Corpus-Driven Splitting of Compound Words

Ralf D. Brown
Language Technologies Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213-3890 USA
ralf+@cs.cmu.edu

Abstract

A method is presented for splitting compound words into their constituents based on cognate words in the other language of a parallel corpus. A minor extension to the method using a bilingual lexicon (which may be statistically derived from the corpus) allows the decompounding of words that do not have cognates in the other language. Further, the algorithm can produce, as a by-product, a mapping from compound words in one language to phrases in the other language.

The method described in this paper is applied to an example-based machine translation (EBMT) by decompounding the training corpus, and training both the EBMT system and the bilingual lexicon it uses for subsentential alignment from the decompounded corpous. Compared with the original corpus, the decompounded corpus substantially reduces the incidence of word-alignment failure, resulting in a modest overall improvement in performance.

1 Introduction

When dealing with parallel English-German texts in the medical domain, a large proportion of the tokens (typically Latin- or Greek-derived medical terms) in the text can be treated as cognates – except that the German version frequently combines the equivalent of multiple English words into one compound word. If one can decompose the compound words, various other processing and uses of the parallel text will be simplified and/or enhanced. Examples of such applications are information retrieval, where it is used in stemming (Braschler & Schäuble 2000), and subsentential alignment as used in statistical and example-based machine translation (Brown et al. 1990; Brown 1999). Any algorithms which assume one-to-one correspondences between words, such as Competitive Linking (Melamed 1997) or automated lexicon extraction (Brown 1997), will also be improved by decompounding of compound words. Finally, text normalization for a variety of applications, such as speech recognition (Adda et al. 1997; Adda-Decker et al. 2000), can benefit from decompounding.

In this paper, we will describe a method which takes advantage of the "cognateness" of portions of a compound word with the corresponding individual words in the other language. We then extend the method to work even when there is no cognate relation, provided that a bilingual lexicon is available or can be statistially extracted from the parallel text.

Cognate Letters

a e	k cqn	Z SC
d tj	K CQN	ä ae
D TJ	p bf	Ä AE
i y	P BF	ö o
I Y	v f	Ö O
j y	V F	ü uo
J Y	z sc	Ü UO

Figure 1: Letter Correspondences from German to English

2 Method

In describing the method, we refer to a "compounding" language L_C and a "noncompounding" language L_N; for the experiments reported below, these are German and English.

To find the point at which to split a word in L_C, concatenate pairs of adjacent L_N words and measure the similarity between the concatenated pair and the (suspected) compound word. If the similarity is above a pre-selected threshold, examine the similarities of various substrings of the compound word with each of the L_N words to determine the point at which to split the compound. Each half may then be submitted recursively to the compound-splitting algorithm to attempt an overall split into more than two parts.

The similarity measure used is a form of Longest Common Substring (LCS). It allows for differing weights, making it what Tiedemann (1999) calls a *Highest Score of Correspondence*. Specifically, a character pair E_i,G_j can have any of three weights:

- full weight if the two characters are identical,
- a reduced weight (0.9) if they are not identical but are listed as related (see Figure 1), and
- an even lower weight (0.5) if the two characters are unrelated but one of them is identical to its predecessor and its predecessor is related to the other character.

The last case accounts for letter doubling in one language which is not present in the other.

The similarity score is computed using the standard dynamic programming approach as described in (Tiedemann 1999).

In addition to the similarity score, we use a coverage score, which is a modified version of the similarity score. Instead of dividing by the length of the longer of the two words, we divide by the length of the L_N (English) word. This provides an indication of the highest possible similarity score with any substring of the L_C (German) compound.

Once a candidate word pair corresponding to the compound has been found, the next task is to find the boundary between the portions of the compound representing each of the words in the pair. First, find the shortest prefix of the compound which maximizes the coverage score for the first word of the pair. Then, find the shortest suffix of the compound with maximizes the coverage score for the second word of the pair. The prefix and suffix define two boundaries, bnd_1 and bnd_2, which occur immediately following the final character of the prefix and immediately preceding the first character of the suffix, respectively. Note that the boundaries occur *between* letters.

The prefix is found by successively dropping the last letter from the (partial) compound until the coverage score decreases; bnd_1 is then set after the last character removed from the compound. Similarly, the suffix is found by successively dropping the first letter from the (partial) compound until the coverage score for the second word of the word pair decreases. bnd_2 is then set prior to the last character removed.

If bnd_1 and bnd_2 are the same, the compound has been split successfully, unless the boundary falls on a point which has been defined as an invalid split, e.g. between the "c" and "h" in "sch" for German. The software can optionally attempt to determine invalid boundaries itself by counting initial and final letter bigrams of words in the corpus, and disallowing the split if neither the two letters before the split nor the two after the split have ever been encountered in the appropriate position in any word of the corpus.

When bnd_1 lies before bnd_2, some letters are not accounted for as cognates of either L_N word. If a hyphen is the first or last letter in the gap between bnd_1 and bnd_2 (and the gap is not too large), set the boundary to the position just after the hyphen. In the absence of a hyphen, a small amount of knowledge about the compounding language is required to determine where in the gap between bnd_1 and bnd_2 to place the split point. For example, if the language inflects words by changing only word endings, we can assign the entire gap (if not too large) to the first word of the pair if bnd_2 is located just prior to a letter which is cognate with the first letter of the second word of the pair. Similarly for the case where the language inflects the beginnings of words. The program can also be given a list of substrings which it should avoid splitting; if one of these crosses bnd_2, the boundary is set to that string's beginning. Similarly, if one of the substrings crosses bnd_1, the boundary is set to the string's end.

Should bnd_1 fall *after* bnd_2, we need to determine how much of the overlap is due to a "missing" cognate letter which just happens to be present in the other half of the compound word. First, if the overlap region contains a hyphen at its beginning or end, set the boundary to just after that hyphen. Next, check whether one of the words in the L_N word pair exactly matches an initial or final string of the compound; if so, set the boundary accordingly. Finally, if enabled by the program's configuration, recompute bnd_1 and bnd_2 after removing the final letter of the first word and/or the first letter of the second word. If the two boundaries are the same, declare that the split point. If they leave a gap, set the split point to be at the beginning or end of the gap if the language inflects on the appropriate end of the word and the letter just outside the gap is cognate.

The extension of the method to handle non-cognates is quite straightforward. For each L_N word, we apply similarity scoring not just to the word itself, but also to each translation into L_C given by a bilingual lexicon. The translations are ordered by decreasing similarity score with the compound, and the original, untranslated word is tried after all translations have been found to be unsuccessful. We may allow both words to be translated, or limit the application of the lexicon to at most one word of each pair examined.

3 Examples

Some examples from the German-English corpus will now be used to illustrate the various cases described above. For each example, subscripted digits indicate the locations of bnd_1 and bnd_2.

Example 1: exact match

English	"abdominal angiography"
German	"Abdominal$_{12}$angiographie"
split	"Abdominal angiographie"

Example 2: gap, but contains hyphen

English	"2nd-line therapy"
German	"2nd-line$_1$-$_2$Therapie"
split	"2nd-line- Therapie"

Example 3: gap, resolved by assuming that inflectional morphology caused the gap

English	"Amniotic membrane"
German	"Amnio$_1$n$_2$membran"
split	"Amnion membran"

Example 4: overlap, but one word matches exactly

English	"heart transplantation"
German	"Herz$_2$t$_1$ransplantation"
split	"Herz transplantation"

Example 5: overlap, resolved by dropping letters

English	"hormone therapy"
German	"Hormon$_2$the$_1$rapie"

(recompute after dropping last letter of first word:)

English	"hormon therapy"
German	"Hormon$_{12}$therapie"
split	"Hormon therapie"

4 Experiments

The data for the experiments reported here consisted of 531,690 paired (English and German) journal article titles retrieved from the PubMed service (http://www.ncbi.-nlm.nih.gov/PubMed/). 8035 title pairs from articles appearing during the calendar year 2000 were held out for testing EBMT performance, and the remaining 523,655 pairs were used for training. For those runs involving a bilingual dictionary, the dictionary was extracted from the training portion of the title collection.

The first of the three experimental conditions to be investigated ("baseline") used only the raw text. Thus, compounds were split solely on the basis of similarity between pairs of English words and a German compound.

The second condition ("dictionary") used the corpus-derived dictionary to allow matches based on the similarity between the German translations of English words and German compounds as well as direct English-German similarity. Two variants of this approach were used – in the first ("dict-single"), only a single word was allowed to be translated, while in the second ("dict-full"), either or both words could be translated prior to measuring similarity.

The third experimental condition ("feedback") first ran the "dictionary" condition to generate an initial decomposition of the text. This decomposed version of the training text was then used to train a new dictionary, which in turn was used to generate the final decomposition. As for "dictionary", the runs could be restricted to translating at most one word of the pair or allowed to translate both words prior to measuring similarity.

For all experiments, the parameters were set as follows:

- minimum length = 6 characters
- similarity threshold = 0.6
- maximum prefix = 0
- maximum suffix = 2
- forbidden boundaries = "chr e", "sc h"
- deprecated boundaries = "sch", "handlung", "sicht", "haltung", "risch"
- known prefixes = "ge", "be", "zu", "k", "ver"
- known suffixes = "gs"

EBMT performance was measured using a test set consisting of 2000 sentence pairs from the held-out year-2000 PubMed data. The system was trained on various decompounded corpora and a bilingual lexicon derived from the same corpus. For translating from German into English, the German halves of each sentence pair (19704 tokens prior to decompounding) were used as input; for translating from English into German, the English halves (23386 tokens) were used.

5 Results

Figures 2 and 3 show the performance of the different variations of the decompounder described in Section 4. For each variation, the number of distinct compounds ("types") which were split are shown, along with the total number of instances of those words in the corpus ("tokens"). The error rate was estimated by manually evaluating a uniform sample of 500 types, counting as an error all compounds for which *any* hypothesized split is incorrect (either extraneous or in the wrong location). This is a strict scoring, resulting in a 100% error rate if the program hypothesizes all possible splits; the error

Run	Types	Tokens	Errors/500	Error Rate
Baseline	66,960	383,120	5	1.0%
Dict-Single	100,540	415,521	23	4.6%
Dict-Full	128,847	665,231	37	7.4%
Feedback-S-S	109,559	482,604	34	6.8%
Feedback-S-F	143,151	828,147	37	7.4%
Feedback-F-S	116,306	644,224	33	6.6%
Feedback-F-F	150,726	943,290	59	11.8%

Figure 2: Performance of the Decompounder

Run	Types	Tokens	Errors/500	Error Rate
Baseline	42,142	188,033	4	0.8%
Dict-Single	58,775	209,735	16	3.2%
Dict-Full	73,531	341,801	34	6.8%
Feedback-S-S	63,274	253,623	24	4.8%
Feedback-S-F	81,203	457,119	45	9.0%
Feedback-F-S	66,369	320,471	34	6.8%
Feedback-F-F	84,961	529,782	58	11.6%

Figure 3: Performance of the Decompounder (with corpus-restricted splitting)

rate as a percentage of hypothesized splits is lower than the error rate shown. The only difference in parameters between corresponding entries in the two figures is that Figure 3 used initial/final-bigram statistics to restrict possible decompositions.

Figures 4 and 5 show how the performance of the baseline system varies with differing thresholds on the similarity score. Again, the two figures differ in whether bigram statistics were used to restrict decompositions. The recall rates were estimated by counting the number of terms in a 1000-element sample of compounds for which splits were proposed; this provides an upper bound, since the program will be credited for a term even if it finds only one of multiple split points. The sampling of compounds was created by taking 2000 uniformly distributed terms from the alphabetically sorted German vocabulary, and manually extracting the first 1000 compounds (approximately 70% of all types are compounds, and even those which can translate as a single word in English were retained). Figure 6 shows the recall-error tradeoff graphically.

Full analysis of the decomposition results is still pending, but some preliminary observations will be proffered. Although the corpus-restricted decomposition does indeed reduce the error rate at any given threshold, the yield is reduced to the point where the same yield can be achieved with a lower error rate by using unrestricted splitting with a higher threshold. A large proportion of the errors in the "feedback" variants are due to the splitting-off of inflectional morphemes such as "s", "ische", and "ischer", which may be due to matching the base form of a word as one half of the supposed compound and an unrelated English word against the inflectional morpheme as the other half.

The final experiment was to use the decompounded corpus and a bilingual lexicon

Threshold	Types	Tokens	Error Rate	Est.Recall
1.0	1,678	12,413	0.6%*	0.5%
0.9	13,361	51,710	0.4%	4.3%
0.8	36,677	151,757	0.4%	11.3%
0.7	51,902	242,913	0.8%	16.9%
0.6	66,960	383,120	1.0%	22.2%
0.5	84,027	815,942	4.2%	27.5%
0.4	104,801	1,135,402	17.4%	34.4%

*All three errors in the sample of 500 were due to erroneous embedded spaces in the word in the English version of the title.

Figure 4: Effects of Varying Thresholds

Threshold	Types	Tokens	Error Rate	Est.Recall
1.0	672	5,440	0.2%*	0.1%
0.9	8,724	29,321	0.2%	3.1%
0.8	23,151	76,475	0.6%	6.8%
0.7	32,780	139,427	0.4%	10.4%
0.6	42,142	188,033	0.8%	13.8%
0.5	52,458	428,756	1.6%	17.3%
0.4	63,580	715,895	9.4%	21.5%

*The lone error in the sample of 500 was due to an erroneous embedded space in the word in the English version of the title.

Figure 5: Effects of Varying Thresholds (with corpus-restricted splitting)

statistically extracted from that corpus to train an example-based machine translation system. Figure 7 shows what percentage of words from the 2000-sentence test input produces phrasal (two words or longer) matches against the corpus, what percentage of the input generates translations after successful subsentential alignment, and the average length of a successfully-aligned match. For German input, the percentage of matches against the corpus varies because the input text (both training and test) changes due to decompounding. As a result of the increased number of tokens after decompounding, the coverage values are not directly comparable, so that the actual performance improvement is smaller than it would appear from Figure 7.

6 Conclusions

Corpus-directed decomposition of compound words can yield a substantial fraction of all possible decompositions with a low error rate, which is useful for a variety of applications. While the additions to the basic method increase the number of words that are decomposed, the error rate quickly becomes excessively high.

Use of a decompounded corpus with an example-based machine translation system provides a small but definite improvement in performance. The percentage of input words for which corpus matches exist but no good alignment from English to German

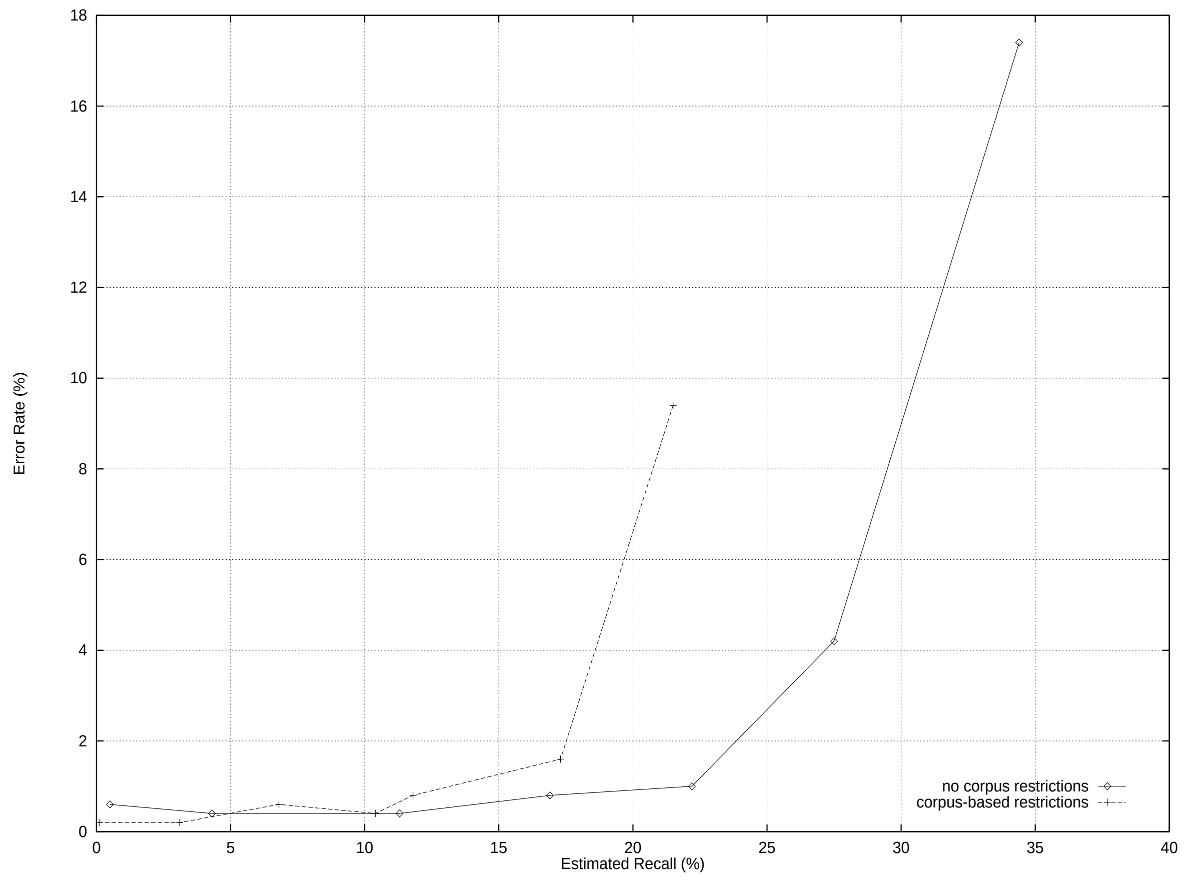

Figure 6: Recall vs. Error Rate

	Corpus Matches	Coverage (words)	Avg Length (words)
Baseline Eng->Ger	90.34%	83.18%	2.985
Decompounded (base)	90.34%	84.05%	2.985
Decompounded (dict-S)	90.34%	84.91%	2.989
Decompounded (feed-S-S)	90.34%	84.93%	2.983
Baseline Ger->Eng	78.17%	71.31%	2.876
Decompounded (base)	80.81%*	74.40%	2.992
Decompounded (dict-S)	81.21%*	75.14%	2.943
Decompounded (feed-S-S)	81.74%*	75.61%	2.963

* see text

Figure 7: EBMT Performance

can be found is reduced by as much as 24.4%.

The decompounding process generates, as a by-product, a mapping from the compounds to phrases in the other language. This could be used as a bilingual phrasal lexicon, but in the current implementation the phrase will be an incomplete translation whenever a split point is missed.

7 Future Work

Many enhancements to the work described here are possible.

The set of letters considered cognate to each letter in the one language is currently specified in a data file; these could be induced automatically from the training corpus. An advantage of automatic induction is that the weight of each possible cognate letter can be set according to the relative frequency of occurrence. Separate weights are supported by the existing code, but have not been used to date.

Similarly, positions where it is not acceptable to split a word are specified manually or using letter bigrams from the corpus; the corpus proved to be too small for the latter to be effective. A better approach may be to prefer split points which generate words that are present in the corpus (including words generated by unambiguous decompositions). This would be an extension of the heuristic of Example 4 in Section 3.

The current system supports one specific case of two-to-one mappings – doubled letters. A more general many-one or even many-many mapping would be able to deal with cases such as a German "dsch" corresponding to an English "j" or English "sh" corresponding to German "sch".

Many of the generated errors are due to cognate-matching against words which are not actually translations. To reduce this source of error, the lexicon should be used (when available) to restrict possible L_N words to check against suspected compound words.

It should also be possible to improve performance by using two passes. In the first pass, use a moderately strict threshold to find constituents with a high degree of confidence. On the second pass, use the constituents found during the first pass to guide the decompounding when using a lower threshold for greater recall.

References

Adda, Gilles, Martine Adda-Decker, Jean-Luc Gauvain & Lori Lamel: 1997, 'Text Normalization and Speech Recognition in French', in *Proceedings of Eurospeech '97*, Rhodes, Greece, pp. 2711–2714.

Adda-Decker, Martine, Gilles Adda & Lori Lamel: 2000, 'Investigating Text Normalization and Pronunciation Variants for German Broadcast Transcription', in *Proceedings of ICSLP-2000*, `ftp://tlp.limsi.fr/public/icslp00_h4ger.ps.Z`.

Braschler, Martin & Peter Schäuble: 2000, 'Experiments with the Eurospider Retrieval System for CLEF 2000', `http://www.iei.pi.cnr.it/DELOS/CLEF/braschler.pdf`.

Brown, Peter, J. Cocke, S. Della Pietra, V. Della Pietra, F. Jelinek, J. Lafferty, R. Mercer & P. Roossin: 1990, 'A Statistical Approach to Machine Translation', *Computational Linguistics*, **16**: 79–85.

Brown, Ralf D.: 1997, 'Automated Dictionary Extraction for "Knowledge-Free" Example-Based Translation', in *Proceedings of the Seventh International Conference on Theoretical and Methodological Issues in Machine Translation (TMI-97)*, Santa Fe, New Mexico, pp. 111–118, `http://www.cs.cmu.edu/~ralf/papers.html`.

Brown, Ralf D.: 1999, 'Adding Linguistic Knowledge to a Lexical Example-Based Translation System', in *Proceedings of the Eighth International Conference on Theoretical and*

Methodological Issues in Machine Translation (TMI-99), Chester, England, pp. 22–32, `http://www.cs.cmu.edu/~ralf/papers.html`.

Melamed, I. Dan: 1997, 'A Word-to-Word Model of Translational Equivalence', in *35th Annual Meeting of the Association for Computational Linguistics (ACL'97)*, pp. 490–497.

Tiedemann, Jörg: 1999, 'Automatic Construction of Weighted String Similarity Measures', in *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*.