Phrase Clustering for Smoothing TM Probabilities – or, How to Extract Paraphrases from Phrase Tables

[1]Roland Kuhn, [1]Boxing Chen, [1]George Foster and [2]Evan Stratford
[1]National Research Council of Canada, Gatineau, Québec, Canada
[2]University of Waterloo, Waterloo, Ontario, Canada
[1]First.Last@nrc.gc.ca; [2]evan.stratford@gmail.com

Abstract

This paper describes how to cluster together the phrases of a phrase-based statistical machine translation (SMT) system, using information in the phrase table itself. The clustering is symmetric and recursive: it is applied both to source-language and target-language phrases, and the clustering in one language helps determine the clustering in the other. The phrase clusters have many possible uses. This paper looks at one of these uses: smoothing the conditional translation model (TM) probabilities employed by the SMT system. We incorporated phrase-cluster-derived probability estimates into a baseline loglinear feature combination that included relative frequency and lexically-weighted conditional probability estimates. In Chinese-English and French-English learning curve experiments, we obtained a gain over the baseline in 29 of 30 tests, with a maximum gain of 0.55 BLEU points (though most gains were fairly small). The largest gains came with medium (200-400K sentence pairs) rather than with small (less than 100K sentence pairs) amounts of training data, contrary to what one would expect from the paraphrasing literature. We have only begun to explore the original smoothing approach described here.

1 Introduction and Related Work

The source-language and target-language "phrases" employed by many statistical machine translation (SMT) systems are anomalous: they are arbitrary sequences of contiguous words extracted by complex heuristics from a bilingual corpus, satisfying no formal linguistic criteria. Nevertheless, phrase-based systems perform better than word-based systems (Koehn 2010, pp. 127-129). In this paper, we look at what happens when we cluster together these anomalous but useful entities.

The work on phrase clusters described here is itself somewhat hard to cluster with related work. It has some affinity with work on paraphrases for SMT. Key papers in this area include (Bannard and Callison-Burch, 2005), which pioneered the extraction of paraphrases from bilingual parallel corpora, (Callison-Burch *et al.*, 2006) which showed that paraphrase generation could improve SMT performance, (Callison-Burch, 2008) and (Zhao *et al.*, 2008) which showed how to improve the quality of paraphrases, and (Marton *et al.*, 2009) which derived paraphrases from monolingual data using distributional information. Paraphrases typically help SMT systems trained on under 100K sentence pairs the most.

The phrase clustering algorithm in this paper outputs groups of source-language and target-language phrases with similar meanings: paraphrases. However, previous work on paraphrases for SMT has aimed at finding translations for source-language phrases in the system's input that weren't seen during system training. Our approach is completely useless in this situation: it only generates new information for target or source phrases that are already in the system's phrase table. Thus, we find paraphrases for many of the source and target phrases that **are** in the phrase table, while the work cited above looks for paraphrases of source phrases that are **not** in the phrase table.

Our work also differs from most work on paraphrases in that information is extracted not

from sources outside the SMT system (*e.g.*, pivot languages or thesauri) but from the system's phrase table. In this respect if no other, it is similar to Chiang's classic work on hierarchical phrase-based systems (Chiang, 2005), though Chiang was mining a very different type of information from phrase tables.

In this paper, our goal is to obtain better estimates for P(**s**|**t**) and P(**t**|**s**), where **s** is a source-language phrase, **t** is a target-language phrase, and phrase pair (**s**,**t**) was seen at least once in training data. The current work is thus related to work on smoothing P(**s**|**t**) and P(**t**|**s**) – see (Foster *et al.*, 2006). Typically, relative frequency estimates for P(**s**|**t**) and P(**t**|**s**) are smoothed with "lexical" estimates found by breaking phrases **s** and **t** into words. We adopt a different idea, that of smoothing P(**s**|**t**) and P(**t**|**s**) with estimates obtained from phrases that have similar meanings to **s** and **t**. The two methods can be combined, and may provide complementary information. *E.g.*, lexical estimates don't work well for non-compositional phrases like "kick the bucket" - our method might cluster this phrase with "die" and "expire". The research that comes closest to ours is the work of Schwenk *et al.* (2007) on continuous space N-gram models, where a neural network is employed to smooth translation probabilities. However, both Schwenk *et al.*'s smoothing technique and the system to which it is applied are quite different from ours.

2 Deriving Conditional Probabilities from Phrase Clusters

Given phrase clusters in the source and target languages, how would one derive estimates for conditional probabilities P(**s**|**t**) and P(**t**|**s**)? We assume that the clustering is "hard": each source phrase **s** belongs to exactly one cluster C(**s**), and each target phrase **t** belongs to exactly one cluster C(**t**). Some of these clusters will contain singleton phrases, and others will contain more than one phrase. Let "#" denote the total number of observations in the training data associated with a phrase or phrase cluster. *E.g.*, suppose the English cluster C_S contains the three phrases "red", "dark red", and "burgundy", with 50, 25, and 10 observations in the training data respectively – then $\#(C_S) = 85$. Also, let $\#(C_S,C_T)$ be the number of co-occurrences in the training data of source-language cluster C_S and target-language cluster C_T.

The phrase-cluster-based probabilities P_{PC} are:

$$P_{PC}(s \mid t) = P(s \mid C(s)) \times P(C(s) \mid C(t)) = \frac{\#(s)}{\#C(s)} \times \frac{\#(C(s),C(t))}{\#C(t)} \quad (1)$$

and

$$P_{PC}(t \mid s) = P(t \mid C(t)) \times P(C(t) \mid C(s)) = \frac{\#(t)}{\#C(t)} \times \frac{\#(C(s),C(t))}{\#C(s)} \quad (2)$$

Note that the P_{PC} will often be non-zero where the corresponding relative frequency estimates P_{RF} were zero (the opposite can't happen). Also, the P_{PC} will be most useful where the phrase being conditioned on was seldom seen in the training data. If **t** was seen 1,000 times during training, the P_{RF}(**s**|**t**) are reliable and don't need smoothing; but if **t** was seen 6 times, P_{PC}(**s**|**t**) may yield valuable extra information. The same argument applies in reverse to estimation of P(**t**|**s**).

3 Clustering Phrases

We used only information "native" to phrase tables to cluster phrases. Two types of similarity metric between phrases or phrase clusters were employed: count-based metrics and edit-based metrics. The former are based on phrase co-occurrence counts; the latter are based on the word sequences that make up the phrases. Each has its advantages. Count-based metrics can deduce from the similar translations of two phrases that they have similar meanings, despite dissimilarity between the two word sequences – *e.g.*, they can deduce that "red" and "burgundy" belong in the same cluster. However, these metrics are unreliable when total counts are low, since phrase co-occurrences are determined by a noisy alignment process. Edit-based metrics are independent of how often phrases were observed. However, sometimes they can be fooled by phrases that have similar word sequences but different meanings (*e.g.*, "the dog bit the man" and "the man bit the dog", or "walk on the beach" and "don't walk on the beach"). In our experiments, we used a combination of count-based and edit-based metrics to cluster phrases (by simply multiplying the metrics together). However, we invested most of our effort in perfecting the count-based component: our edit-based metric was fairly naïve.

If we rely mainly on count-based similarity between phrases to cluster them, and this kind of similarity is most reliable when phrases have high counts, yet we need phrase-cluster-based

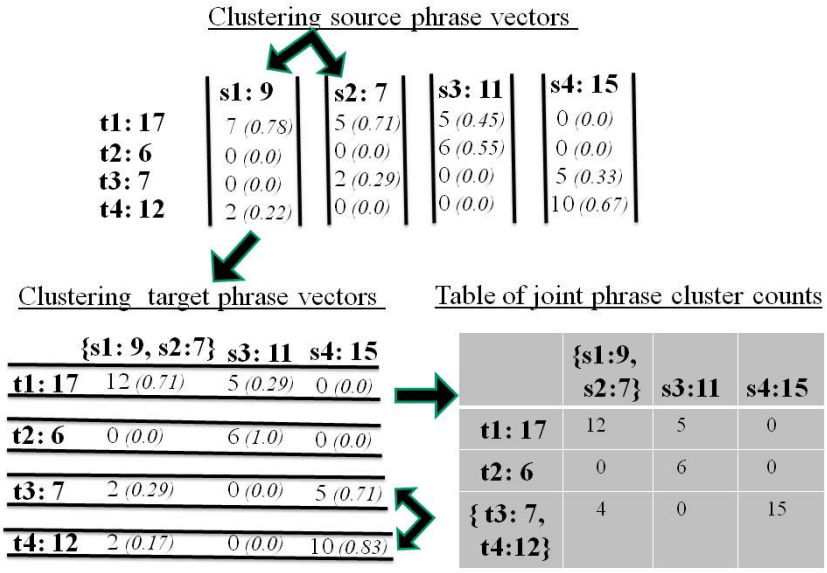

Figure 1: Example of phrase clustering

estimates most for phrases with low counts, aren't we carrying out clustering on the phrases that need it least? Our hope was that there is a class of phrases with intermediate counts (*e.g.*, with 3-15 observations in the training data) that can be clustered reliably, but still benefit from phrase-cluster-based probability estimates.

3.1 Count-based clustering: overview

Figure 1 shows count-based phrase clustering. One first arbitrarily picks a language (either source or target) and then clusters together some of the phrases in that language. One then switches to the other language and clusters phrases in that language, then switches back to the first one, and so on until enough clustering has taken place.

Each phrase or phrase cluster is represented by the vector of its co-occurrence counts. To calculate the similarity between two phrase clusters, one first normalizes their count vectors. At the top of **Figure 1**, source phrase **s1** occurred 9 times: 7 times aligned with target phrase **t1**, 2 times aligned with **t4**. For source similarity computation, the entry for (**s1**,**t1**) is normalized to 7/9 = 0.78 and the entry for (**s1**,**t4**) is normalized to 2/9 = 0.22 (these normalized values are shown in brackets and italics after the counts).

The two most similar normalized vectors at the top of **Figure 1** are those associated with phrases **s1** and **s2**. These phrases are merged by adding corresponding counts, yielding a new vector associated with the new phrase cluster {**s1**, **s2**}. In real life, one would now do more source-language clustering on the source language side; in this example, we immediately proceed to target-language clustering (carried out in target language space). Note that the target similarity calculations are affected by the previous source clustering (because **s1** and **s2** are now represented by the same coordinate, **t3** and **t4** are now closer than they were in the initial table). In this manner, we can iterate back and forth between the two languages. The final output is a table of joint phrase cluster counts, which is used to estimate the P_{PC} (see previous section).

3.2 Count-based clustering: details

Count-based similarity is computed as follows:

1. Phrase alignment is a noisy process, so we first apply the *tf-idf* transformation to phrase cluster counts. For source similarity computation, each co-occurrence count $\#(C_S,C_T)$ between source cluster C_S and target cluster C_T is multiplied by a factor that reflects the information content of C_T. Let $\#\text{diff}(C_S)$ be number of clusters on the source side, and let $\#[C_T>0]$ for a particular target cluster C_T be the number of source clusters C_S that co-occur with C_T. Then let $\#'(C_S,C_T)=\#(C_S,C_T)\times\log(\#diff(C_S)/\#[C_T>0])$. Similarly, for target similarity computation, let

$\#'(C_S, C_T) = \#(C_S, C_T) \times \log(\#diff(C_T)/\#[C_S > 0])$. *E.g.*, in source similarity computation, if C_T co-occurs with all source clusters, its contribution will be set to zero (because it carries little information).

2. We normalize by dividing each vector of *tf-idf* counts $\#'(C_S, C_T)$ by the total number of observations in the vector.
3. We compute the similarity between each pair of *tf-idf* vectors using either the cosine measure (Salton and McGill, 1986) or one of a family of probabilistic metrics described below.
4. We cluster together the most similar vectors; this involves summing the unmodified counts $\#(C_S, C_T)$ of the vectors (*i.e.*, the *tf-idf* transformation is only applied for the purposes of similarity calculation and is not retained).

Now, we'll describe the probabilistic metrics we considered. For a count vector of dimension D, $\underline{\mathbf{u}} = (u_1, u_2, \ldots, u_D)$, define a function $I(\underline{\mathbf{u}}) = u_1 \times \log(u_1 / \sum_i u_i) + \ldots + u_D \times \log(u_D / \sum_i u_i)$. $I(\underline{\mathbf{u}})$ is a measure of how well the data in $\underline{\mathbf{u}}$ are modeled by the normalized vector $(u_1/\Sigma_i u_i, \ldots, u_D/\Sigma_i u_i)$. Thus, when two count vectors $\underline{\mathbf{u}}$ and $\underline{\mathbf{v}}$ are merged (by adding them) we have the following measure of the loss in modeling accuracy:

Probability Loss (PL):

$$PL(\underline{\mathbf{u}}, \underline{\mathbf{v}}) = I(\underline{\mathbf{u}}) + I(\underline{\mathbf{v}}) - I(\underline{\mathbf{u}} + \underline{\mathbf{v}}) . \quad (3)$$

However, if we choose merges with the lowest PL, we will usually merge only vectors with small counts. We are more interested in the average impact of a merge, so we define

Average Probability Loss (APL):

$$APL(\underline{\mathbf{u}}, \underline{\mathbf{v}}) = (I(\underline{\mathbf{u}}) + I(\underline{\mathbf{v}}) - I(\underline{\mathbf{u}} + \underline{\mathbf{v}}))/(\sum_i u_i + \sum_i v_i) . \quad (4)$$

In our initial experiments, APL worked better than PL. However, APL had a strange side-effect. Most of the phrase clusters it induced made intuitive sense, but there were typically three or four clusters with large numbers of observations on both language sides that grouped together phrases with wildly disparate meanings. Why does APL induce these "monster clusters"?

Consider two count vectors $\underline{\mathbf{u}}$ and $\underline{\mathbf{v}}$. If $\Sigma_i u_i$ is very big and $\Sigma_i v_i$ is small, then $I(\underline{\mathbf{u}})$ and $I(\underline{\mathbf{u}} + \underline{\mathbf{v}})$ will be very similar, and APL will be approximately $I(\underline{\mathbf{v}}) / [\Sigma_i u_i + \Sigma_i v_i]$ which will be close to zero. Thus, the decision will probably be made to merge $\underline{\mathbf{u}}$ and $\underline{\mathbf{v}}$, even if they have quite different semantics. The resulting cluster, whose counts are represented by $\underline{\mathbf{u}} + \underline{\mathbf{v}}$, is now even bigger and even more likely to swallow up other small count vectors in the next rounds of merging: it becomes a kind of black hole. Worse, a monster cluster creates noise in the other language by merging coordinates that ought to be kept separate, making monster clusters in the other language likely.

To deal with this problem, we devised another metric. Let $I(\underline{\mathbf{u}} \mid \underline{\mathbf{v}}) = u_1 \times \log(v_1 / \sum_i v_i) + \ldots + u_D \times \log(v_D / \sum_i v_i)$. This is a measure of how well the counts in $\underline{\mathbf{v}}$ predict the distribution of counts in $\underline{\mathbf{u}}$. Then let

Maximum Average Probability Loss (MAPL):

$$MAPL(\underline{\mathbf{u}}, \underline{\mathbf{v}}) = \max\left(\frac{I(\underline{\mathbf{u}}) - I(\underline{\mathbf{u}} \mid \underline{\mathbf{u}} + \underline{\mathbf{v}})}{\sum_i u_i}, \frac{I(\underline{\mathbf{v}}) - I(\underline{\mathbf{v}} \mid \underline{\mathbf{u}} + \underline{\mathbf{v}})}{\sum_i v_i}\right) . \quad (5)$$

The first term inside the maximum indicates the average probability loss for an observation in $\underline{\mathbf{u}}$ when it is modeled by $\underline{\mathbf{u}}+\underline{\mathbf{v}}$ instead of $\underline{\mathbf{u}}$; similarly, the second term indicates the average probability loss for an observation in $\underline{\mathbf{v}}$. If we merge vector pairs with the lowest values of MAPL, we will never merge vectors in a way that will cause a large loss to either of the two parents.

In practice, we found that all these metrics worked better when multiplied by the Dice coefficient based distance. For $\underline{\mathbf{u}}$ and $\underline{\mathbf{v}}$, this is $Dice(\underline{\mathbf{u}}, \underline{\mathbf{v}}) = 1 - \frac{2 \times |\underline{\mathbf{u}} \cap \underline{\mathbf{v}}|}{|\underline{\mathbf{u}}| + |\underline{\mathbf{v}}|}$, where "$|\underline{\mathbf{u}}|$" means the number of non-zero count entries in $\underline{\mathbf{u}}$, and "$|\underline{\mathbf{u}} \cap \underline{\mathbf{v}}|$" is the number of count entries that are non-zero in $\underline{\mathbf{u}}$ and $\underline{\mathbf{v}}$.

3.3 Edit-based similarity

In most of our experiments, count-based metrics were combined with edit-based metrics; we put little effort into optimizing the edit metrics. Let MCWS stand for "maximum common word sequence". For phrases p_1 and p_2, we define

$$Edit(p_1, p_2) = 1 - \frac{2 \times len(MCWS(p_1, p_2))}{len(p_1) + len(p_2)} . \quad (6)$$

where *len*() returns the number of words.

This metric does not allow for costs that depend on word identity. In future work, we may penalize differences involving content or negative words more than other kinds of differences.

We also defined an edit-based metric for distance between phrase clusters. There are many ways of doing this. Let cluster 1 contain the

phrases "red" (10); "burgundy" (5); "somewhat resembling scarlet" (2) and cluster 2 contain "dark burgundy" (7); "scarlet" (3) (each number in brackets is how often a phrase was observed). What is the edit distance between clusters 1 and 2? We chose a naïve approach where the edit distance is taken between the two phrases that have the most observations in each cluster. Thus, the distance between cluster 1 and cluster 2 would be Edit("red", "dark burgundy")=1.0. Obviously, more sophisticated definitions of edit distance for phrase clusters are possible.

3.4 Examples of phrase clusters

Figure 2 shows an English phrase cluster learned in the course of our Chinese-English experiments. Each phrase is followed by its count in brackets; we do not show phrases with low counts. Since our edit distance regards words as atoms (it doesn't know about morphology), the phrases containing "emancipating" were clustered with phrases containing "emancipation" based on similarities in their Chinese translations, rather than because of the common stem.

Figure 3 shows part of a French phrase cluster learned during French-English experiments. The surface forms are quite varied, but most of the phrases mean "to assure or to guarantee that something will happen". An interesting exception is "pas faire" – it means not to do something ("pas" is negative). This illustrates why we need a better edit distance that heavily weights negative words.

```
emancipating (247), emancipate
(167), emancipate our (73),
emancipating thinking (67),
emancipate our minds (46), fur-
ther emancipate (45), emanci-
pate the (38), emancipate the
mind (38), emancipating minds
(33), emancipate their (32),
emancipate their minds (27),
emancipating our minds (24),
emancipating our (21), emanci-
pate our mind (21), further
emancipate our (19), emancipate
our thinking (14), further
emancipate their (11), emanci-
pating the minds (9), emanci-
pate thinking (8), unfettering
(8) ...
```

Figure 2: partial English phrase cluster

```
garantir que (64), assurer que
(46), veiller à ce que (27),
afin de garantir (24), faire en
sorte (19), de garantir que
(16), afin de garantir que
(14), faire des (14), de veil-
ler à ce (14), s' assurer que
(13), de veiller à ce que (13),
pour garantir que (13), de fai-
re en sorte (8), de faire en
sorte que (7), à garantir que
(6), pas faire (5), de veiller
(5), de veiller à (5) …
```

Figure 3: partial French phrase cluster

4 Experiments

We carried out experiments on a standard one-pass phrase-based SMT system with a phrase table derived from merged counts of symmetrized IBM2 and HMM alignments; the system has both lexicalized and distance-based distortion components (there is a 7-word distortion limit) and employs cube pruning (Huang and Chiang, 2007). The baseline is a loglinear feature combination that includes language models, the distortion components, relative frequency estimators $P_{RF}(\mathbf{s}|\mathbf{t})$ and $P_{RF}(\mathbf{t}|\mathbf{s})$ and lexical weight estimators $P_{LW}(\mathbf{s}|\mathbf{t})$ and $P_{LW}(\mathbf{t}|\mathbf{s})$. The $P_{LW}()$ components are based on (Zens and Ney, 2004); Foster *et al.* (2006) found this to be the most effective lexical smoothing technique. The phrase-cluster-based components $P_{PC}(\mathbf{s}|\mathbf{t})$ and $P_{PC}(\mathbf{t}|\mathbf{s})$ are incorporated as additional loglinear feature functions. Weights on feature functions are found by lattice MERT (Macherey *et al.*, 2008).

4.1 Data

We evaluated our method on Chinese-to-English and French-to-English tasks. For each pair, we carried out experiments on training corpora of different sizes. Chinese-to-English data were from the NIST[1] 2009 evaluation; all the allowed bilingual corpora except the *UN corpus*, *Hong Kong Hansard and Hong Kong Law corpus* were used to estimate the translation model. For Chinese-to-English, we trained two 5-gram language models: the first on the English side of the parallel data, and the second on the English *Gigaword corpus*.

Our Chinese-English development set is made up mainly of data from the NIST 2005 test set; it also includes some balanced-genre web-text

[1] http://www.nist.gov/speech/tests/mt

from the NIST training material. Evaluation was performed on the NIST 2006 and 2008 test sets. **Table 1** gives figures for training, development and test corpora for Chinese-to-English tasks; |S| is the number of sentences, and |W| is the number of words. Four references are provided for all dev and test sets.

			Chi	Eng
All parallel Train		|S|	3.3M	
		|W|	68.2M	66.5M
Dev		|S|	1,506	1,506×4
Test	NIST06	|S|	1,664	1,664×4
	NIST08	|S|	1,357	1,357×4
Gigaword		|S|	-	11.7M

Table 1: Statistics for Chinese-to-English tasks.

			Fre	Eng
Train	Europarl	|S|	1.6M	
		|W|	51.3M	46.6M
Dev	2008	|S|	2,051	
Test	2009	|S|	2,525	
	2010	|S|	2,489	
GigaFrEn		|S|	-	22.5M

Table 2: Statistics for French-to-English tasks.

Lang (#sent)		Ch-En (3.3M)		Fr-En (1.6M)	
		#count-1	#other	#count-1	#other
Src	Before clustering	11.3M	5.7M	28.1M	21.2M
	After clustering	11.3M	5.3M	28.1M	19.3M
	#clustered	0	0.4M	0	1.9M
Tgt	Before clustering	11.9M	6.0M	25.6M	20.4M
	After clustering	11.9M	5.6M	25.6M	18.5M
	#clustered	0	0.4M	0	1.9M

Table 3: # phrase classes before & after clustering.

For French-to-English tasks, we used WMT 2010[2] French-to-English track data sets. Parallel *Europarl* data are used for training; the WMT Newstest 2008 set is used as the dev set, and WMT Newstest 2009 and 2010 are used as test sets. One reference is provided for each source input sentence. Two language models are used in this task: one is the English side of the parallel data, and the second is the English side of the *GigaFrEn* corpus. **Table 2** gives figures for training, development and test corpora for French-to-English tasks.

4.2 Amount of clustering and metric

Parameters for clustering were determined by preliminary tests involving the FBIS corpus (about 8% of the Chinese-English data). For Chinese-English, we decided to first cluster Chinese by a small amount, then English by about the same amount. For French-English, we first clustered French, then English, using the same clustering parameters as for Chinese-English. In both cases, it looked as though phrases that occurred only once in training data couldn't be clustered reliably, so we prevented these "count 1" phrases from participating in clustering.

Table 3 shows the results. Only 2-4% of the total phrases in each language end up in a cluster (that's 6.5-9% of eligible phrases, *i.e.*, of phrases that aren't "count 1"). However, about 20-25% of translation probabilities are smoothed for both language pairs. Based on these preliminary tests, we decided to use $Edit \times Dice \times MAPL$ ($Edit \times DMAPL$) as our metric (though $Edit \times Cosine$ was a close runner-up).

4.3 Results and discussion

Our evaluation metric is IBM BLEU (Papineni *et al.*, 2002), which performs case-insensitive matching of n-grams up to $n = 4$. Our first experiment evaluated the effects of the phrase clustering features given various amounts of training data. **Figure 4** gives the BLEU score improvements for the two language pairs, with results for each pair averaged over two test sets (training data size shown as #sentences). The improvement is largest for medium amounts of training data. Since the French-English training data has more words per sentence than Chinese-English, the two peaks would have been closer together if we'd put #words on the x axis: improvements for both tasks peak around 6-8 M English words. For more details, refer to **Table 4** and **Table 5**. The biggest improvement is 0.55 BLEU for the NIST06 test. More importantly, cluster features help performance in 29 of the 30 experiments.

[2] http://www.statmt.org/wmt10/

Data size	Nist06			Nist08		
	Baseline	+phrase-clustering	Improv.	Baseline	+phrase-clustering	Improv.
25K	21.66	21.88	0.22	15.80	15.99	0.19
50K	23.23	23.43	0.20	17.69	17.84	0.15
100K	25.83	26.24	0.41	20.08	20.27	0.19
200K	27.80	28.26	0.46	21.28	21.58	0.30
400K	29.61	30.16	**0.55**	23.37	23.75	**0.38**
800K	30.87	31.17	0.30	24.41	24.65	0.24
1.6M	32.94	33.10	0.16	25.61	25.72	0.11
3.3M	33.59	33.64	0.05	26.84	26.85	0.01

Table 4: BLEU(%) scores for Chinese-to-English task with the various training corpora, including baseline results, results for with phrase clustering, and the absolute improvements. Corpus size is measured in sentences.

Data size	Newstest2009			Newstest2010		
	Baseline	+phrase-clustering	Improv.	Baseline	+phrase-clustering	Improv.
25K	20.21	20.37	0.16	20.54	20.73	0.19
50K	21.25	21.44	0.19	21.95	22.11	0.16
100K	22.56	22.86	0.30	23.44	23.69	0.25
200K	23.67	24.02	**0.35**	24.31	24.71	**0.40**
400K	24.36	24.50	0.14	25.28	25.46	0.18
800K	24.92	24.97	0.05	25.80	25.90	0.10
1.6M	25.47	25.47	0.00	26.35	26.37	0.02

Table 5: BLEU(%) scores for French-to-English task with the various training corpora, including baseline results without phrase clustering feature, results for phrase clustering, and the absolute improvements.

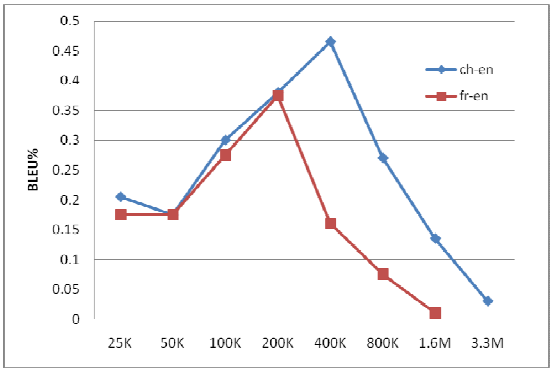

Figure 4: Average BLEU improvement for Chinese-to-English and French-to-English tasks (each averaged over two tests) *vs.* #training sent.

In the research on paraphrases cited earlier, paraphrases tend to be most helpful for small amounts of training data. By contrast, our approach seems to be most helpful for medium amounts of training data (200-400K sentence pairs). We attribute this to the properties of count-based clustering. When there is little training data, clustering is unreliable; when there is much data, clustering is reliable but unneeded, because most relative frequencies are well-estimated. In between, phrase cluster probability estimates are both reliable and useful.

Finally, we carried out experiments to see if some of our earlier decisions were correct. Were we right to use DMAPL instead of cosine as the count-based component of our metric? Experiments with $Edit \times DMAPL$ *vs.* $Edit \times Cosine$ on 400K Chinese-English (tested on NIST06 and NIST08) and on 200K French-English (tested on Newstest2009 and 2010) showed a tiny advantage for $Edit \times DMAPL$ of about 0.06 BLEU. So we probably didn't make the wrong decision here (though it doesn't matter much). Were we right to include the *Edit* component? Carrying out analogous experiments with $Edit \times DMAPL$ *vs*. *DMAPL*, we found that dropping *Edit* caused a loss of 0.1-0.2 BLEU for all four test sets. Here again, we made the right decision.

In a final experiment, we allowed "count 1" phrases to participate in clustering (using $Edit \times DMAPL$). The resulting Chinese-English system had somewhat more clustered phrases than the previous one (for both Chinese and English, about 3.5% of phrases were in clusters compared to 2.5% in the previous system). To our surprise, this led to a slight improvement in

BLEU: the 400K Chinese-English system now yielded 30.25 on NIST06 (up 0.09) and 23.88 on NIST08 (up 0.13). The French-English system where "count 1" clustering is allowed also had more phrases in clusters than the system where it's prohibited (the former has just under 10% of French and English phrases in clusters *vs.* 4% for the latter). For French-English, the 200K system allowing "count 1" clustering again yielded a slightly higher BLEU: 24.07 on Newstest2009 and 24.76 on Newstest2010 (up 0.05 in both cases). Thus, our decision not to allow "count 1" phrases to participate in clustering in the Table 4 and 5 experiments appears to have been a mistake. We suspect we can greatly improve handling of "count 1" phrases – *e.g.*, by weighting the Edit component of the similarity metric more heavily when assigning these phrases to clusters.

5 Conclusion and Future Work

We have shown that source-language and target-language phrases in the phrase table can be clustered, and that these clusters can be used to smooth "forward" and "backward" estimates P(**t**|**s**) and P(**s**|**t**), yielding modest but consistent BLEU gains. Though our experiments were done on a phrase-based system, this method could also be applied to hierarchical phrase-based SMT and syntactic SMT systems. There are several possibilities for future work based on new applications for phrase clusters:

- In the experiments above, we used phrase clusters to smooth P(**t**|**s**) and P(**s**|**t**) when the pair (**s**,**t**) was observed in training data. However, the phrase clusters often give non-zero probabilities for P(**t**|**s**) and P(**s**|**t**) when **s** and **t** were both in the training data, but didn't co-occur. We could allow the decoder to consider such "invented" phrase pairs (**s**,**t**).
- Phrase clusters could be used to construct target language models (LMs) in which the basic unit is a phrase cluster rather than a word. For instance, a tri-cluster model would estimate the probability of phrase **p** at time **i** as a function of its phrase cluster, $\boldsymbol{C_i}(\mathbf{p})$, and the two preceding phrase clusters $\boldsymbol{C_{i-1}}$ and $\boldsymbol{C_{i-2}}$: $P(\mathbf{p}) = f(\mathbf{p} \mid C_i(\mathbf{p})) \times f(C_i \mid C_{i-1}C_{i-2})$.
- Lexicalized distortion models could be modified so as to condition distortion events on phrase clusters.
- We could build SMT grammars in which the terminals are phrases and the parents of terminals are phrase clusters.

The phrase clustering algorithm described above could be improved in several ways:

- In the above, the edit distance between phrases and between phrase clusters was crudely defined. If we improve edit distance, it will have an especially large impact on "count 1" phrases, for which count-based metrics are unreliable and which are a large proportion of all phrases. The edit distance between two phrases weighted all words equally: preferably, weights for word substitution, insertion, or deletion would be learned from purely count-derived phrase clusters (content words and negative words might have heavier weights than other words). The edit distance between two phrase clusters was defined as the edit distance between the phrases with the most observations in each cluster. *E.g.*, distance to the phrase cluster in **Figure 2** is defined as the phrase edit distance to "emancipating". Instead, one could allow a cluster to be characterized by (*e.g.*) up to three phrases, and let distance between two clusters be the minimum or average pairwise edit distance between these characteristic phrases.
- To cluster phrases, we only used information derived from phrase tables. In future work, we could also use the kind of information used in work on paraphrases, such as the context surrounding phrases in monolingual corpora, entries in thesauri, and information from pivot languages.
- The phrase clustering above was "hard": each phrase in either language belongs to exactly one cluster. We could modify our algorithms to carry out "soft" clustering. For instance, we could interpolate the probabilities associated with a phrase with probabilities from its neighbours.
- Clustering is a primitive way of finding latent structure in the table of joint phrase counts. One could apply principal component analysis or a related algorithm to this table.

References

C. Bannard and C. Callison-Burch. "Paraphrasing with Bilingual Parallel Corpora". *Proc. ACL*, pp. 597-604, Ann Arbor, USA, June 2005.

C. Callison-Burch, P. Koehn, and M. Osborne. "Improved Statistical Machine Translation Using Paraphrases". *Proc. HLT/NAACL*, pp. 17-24, New York City, USA, June 2006.

C. Callison-Burch. "Syntactic Constraints on Paraphrases Extracted from Parallel Corpora". *Proc. EMNLP*, pp. 196-205, Honolulu, USA, October 2008.

D. Chiang. "A hierarchical phrase-based model for statistical machine translation". *Proc. ACL*, pp. 263-270, Ann Arbor, USA, June 2005.

G. Foster, R. Kuhn, and H. Johnson. "Phrasetable smoothing for statistical machine translation". *Proc. EMNLP*, pp. 53-61, Sydney, Australia, July 2006.

L. Huang and D. Chiang. "Forest Rescoring: Faster Decoding with Integrated Language Models". *Proc. ACL*, pp. 144-151, Prague, Czech Republic, June 2007.

P. Koehn. 2010. *Statistical Machine Translation.* Cambridge University Press, Cambridge, UK.

W. Macherey, F. Och, I. Thayer, and J. Uszkoreit. "Lattice-based Minimum Error Rate Training for Statistical Machine Translation". *Proc. EMNLP*, pp. 725-734, Honolulu, USA, October 2008.

Y. Marton, C. Callison-Burch, and Philip Resnik. "Improved Statistical Machine Translation Using Monolingually-Derived Paraphrases". *Proc. EMNLP*, pp. 381-390, Singapore, August 2009.

K. Papineni, S. Roukos, T. Ward, and W. Zhu. "Bleu: a method for automatic evaluation of machine translation". *Proc. ACL*, pp. 311–318, Philadelphia, July 2002.

G. Salton and M. McGill. 1986. *Introduction to Modern Information Retrieval.* McGraw-Hill Inc., New York, USA.

H. Schwenk, M. Costa-jussà, and J. Fonollosa. "Smooth Bilingual N-gram Translation". *Proc. Joint EMNLP/CoNLL*, pp. 430-438, Prague, Czech Republic, June 2007.

R. Zens and H. Ney. "Improvements in phrase-based statistical machine translation". *Proc. ACL/HLT*, pp. 257-264, Boston, USA, May 2004.

S. Zhao, H. Wang, T. Liu, and S. Li. "Pivot Approach for Extracting Paraphrase Patterns from Bilingual Corpora". *Proc. ACL/HLT*, pp. 780-788, Columbus, USA, June 2008.