Unsupervised identification of compounds

Suzanne Aussems S.H.J.A.AUSSEMS@TILBURGUNIVERSITY.EDU
Bas Goris B.C.C.GORIS@TILBURGUNIVERSITY.EDU
Vincent Lichtenberg V.J.J.LICHTENBERG@TILBURGUNIVERSITY.EDU
Nanne van Noord NANNE@TILBURGUNIVERSITY.EDU
Rick Smetsers R.H.A.M.SMETSERS@TILBURGUNIVERSITY.EDU
Menno van Zaanen MVZAANEN@TILBURGUNIVERSITY.EDU
Tilburg University, Tilburg, The Netherlands

Keywords: automatic compound recognition, point-wise mutual information, edge detection, unsupervised machine learning

Abstract

In a variety of languages, compounds (i.e., lexemes consisting of more than one stem) are written as one token. Identifying such compounds together with their compound boundaries can help improve the quality of computational linguistic tasks such as machine translation and spelling correction. In order to create annotated compound datasets, we need to be able to identify compounds in various languages. Since manual identification is very time consuming, we propose novel, language-independent approaches to the identification of compounds in sets of words. A range of methods has been explored, including unsupervised machine learning approaches. The most successful approach focuses on the identification of compound boundaries by identifying irregularities in letter combinations, exploiting point-wise mutual information values between letter n-grams. The results of applying of our methods to a collection of Dutch words show major improvements over a word-based compound identifier.

1. Introduction

In languages such as German, Russian, Finnish, Icelandic, Afrikaans and Dutch, the process of compounding is highly productive. New compounds can be created by combining two or more simplex words into a new word (van Huyssteen & van Zaanen, 2004). The meaning of the compound depends on the meaning of its parts. For instance, in Dutch, the compound *slakom* '*lettuce bowl*' consists of the simplex words *sla* '*lettuce*' and *kom* '*bowl*'. The process of compounding can be repeated, leading to compounds that have more than two simplex components, such as *slakomverkoper* '*lettuce bowl vendor*'.

Due to the productive nature of the process of compounding, fixed word lists will always be a limiting factor in describing the dictionaries of languages that allow for compounding. Automatic identification of compounds and compound boundaries can help improve upon the quality of linguistic applications such as machine translators and spelling checkers. For instance, being able to identify the parts of a compound allows for the translation of the parts (for instance, if *fietspad* '*bicycle path*' is not in the dictionary, but the compound boundary (**fiets + pad**) and the simplex words *fiets* '*bicycle*' and *pad* '*path*' are known, the word may still be translated). Similarly, spelling correctors require knowledge of the process of compounding to accept valid compounds.

In certain cases, the concatenation of simplex words into compounds requires a form of "glue". For instance, *instellingenmenu* '*setup menu*' contains the simplex words *instelling* '*setup*' and *menu* '*menu*' with the morpheme *en* serving as a glue. Such morphemes are called *linking morphemes* (Booij, 1996; Wiese, 1996).

Note that in other languages, such as English, components of compounds are written as separate tokens. The identification of such multi-word compound units

Appearing in *Proceedings of BENELEARN 2013*.

is not trivial (Mima & Ananiadou, 2000). While there are certain similarities between multi-word compound units and single token compounds, the differences are to such an extent that they require a different approach (Ryder, 1994).

Previous research concerned with splitting compounds has been conducted primarily in the context of machine translation (Koehn & Knight, 2003; Garera & Yarowsky, 2008; Macherey et al., 2011; Alfonseca et al., 2008). In previous studies, monolingual and bilingual approaches can be distinguished.

Several bilingual approaches have successfully used information from parallel corpora to identify compound boundaries. Koehn and Knight (2003) break up German compounds so that a one-to-one correspondence to English content words can be established. Part-of-speech tags and frequency information are used to align German compound parts to their English translations. Similarly, Macherey et al. (2011) apply dynamic programming to find one-to-one mappings to English translations of German and Greek compound parts. The emphasis on finding correct translations for compounds is also evident in Garera and Yarowsky (2008). While their approach does not require bilingual training text, the authors use cross language compound evidence obtained from bilingual dictionaries; an approach similar to Koehn and Knight (2003).

The purpose of these bilingual approaches is to improve the quality of machine translation systems. For their tasks, parallel corpora augmented with metadata are readily available. Monolingual methods aim to identify compound boundaries without the need of such information. Alfonseca et al. (2008) apply frequency and probability based methods, and search for compound components that occur in different anchor texts pointing to the same document. The combined features are used in a supervised machine learning classifier.

Research has been conducted on the segmentation of words in morphemes. The *Morfessor* system (Creutz & Lagus, 2005) implements an unsupervised method for producing a segmentation of morphemes. Its task is to model a language that consists of a lexicon of morphemes, by looking at high-frequency n-grams in unannotated corpus text. Segmentation is applied by matching morphemes in the constructed lexicon to words. As such, its approach is similar to our compound component approach, described in Section 3.1. Authors report precision scores as high as 63% and recall as high as 75% for English word type boundaries, and precision of 85% and recall of 53% for Finnish word type boundaries.

Our research is different from previous work in several regards. In the first place, our task is different as we aim to identify compounds and not compound boundaries. To achieve this, we investigate several unsupervised methods. In the second place, our monolingual data consists solely of individual lemmas, without additional frequency data, POS tags or other meta-information. As such, our methods can be applied to identify compounds in languages where such information is not readily available.

The task of identifying compounds in non-trivial. There are even cases in which it is unclear whether a word is a compound (without further context). For instance, the word *minister* can mean '*minister*' or '*mini star*'. In the first case, the word is a simplex word, whereas in the second situation, the word is a compound **mini + ster**. Such situations happen regularly as certain affixes can also serve as simplex words. For instance, *vergeven* '*to forgive*' is not a compound, but can be analyzed as one **ver + geven** '*far to give*'. Another such affix is *-lijk*, for instance in *mannelijk* '*manly*', but the simplex word *lijk* translates to '*corpse*'.

van Huyssteen and van Zaanen (2004) propose two compound analysis systems for compounds in Afrikaans, that recognize compound boundaries and linking morphemes if present. The first system is based on a longest string-matching algorithm that searches for known (simplex) words at the beginning and at the end of a potential compound. If the analysis of a potential compound shows that it consists of valid and shorter words from a word list, possibly glued together using valid linking morphemes, the word is considered a compound. This algorithm forms the basis of the compound component identifier as described in Section 3.1.

The second system is an unsupervised clustering approach that, based on k-means, groups words in either a compound or non-compound class. The clustering is performed based on shallow, word-driven features.

The final system is an unsupervised machine learning classification approach that decides whether there is a compound boundary or linking morpheme boundary between any of the letters in the word. Using a sliding window, all positions between letters in the word are considered. The letters left and right of the between-letter positions are used as features in the machine learning classifier. The systems described in this paper are unsupervised and as such do not require any annotated data.

2. Problem description

Ultimately, we would like to be able to develop systems that automatically identify compound and linking morpheme boundaries given a compound. However, developing and evaluating such systems requires datasets annotated with compound boundaries.

Unfortunately, only a very limited number of annotated compounds datasets exist. For instance, the CKarma dataset (CText, 2005; Pilon & Puttkammer, 2008) contains 72,849 annotated Afrikaans compounds and for Dutch, the e-Lex dataset (see Section 4.1) is available, containing 88,713 annotated compounds.

To create larger datasets of (manually) annotated compounds, or to create such datasets for languages for which no such datasets yet exist, access to a dataset containing (only) compounds is required. The research described here aims to tackle the problem of automatically identifying compounds in large plain text corpora.

Following the identification of compounds, the annotation task is to manually identify the compound and linking morpheme boundaries. Manually annotating such a list of types is an expertise, time and resource intensive task as each word has to be considered.

Systems that identify compounds in corpora should aim for the identification of as many compounds as possible (high recall), while limiting the incorporation of non-compounds as much as possible (high precision). Compounds that are not identified by the system will never be considered for manual annotation at all, but non-compounds can still be discarded during the compound boundary annotation.

3. System description

We have experimented with a variety of systems to identify compounds given a set of words. First, we describe the compound component identifier, which is a word-based approach. Second, we propose an unsupervised clustering approach, which aims to identify compounds based on a combination of shallow features. Finally, we describe approaches that aim to identify potential compound boundaries based on the pointwise mutual information values between letters, which indicate the regularity of letters occurring next to each other. Letter combinations that do not co-locate regularly are likely to be compound boundaries.

The systems described here receive only a set of types as input. No frequency information is given, because often no reliable counts can be obtained. For instance, for Afrikaans, no publicly available large scale corpus exists. In case a system requires additional information, this is indicated in the description of the system. Ideally, the developed systems are completely unsupervised and language independent.

3.1. Compound component identifier

Based on the idea that compounds consist of two or more meaningful lexical stems, van Huyssteen and van Zaanen (2004) developed a system, called *longest string-matching*, that identifies compound boundaries in compounds. The algorithm takes a compound as input and recursively searches a dictionary (containing simplex words) for words that can be concatenated to form a compound. The system is also able to deal with linking morphemes, which are provided beforehand.

The compound component (CC) identifier follows the idea of the longest string-matching approach closely. However, in contrast to the longest string-matching algorithm, the CC identifier does not have access to a list of simplex words. The CC identifier solves this by making use of all the words in the input dataset. To be more precise, the CC identifier receives a comprehensive set of input data containing both simplex and compound types. The CC identifier then searches for types that can be constructed by concatenating two or more other types from the set. If the CC identifies such a construction, the system classifies it as a compound.

The CC identifier is provided with a list of valid linking morphemes beforehand. This information is language dependent, so the system requires minor supervision by an expert. We have taken into account that very short simplex words lead to over-generation of the system (identifying too many words as compounds). Specifically two letter words, which are typically function words, such as determiners, prepositions or conjunctions, (for instance *in* '*in*', *op* '*on*', *na* '*after*', or *en* '*and*') have a large negative impact. Despite being valid simplex words, they can be used incorrectly as a stem in a compound. For instance, the non-compound *inbedden* '*to embed*' can be split into **in + bedden** '*in + beds*'. To solve this problem, we limit the minimum component length to three letters.

3.2. k-Means identifier

The idea behind the k-means identifier (KM) is that a combination of surface properties of words may be enough to identify words as compounds. For instance, because compounds are constructed using several simplex words, on average they tend to be longer than simplex words or contain more syllables. Obviously, each feature by itself does not provide enough infor-

mation to decide whether a word is a compound or not, but a combination of features may yield useful results.

The input of the KM identifier is, like the other systems we describe here, a set of types. Shallow features are extracted, leading to a feature vector for each type. The collection of feature vectors serves as the input of a k-means clustering algorithm. The value of k describes the number of classes, which is set manually beforehand. All possible combinations of the following features have been used in the experiments.

word length the number of characters in the word;

longest vowel cluster length the length of the longest sequence of adjacent vowels in the word;

vowel cluster count the number of vowel clusters in the word (serving as a rough approximation of syllable count);

cumulative bi-gram probability the sum of the letter bi-gram log probabilities of the word (n-gram probabilities are always smaller than 1, so we take the absolute values of the log probabilities);

cumulative tri-gram probability the sum of the absolute values of the letter tri-gram log probabilities in the word;

cumulative 4-gram probability the sum of the absolute values of the letter 4-gram log probabilities in the word;

cumulative 5-gram probability the sum of the absolute values of the letter 5-gram log probabilities in the word;

valid word regex classification boolean classification based on a regular expression match that yields false if a substring of the word consists of a sequence of three non-alphabetic characters, otherwise the word is classified as true. This feature was added to reduce noise and the influence of digit combinations in datasets.

3.3. Point-wise mutual information identifier

The approach we describe in this section is based on the idea that compound boundaries can be found where "unusual" letter combinations occur. The underlying idea is based on the following observed properties of language:

1. the unequal distribution of character n-grams found in languages such as English and Chinese (Ha et al., 2003);

2. our observation that in simplex and non-compound words certain letter combinations occur more often in isolation than together.

These properties may be different between simplex words and compounds, since compounds contain letter combinations that disregard the unequal distribution of letter sequences (specifically around the boundary of the simplex words in the compound). To clarify, a compound can consist of two simplex words each by itself adheres to the non-linear distribution of letter combinations (n-grams). However, the compound boundary does not, because compound boundary letter combinations simply consist of the beginning and ending of the simplex components.

To measure the regularity of letter combinations, we use the point-wise mutual information (PMI) metric. The following paragraphs explain the process of calculating these values, but let us exemplify its use first with the following example. The word *boekenbeurs* '*book convention*' has a compound boundary between *boeken* and *beurs*.[1] Figure 1 shows that the letter combinations on compound boundary *enbe*, *en* and *be*, occur more often in isolation than together. In other words, the PMI value for the two bi-grams is low, which shows that it is surprising that these characters occur together.

The calculation of the point-wise mutual information (PMI) metric between two letter n-grams, A and B is calculated according to Equation 1. The probability of the concatenated n-grams (AB) is divided by the product of the probabilities of the n-grams by themselves. The results described here are all generated using $n = 2$. Note that the results of the fraction is always smaller than or equal to 1. Taking the log leads to a negative result where smaller fractions lead to larger PMI values. The calculation of the probability of a letter n-gram is computed using the relative frequency (see Equation 2).

$$pmi(A, B) = \log \frac{P(AB)}{P(A) * P(B)} \quad (1)$$

$$P(X) = \frac{\text{count}(X)}{|\text{corpus}|} \quad (2)$$

[1] The simplex *boeken* contains a linking morpheme boundary, but this is irrelevant when identifying compounds.

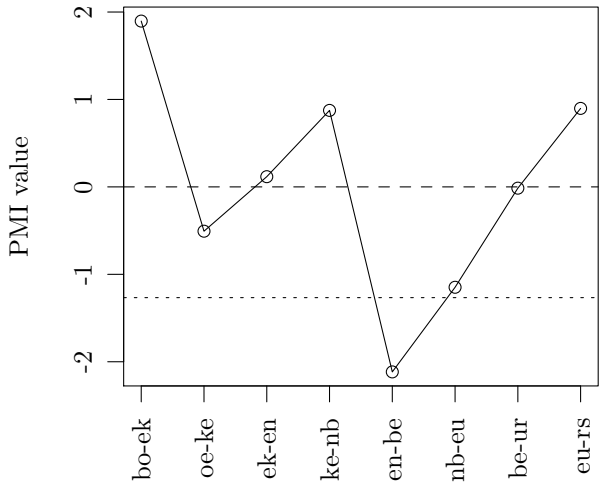

Figure 1. PMI values for the boundaries in *boekenbeurs* '*book convention*'. The dashed line is the mean, the dotted line is one standard deviation below the mean.

Given a word, PMI values are computed by applying a sliding window over the word. Equation 1 is applied to each letter $n*2$-gram, where the relative frequency of the two adjacent letter n-grams constitute A and B respectively. For a word of length k, this procedure yields a set of $k-(n*2)$ PMI values (see Equation 3). Here pmi_0 represents the application of the *pmi* function on the first letter $n*2$-gram in the word. For example, PMI for letter bi-grams in *fietspad* '*bicycle path*' yields a set of values for letter 4-grams: *fiet*, *iets*, *etsp*, *tspa* and *spad*.

$$PMI = \{pmi_i\}_{i=0}^{k-(n*2)} \tag{3}$$

After calculating PMI the resulting values are standardized according to Equation 4, where μ is the mean of the values and σ is the standard deviation. The standardized PMI values (z) are evaluated on their distance to μ. If a value exceeds a given threshold parameter t, which signifies the number of standard deviations from the mean, a word is identified as being a compound.

$$z = \frac{x-\mu}{\sigma} \tag{4}$$

3.4. PMI edge detection identifier

To improve compound identification performance, edge detection filters are applied to PMI values obtained from the system described in Section 3.3. Edge detection is a technique widely used in the field of signal processing to detect discontinuities in multi-dimensional signals. A small matrix, called a *kernel*, is applied to the signal, to either sharpen or smoothen the input. This is accomplished by means of convolution between the kernel and the input signal (Ziou & Tabbone, 1998). Since our data is of a one-dimensional nature, we utilize vector kernels instead of matrices. It is expected that compound boundaries (i.e., dips in PMI values) are more easily detectable after application of an edge detection kernel.

Kernel convolution usually requires values from outside the range of the PMI values. Therefore, "edges" (i.e., the first and last PMI values) are extended with length k, which is calculated according to Equation 5. Then, a sliding window is applied and dot products of the kernel and the PMI values are calculated according to Equation 6. Resulting values are standardized according to Equation 4 and evaluated on the number of standard deviations they differ from the mean. As such, evaluation is identical to that of "regular" PMI values, described in Section 3.3.

$$k = \lfloor \frac{|\text{kernel}|}{2} \rfloor \tag{5}$$

$$\text{PMI}_{\text{filtered}} = \left\{ \sum_{i=0}^{|\text{kernel}|} \text{pmi}_{j+(i-k)} \text{kernel}_i \right\}_{j=0}^{|\text{PMI}|} \tag{6}$$

Two types of convolution kernels are evaluated here, based on *Gaussian* (Equation 7) and *sigmoid* (Equation 8) functions respectively. Kernel values are calculated based on a template and an input parameter p. To ensure that the output values are of the same relative magnitude as the input values, the kernel values are normalized so that the sum of their values equals 1. All kernels we evaluated are of length three (yielding context $k = \lfloor \frac{3}{2} \rfloor = 1$).

$$\text{kernel}_{\text{Gaussian},p} = \left\{ \frac{1}{3} - p, \frac{1}{3} + 2p, \frac{1}{3} - p \right\} \tag{7}$$

$$\text{kernel}_{\text{sigmoid},p} = \left\{ \frac{1}{3} - p, \frac{1}{3}, \frac{1}{3} + p \right\} \tag{8}$$

As shown in Figure 2 (compared to Figure 1) differences between PMI values get larger when edge-detection is applied to the PMI of the word *boekenbeurs*. It is expected that edge detection techniques

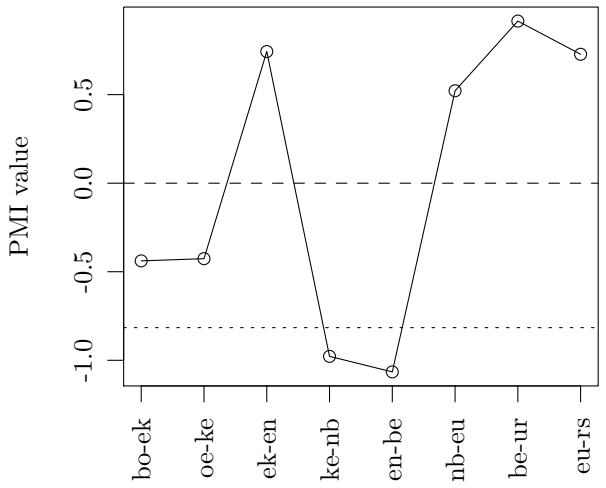

Figure 2. Sigmoid edge detection with $p = 1$ applied to PMI values for the boundaries of the word *boekenbeurs*. The dashed line is the mean, the dotted line is one standard deviation below the mean.

amplify compound boundaries and help identifying compounds.

4. Results

4.1. Dataset

All systems are tested on the Dutch e-Lex dataset (`http://tst-centrale.org/nl/producten/lexica/e-lex/7-25`), which contains a total of 96,219 morphologically segmented lemma entries. The morphological segmentations of the lemmata were used during evaluation of our systems to determine which words were compounds. We identified compounds by parsing hierarchical morphological segmentations. A total of 68,686 lemma entries that contain compositional morphology were identified as compounds. The remaining 27,533 words are simplex words possibly annotated with derivational morphology structure.

4.2. System results

The results of all systems can be found in Table 1. The compound component identifier has the highest precision, which was expected as it exploits the fact that compounds are built from simplex words and e-Lex contains a substantial amount of simplex words. The CC identifier does not have any manually tunable parameters. As such, it can not be modified to improve recall.

Table 1. Results for all systems on the e-Lex dataset. P is precision, R is recall and F_1 is F_1 score. For k-means experiments, feature combinations for optimal F_β scores are selected. For the PMI experiments, N, G and S indicates that no edge detection, Gaussian or sigmoid filters have been applied, p indicates the kernel parameter value and t refers to the threshold used for classification. Italics indicate system dependent best results. Overall best results are shown in boldface.

	P	R	F_1
Compound component identifier			
	89.603	*63.268*	*74.166*
k-means identifier			
max $F_{0.5}$, $k = 2$	*83.842*	61.573	71.002
max F_1, $k = 2$	76.236	*80.134*	*78.136*
max F_2, $k = 2$	76.236	*80.134*	*78.136*
Point-wise mutual information identifier			
N, $t = 1$	*74.683*	73.842	74.260
N, $t = 0.1$	73.092	**99.999**	84.454
N, $t = 0.3$	73.103	99.987	*84.457*
G, $p = 0.3, t = 1$	*74.754*	78.051	76.367
G, $p = 0.5, t = 0.2$	73.096	**99.999**	84.456
G, $p = 0.7, t = 0.2$	73.118	99.975	**84.463**
S, $p = 0.3, t = 1$	*73.816*	88.989	80.695
S, $p = -1.9, t = 0.1$	73.093	**99.999**	*84.455*

The k-means identifier is applied to the data with $k = 2$ target classes. In each experiment, the class that fits the compound class best, is selected as the compound class. The best results are found when using only a sub-set of the features. When optimizing on $F_{0.5}$, the longest vowel cluster length and cumulative 5-gram probability features are used. When optimizing F_1 or F_2, only the vowel cluster count feature is used.

In an attempt to improve precision and recall values separately, experiments have been conducted by maximizing commonly used F_β measures. The $F_{0.5}$ measure weights precision higher than recall and the F_2 measure puts more emphasis on recall than precision as shown in Equation 9. It is interesting to note that the $F_{0.5}$ optimized k-means identifier is able to produce results comparable to the compound component identifier, even though the sources of information are completely different.

$$F_\beta = (1 + \beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + \text{recall}} \qquad (9)$$

The threshold t for evaluating PMI values is set to values between 0.1 and 3.5 standard deviations from the mean. The best F_1 and recall scores for all three PMI variants (unfiltered, sigmoid and Gaussian) are found

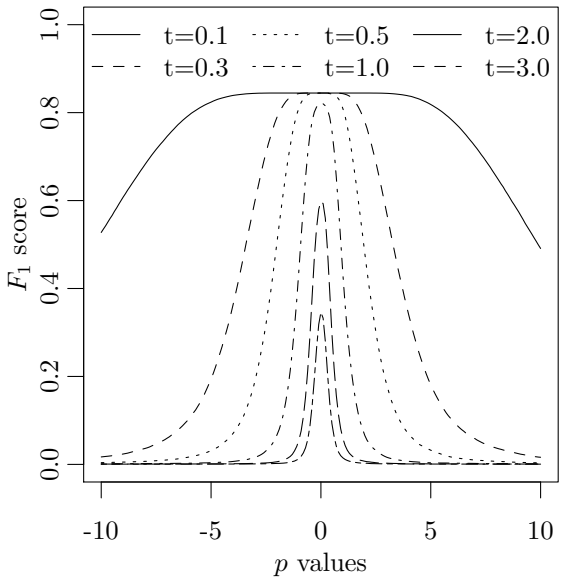

Figure 3. F_1 score results for the sigmoid filter with varying t and p values.

with t values between 0.1 and 0.3. The best results of all three systems are comparable. Best precision is obtained with slightly higher thresholds, ranging between 0.5 and 1. Increasing the threshold to values higher than 1 rapidly degrades performance: unfiltered PMI values evaluated to a threshold of 1.5 already result in a F_1 score of 45.570, while a threshold of 3.5 only yields a F_1 score of 1.744, which is similar to the Gaussian and sigmoid results.

Precision obtained by PMI variants never reach those generated by the compound component and k-means methods.

Even though the Gaussian, sigmoid and unfiltered PMI settings all yield significantly different distributions of F_1 scores , their behavior is similar when varying the values of the parameter p (varying the impact of the values in the context). Both increasing or decreasing p to a certain extent lead to similar precision, recall, and F_1 scores. The best F_1 scores are obtained with p around 0. Performance rapidly decreases with larger absolute p values. This is illustrated in Figure 3. Here we see the F_1 score results of the sigmoid filter with various values for t and p. This indicates that essentially the application of the kernels does not lead to improved results.

5. Conclusion

In this paper we have investigated different approaches to the identification of compounds in a set of words. These systems will be used to select compounds that will be structurally annotated for compound boundaries by human annotators in a later stage.

The compound component (CC) system is based on a similar system that has been used in the past to identify compound boundaries automatically. The CC system identifies compounds in a set of words by identifying components (which are simplex words) in the given set of words. This system leads to a high precision. A second approach, based on the unsupervised k-means clustering using shallow features of words leads to similar performance using different information.

We compared these results against a third approach, which is based on point-wise mutual information (PMI) values of the consecutive letter combinations in the word. These systems result in the highest recall (near 100%) and F_1 scores (over 84%).

As future work, we are interested in using the PMI system to identify the actual compound boundaries (similarly to the use of the original CC system and others described as previous research earlier). However, preliminary results show that the PMI systems also identify boundaries that are not actual compound boundaries, but may correspond to other morphological boundaries. In the setting described here, this is not a problem. A word is considered a compound if *at least* one boundary is found. However, identifying too many boundaries has a negative effect on the precision of the identification of compound boundaries. Future work will concentrate on the development of systems that identify compound boundaries based on the compound identification systems.

Ultimately, we would like to use these systems to identify potential compounds from large plain text corpora. For Dutch we plan to identify compounds in the SoNaR corpus (Oostdijk et al., 2008; Oostdijk et al., In press). SoNaR is a large collection of written contemporary Dutch texts. It contains approximately 500 million tokens and as such it provides a good starting point to identify naturally occurring compounds in Dutch.

Acknowledgments

The research described in this paper has been performed in the context of the Automatic Compound Processing (AuCoPro) project. This is a collaboration between researchers from North-West Univer-

sity (Potchefstroom, South Africa), the University of Antwerp (Belgium) and Tilburg University (The Netherlands). The project concentrates on the identification of compound boundaries (Tilburg) and the semantic relations between the elements in compounds (Antwerp). This research is performed both on Dutch and Afrikaans (Potchefstroom).

The project is co-funded by a joint research grant of the Nederlandse Taalunie (Dutch Language Union) and the Department of Arts and Culture (DAC) of South Africa and a grant of the National Research Foundation (NRF) (grant number 81794).

We would also like to thank Sylvie Bruys, who helped during the initial phase of the project and who has manually annotated compound data.

References

Alfonseca, E., Bilac, S., & Pharies, S. (2008). German decompounding in a difficult corpus. *CICLing Conference on Computational Linguistics and Intelligent Text Processing* (pp. 128–139). Berlin, Heidelberg: Springer.

Booij, G. (1996). Verbindingsklanken in samenstellingen en de nieuwe spellingregeling. *Nederlandse Taalkunde*, *2*, 126–134.

Creutz, M., & Lagus, K. (2005). *Unsupervised morpheme segmentation and morphology induction from text corpora using morfessor 1.0* (Technical Report). Helsinki University of Technology.

CText (2005). *Ckarma: C5 kompositumanaliseerder vir robuuste morfologiese analise* (Technical Report). Centre for Text Technology (CText), North-West University, Potchefstroom, South Africa.

Garera, N., & Yarowsky, D. (2008). Translating compounds by learning component gloss translation models via multiple languages. *Proceedings of the 3rd International Conference on Natural Language Processing (IJCNLP)* (pp. 403–410).

Ha, L. Q., Sicilia-garcia, E. I., Ming, J., & Smith, F. J. (2003). Extension of Zipf's law to word and character n-grams for English and Chinese. *Journal of Computational Linguistics and Chinese Language Processing*, *8*, 77–102.

Koehn, P., & Knight, K. (2003). Empirical methods for compound splitting. *Proceedings of the 11th Annual Meeting of the European Chapter of the Association for Computational Linguistics (EACL); Dublin, Ireland* (pp. 187–194).

Macherey, K., Dai, A. M., Talbot, D., Popat, A. C., & Och, F. (2011). Language-independent compound splitting with morphological operations. *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics; Portland, OR, USA* (pp. 1395–1404). New Brunswick:NJ, USA: Association for Computational Linguistics.

Mima, H., & Ananiadou, S. (2000). An application and evaluation of the c/nc-value approach for the automatic term recognition of multi-word units in japanese. *Terminology*, *6*, 175–194. Special issue on Japanese Term Extraction.

Oostdijk, N., Reynaert, M., Hoste, V., & Schuurman, I. (In press). The construction of a 500-million-word reference corpus of contemporary written dutch. In P. Spyns and J. Odijk (Eds.), *Essential speech and language technology for dutch: Results by the stevin-programme*, chapter 13. Berlin Heidelberg, Germany: Springer-Verlag.

Oostdijk, N., Reynaert, M., Monachesi, P., van Noord, G., Ordelman, R., Schuurman, I., & Vandeghinste, V. (2008). From D-Coi to SoNaR: A reference corpus for dutch. *Proceedings on the sixth international conference on language resources and evaluation (LREC 2008); Marrakech, Marokko* (pp. 1437–1444). ELRA.

Pilon, S., & Puttkammer, M. J. andVan Huyssteen, G. B. (2008). Die ontwikkeling van 'n woordafbreker en kompositumanaliseerder vir afrikaans (the development of a hyphenator and compound analyser for afrikaans). *Literator*, *29*, 21–41.

Ryder, M. E. (1994). *Ordered chaos: The interpretation of english noun-noun compounds*, vol. 123. University of California press.

van Huyssteen, G. B., & van Zaanen, M. M. (2004). Learning compound boundaries for afrikaans spelling checking. *Pre-Proceedings of the Workshop on International Proofing Tools and Language Technologies; Patras, Greece* (pp. 101–108).

Wiese, R. (1996). *The phonology of german*. New York: NY, USA: Oxford University Press.

Ziou, D., & Tabbone, S. (1998). Edge detection techniques - an overview. *International Journal of Pattern Recognition and Image Analysis*, *8*, 537–559.