Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL), Philadelphia, July 2002, pp. 409-416.

Using Similarity Scoring To Improve the Bilingual Dictionary for Word Alignment

Katharina Probst
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA, USA, 15213
kathrin@cs.cmu.edu

Ralf Brown
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA, USA, 15213
ralf@cs.cmu.edu

Abstract

We describe an approach to improve the bilingual cooccurrence dictionary that is used for word alignment, and evaluate the improved dictionary using a version of the Competitive Linking algorithm. We demonstrate a problem faced by the Competitive Linking algorithm and present an approach to ameliorate it. In particular, we rebuild the bilingual dictionary by clustering similar words in a language and assigning them a higher cooccurrence score with a given word in the other language than each single word would have otherwise. Experimental results show a significant improvement in precision and recall for word alignment when the improved dicitonary is used.

1 Introduction and Related Work

Word alignment is a well-studied problem in Natural Language Computing. This is hardly surprising given its significance in many applications: word-aligned data is crucial for example-based machine translation, statistical machine translation, but also other applications such as cross-lingual information retrieval. Since it is a hard and time-consuming task to hand-align bilingual data, the automation of this task receives a fair amount of attention. In this paper, we present an approach to improve the bilingual dictionary that is used by word alignment algorithms. Our method is based on similarity scores between words, which in effect results in the clustering of morphological variants.

One line of related work is research in clustering based on word similarities. This problem is an area of active research in the Information Retrieval community. For instance, Xu and Croft (1998) present an algorithm that first clusters what are assumedly variants of the same word, then further refines the clusters using a cooccurrence related measure. Word variants are found via a stemmer or by clustering all words that begin with the same three letters. Another technique uses similarity scores based on N-grams (e.g. (Kosinov, 2001)). The similarity of two words is measured using the number of N-grams that their occurrences have in common. As in our approach, similar words are then clustered into equivalence classes.

Other related work falls in the category of word alignment, where much research has been done. A number of algorithms have been proposed and evaluated for the task. As Melamed (2000) points out, most of these algorithms are based on word cooccurrences in sentence-aligned bilingual data. A source language word s_i and a target language word t_i are said to cooccur if s_i occurs in a source language sentence and t_i occurs in the corresponding target language sentence. Cooccurrence scores then are then counts for all word pairs s_j and t_k, where s_j is in the source language vocabulary and t_k is in the target language vocabulary. Often, the scores also take into account the marginal probabilites of each word and sometimes also the conditional probabilities of one word given the other.

Aside from the classic statistical approach of

(Brown et al., 1990; Brown et al., 1993), a number of other algorithms have been developed. Ahrenberg et al. (1998) use morphological information on both the source and the target languages. This information serves to build equivalence classes of words based on suffices. A different approach was proposed by Gaussier (1998). This approach models word alignments as flow networks. Determining the word alignments then amounts to solving the network, for which there are known algorithms. Brown (1998) describes an algorithm that starts with 'anchors', words that are unambiguous translations of each other. From these anchors, alignments are expanded in both directions, so that entire segments can be aligned.

The algorithm that this work was based on is the Competitive Linking algorithm. We used it to test our improved dictionary. Competitive Linking was described by Melamed (1997; 1998; 2000). It computes all possible word alignments in parallel data, and ranks them by their cooccurrence or by a similar score. Then links between words (i.e. alignments) are chosen from the top of the list until no more links can be assigned. There is a limit on the number of links a word can have. In its basic form the Competitive Linking algorithm (Melamed, 1997) allows for only up to one link per word. However, this one-to-one/zero-to-one assumption is relaxed by redefining the notion of a word.

2 Competitive Linking in our work

We implemented the basic Competitive Linking algorithm as described above. For each pair of parallel sentences, we construct a ranked list of possible links: each word in the source language is paired with each word in the target language. Then for each word pair the score is looked up in the dictionary, and the pairs are ranked from highest to lowest score. If a word pair does not appear in the dictionary, it is not ranked. The algorithm then recursively links the word pair with the highest cooccurrence, then the next one, etc. In our implementation, linking is performed on a sentence basis, i.e. the list of possible links is constructed only for one sentence pair at a time.

Our version allows for more than one link per word, i.e. we do not assume one-to-one or zero-to-one alignments between words. Furthermore, our implementation contains a threshold that specifies how high the cooccurrence score must be for the two words in order for this pair to be considered for a link.

3 The baseline dictionary

In our experiments, we used a baseline dictionary, rebuilt the dictionary with our approach, and compared the performance of the alignment algorithm between the baseline and the rebuilt dictionary. The dictionary that was used as a baseline and as a basis for rebuilding is derived from bilingual sentence-aligned text using a count-and-filter algorithm:

- **Count**: for each source word type, count the number of times each target word type cooccurs in the same sentence pair, as well as the total number of occurrences of each source and target type.

- **Filter**: after counting all cooccurrences, retain only those word pairs whose cooccurrence probability is above a defined threshold. To be retained, a word pair W_s,W_t must satisfy

  $$min(P(W_s|W_t), P(W_t|W_s)) \geq thr_{C(W_s,W_t)}$$

  where $C(W_s, W_t)$ is the number of times the two words cooccurred.

By making the threshold vary with frequency, one can control the tendency for infrequent words to be included in the dictionary as a result of chance collocations. The 50% cooccurrence probability of a pair of words with frequency 2 and a single co-occurrence is probably due to chance, while a 10% cooccurrence probability of words with frequency 5000 is most likely the result of the two words being translations of each other. In our experiments, we varied the threshold from 0.005 to 0.01 and 0.02.

It should be noted that there are many possible algorithms that could be used to derive the baseline dictionary, e.g. χ^2, pointwise mutual information, etc. An overview of such approaches can be found in (Kilgarriff, 1996). In our work, we preferred to use the above-described method, because it this method is utilized in the example-based MT system being developed in our group (Brown, 1997). It has proven useful in this context.

4 The problem of derivational and inflectional morphology

As the scores in the dictionary are based on surface form words, statistical alignment algorithms such as Competitive Linking face the problem of inflected and derived terms. For instance, the English word *liberty* can be translated into French as a noun (*liberté*), or else as an adjective (*libre*), the same adjective in the plural (*libres*), etc. This happens quite frequently, as sentences are often restructured in translation. In such a case, *liberté*, *libre*, *libres*, and all the other translations of *liberty* in a sense share their cooccurrence scores with *liberty*. This can cause problems especially because there are words that are overall frequent in one language (here, French), and that receive a high cooccurrence count regardless of the word in the other language (here, English). If the cooccurrence score between *liberty* and an unrelated but frequent word is higher than *libres*, then the algorithm will prefer a link between *liberty* and *le* over a link between *liberty* and *libres*, even if the latter is correct.

As for a concrete example from the training data used in this study, consider the English word *oil*. This word is quite frequent in the training data and thus cooccurs at high counts with many target language words [1]. In this case, the target language is French. The cooccurrence dictionary contains the following entries for *oil* among other entries:

oil - *et* 543

oil - *dans* 118

. . .

oil - *pétrole* 259

oil - *pétrolière* 61

oil - *pétrolières* 61

It can be seen that words such as *et* and *dans* receive higher coccurrence scores with *oil* than some correct translations of *oil*, such as *pétrolière*, and *pétrolières*, and, in the case of *et*, also *pétrole*. This will cause the Competitive Linking algorithm to favor a link e.g. between *oil* and *et* over a link between *oil* and *pétrole*.

In particular, word variations can be due to inflectional morphology (e.g. adjective endings) and derivational morphology (e.g. a noun being translated as an adjective due to sentence restructuring). Both inflectional and derivational morphology will result in words that are similar, but not identical, so that cooccurrence counts will score them separately. Below we describe an approach that addresses these two problems. In principle, we cluster *similar* words and assign them a new dictionary score that is higher than the scores of the individual words. In this way, the dictionary is rebuilt. This will influence the ranked list that is produced by the algorithm and thus the final alignments.

[1]We used Hansards data, see the evaluation section for details.

5 Rebuilding the dictionary based on similarity scores

Rebuilding the dictionary is based largely on similarities between words. We have implemented an algorithm that assigns a similarity score to a pair of words $t_i :: t_j$. The score is higher for a pair of similar words, while it favors neither shorter nor longer words. The algorithm finds the number of matching characters between the words, while allowing for insertions, deletions, and substitutions. The concept is thus very closely related to the Edit distance, with the difference that our algorithm counts the matching characters rather than the non-matching ones. The length of the matching substring (which is not necessarily continguous) is denoted by *MatchStringLength*). At each step, a character from t_i is compared to a character from t_j. If the characters are identical, the count for the *MatchStringLength* is incremented. Then the algorithm checks for reduplication of the character in one or both of the words. Reduplication also results in an incremented *MatchStringLength*. If the characters do not match, the algorithm skips one or more characters in either word.

Then the longest common substring is put in relation to the length of the two words. This is done so as to not favor longer words that would result in a higher *MatchStringLength* than shorter words. The similarity score of t_i and t_j is then computed using the following formula:

$$\frac{MatchStringLength}{\frac{1}{2}\left(length_{t_i}+length_{t_j}\right)}$$

This similarity scoring provides the basis for our newly built dictionary. The algorithm proceeds as follows: For any given source language word s_i, there are n target language words $t_1 \ldots t_n$ such that the cooccurrence score $cooc(s_i,t_j)$ is greater than 0.

Note that in most cases n is much smaller than the size of the target language vocabulary, but also much greater than 0. For the words $t_1 \dots t_n$, the algorithm computes the similarity score for each word pair (t_j, t_k), where $1 \leq j, k \leq n, j \neq k$. Note that this computation is potentially very complex. The number of word pairs grows exponentially as n grows. This problem is addressed by excluding word pairs whose cooccurrence scores are low, as will be discussed in more detail later.

In the following, we use a greedy bottom-up clustering algorithm (Manning and Schütze, 1999) to cluster those words that have high similarity scores. The clustering algorithm is initialized to n clusters, where each cluster contains exactly one of the words $t_1 \dots t_n$. In the first step, the algorithm clusters the pair of words with the maximum similarity score. The new cluster also stores a similarity score $SimAve(merged)$, which in this case is the similarity score of the two clustered words. In the following steps, the algorithm again merges those two clusters that have the highest similarity score $SimAve(merged)$. The clustering can occur in one of three ways:

1. Merge two clusters that each contain one word. Then the similarity score $SimAve_{merged}$ of the merged cluster will be the similarity score of the word pair.

2. Merge a cluster c_i that contains a single word t_i and a cluster c_j that contains k words $t_1 \dots t_k$ and has $SimAve(merged_{c_i,c_j})$. Then the similarity score of the merged cluster is the average similarity score of the k-word cluster, averaged with the similarity scores between the single word and all k words in the cluster. This means that the algorithm computes the similarity score between the single word t_i in cluster c_i and each of the k words in cluster c_j, and averages them with $SimAve(c_j)$:

$$\frac{\left(\sum_{j=1}^{k} sim(t_i,t_j)\right)+\left(\binom{k}{2} SimAve(c_j)\right)}{k+\binom{k}{2}}$$

3. Merge two clusters that each contain more than a single word. In this case, the algorithm proceeds as in the second case, but averages the added similarity score over all word pairs. Suppose there exists a cluster c_i with l words $t_1 \dots t_l$ and $SimAve(c_i)$ and a cluster c_j with k words $t_1 \dots t_k$ and $SimAve(c_j)$. Then $SimAve(merged_{c_i,c_j})$ is computed as follows:

$$\frac{\left(\sum_{i=1}^{l}\sum_{j=1}^{k} sim(t_i,t_j)\right)+\left(\binom{k}{2} SimAve(c_j)\right)+\left(\binom{l}{2} SimAve(c_i)\right)}{(k*l)+\binom{k}{2}+\binom{l}{2}}$$

Clustering proceeds until a threshold, $minsim$, is exhausted. If none of the possible merges would result in a new cluster whose average similarity score $SimAve(merged)$ would be at least $minsim$, clustering stops. Then the dictionary entries are modified as follows: suppose that words $t_k \dots t_m$ are clustered, where all words $t_k \dots t_m$ cooccur with source language word s_i. Furthermore, denote the cooccurrence score of the word pair s_i and t_k by $cooc(s_i, t_k)$. Then in the rebuilt dictionary the entry

$$s_i, t_j : cooc(s_i, t_j)$$

will be replaced with

$$s_i, t_j : \sum_{l=k}^{m} cooc\,(s_i, t_l) \text{ if } t_j \in t_k \dots t_m$$

Not all words are considered for clustering. First, we compiled a stop list of target language words that are never clustered, regardless of their similarity and cooccurrence scores with other words. The words on the stop list are the 20 most frequent words in the target language training data. Section 4 argues why this exclusion makes sense: one of the goals of clustering is to enable variations of a word to receive a higher dictionary score than words that are very common overall.

Furthermore, we have decided to exclude words from clustering that account for only few of the cooccurrences of s_i. In particular, a separate threshold, $coocsratio$, controls how high the cooccurrence score with s_i has to be in relation to all other scores between s_i and a target language word. $coocsratio$ is expressed as follows: a word t_j qualifies for clustering if

$$\frac{cooc(s_i,t_j)}{\sum_{k=1}^{n} cooc(s_i,t_k)} > coocsratio$$

As before, $t_1 \dots t_n$ are all the target language words that cooccur with source language word s_i.

Similarly to the most frequent words, dictionary scores for word pairs that are too rare for clustering remain unchanged.

This exclusion makes sense because words that cooccur infrequently are likely not translations of each other, so it is undesirable to boost their score by clustering. Furthermore, this threshold helps keep the complexity of the operation under control. The fewer words qualify for clustering, the fewer similarity scores for pairs of words have to be computed.

6 Evaluation

We trained three basic dictionaries using part of the Hansard data, around five megabytes of data (around 20k sentence pairs and 850k words). The basic dictionaries were built using the algorithm described in section 3, with three different thresholds: 0.005, 0.01, and 0.02. In the following, we will refer to these dictionaries as as Dict0.005, Dict0.01, and Dict0.02.

50 sentences were held back for testing. These sentences were hand-aligned by a fluent speaker of French. No one-to-one assumption was enforced. A word could thus align to zero or more words, where no upper limit was enforced (although there is a natural upper limit).

The Competitive Linking algorithm was then run with multiple parameter settings. In one setting, we varied the maximum number of links allowed per word, $maxlinks$. For example, if the maximum number is 2, then a word can align to 0, 1, or 2 words in the parallel sentence. In other settings, we enforced a minimum score in the bilingual dictionary for a link to be accepted, $minscore$. This means that two words cannot be aligned if their score is below $minscore$. In the rebuilt dictionaries, $minscore$ is applied in the same way.

The dictionary was also rebuilt using a number of different parameter settings. The two parameters that can be varied when rebuilding the dictionary are the similarity threshold $minsim$ and the cooccurrence threshold $coocsratio$. $minsim$ enforces that all words within one cluster must have an average similarity score of at least $minsim$. The second threshold, $coocsratio$, enforces that only certain words are considered for clustering. Those words that are considered for clustering should account for more than $100 * coocsratio\%$ of the cooccurrences of the source language word with any target language word. If a word falls below threshold $coocsratio$, its entry in the dictionary remains unchanged, and it is not clustered with any other word. Below we summarize the values each parameter was set to.

- *maxlinks* Used in Competitive Linking algorithm: Maximum number of words any word can be aligned with. Set to: 1, 2, 3.

- *minscore* Used in Competitive Linking algorithm: Minimum score of a word pair in the dictionary to be considered as a possible link. Set to: 1, 2, 4, 6, 8, 10, 20, 30, 40, 50.

- *minsim* Used in rebuilding dictionary: Minimum average similarity score of the words in a cluster. Set to: 0.6, 0.7, 0.8.

- *coocsratio* Used in rebuilding dictionary: $100 * coocsratio$ is the minimum percentage of all cooccurrences of a source language word with any target language word that are accounted for by one target language word. Set to: 0.003.

Thus varying the parameters, we have constructed various dictionaries by rebuilding the three baseline dictionaries. Here, we report on results on three dictionaries where *minsim* was set to 0.7 and *coocsratio* was set to 0.003. For these parameter settings, we observed robust results, although other parameter settings also yielded positive results.

Precision and recall was measured using the hand-aligned 50 sentences. Precision was defined as the percentage of links that were *correctly* proposed by our algorithm out of all links that were proposed. Recall is defined as the percentage of links that were found by our algorithm out of all links that should have been found. In both cases, the hand-aligned data was used as a gold standard. The F-measure combines precision and recall: $f\text{-}measure = \frac{2*precision*recall}{precision+recall}$.

The following figures and tables illustrate that the Competitive Linking algorithm performs favorably when a rebuilt dictionary is used. Table 1 lists the improvement in precision and recall for each of the dictionaries. The table shows the values when the *minscore* score is set to 50, and up to 1 link was allowed per word. Furthermore, the p-values of a 1-tailed t-test are listed, indicating these performance boosts are in mostly highly statistically significant

	Dict0.005	Dict0.01	Dict0.02
P Improvement	0.060	0.067	0.057
P p-value	0.0003	0.0042	0.0126
R Improvement	0.094	0.11	0.087
R p-value	0.0026	0.0008	0.0037

Table 1: Percent improvement and p-value for recall and precision, comparing baseline and rebuilt dictionaries at *minscore* 50 and *maxlinks* 1.

for these parameter settings, where some of the best results were observed.

The following figures (figures 1-9) serve to illustrate the impact of the algorithm in greater detail. All figures plot the precision, recall, and f-measure performance against different *minscore* settings, comparing rebuilt dictionaries to their baselines. For each dictionary, three plots are given, one for each *maxlinks* setting, i.e. the maximum number of links allowed per word. The curve names indicate the type of the curve (Precision, Recall, or F-measure), the maximum number of links allowed per word (1, 2, or 3), the dictionary used (Dict0.005, Dict0.01, or Dict0.02), and whether the run used the baseline dictionary or the rebuilt dictionary (Baseline or Cog7.3).

It can be seen that our algorithm leads to stable improvement across parameter settings. In few cases, it drops below the baseline when *minscore* is low. Overall, however, our algorithm is robust - it improves alignment regardless of how many links are allowed per word, what baseline dictionary is used, and boosts both precision and recall, and thus also the f-measure.

To return briefly to the example cited in section 3, we can now show how the dictionary rebuild has affected these entries. In dictionary $Sim.7.3.$ they now look as follows:

oil - *et*	262
oil - *dans*	118
...	
oil - *pétrole*	434
oil - *pétrolière*	434
oil - *pétrolières*	434

The fact that *pétrole*, *pétrolière*, and *pétrolières* now receive higher scores than *et* and *dans* is what causes the alignment performance to increase.

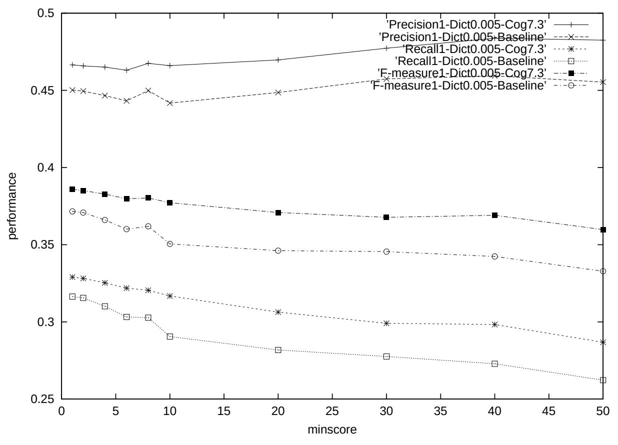

Figure 1: Performance of dictionaries Dict0.005 for up to one link per word

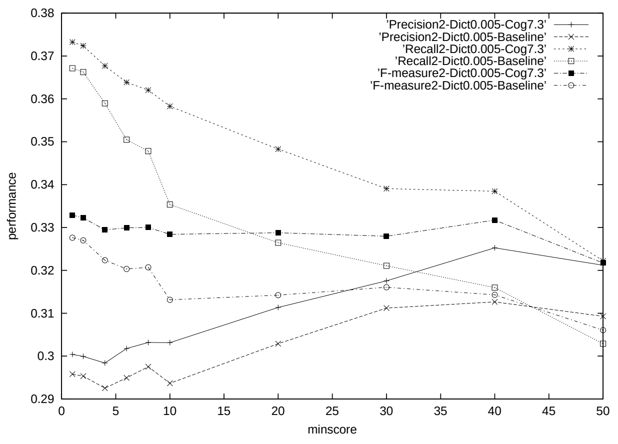

Figure 2: Performance of dictionaries Dict0.005 for up to two links per word

7 Conclusions and Future Work

We have demonstrated how rebuilding a dictionary can improve the performance (both precision and recall) of a word alignment algorithm. The algorithm proved robust across baseline dictionaries and various different parameter settings. Although a small test set was used, the improvements are statistically significant for various parameter settings. We have shown that computing similarity scores of pairs of words can be used to cluster morphological variants of words in an inflected language such as French.

It will be interesting to see how the similarity and clustering method will work in conjunction with other word alignment algorithms, as the dictionary

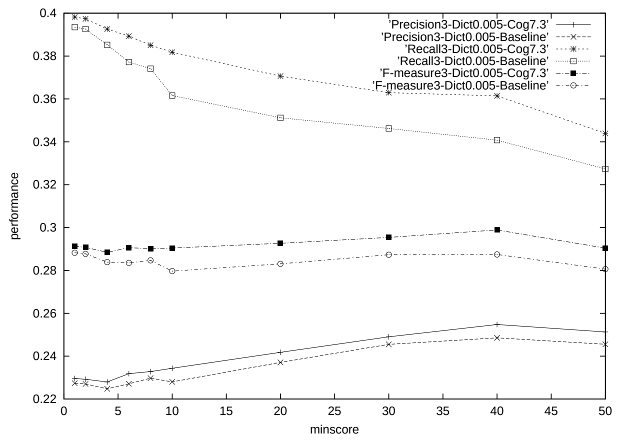

Figure 3: Performance of dictionaries Dict0.005 for up to three links per word

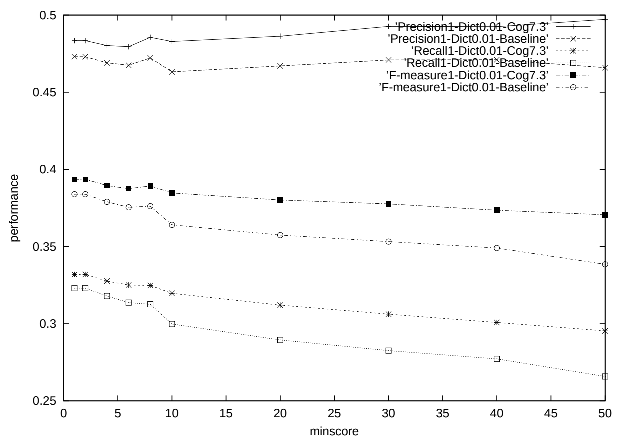

Figure 4: Performance of dictionaries Dict0.01 for up to one link per word

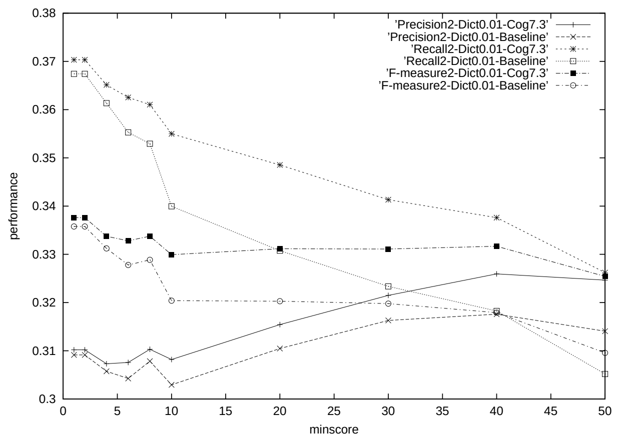

Figure 5: Performance of dictionaries Dict0.01 for up to two links per word

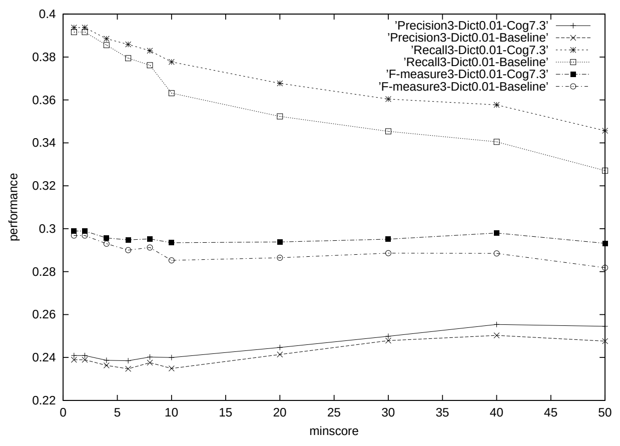

Figure 6: Performance of dictionaries Dict0.01 for up to three links per word

rebuilding algorithm is independent of the actual word alignment method used.

Furthermore, we plan to explore ways to improve the similarity scoring algorithm. For instance, we can assign lower match scores when the characters are not identical, but members of the same equivalence class. The equivalence classes will depend on the target language at hand. For instance, in German, *a* and *ä* will be assigned to the same equivalence class, because some inflections cause *a* to become *ä*. An improved similarity scoring algorithm may in turn result in improved word alignments.

In general, we hope to move automated dictionary extraction away from pure surface form statistics and toward dictionaries that are more linguistically motivated.

References

Lars Ahrenberg, M. Andersson, and M. Merkel. 1998. A simple hybrid aligner for generating lexical correspondences in parallel texts. In *Proceedings of COLING-ACL'98*.

Peter Brown, J. Cocke, V.D. Pietra, S.D. Pietra, J. Jelinek, J. Lafferty, R. Mercer, and P. Roossina. 1990. A statistical approach to Machine Translation. *Computational Linguistics*, 16(2):79–85.

Peter Brown, S.D. Pietra, V.D. Pietra, and R. Mercer. 1993. The mathematics of statistical Machine Translation: Parameter estimation. *Computational Linguistics*.

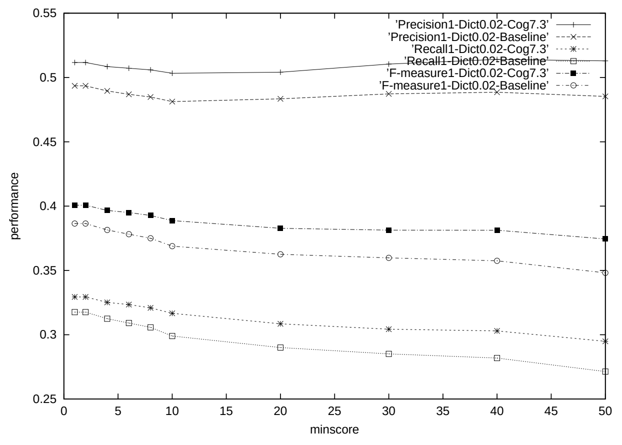

Figure 7: Performance of dictionaries Dict0.02 for up to one link per word

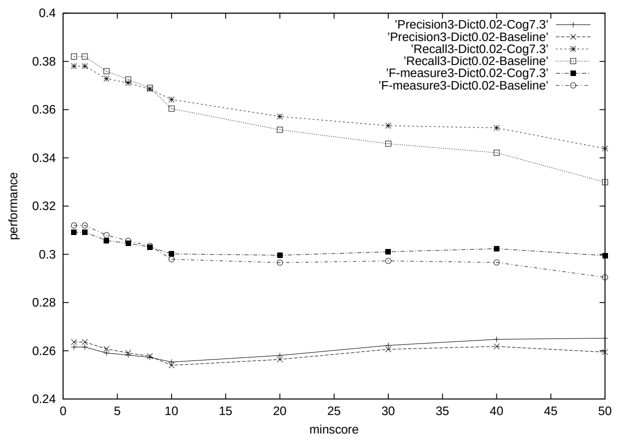

Figure 9: Performance of dictionaries Dict0.02 for up to three links per word

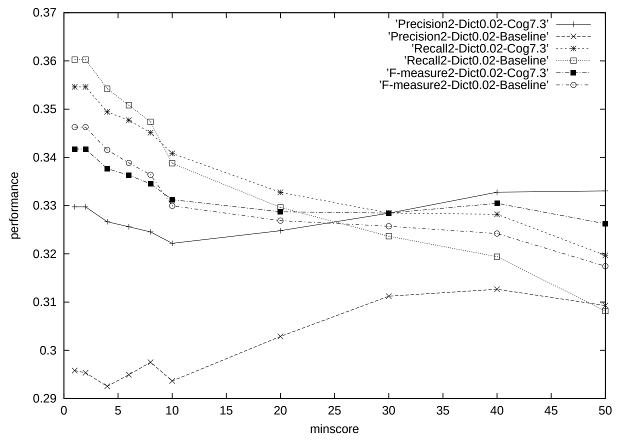

Figure 8: Performance of dictionaries Dict0.02 for up to two links per word

Ralf Brown. 1997. Automated dictionary extraction for 'knowledge-free' example-based translation. In *Proceedings of TMI 1997*, pages 111–118.

Ralf Brown. 1998. Automatically-extracted thesauri for cross-language IR: When better is worse. In *Proceedings of COMPUTERM'98*.

Eric Gaussier. 1998. Flow network models for word alignment and terminology extraction from bilingual corpora. In *Proceedings of COLING-ACL'98*.

Adam Kilgarriff. 1996. Which words are particularly characteristic of a text? A survey of statistical approaches. In *Proceedings of AISB Workshop on Language Engineering for Document Analysis and Recognition*.

Serhiy Kosinov. 2001. Evaluation of N-grams conflation approach in text-based Information Retrieval. In *Proceedings of International Workshop on Information Retrieval IR'01*.

Christopher D. Manning and Hinrich Schütze, 1999. *Foundations of Statistical Natural Language Processing*, chapter 14. MIT Press.

Dan I. Melamed. 1997. A word-to-word model of translation equivalence. In *Proceedings of ACL'97*.

Dan I. Melamed. 1998. Empirical methods for MT lexicon development. In *Proceedings of AMTA'98*.

Dan I. Melamed. 2000. Models of translational equivalence among words. *Computational Linguistics*, 26(2):221–249.

Jinxi Xu and W. Bruce Croft. 1998. Corpus-based stemming using co-occurrence of word variants. *ACM Transactions on Information Systems*, 16(1):61–81.