the form of the returned component. This lets us tease apart the impact of finding component elements from the impact of morphology. The last column in Table 5 shows accuracy of guesser string segmentation as compared to string segmentation performed by regular dynamic decompounding; in this respect the guesser's performance is indeed much better. These results are encouraging, showing that we can recover correct components in up to 79% of cases, which is a very useful improvement for the purposes of information retrieval and search. While some recall is lost by returning strings instead of lemmas, we are planning to add a second step that would employ a lemma "guesser", in order to produce the most likely dictionary form from the recovered unknown string.

language	max elements	corr. with score	min length of compound	corr. with score	min length of element	corr. with score	min length of unknown element	corr. with score
Danish	2	-0.19	3-7	-0.46	4	0.43	3	-0.09
Dutch	2	-0.21	8	-0.47	5	0.51	3	-0.06
Estonian	2	-0.15	3-7	-0.57	4	0.33	3	-0.08
Farsi	2	-0.17	3-5	-0.51	3	0.11	2	-0.24
Finnish	2-10	-0.07	3-8	-0.49	5	0.41	3	0
German	2	-0.26	8	-0.43	5	0.5	3	-0.06
Hungarian	2-16	0	1-6	-0.58	4	0.39	2	-0.33
Korean	2-10	-0.07	3	-0.14	2	-0.07	1	-0.11
Norwegian	2-10	-0.18	3-8	-0.61	5	0.56	3	0.01
Swedish	2	-0.23	7	-0.45	4	0.49	3	-0.04
Average		**-0.15**		**-0.47**		**0.37**		**-0.1**

Table 6. Optimal guesser settings and their correlations of settings with the guesser score.

Finally, Table 6 shows the correlation (Pearson's *r*) of guesser settings (or their ranges) and the resulting scores. As can be seen, the strongest correlation holds for the minimum length of compound (average -0.47) and minimum length of element (0.37). In the former case, the correlation is inverse, which means the higher the value, the lower the final score; this is caused by the fact that our test set contains only compounds, so returning the whole unsplit string will never be the right result. The second correlation reflects the fact that it is safer to exclude very short elements from appearing as components, a finding that confirms earlier research.

5 Error analysis

A considerable percentage of mismatch errors when guessing the unknown components of compounds is caused by the connectors. Our current guesser settings return the whole unknown string, without attempting to identify any potential connectors on its edges. This seems like an obvious area for improvement, as it would let us return more correct decompositions for cases shown in Table 7 (unknown strings are enclosed in square brackets and are currently returned whole).

language	**token**	**dynamic**	**guesser**	**translation**
Norwegian	kjærlighetsbrev	kjærlighet#brev	kjærlighet#[**s** + brev]	love letter
Danish	ungdomshus	ungdom#hus	ungdom#[**s** + hus]	youth
German	sklavenmoral	sklave#moral	sklave#[**n** + moral]	slave morality
Swedish	kvinnoförbund	kvinna#förbund	kvinn#[**o** + förbund]	women's alliance

Table 7. Examples of connector mismatches between dynamic decompounding and the guesser.

As could be expected, most errors are nevertheless caused by the guesser splitting unknown strings into smaller known chunks; several typical examples are shown in Table 8.

language	**token**	**dynamic**	**guesser**	**translation**
Danish	populærkulturen	populær#kultur	populær#kult#uren	popular culture
Dutch	kunstschilders	kunst#schilder	kunst#schil#ders	painters
Finnish	rockmuusikot	rock#muusiko	rock#muusi#kot	rock music
Swedish	radioversion	radio#version	radio#vers#ion	radio version

Table 8. Examples of incorrect splitting of unknown strings.

6 Conclusion and future work

In this paper, we have shown a dictionary-based compound analyzer, augmented with the function to handle unknown substrings. A cross-linguistic evaluation against the gold standard containing component lemmas shows that the correct handling of unknown compound elements is a difficult issue especially if we try to match dictionary lemmas; however, a more detailed evaluation of the string segmentation and boundary detection shows fairly good results. Being able to decompose unknown compounds and match the components to known lemmas to increase recall is crucial to many NLP applications, such as information retrieval or Machine Translation. A correct segmentation is of fundamental importance, but the question remains how we can match the unknown, possibly inflected, substring to known lemmas. In the future, we plan to address this question by (1) adding the option to separate out connectors from unknown strings, and (2) build a lemma "guesser" that would try to construct a probable dictionary representation for the unknown string, in effect building a pipeline that would more fully mirror the process of regular dynamic decompounding.

Acknowledgements
We would like to thank the rest of the Oracle Language Technology team, in particular Elena Spivak and Rattima Nitisaroj, for their help with compound examples.

References

Alfonseca, Enrique, Slaven Bilac and Stefan Pharies. 2008. German Decompounding in a Difficult Corpus. In *Computational Linguistics and Intelligent Text Processing*, A. Gelbukh (ed.). Springer Verlag, Berlin and Heidelberg, 128-139.

Aussems, Suzanne., Bas Goris., Vincent Lichtenberg, Nanne van Noord, Rick Smetser, and Menno van Zaanen. 2013. Unsupervised identification of compounds. In *Proceedings of the 22nd Belgian-Dutch conference on machine learning,* A. van den Bosch, T. Heskes, & D. van Leeuwen (Eds.), Nijmegen, 18-25.

Bauer, Laurie. 2009. Typology of Compounds. In *The Oxford Handbook of Compounding*, Rochelle Lieber and Pavol Štekauer (eds.). Oxford University Press, Oxford.343-356.

Holz, Florian and Chris Biemann. 2008. Unsupervised and Knowledge-Free Learning of Compound Splits and Periphrases. *CICLing'08 Proceedings of the 9th international conference on Computational linguistics and intelligent text processing*, A. Gelbukh (ed.). Springer Verlag, Berlin and Heidelberg, 117-127.

Koehn, Philipp and Kevin Knight. 2003. Empirical Methods for Compound Splitting. *Proceedings of the 10th conference of the European Chapter of the Association for Computational Linguistics*, Vol. 1, 187-193.

Krott, Andrea, Robert Schreuder, R. Harald Baayen and Wolfgang U. Dressler. 2007 Analogical effects on linking elements in German compounds. *Language and Cognitive Processes*, 22(1):25-57.

Larson, Martha, Daniel Willett, Joachin Köhler and Gerhard Rigoll. 2000. Compound splitting and lexical unit recombination for improved performance of a speech recognition system for German parliamentary speeches In *INTERSPEECH*, 945-948.

Macherey, Klaus, Andrew M. Dai, David Talbot, Ashok C. Popat and Franz Och. 2011. Language-independent compound splitting with Morphological Operations. *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, 1395-1404.

Monz, Christof and Maarten de Rijke. 2002. Shallow Morphological Analysis in Monolingual Information Retrieval for Dutch, German and Italian. In Evaluation *of Cross-Language Information Retrieval Systems*. Carol Peters, Martin Braschler, Julio Gonzalo and Michael Kluck (eds.). Springer Verlag, Berlin and Heidelberg, 262-277.

van Huyssteen, Gerhard and Menno van Zaanen. 2004. Learning Compound Boundaries for Afrikaans Spelling Checking. In *Pre-Proceedings of the Workshop on International Proofing Tools and Language Technologies*; Patras, Greece. 101–108.

van Zaanen, Menno, Gerhard van Huyssteen, Suzanne Aussems, Chris Emmery, and Roald Eiselen. 2014. The Development of Dutch and Afrikaans Language Resources for Compound Boundary Analysis. In *Proceeding of LREC 2014.*