Splitting of Compound Terms in non-Prototypical Compounding Languages

Elizaveta Clouet and **Béatrice Daille**
LINA, University of Nantes
{elizaveta.clouet,beatrice.daille}@univ-nantes.fr

Abstract

Compounding is present in a large variety of languages in different proportions. Compound rate in the text obviously depends on the language, but also on the genre and the domain. Scientific and technical texts are especially conducive to compounding, even in the languages that are not traditionally admitted as highly compounding ones. In this article we address compound splitting of specialized terms. We propose a multi-lingual method of compound recognition and splitting, which uses corpus frequencies, lexical data and optionally linguistic rules. This is a supervised method which requires a small amount of segmented compounds as input. We evaluate the method on two languages that rarely serve as a material for automatic splitting systems: English and Russian. The results obtained are competitive with those of a state-of-the-art corpus-driven approach.

1 Introduction

Compounding is a method of word formation consisting of a combination of two (or more) lexical elements that form a unit of meaning. In this work we only handle so called "closed compounds" (Macherey et al., 2011), i.e. those forming also a graphical unit. A great number of languages resort to this word formation. In some of them such as German, Dutch (Germanic family), Estonian or Finnish (Uralic family) compounding is very regular and well described. In other languages it is less productive (e.g. Slavic family), or even marginal (most of Romance languages). This phenomenon is particularly productive in specialized domains because of the necessity to denote the domain concepts in a very concise and precise way. In addition, specialized texts contain many neoclassical compounds (Namer, 2009), i.e. compounds with some elements of Greek or Latin etymological origin: *hydro* + *logy* = *hydrology*.

In this article we discuss processing of compound terms, and we carry out the experiments with English and Russian, which are not prototypical compounding languages. As a matter of fact, compounding in English is rather productive and widely investigated in linguistic studies. But this language is rarely subject to experiments in automatic compound splitting. The first reason is that most of English compounds are formed by simple concatenation (*airfoil* = *air* + *foil, streamtube* = *stream* + *tube*), so their splitting is supposed to be straightforward. The second reason is that many compounds in highly compounding languages should be translated into English as multi-word expressions. That is why the works addressing automatic compound splitting in the context of machine translation often admit that English contains only few closed compounds (Koehn and Knight, 2003; Macherey et al., 2011). In these works, the use of English parallel texts helps to extract the multi-word equivalents of compounds from the texts in highly compounding languages. We assume that English compound terms are still worth splitting and analyzing. The first reason is that the assumption of independent occurring of English compound elements fails when we consider neoclassical compounds. The second ground is that distinguishing between compounds and non-compound out-of-dictionary words (named entities, derivative forms, etc.) can be problematic.

Proceedings of the First Workshop on Computational Approaches to Compound Analysis, pages 11–19,
Dublin, Ireland, August 24 2014.

In the Russian language the elements of compounds do not often appear as independent words in texts, for example:

водоснабжение 'water supply'
vodosnabzhenie[1] = voda 'water' + snabzhenie 'supply'

Here the inflection "a" of the first component is omitted and the linking morpheme "o" is inserted.

Compounding in Russian is less regular than, for instance, in German. Therefore most of NLP systems for Russian, to our knowledge, store usual compound parts in the lexicon. For specialized vocabularies this solution does not seem to be sufficient, since the new compounds terms constantly appear.

To handle compounds in typologically different languages, including languages with non-independent components, we propose a corpus-driven splitting system using also string similarity and able to integrate language-specific rules[2].

The article has the following structure. Section 2 gives a review of some compound splitting methods proposed in the literature. Section 3 presents our splitting method. In Section 4 the experiments and data are described. We discuss the results and analyse the errors. We also compare our system to the state-of-the-art corpus-based method of Koehn and Knight (2003). We conclude with Section 5.

2 Related Works

Compound splitting was addressed in many NLP works, as a standalone task or in the pipeline of another application: machine translation (Koehn and Knight, 2003; Macherey et al., 2011; Stymne et al., 2013), information retrieval (Braschler and Ripplinger, 2004; Chen and Gey, 2001), etc.

To deal with the non-independent compound elements in morphologically rich languages, different solutions have been proposed. It is possible to store separately the compound stems and the linking morphemes (the morphemes that are inserted at the component boundaries to form a compound), and to have a grammar to combine them. This solution is realized in the morphological analyser for German SMOR (Schmid et al., 2004). The construction of such a finite-state morphology for a new language is a costly task in terms of time and efforts.

Another approach is to formalize the component modifications as a set of rules describing addition, but also deletion and substitution of some character sequences on the component boundaries within a compound. Thus, the compound splitter BananaSplit (Ott, 2005) uses a set of linguistic rules for German to restore independent forms from compound forms, and validates the restored forms with the help of a monolingual dictionary.

Other methods resort to the corpora to validate the analyses. A pioneering work using corpus statistics for compound splitting was done by Koehn and Knight (2003). The algorithm generates all possible segmentations for a given word (taking into account some linking morphemes), and gives a probability for each segmentation, estimated from the geometric mean of the component frequencies in the corpus. The segmentation with the highest score is classed as the best.

Probabilistic splitting methods using machine learning technologies have been proposed (Dyer, 2009; Hewlett and Cohen, 2011; Macherey et al., 2011). Actually they are less precise than the language-specific methods, but their advantage is the usability for any language. Statistical methods also tend to integrate some linguistic knowledge (e.g. list of linking morphemes).

3 Compound Splitting Method

Our concern was to design a corpus-driven system that could be applied to different languages, but also able to integrate linguistic knowledge. For a given word, the system makes the decision as to whether it is a compound, and for compounds it gives one or several candidate analyses ranked by their scores (in this work we use up to five candidates). Figure 1 illustrates the splitting mechanism, as well as the system training needed to set up splitting parameters.

[1] Here and further for Russian examples transliteration is given.

[2] https://logiciels.lina.univ-nantes.fr/redmine/projects/compost

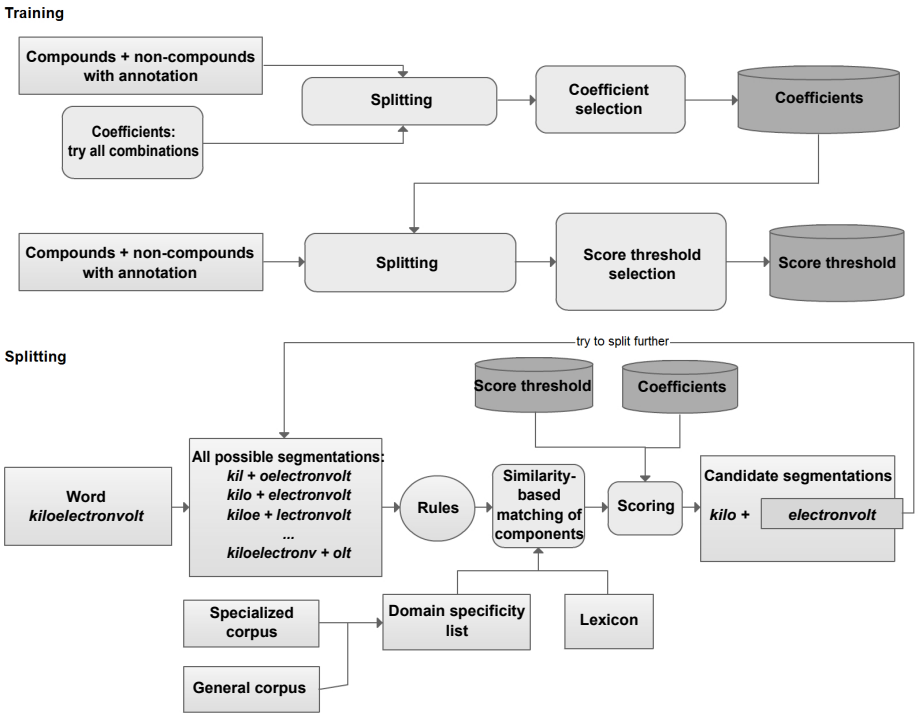

Figure 1: Parameters setting and compound splitting

3.1 General Method Description

To split a candidate compound, we start generating all its possible two-part segmentations beginning with the components of minimum permitted length: we used the minimum length of 3 characters, which is a frequent choice for compound splitting systems (Koehn and Knight, 2003; Dyer, 2009).

RU ветротурбина 'wind turbine'
vetroturbina → vet + roturbina
vetroturbina → vetr + oturbina
vetroturbina → vetro + turbina
...
vetroturbina → vetroturb + ina

If we can provide rules for this language to restore independent lexemes from non-independent components, we apply them to the component candidates. We will further refer to these rules as "linguistic rules" or simply "rules". For English only a few rules for correcting lemmatization were needed, for instance "ed" → "e": *health-based → health + base*. For Russian we defined a set of 15 rules to transform the left component and 14 rules to transform the right component. In the example above, the rule *"o" → ""* can be applied to the 3rd segmentation candidate: *vetro → vetr*.

If the rules are not available or not sufficient to cover all modifications, the potential lemmas are proposed using a similarity measure. We use normalized Levenshtein distance (Frunza and Inkpen, 2009) as similarity measure. Thus, *vetr* is not an independent word, and some candidate lemmas were found: *veter* 'wind', *veto* 'veto', and *vetka* 'branch', all with the similarity value of 0.8.

For each candidate segmentation, the lemmas for both components are matched with a lexicon and with a word list extracted from a specialized monolingual corpus. The lexicon contains a monolingual dictionary filtered by POS and combined with a list of neoclassical roots and prefixes to process neoclassical compounds. In this work, we kept only nouns, adjectives, verbs and adverbs in the lexicon to reduce prospective errors, even if we are aware that compound components can belong to other POS (pronouns, numerals), but in the languages that we investigated such formations are minor. Then, the segmentation score is calculated (see below).

After that we try to split the right side component further in a recursive manner, and so on up to a certain level. This level is a parameter corresponding to the maximum expected number of components, for instance:

EN kiloelectronvolt
kiloelectronvolt → kilo + electronvolt
electronvolt → electron + volt

Finally, the algorithm returns a top 5 of the best segmentations ordered by their score. For RU ветротурбина the output is:

veter turbina 0.81
veto turbina 0.8
vetka turbina 0.8

The correct split is *veter* 'wind' + *turbina* 'turbine', and indeed it has the best score given by the program.

3.2 Score Calculation

At each level of segmentation recurrence (at the first level the word is divided into 2 components, at the second level into 3 components and so on), the segmentation score is calculated as follows:

$$Score(segm) = \begin{cases} \frac{Score(compA)+Score(compB)}{2} & \text{if exact match} \\ \frac{Score(compA)+Score(compB)}{nbComp} & \text{otherwise} \end{cases}$$

where *nbComp* is the number of components in the word at this level of recurrence. "Exact match" means that all components are found "as is" in the dictionary or corpus. We consider that two-part segmentations are more frequent and more plausible than those containing three and more parts, that is why we favor the splits into two components. For example for the RU ветроколеса vetrokolesa (plural form of 'windwheel'):

Correct split is vetro.kolesa = veter 'wind' + koleso 'wheel'

$$Score(vetro + kolesa) = \frac{Score(veter) + Score(koleso)}{2}$$

Incorrect split is vetro.ko.lesa = veter 'wind' + ko 'co-, prefix' + les 'wood'

$$Score(vetro + ko + lesa) = \frac{Score(veter) + \frac{Score(ko)+Score(les)}{2}}{3}$$

An exception occurs for the splits in which the components exactly match the dictionary or corpus (e.g. EN *kiloelectronvolt*). These splits are realistic and we do not penalize them dividing by *nbComp*.

$$Score(kilo + electron + volt) = \frac{Score(kilo) + \frac{Score(electron)+Score(volt)}{2}}{2}$$

This way of calculating the segmentation score at each splitting level corresponds also to the theoretical principle defended by Benveniste (1974) and that we share: a compound word is always formed by two components. Among these two components, one can be a compound itself, but even in this case only two components take part in a compound formation. However, we have noticed in a separate experiment that the score obtained in such a way is rather close to the arithmetic mean of the component scores.

The score of a component is calculated by a linear interpolation:

$$Score(comp) = \alpha\ sim(comp, lemma) + \beta\ inDico + \gamma\ inCorpus + \delta\ DSpec \quad (1)$$

where *sim(comp,lemma)* means similarity between the component and a candidate lemma (from 0 to 1), *inDico* and *inCorpus* indicate the existence of the lemma in the lexicon and in the corpus (0 or 1), *DSpec* means domain specificity value for this lemma (see below). The sum of coefficients α, β, γ and δ is 1.

If a lemma in the lexicon is neoclassical, which means it does not occur individually in the corpus, we assign it the value *inCorpus* = 1, otherwise the score would be penalizing for all neoclassical compounds. The coefficients α, β, γ and δ are parameters and should be learned for each language using a small amount of training data (about a hundred of compounds per language).

3.3 Domain specificity

Since we are dealing with specialized vocabularies, we exploit the notion of domain specificity of a lexical unit, or in other terms, the relevance of the lexical unit for a given domain. Our domain specificity is based on Ahmad's "weirdness ratio" (Ahmad et al., 1992) which is calculated as the ratio between the term frequency in a specialized corpus and the term frequency in a general corpus. As we use a linear interpolation formula, each component of the sum should be between 0 and 1. So we normalize a weirdness ratio of a lemma dividing it by the maximum weirdness ratio found for this specialized corpus.

Domain specificity helps to disambiguate splitting variants. So, for our Russian example ветротурбина *vetroturbina*, we would like to rank the analysis *veter turbina* better than *veto turbina* or *vetka turbina*, and we rely on the fact that the word *veter* 'wind' is more specific for the given wind energy domain than the words *veto* 'veto' and *vetka* 'branch'.

Our method can also be used to process general language compounds. In this case, one should exploit a list of the words extracted from a general language corpus, with their relative frequency in this corpus instead of the domain specificity list extracted from a specialized corpus.

3.4 Training phase

The training phase consists of two steps: optimizing the coefficients used for component scoring, and defining a score threshold to recognize whether an out-of-dictionary word is a compound.

Firstly, for a given language we train our algorithm on a certain number of words annotated with their segmentations (training dataset) in order to find the best coefficients α, β, γ and δ (see equation (1)). We try all the possible coefficients from 0 to 1 with a step of 0.1 (the sum of all coefficients equals one) and we retain the combination that results in the highest recall and precision. In this step, the parameters giving the highest recall are generally the same as the ones giving the best precision.

Recall for Top N is calculated as the ratio between the number of words that have a correct split among N segmentations ranked as the best ones by the algorithm and the total number of analysed compounds:

$$Recall = \frac{nbSegmentedCorrectly}{nbCompounds} \quad (2)$$

Precision for Top N is calculated as the ratio between the number of words having a correct split among N segmentations ranked as the best ones by the algorithm and the number of words split by the algorithm:

$$Precision = \frac{nbSegmentedCorrectly}{nbSegmented} \quad (3)$$

Note that the denominator here is the number of words which were split, and not the number of all candidates produced.

Secondly, we apply again the same algorithm with the selected coefficients to a training dataset. The current objective is to determine an optimal score threshold over which the given word is probably a compound. To do that, we try all the thresholds from 0 to 1 with a step of 0.05 and calculate recall and

Language	Specialized Corpus	General Corpus	Dictionary	NCP-list
EN	314,549	5,001,609	145,542	437
RU	323,929	109,115,810	526,876	132

Table 1: Resources Statistics

precision for Top 1 and Top 5 of segmentations ranked as the best ones by the algorithm and with a score higher than this threshold.

The user can therefore choose the threshold that he considers as optimal depending on the target application. For instance, for lexicon acquisition task splitting precision will be more important than recall, whereas for the supervised translation task splitting recall may turn out more important. In this work, we optimize the threshold for a better precision. This allows us to compare the results to those of a state-of-the-art method (Koehn and Knight, 2003) which privileges precision rather than recall.

4 Experiments and Results

We use a unified strategy for the two major types of compounds, native and neoclassical. We also consider in our experiments some prefixed words. Standard prefixation is, of course, a subtype of derivation, and not of compounding. However, in some boundary cases it is difficult to catch the difference between prefixed formations and neoclassical or native compounds: compare prefix *bi-* to neoclassical root *uni-* according to Béchade (1992). Moreover, from the operational point of view, some prefixed words can be split in the same manner as compounds.

4.1 Data

Table 1 summarizes the size of the resources used in this work. Wind Energy corpora[3] were crawled from Web pages by means of Babouk (Groc, 2011), a tool dedicated to automatic compilation of domain-specific corpora, with the use of a key-term list. To compute the domain specificity of a term, we needed its frequency in the general language corpus. For the English language, we used a subpart of the *New York Times* corpus [4]. For Russian, we used a frequency list computed from The Russian National Corpus[5].

Concerning the dictionaries, we exploited the monolingual parts of the following bilingual general language dictionaries: FR-EN from the ELRA catalogue[6] for the English part, RU-EN from *English-Russian full dictionary*[7] for the Russian part.

To obtain the lists of neoclassical roots and prefixes, we firstly took the English and Russian equivalents of neoclassical elements enumerated in (Béchade, 1992). Secondly, we completed these lists with some prefixes of Latin, Greek or another origin that have equivalent in numerous languages (co-, pre-, post-, trans-, etc.). For English we consulted publicly available morpheme translation tables[8]. We will refer to this source as *NCP-list*.

We extracted from the specialized corpus a lexicon which contains only the words that (1) are either nouns or adjectives, because most compounds belong to these two categories (about 80% according to cross-language study reported in (Scalise and Fabregas, 2010)); (2) have frequencies greater than 2; (3) are 6 characters and more long; and (4) do not appear in the general language dictionary used. We applied these criteria to eliminate the words that are not compounds in order not to split them. For POS filtering, TreeTagger [9] was used.

The lexicons were randomly sorted and manually annotated until we obtained between 200 and 300 compounds. Each compound word was annotated with the correct segmentation(s), other words were marked as non-compounds. The annotated lexicons were then divided into two parts: a training dataset

[3] http://www.lina.univ-nantes.fr/?Ressources-linguistiques-du-projet.html
[4] http://catalog.ldc.upenn.edu/LDC2008T19.
[5] http://corpus.leeds.ac.uk/serge/frqlist/rnc-modern-lpos.num.html
[6] http://catalog.elra.info/product_info.php?products_id=667
[7] http://dicto.org.ru/xdxf.html
[8] http://www.lina.univ-nantes.fr/?Linguistic-resources-from-the,1676.html
[9] http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/

Language	No. of lemmas	No. of compounds	Compound rate
	Training dataset		
EN	404	106	27%
RU	485	116	24%
	Test dataset		
EN	401	100	25%
RU	700	170	24%

Table 2: Extracted Lexicon Size and Compound Rate

used to train our system and a test dataset used to evaluate the splitting quality. Table 2 shows the lexicon sizes for each language, as well as the number of compounds found.

4.2 Experiments

The application of our method to the training data allowed us to obtain the parameters required for splitting algorithm (see Table 3). The selected coefficients were then used to analyse a test dataset for each language. The evaluation in terms of recall, precision, and F-measure is presented in Table 4.

The results vary according to the language. For Russian the precision turns out to be higher than for English: 81% for Top 1 and 82% for Top 5 for Russian against 74% for Top 1 and 78% for Top 5 for English. However, the recall is much higher for English than for Russian: 87% for Top 1 and 91% for Top 5 against 52% for Russian. It can be explained by the abundance of component modifications in the latter and by the lack of correct lemmatization for many terms. We can also notice that for Russian the gain obtained in the Top 5 of segmentation candidates is very small compared to the Top 1, which means that for this language if the correct candidate was found, it was almost always ranked as the best one.

4.3 Error Analysis

Pre-processing is very important for proper splitting and recognition of compounds and non-compounds. In our experiments lemmatization was performed by a probabilistic tool, trained on general language data, so some compound and highly specialized non-compound terms were not lemmatized correctly. The rules that we have introduced in order to deal with such cases help to properly split non-lemmatized compounds, but they do not help to recognize non-compounds. Correction of lemmatization should be done before dictionary filtering, it will enable the identification of some words as in-dictionary ones, and enable them to be kept unsplit. This will decrease the number of false segmentations and consequently improve precision.

Some false positive segmentations were due to the splitting of named entities: EN *Cambridge* split into *cam* + *bridge*. Named entity recognition would decrease the number of erroneous segmentations.

Among the errors intrinsic to our method, we can state those related to the use of string similarity, so certain affixes or combinations of affixes may be confused with the independent words, e.g.,

RU керосиновый 'kerosene_ADJ', non-compound word
Erroneous segmentation: kerosinovyj = kerosin 'kerosene_N' + novyj 'new'

To avoid this problem, a large set of morphemes for each morphologically rich language is needed.

4.4 Comparison to a State-of-the-art Method

We compared our method to the corpus-driven approach proposed by Koehn and Knight (2003) that became a state-of-the-art method of compound splitting. We applied it to our experimental data with the

Language	Coefficients (α β γ δ)	Score threshold
EN	0.7 0.1 0.1 0.1	0.85
RU	0.3 0.1 0.4 0.2	0.8

Table 3: Parameters Learned on the Training Dataset

Language	Top 1			Top 5		
	R	P	F	R	P	F
EN	**87**	74	**80**	**91**	78	**84**
RU	**52**	**81**	**63**	52	**82**	64

Table 4: Splitting quality evaluation in terms of R(ecall), P(precision) and F(-measure).
Highlighting corresponds to the experiments in which our method outperforms (Koehn and Knight 2003).

Language	Top 1			Top 5		
	R	P	F	R	P	F
EN	71	87	78	74	83	78
RU	48	69	57	62	68	65

Table 5: Splitting quality by (Koehn and Knight 2003) method.

usage of the same rules, stop lists and NCP-lists as we have used in our method. For the elements from the NCP-list, a corpus frequency of 1 was artificially assigned, otherwise they had no chance of being split correctly by this corpus-based method because they never independently occur in the texts. The results are presented in Table 5.

We should notice that this method can produce several segmentation candidates, including the original word kept unsplit. The unsplit word can be ranked as first, second and so on, unlike our method which first of all takes the decision whether a word is a compound or not. If for a given compound the algorithm returned the unsplit word as the first candidate, but a correct segmentation was among the first 5 candidates (e.g. *monopile* → 1) *monopile* 2) *mono pile*), we evaluated this analysis as correct for the Top 5, otherwise the evaluation would be penalizing for this method. But according to the same logic, if for a non-compound the first candidate was unsplit, and the second was split (e.g. *district* → 1) *district* 2) *di strict*), we evaluated it as correct for Top 1 and incorrect for Top 5. The consequence of such an evaluation is that the precision can be lower for the Top 5 experiments than for the Top 1 experiments, as we actually stated.

The method we proposed was more precise than Koehn and Knight's one for Russian (up to 14 points), but less precise for English (up to 13 points). Our method segmented a larger number of compounds in all experiments, except for the Top 5 for Russian. F-measure was also higher with our method (from 2 to 6 points) except for the Top 5 for Russian (1 point difference).

Koehn and Knight's method is based on corpus frequency, and it is possible to integrate linguistic rules too. Our method, besides the corpus evidence and linguistic rules, takes into account the evidence from a monolingual dictionary and the string similarity between compound elements and their lemmas.

The fact that Koehn and Knight's method does not exploit string similarity guards it against producing implausible segmentations and allows achieving, when combined with linguistic rules, a high precision, especially for the English language. However, the use of string similarity allows our method to detect some components not appearing as independent words in the texts/dictionaries even if the rules used do not cover their transformation into lemmas, cf. RU example *vetroturbina*, which was not split by Koehn and Knight's method.

Koehn and Knight's method is also known to keep some compounds unsplit if they occur more often in the corpus than their components, like the example of EN *monopile*. The neoclassical compounds also tend to remain unsplit with this method.

5 Conclusion

In this article, we have investigated the compound splitting of lexical units from specialized domains. We proposed a method for compound recognition and splitting that exploits language-independent features (corpus frequency, string similarity), lexical data (monolingual dictionary, list of neoclassical elements and prefixes) and linguistic rules. We tested this method on two languages that are not traditionally considered as highly compounding ones, but as we have seen, the specialized texts in these languages

contain many compounds.

Our method turns out to be competitive with the state-of-the-art corpus-driven approach of Koehn and Knight (2003). The advantage of our method is its domain-orientation, which enables to boost correct segmentations containing specialized words into the Top 1. The use of string similarity may introduce some false positive splitting, but at the same time, it allows detection of additional components not covered by the rules. This point is particularly important to handle compounds in morphologically rich languages in which retrieval of the independent forms from compound elements is not straightforward, such as Russian. The results of both methods tested are lower for Russian than for English. This confirms that processing of Russian compounds is actually challenging for NLP systems and worth further investigating.

References

Khurshid Ahmad, Andrea Davies, Heather Fulford, and Margaret Rogers. 1992. What is a term? The semi-automatic extraction of terms from text. In *Translation Studies: An Interdiscipline*, pages 267–278, Amsterdam/Philadelphia. John Benjamins.

Hervé-D. Béchade. 1992. *Phonétique et morphologie du français moderne et contemporain*. Presses Universitaires de France.

Emile Benveniste. 1974. *Problèmes de linguistique générale*. Gallimard, Paris.

M. Braschler and B. Ripplinger. 2004. How effective is stemming and decompounding for german text retrieval. In *Information Retrieval*, pages 291–316.

A. Chen and F.C. Gey. 2001. Translation term weighting and combining translation resources in cross-language retrieval. In *Proceedings of TREC Conference*.

Chris Dyer. 2009. Using a maximum entropy model to build segmentation lattices for mt. In *Proceedings of HLT-NAACL 2009*.

0. Frunza and D. Inkpen. 2009. Identification and disambiguation of cognates, false friends, and partial cognates using machine learning techniques. In *International Journal of Linguistics*.

Clément De Groc. 2011. Babouk : Focused Web Crawling for Corpus Compilation and Automatic Terminology Extraction. In *The IEEEWICACM International Conferences on Web Intelligence*, pages 497–498, Lyon, France.

D. Hewlett and P. Cohen. 2011. Fully unsupervised word segmentation with BVE and MDL. In *Proceedings of ACL 2011*, pages 540–545, Portland, Oregon.

P. Koehn and K. Knight. 2003. Empirical methods for compound splitting. In *Proceedings of EAC 2003*, Budapest, Hungary.

K. Macherey, A.M. Dai, D. Talbot, A.C. Popat, and F. Och. 2011. Language-independent compound splitting with morphological operations. In *Proceedings of ACL 2011*, pages 1395–1404, Portland, Oregon.

Fiammetta Namer. 2009. *Morphologie, lexique et traitement automatique des langues*. Lavoisier, Paris.

N Ott. 2005. Measuring semantic relatedness of German compounds using GermaNet.

Sergio Scalise and Antonio Fabregas. 2010. Why compounding? In Sergio Scalise and Irene Vogel, editors, *Cross-disciplinary issues in compounding*, volume 311 of *Current issues in linguistic theory*, pages 1–18. John Benjamins Publishing Company, Amsterdam/Philadelphia.

Helmut Schmid, Arne Fitschen, and Ulrich Heid. 2004. SMOR: A german computational morphology covering derivation, composition, and inflection. In *Proceedings of LREC 2004*, pages 1263–1266, Lisbon, Portugal.

Sara Stymne, Nicola Cancedda, and Lars Ahrenberg. 2013. Generation of compound words in statistical machine translation into compounding languages. *Computational Linguistics*, 39(4):1067–1108.