PORTAGE in the NIST 2009 MT Evaluation

George Foster, Boxing Chen, Eric Joanis,
Howard Johnson, Roland Kuhn, and Samuel Larkin

31 August 2009

Abstract

This report describes experiments performed during preparations for the NIST 2009 MT evaluation, where we participated in the constrained Chinese-English condition with PORTAGE. The aim is to publicize new findings about the optimum configuration, and to suggest some avenues for further improvement.

Contents

1 Overview

For this year's NIST evaluation, we decided to participate only in the Chinese-to-English constrained task with PORTAGE.[1] Other possible tasks were Arabic-to-English and Urdu-to-English; all tasks could be performed either with constrained training data (limited to a prescribed list) or unconstrained; and in either a single-system or system-combination track. Unfortunately, Chinese-English was evaluated only on a progress test set this year—this is sequestered data re-used from year to year, whose reference translations are not released by NIST. Since first-time NIST participants are not automatically allowed to participate in progress tests, this may have meant fewer participants for this task. See
`www.itl.nist.gov/iad/mig/tests /mt/2009` for more information about the NIST evaluation.

Our results were quite good: we placed 3rd out of 19 participants in our track, and 8th out of all 55 primary and contrastive submissions in the

[1] We also participated in the unconstrained Chinese-English condition with a Systran serial combination, not discussed here.

merged single-system/system-combination constrained tracks. We also improved by 3.3 BLEU percentage points over last year's results, which was the largest increase registered by any site in the Chinese-English progress test. However, to put this in perspective, our BLEU score was 28.11, compared to 32.25 for the top system (ISI/Language Weaver) and 27.90 for the system immediately below us (SRI). Also, a number of traditionally strong sites, such as Google, IBM, and Edinburgh, did not field a Chinese-English system this year.

A nice aspect of this year's system was that many of the improvements to PORTAGE undertaken during the previous year bore fruit. The results are therefore largely a group effort; certainly each of the authors of this report is directly responsible for some of the BLEU score improvement over last year's results.

The main purpose of this report is to describe the configuration used for the NIST 2009 evaluation and the experiments run to establish it (including many unsuccessful ones). It should be stressed that these are research results and cannot be interpreted as representing "best practice" for PORTAGE.[2] Another aim of the report is to identify areas where further investigation seems warranted. These are indicated with a $\bigoplus$ symbol in the text.

Preparation for an evaluation is essentially a greedy search, so there is no single common baseline for most of the experiments presented below. This "evolving baseline" phenomenon was compounded by the introduction of new HMM code and a slightly new phrase extraction algorithm mid-stream. Further complications resulted from the use of three different versions of the training corpora: nist08, nist09-illegal, and nist09-final (see section 2 for more details). Experiments based on nist08 are marked with ** and those based on nist09-illegal are marked with * in all tables of results. To save time, most experiments were run only with the decoder, with no nbest rescoring step. Whenever possible, we also avoided the use of MERT by substituting alternative models (phrase tables, LMs, etc) while keeping log-linear weights fixed. Unless otherwise stated, all results are measured on lowercase output using our in-house BLEU program (`bleumain`), which implements the same algorithm as this year's official scoring script (`mteval-v13.pl`).[3]

As shown in the table of contents, the remainder of this report is orga-

[2]The best configuration is highly dependent on setting, and the setting for NIST is atypical in many ways. The performance of a configuration can also change substantially due to new features being added to PORTAGE.

[3]This algorithm uses the closest-length-match reference, rather than the shortest reference, in the brevity penalty calculation. bleumain is much faster than mteval, and unlike mteval, it takes tokenized text as input.

nized so as to roughly follow PORTAGE's processing chain: data, baseline, preprocessing, LMs, TMs and DMs, decoding, rescoring, and truecasing and detokenization. The final section presents some conclusions.

2 Data

2.1 Interim Versions

We used two interim versions of the training data during preparation for the evaluation:

- nist08. This was essentially last year's corpus, minus one corpus (LDC2002E17 - Chinese Treebank translation) that does not appear on the list of allowed resources for this year (it is superceded by LDC2003E07 - Chinese Treebank English Parallel Corpus, but we did not have a preprocessed version of that corpus available when we wanted to begin training). Results based on this corpus are identified with **.

- nist09-illegal. This is identical to the nist09-final corpus described in the next section except for the inclusion of LDC2009E10 - 2008 English-Chinese eval set among the training material. A few weeks before the evaluation, NIST sent out a message announcing that LDC2009E10 was out of bounds because it overlapped the epoch of the progress eval set, so we had to back it out of the system.[4] This corpus uses different preprocessing from nist08. Results based on it are identified with *.

2.2 Final Version

Table 1 lists the sub-corpora that comprise the nist09-final corpus, used for the evaluation. These are a subset of the full list of resources permitted for the Chinese constrained track, which can be categorized into:

- parallel corpora: we used all available except for the 2008 English-Chinese eval set (LDC2009E10), which was prohibited as described above; and the Hong Kong Parallel Text (LDC2004T08), which we presumed to be equivalent to the existing Hong Kong parallel corpora (LDC2000T50, LDC2000T47, and LDC2000T46).

[4] One might wonder why it was included among the allowed resources at all if this were the case. But perhaps the intent was that it could be useful as a test corpus—similar to what we assume is a legitimate use for the 2008 Chinese-English eval set.

- monolingual corpora: we used only the English Gigaword, not the Chinese Gigaword (LDC2007T38), nor the Google ngrams (LDC2006T13), nor several smaller English corpora (LDC2002T31, LDC2005T28, LDC2005T35, LDC1993T1, LDC1993T3A, LDC1995T21, LDC1995T6, and LDC1998T30) which are possibly already contained in the Gigaword.

- other resources: we used the named entity list (LDC2005T34) and bilingual lexicon (LDC2002L27), treating them as ordinary parallel corpora, as shown in table 1; we eschewed the Chinese propbank (LDC2005T23), Chinese treebank (LDC2007T36), tagged Chinese Gigaword (LDC2007T03), the English treebank (LDC1999T42), the English Chinese translation treebank (LDC2007T02), manual word alignments (LDC2006E86 and LDC2006E93), and OntoNotes (LDC2007T21).

As shown in table 1, the dev corpus (used for MERT and rescoring) was partly taken from the NIST05 eval set, and partly from GALE webtext corpora. We tried to match the balance of genres from the NIST08 eval set (roughly equal amounts of newswire and webtext), but found that last year's dev set, which contained about 70% webtext, gave better results. ⊕ Table 2 shows the sizes and compositions of the parallel corpora used for training, dev, and test.

2.3 File Locations

All files pertaining to the NIST 2009 system are stored on balzac below `/home/portage`.

Raw corpora are stored in `corpora/GALE_distn` and intermediate preprocessed versions are in `corpora/GALE_prepd`. Final preprocessed versions are in `corpora/NIST09-Chinese`: the `collect` subdirectory contains scripts for installing various versions from `GALE_prepd` into `NIST09-Chinese`, performing some further preprocessing and reorganization along the way.

Systems are in `models/NIST09-Chinese`: subdirectories tend to correspond to different versions of the data, eg, for the versions listed in the precedeing sections: `v1-nist08` for nist08, `v3-newproc-full` for nist09-illegal, and `v4-no-e2c` for nist09-final.

corpus	genre	usage	size
LDC2007T07 English Gigaword Third Edition	in	trg lm	172,125,525
LDC2005T34 Chinese English Name Entity Lists		trg	1,192,305
LDC2002L27 Translation Lexicon Version 3.0		trg	82,098
LDC2005T10 Chinese English News Mag. Parallel Text	in	trg	281,687
LDC2005T06 Chinese News Translation Text Part 1	in	trg	10,317
LDC2003E07 Chinese Treebank English Parallel Corpus	in	trg	4,172
LDC2003E14 FBIS Multilanguage Texts	in	trg	249,867
LDC2002E18 Xinhua Parallel News Text Version 1	in	trg	109,746
LDC2007T09 ISI Automatically Extracted Parallel Text	in	trg	558,378
LDC2004E12 UN Chinese English Parallel Text	out	trg	4,979,857
LDC2000T50 Hong Kong Hansards Parallel Text	out	trg	1,297,205
LDC2000T47 Hong Kong Laws Parallel Text	out	trg	400,667
LDC2000T46 Hong Kong News Parallel Text		trg	702,071
LDC2006E26 GALE Financial News		trg	90,302
LDC2005E83 GALE Y1 Q1 Release - Translations	in	trg/dev	6,437
LDC2006E34 GALE Y1 Q2 Release - Translations V2.0	in	trg/dev	9,828
LDC2006E85 GALE Y1 Q3 Release - Translations	in	trg/dev	11,990
LDC2006E92 GALE Y1 Q4 Release - Translations	in	trg/dev	28,843
LDC2006E24 GALE Y1 Interim Release - Translations	in	trg/dev	20,842
LDC2002T01 Multiple-Translation Chinese Corpus	in	trg	10,923
LDC2003T17 Multiple-Translation Chinese Part 2	in	trg	3,512
LDC2004T07 Multiple-Translation Chinese Part 3	in	trg	3,740
LDC2006T04 Multiple Translation Chinese Part 4	in	trg	3,676
LDC2006E43 NIST 2004 Chinese evaluation set	in	test	1,788
LDC2006E38 NIST 2005 Chinese evaluation set	in	dev	1,082
LDC2007E59 NIST 2006 Chinese evaluation set	in	test	1,664
LDC2009E09 NIST 2008 Chinese evaluation set	in	test	1,357
total size (parallel corpora)			10,071,790

Table 1: Corpora used for NIST 2009: all are parallel except for the English Gigaword on the first line. The *genre* column indicates whether corpora are in-domain (news and webtext), out of domain, or unknown/neutral. The *usage* column indicates whether corpora were used for training, dev, or test. The *size* column gives the number of sentence pairs.

corpus	use	composition	size
train	LM/TM training	all except devtest	10,063,395
dev1	decoding MERT	nist05 + GALE Y1 newsgroup+weblog	1,506
dev2	rescoring MERT	nist05 + GALE Y1 newsgroup+weblog	2,080
nist04	test, reserved	–	1,788
nist06	main test	–	1,664
nist08	main test	–	1,357

Table 2: Division of parallel corpora.

3 Baseline Configuration

The baseline configuration uses the following major features that have been helpful in previous evaluations:

- 5gram static Gigaword LM
- 4gram mixture-adapted LM: train LM's on the source and target sides of all training corpora in table 1, and use EM to find the linear weighting of source-side models that maximizes likelihood of the current source text (all documents combined); apply this weighting to the corresponding target-side models during decoding as described in [6].
- lexicalized HMMs: condition HMM jump probabilities on the current hidden-sequence word, using MAP smoothing to back off to global jump probabilities, as in [9].
- "domain adapted" phrase probabilities: extract phrase pairs from the whole training corpus, but calculate Zens-Ney [23] conditional estimates using the ttable from an HMM model trained on only the in-domain and neutral domain material from table 1.

The resulting system has 8 loglinear features: 4 phrase probabilities (relative frequency and Zens-Ney estimates in both directions), 2 LM probabilities, word count, and distortion penalty. Table 3 shows the contributions of some of the major features.

configuration	nist06	nist08
0. 2008 best	32.66	26.57
1. *init baseline	29.26	23.65
2. mix LM	30.46	24.64
3. mix LM+ lex HMM	30.72	24.99
4. mix LM+ lex HMM + adapted TM	31.62	25.42
5. mix LM+ giga LM + lex HMM	32.15	25.79
6. mix LM+ giga LM + lex HMM + adapted TM	32.68	26.00
7. baseline	**33.12**	**26.65**

Table 3: Baseline system performance, showing contributions of major features. The first line shows best results from last year's system, which are not comparable in many respects (corpora, preprocessing, rescoring, etc.) to the other configurations in the table. The last two lines are identical configurations except for the use of an improved MERT algorithm.

configuration	nist06	nist08
** baseline, nist08 preproc	32.33	25.98
baseline, nist09 preproc + ntreebank	**33.12**	**26.65**

Table 4: Effects of Preprocessing. Both lines correspond to the *baseline* configuration from table 3, but the 1st line was generated using NIST 08 preprocessing.

4 Preprocessing

In contrast to previous years, we did not undertake a major preprocessing initiative this year. Work was limited to an extensive overhaul of the FBIS corpus (better alignment and encoding handling), new punctuation normalization procedures developed for GALE, and cleaning up some GALE speech-transcription data by resolving "./?" ambiguities at the ends of lines. These measures helped significantly, as shown in table 4.[5]

configuration	nist06	nist08
baseline	32.15	25.79
filt 100	32.11	25.77
split 100	32.18	25.78
split 50	32.21	25.72
split 25	**32.43**	**25.97**

Table 5: Corpus splitting and filtering results. The baseline corresponds to line 5 in table 3, the *filt* line is hard filtering using a 100-word threshold and a 9:1 length ratio, and *split* n is Xu splitting with a length limit of n.

In addition to character-level preprocessing, we also looked at filtering the corpus with a hard length limit (100 words) and sentence-length ratio threshold (9:1), and splitting sentences using an algorithm due to Xu et al [21]. The Xu et al algorithm recursively splits sentence pairs using IBM-model probabilities until they are shorter than a given length limit. In practice, due to the presence of noisy input, this condition is impossible to satisfy (eg, if an input pair consists of a very long sentence aligned with a very short one), so we apply hard filtering as a postprocessing step. As shown in table 5, splitting with a limit of 25 improves BLEU score somewhat. It

[5]Note that, as indicated by the asterisks, the comparison in this table is unfortunately confounded with the use of the LDC2003E07 corpus in the nist09-final data set, used with the nist09 preprocessing on line 2.

also decreases training time, which is important due to the HMM slowdown discussed in section 6. The non-filtered and non-split version was used for the evaluation because the positive results with length=25 were not yet available.⊕

5 Language Models

Experiments with LMs involved the use of two different kinds of adaptation as well as higher-order ngrams. We did not have time to try out dynamic number-mapping LMs, nor Google ngrams.⊕

5.1 Dynamic Language Models for Mixture Adaptation

The baseline adapted LM is a mixture of LMs trained on the target side of each of the parallel training corpora listed in table 1,[6] along with a global model trained on the aggregate of all of these target corpora (including the latter gives a consistent gain over just using models trained on the individual corpora). To set weights, we train corresponding models on the *source* sides of all corpora, then use the EM algorithm to minimize perplexity on the current source text. With the exception of including the global model, this is exactly the procedure described in [6]. As was found in [6], it is not beneficial to learn weights that are specific to individual documents or even genres, so we learn a single set of weights per test set (eg one for nist06, one for nist08, etc). Even this granularity may not be optimal, as exchanging these weights does not lead to systematically worse results.

In the past, we have experimented with a different kind of adapted LM, trained on a set of sentences selected using IR matching techniques applied to the current source text. Essentially, each sentence in the text is treated as an IR query, and n matching source sentences are identified in the parallel corpus. The corresponding target sentences are pooled (across all queries) to form a *match* corpus, on which a *dynamic LM* is trained. Originally we used the dynamic LM as a feature within the global log-linear combination, but it did not help when mixture model adaptation was also being used. This year, we hypothesized that it might be more effective to include it *within* the adapted mixture by treating the match corpus as an additional component of the parallel corpus.

[6] The GALE Y1 were merged into a single corpus, then split according to genre: broadcast conversation (bc), broadcast news (bn), newsgroups (ng), and weblogs (wl). As described above, some of this material was also reserved for the development corpus.

configuration	nist06	nist08
*baseline	33.08	26.67
*lemur 50	33.04	26.53
*lemur 100	33.08	26.58
*lemur 100 / 0.05	33.06	26.85
*lemur 100 / 0.10	**33.26**	26.90
*lemur 100 / 0.15	33.20	26.98
*lemur 100 / 0.20	33.11	**27.00**

Table 6: Results for adapted mixture LMs including a dynamic component. The baseline is equivalent to line 7 in table 3. The *lemur n* lines use the best n matches for each sentence according to Lemur. The */ w* lines assign a weight of w to the dynamic LM instead of the weight assigned by the EM algorithm.

To perform sentence matching, we used the freely-available *Lemur* toolkit (`www.lemurproject.org`) as well as an in-house ngram-matching program (see `tmem/README` in the PORTAGE hierarchy) [16].[7] For each matching technique, we generated the top 100 matching sentence pairs for each source sentence, ranked in order of descending match score. The top half of table 6 shows the results for experiments with Lemur, using different numbers of retained matching target sentences to form the match corpus. Log-linear weights were held fixed across these experiments in order to remove the effect of MERT noise [7]. Clearly there is no significant gain from using the match corpus as an additional component in the baseline mixture adaptation algorithm.

To analyze these results, we looked at the weights on the various corpus components, shown in table 7.[8] The pattern is similar for both test corpora: the match corpus receives a very high weight, with the relative weighting on other corpus components being roughly preserved from the baseline configuration. The EM algorithm appears to have a bias toward the match corpus, perhaps due to the fact that Lemur uses ngrams in its matching algorithm. That this does not reflect the usefulness of the match corpus in general is obvious from table 8, which shows a sample of best-match sentences selected

[7] The key parameter to Lemur is `-rule=method:dirichlet,mu:2500`. For the `tmem` matches, an ngram length of 5 was used.

[8] The weights for the *+dyn* columns do not sum to one in this table. The original mixture included the match corpora for all devtest corpora instead of only the current one, by mistake. These typically received very low weights, and have been removed from the table to make it smaller.

component	nist06		nist08	
	baseline	+dyn	baseline	+dyn
fbis	0.093537	0.043937	0.108907	0.050078
financial	0.001643	0.001072	0.002083	0.000844
hkh	0.000707	0.000835	0.002638	0.002388
hkl	0.000014	0.000000	0.000001	0.000018
hkn	0.000158	0.000216	0.000849	0.000590
isi	0.230176	0.125781	0.164057	0.076060
lexicon	0.002263	0.002539	0.003365	0.002863
multip1-merge	0.000061	0.000040	0.000045	0.000019
multip2-merge	0.000274	0.000121	0.000401	0.000147
multip3-merge	0.000121	0.000061	0.000871	0.000367
multip4-merge	0.000482	0.000324	0.000987	0.000381
ne	0.000029	0.000010	0.000099	0.000044
news	0.015327	0.006387	0.021050	0.007754
nist08rev-merge	0.003730	0.002674	0.010891	0.009700
ntreebank	0.000125	0.000040	0.000441	0.000099
sinorama	0.026757	0.004358	0.145991	0.035959
unv2	0.000845	0.020414	0.000720	0.014114
xinhua	0.006535	0.000836	0.007234	0.000523
y1-bc	0.029609	0.020596	0.045019	0.026828
y1-bn	0.122091	0.084546	0.025094	0.013387
y1-ng-90	0.025692	0.014639	0.027456	0.016234
y1-wl-90	0.007697	0.006460	0.030752	0.021734
all	0.432140	0.198651	0.401048	0.183030
match	—	0.445767	—	0.518399

Table 7: Mixture LM weights for parallel corpus components.

score	match	reference
-3.37	MR ANDREW CHENG : Mr President , I would like to begin my speech with a joke on the judiciary of China in a bid to harmonize the atmosphere .	Jokes
-6.92	Ku Chin-t'ien goes to flea markets in his spare time , advising designers to spend their time hunting for what could be useful articles .	I wanted to understand more about what we should be aware of at railroad stations in various places .
-7.30	After the statement was issued in Beijing today , the head of the Chinese delegation , Long Yongtu , told reporters that China 's entry into the WTO requires agreement on market access by China and all its trade partners , [. . .]	Director-general Lamy of the WTO called these suggestions " a fair and reasonable foundation for reaching an agreement which has major long-term targets and balance and is development-oriented . "
-7.52	It noted that only five drugs have been approved for treating pre-cancers , including those for treatment of skin , bladder , breast and colon pre-cancers . Even though surgical treatment exist for many types of pre-cancers , [. . .]	The research also discovered that one of the genes , while increasing people 's risk of suffering from the cancer of the prostate gland , also lowered the risk of suffering from Type 2 diabetes .
-7.66	Even though Hong Kong can hardly compare with the United States , our ratio in this respect still should not be too low . Otherwise , our associate degree programme graduates will just be " prospect-less " graduates .	The man refused , and became argumentative .
-7.82	Commander Liu introduced them to me in a very familiar way ; it was not at all easy for so many parameters to be accurate , and I was amazed at his memory ; I even wondered whether or not I needed to put so many hard to remember figures into a microcomputer .	But the driver involved isn't that lucky . He has to remember , and very likely will remember it all his life .

Table 8: Best-match sentences for nist08 retrieved by Lemur, compared to the corresponding reference translation.

at random for various levels of the matching score. Clearly, most matches contain very few words in common with the reference translations.[9] Some matches are better than others, but the match score does not appear to be a reliable indicator of these.⊕

We attempted to compensate for the EM bias toward the match corpus by setting the weight on the dynamic LM discriminatively, ie simply tuning its value to achieve the best score on the test corpora (without renormalizing the mixture). The results in table 6 (below the horizontal line) show small gains of around 0.2 BLEU for weights that are much lower than the EM-assigned values. We chose 0.15 as a good compromise for optimum weight value across the two test sets.

Future work in this direction could include finer-grained tuning, the use of source-sentence-specific dynamic LMs (instead of just one per source text), tmem matches instead of Lemur ones, and discriminative assignment of all mixture weights instead of only the one on the dynamic LM.⊕

5.2 Gigaword Mixture Adaptation

Motivated by positive results for LM mixture adaptation using the parallel corpus, we tried a similar approach with the unilingual Gigaword corpus, which consists of six distinct sub-corpora drawn from different news agencies, as listed in table 9. Since Chinese versions of these corpora were not available for learning weights specific to each new source text as in the usual method, we learned a single set of weights using the target side of the `dev1` dev set, then held these fixed for subsequent translation. Weights were learned using EM to minimize perplexity as usual. Table 9 shows that there are clear (and unsurprising) preferences for certain sub-corpora over others. Table 10 shows the results of substituting the Gigaword mixture for the static Gigaword LM, which are substantially worse than the baseline. We currently have no explanation for why the drop is so large. ⊕

5.3 Effects of Ngram Order

Experiments in the previous sections were performed using a 4gram LM for the mixture model derived from the parallel corpus, and a 5gram LM trained on the Gigaword corpus. Table 11 shows the results of using higher-order

[9] Some matches appear to contain *no* words in common, but it is important to remember that matching is done on the source side. It should also be noted that only the single best matches for each sentence are shown here; the other 99 matches may contain complementary information that makes the aggregate match corpus more useful.

corpus	dev1 wt
Agence France-Presse, English Service (afp)	0.120447
Associated Press Worldstream, English Service (apw)	0.060999
Central News Agency of Taiwan, English Service (cna)	0.080731
Los Angeles Times/Washington Post Newswire Service (ltw)	0.129831
New York Times Newswire Service (nyt)	0.173202
Xinhua News Agency, English Service (xin)	0.434791

Table 9: Mixture LM weights for Gigaword corpus components.

configuration	nist06	nist08
baseline	**33.12**	**26.65**
giga mix	32.54	26.03
giga mix + global	32.69	26.31

Table 10: Results for Gigaword mixture adaptation. The first line is identical to line 7 in table 3. The *+global* line includes the global Gigaword LM as one of the mixture components.

ngrams for each of these models (holding log-linear weights fixed as usual). The results are mildly positive, but almost certainly not significantly so. This is similar to our findings from previous years. Given that the higher order models don't hurt, we prefer to use them on the off chance that they will match long sequences from the evaluation corpus.

configuration	nist06	nist08
1. *mix4 + giga5	33.08	26.67
2. *mix4 + giga6	**33.24**	26.63
3. *mix5 + giga5	33.06	**26.75**
4. *mix5 + giga6	33.15	26.70
5. *lemur 100 / 0.15	**33.20**	26.98
6. *lemur 100 / 0.15 mix5 + giga6	33.17	**26.99**

Table 11: Results for higher-order ngram models. The 1st line is equivalent to line 7 in table 3, and the 5th is taken from table 6.

6 Translation and Distortion Models

Experiments related to translation models involved new HMM models, a new approach to the "phrase holes" problem, an improved phrase-extraction algorithm, and the addition of lexicalized distortion and phrase-association features.

6.1 New HMM Models

We discovered late in 2008 that there was a conceptual problem with our implementation of an "end distribution" in HMM alignment models. Recall that the generative process for HMM alignment handles each word in the "observed" sequence in succession, linking it to either a word in the "hidden" sequence or a special null word.[10] The probability of the current link in this process depends on the distance between its hidden-sequence word and the hidden-sequence word from the previous link (as well as the identities of the words being linked, and, for the lexicalized model, the identity of the hidden-sequence word from the previous link). Clearly, the probabilities associated with different distances (or jumps) will vary as the process moves along the observed sequence: negative jumps, for example, are impossible at the very beginning of the sequence, become more probable through the middle, and are likely less prevalent again toward the end. To capture these tendencies, we optionally use two special jump distributions: a *start* distribution for the first word of the observed sequence, and a *final* distribution for its last word. Jumps at other positions are governed by a single intermediate distribution. In our previous implementation, the final distribution was not conditioned on observed-sequence position but rather on the jump distance itself and the fact that the jump landed on the final anchor in the hidden sequence. This caused normalization to be done in an incoherent fashion.

As part of the fix to this implementation, we also added the possibility to condition the current link on its hidden-sequence word or its observed-sequence word, and made some other conditioning choices orthogonal. The conditioning configurations for HMMs can now be summarized as:

- use special start jump distribution or not
- use special final jump distribution (correctly conditioned on observed-sequence position) or not

[10] Alignment is performed in both directions, so the source and target languages both play the observed and hidden roles, depending on direction.

model	code	lex conditioning	nist06	nist08
mix+giga+lex	old	prev hidden	32.15	**25.79**
mix+giga+lex	new	prev hidden	31.98	25.64
mix+giga+lex	new	prev observed	31.80	25.43
mix+giga+lex	new	curr observed	**32.18**	25.73
mix+giga+lex+adapt tm	old	prev hidden	**33.12**	26.65
mix+giga+lex+adapt tm	new	prev hidden	33.04	**26.71**
mix+giga+lex+adapt tm	new	prev observed	32.61	26.29
mix+giga+lex+adapt tm	new	curr observed	32.93	26.33

Table 12: Results for new HMM options. The configuration above the line is equivalent to line 5 in table 3; the configuration below is equivalent to line 7. All experiments used MAP smoothing for lexical conditioning, with -map-tau 100 and -lex-prune-ratio 0.2; additionally, the -max-jump parameter was set to 30. Default values were used for all other HMM parameters.

- use end anchor (special symbol appended to both hidden and observed sequences, intended to encourage end punctuation to line up) or not
- lexical or lexical-class conditioning of jump probabilities on one of: previous hidden-sequence word, previous observed-sequence word, or current observed-sequence word

(There are many other options for HMMs that can be listed using `train_ibm -h`. Note that the -newhmm option is required to use the corrected final jump distribution.)

As the new code runs considerably slower than the previous HMM code, we tested only a limited number of options, shown in table 12. All tested options included the start and final distributions; we varied only the lexical conditioning, and did not test the end anchor. The *prev hidden* lines in table 12 are direct comparisons between the old and new code, and show that the new code produces slightly (≈ 0.1) lower BLEU scores for most experiments. One explanation for this may be that the new code chops up sentences that are longer than 200 words into successive chunks for faster alignment. Conditioning on the observed-sequence words (previous and current) does not seem to yield systematic benefit. However, it was discovered after the evaluation that there was a bug with these variants, so these experiments need to be re-run. $\bigoplus$ Given the slightly poorer results with the new implementation of HMMs, we decided to use the old one for subsequent

experiments.[11]

Another system change motivated by the new HMM implementation was the stabilization of `gen-jpts-parallel.sh`, the program that allows phrase extraction to be run in parallel for large corpora. Essentially this breaks large parallel files into smaller pieces, extracts joint phrase tables (jpts) from the pieces in parallel, then combines the small jpts by adding joint counts for all phrase pairs. This is typically used with an option to add IBM1 translations for words that occur in the parallel file but have no translations in the extracted phrase table. Unfortunately, the "parallel files" for this operation were taken to be the pieces of the original file, so the set of added pairs depended on how many pieces there were. In the past, we "stabilized" this procedure by never changing the number of pieces, but with slower HMM alignment we needed to boost the parallelism, and hence implemented a fix to make the added pairs depend only on the original file. A side effect is a slight change compared to the basic algorithm currently implemented by `gen_phrase_tables`, resulting in a slightly improved set of added pairs. This change was introduced after the new HMM code was installed, and its effects have not yet been measured, even though it was used in subsequent experiments for the evaluation.⊕

6.2 Estimating Probabilities for Phrase Table Combination

It can be advantageous to distinguish phrase pairs arising from different sources, such as different parts of the training corpus, alternate phrase extraction techniques, bilingual lexicons, etc, by placing them in separate phrase tables and learning a log-linear weight for each table. This creates the problem of assigning probabilities to the "holes" that occur when one phrase table doesn't contain phrase pairs that occur in some other table(s). Canoe deals with this by assigning a small, fixed, epsilon value. This solution is not perfect, however, because it creates a bias against smaller high precision tables, which will be assigned a low weight to compensate for their tendency to consider many valid phrase pairs highly improbable.

We implemented a better solution in the program `joint2multi_cpt` (henceforth joint2multi), a partial replacement for `joint2cond_phrase_tables` (henceforth joint2cond). Joint2cond produces a conditional phrase table (cpt) from an input joint frequency phrase table (jpt) by applying selected smoothing techniques. If a set of distinct jpts are available, then each is run

[11] More accurately, we used HMM models that had been trained with the old implementation, but used the new implementation for subsequent alignment and lexicalized-distortion experiments described in sections 6.3 and 6.4.

separately through joint2cond, resulting in a set of cpts that will be used with canoe.[12] Joint2multi, on the other hand, produces a single, multicolumn, cpt from a set of input jpts. Since it has all of the input tables on hand when making probability estimates, it can assign more sensible values to holes than canoe's fixed epsilon strategy.

For NIST09, we tried only a very simple method for combining the evidence from multiple jpts in order to fill holes. Assuming that conditional probabilities for target given source language are desired (the other direction works the same way), we distinguish three cases:

1. the phrase pair occurs in the current table: make (smoothed) relative-frequency estimate as usual

2. only the source phrase occurs in the current table: assign a small (non-constant) probability

3. the source phrase does not occur in the current table: assign uniform probability over all translations of the source phrase found in any table

Motivated by theoretical arguments in favour of MAP smoothing (see appendix A), we used a simple add-alpha smoothing scheme for relative-frequency (RF) estimates (used for both case 1 and 2 above):

$$p(t|s) = \frac{f(s,t) + \alpha}{\sum_{t:F(s,t)>0} f(s,t) + \alpha} = \frac{f(s,t) + \alpha}{f(s) + \alpha N(s,*)}$$

where (s,t) is a phrase pair, $p(t|s)$ is the smoothed RF estimate in the current table, $f(s,t)$ and $f(s)$ are frequencies in the current table, $F(s,t)$ is a frequency across all tables, and $N(s,*)$ is the number of translations of s across all tables. This formula is used for cases 1 and 2 above. For case 3, $p(t|s) = 1/N(s,*)$.

There is a potential problem for case 2: as the number of known translations increases (assuming a fixed source-phrase frequency), the probability assigned to an unseen translation within the current table *decreases*. This seems opposite to the intuition captured in the widely-used Kneser-Ney smoothing scheme for ngram LMs [12], and may well be sub-optimal in the current context. $\bigoplus$

[12] It is also possible to create a single cpt from many input jpts with joint2cond. However, this is exactly equivalent to summing frequencies across the jpts to produce a single combined jpt, then using that as input. In this case, splitting into separate jpts is only done in order to parallelize the expensive phrase extraction step.

configuration			nist06	nist08
split	comb method	α		
** none	–	–	31.80	25.54
** model	canoe	–	31.78	25.75
** model	multi	0.00	31.73	25.36
** model	multi	0.01	31.70	25.63
** model	multi	0.10	31.28	25.46
** corp	canoe	–	31.90	25.64
** corp	multi	0.00	**32.51**	**26.02**
** corp	multi	0.01	31.68	25.36
** corp	multi	0.10	31.71	25.51
** both	canoe	–	32.09	26.01
** both	multi	0.00	32.14	25.59
** both	multi	0.01	32.30	25.77
** both	multi	0.10	32.02	25.34
** none	–	–	32.33	25.98
** model	canoe	–	32.40	25.99
** model	multi	0.0	32.10	26.14
** corp	canoe	–	32.42	26.02
** corp	multi	0.0	**32.78**	**26.15**
** both	canoe	–	31.80	25.62
** both	multi	0.0	32.29	25.64

Table 13: Different methods for combining phrase tables. The *split* column describes the phrase tables to be combined: *none* means a single large table (no combination necessary); *model* means one IBM2 phrase table and one HMM phrase table; *corp* means one table trained on the "good" part of the corpus, one on the less good part; and *both* means tables split by both model and corpus (4 tables in all). The *comb method* column describes how the tables are combined: either by creating separate cpts for canoe, or creating a single multi-column cpt (the new method described in this section). Results in the top part of the table pertain to a configuration that is equivalent to line 5 in table 3; results in the bottom part pertain to a configuration equivalent to line 7 in table 3.

The results of experiments with different combination schemes are shown in table 13. In general, the new joint2multi technique (*multi* lines in the table) does better than the old joint2cond one (*canoe* lines), though the differences are quite small and are not systematic. The optimum value for α is 0 (assigning 0 probability to holes where the source phrase appears in the current table), which may be related to the point in the previous paragraph. In contrast to findings from previous years, splitting by corpus alone (instead of by model and corpus) seems to give slightly better results, particularly for the stronger configuration in the bottom part of the table. For this configuration, the best gain over the baseline (obtained by the joint2multi corpus split) was only 0.13 for the valuable nist08 test set, which we deemed not compelling enough to justify the inconvenience of splitting the phrase table in future nist09 configurations.

6.3 The Insect

In comparisons to phrase tables generated by Moses from the FBIS corpus, we noticed that PORTAGE's phrase tables contained only about one third as many entries. Further investigation revealed that PORTAGE's phrase-extraction algorithm was subtly different from Moses' in that it permitted phrase pairs in which none of the words was linked to any others. For instance, in the sentence pair:

> The Newfoundland Government came up with **a** number **of** suggestions in **the** Green Paper .
>
> Le gouvernement **de** Terre-Neuve a présenté diverses solutions dans **son** Livre vert .

suppose the words in bold are not aligned to any others. Then PORTAGE would extract the phrase pairs a/de, a/son, of/de, of/son, etc, while Moses would not. We call this behaviour, which has been part of PORTAGE since its inception, the "insect".[13] We speculate that its presence has led us to tune our symmetrized word alignment algorithms in such a way as to minimize the number of unaligned words in order to avoid polluting the phrase tables with large numbers of noisy entries such as the ones above.

Removing the insect (by requiring extracted phrase pairs to contain at least one link) permits the use of sparser and more natural word alignments that lead to larger phrase tables with less noise. Unfortunately, it also necessitates re-optimizing our word-alignment strategy to take advantage of

[13] Not quite a bug, but something related.

configuration	num phrases	nist06	nist08
1. mix+giga+lex p0=0.0	93m	32.15	25.79
2. mix+giga+lex p0=0.5	199m	32.55	26.10
3. mix+giga+lex p0=0.6	232m	**32.60**	**26.14**
4. mix+giga+lex+adapt p0=0.0	93m	33.12	26.65
5. mix+giga+lex+adapt p0=0.5	199m	**33.26**	27.05
6. mix+giga+lex+adapt p0=0.6	232m	33.19	**27.16**

Table 14: Effects of sparser word alignments obtained by varying `p0` during word alignment. The configuration above the line is equivalent to line 5 in table 3; the configuration below is equivalent to line 7.

this. For lack of time, we adusted only one word-alignment parameter: the `p0` component of the jump-to-null probability for HMM-based alignments. As its name implies, this affects the probability that the current observed word will align to the null word in the HMM generative process. (Whether or not the word is actually left unaligned in the final symmetrized alignment depends also on many other factors, such as lexical associations, alignment monotonicity, etc.) Note that, for speed and convenience, we adjusted this parameter only during HMM alignment; for HMM training, we retained its default value of 0.0.[14] We also used the same value in both alignment directions, despite some evidence on GALE data that it can be beneficial to tune different values.

The results in table 14 show that this tuning leads to much larger phrase tables and significant increases in BLEU score for most configurations. The highest `p0` value we tested (0.6) gave the best results: an average gain of 0.35 BLEU over the baseline, with a 250% increase in phrase table size. More investigation is required to determine the optimum parameters for HMM/IBM2 training and alignment, and for symmetrized alignment. ⨁

6.4 Lexicalized Distortion

Having a distortion model that takes lexical context into account has been shown to be important for language pairs like Chinese-English that involve substantial reordering [24]. We implemented a lexicalized distortion model similar to Moses' that characterizes the position of the current source phrase relative to the previous one as either *monotonic*, *swapped* or *discontinuous*, according as it immediately follows, immediately precedes, or occurs

[14]The complete jump-to-null probability is `p0 + up0/(I+1)`, where I is the length of the observed sequence and we set `up0` to 1.

anywhere else. (One obvious enhancement is to split discontinuous into discontinuous-following and discontinuous-preceding. $\bigoplus$) The probability of these three orientations is conditioned on both the previous and the current phrase pairs, giving two different estimates:

- $p(o|pp)$ - the probability that the previous phrase pair pp has orientation o with respect to the following pair; and
- $p(o|cp)$ - the probability that the current phrase pair cp has orientation o with respect to the previous pair.

where o ranges over the values m, s, and d for monotonic, swapped, and discontinuous orientations.

All of this information (previous pair, current pair, and orientation) is available when placing the current phrase pair during normal decoding, so it is straightforward to incorporate it into decoder features. We implemented two different feature schemes:

- A 2-feature scheme with values $\log p(o|pp)$ and $\log p(o|cp)$ as above.
- A 6-feature scheme with values $\delta(o, o') \log p(o'|pp)$ and $\delta(o, o') \log p(o'|cp)$ for $o' \in \{m, s, d\}$. In this scheme, only the two features corresponding to the current orientation are non-zero in any given context.

Estimating $p(o|pp)$ and $p(o|cp)$ is a two-step procedure: first count the number of times each phrase pair occurs in all possible orientations, then estimate probabilities from the counts. To establish orientations, we use a version of the Moses technique illustrated in figure 1. This is applied during standard phrase extraction, and relies on the alignment of the *words* that immediately precede and follow the current target phrase as approximations for alignments of the preceding and following *phrases*, which are usually ambiguous. To make the approximation slightly more accurate, we imposed some additional partition conditions: the m link in figure 1 is taken to indicate monotonicity only if it is the *earliest* link for *next*, the s link indicates swapping only if it is the *latest* link for *next*; and similar conditions for *prev*. Also, we counted orientations at the beginning and ends of sentence pairs, assuming aligned pairs of dummy words in those positions.

Estimating conditional phrase-pair probabilities is a problem that benefits from smoothing [8], so clearly estimating distortion probabilities—with sparser data due to fragmentation of original phrase-pair counts into three different categories—should also benefit. We implemented a MAP-based

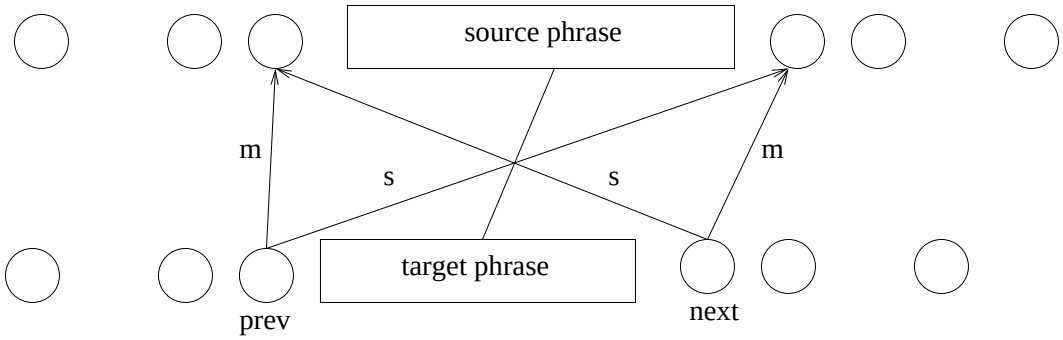

Figure 1: Assigning orientations for lexicalized distortion. The current phrase pair is deemed to have a monotonic orientation with respect to the following one if the m link exists for word *next*, a swapped orientation if the s link exists for *next*, or a discontinuous orientation if neither link exists. Orientation with respect to the previous pair is derived from *prev* in a similar way.

smoothing scheme (see appendix A) that uses a hierarchy of prior distributions. First, a global distribution is smoothed using a uniform distribution:

$$p_g(o|\cdot) = \frac{f(o|\cdot) + \alpha_u p_u(o|\cdot)}{f(\cdot) + \alpha_u},$$

where $f()$ is frequency, and "$\cdot$" matches any context. Next, source- and target-dependent distributions are smoothed using the global distribution:

$$p_s(o|s) = \frac{f(o|s) + \alpha_g p_g(o|\cdot)}{f(s) + \alpha_g} \quad \text{and} \quad p_t(o|t) = \frac{f(o|t) + \alpha_g p_g(o|\cdot)}{f(t) + \alpha_g},$$

and the final distribution is smoothed using both of these:

$$p(o|s,t) = \frac{f(o|s,t) + \alpha_s p_s(o|s) + \alpha_t p_t(o|t)}{f(s,t) + \alpha_s + \alpha_t}.$$

The four hyperparameters α_u, α_g, α_s, and α_t are shared by the previous and current models, $p(o|pp)$ and $p(o|cp)$. To tune these values, we minimized the perplexity of $p(o|pp)p(o|cp)$ on counts derived from the dev1 corpus. A rough grid search yielded an optimal value of 10 for all four parameters.

Results from using lexicalized distortion with a strong baseline model are shown in table 15. The MERT column is included to show the large variation in BLEU scores due to different MERT procedures. Since the two sets of baseline weights give conflicting results, we average them to get

configuration	MERT	nist06	nist08
1. baseline	old wts	33.16	27.57
2. baseline	std	33.42	26.96
3. lex-dist 2	std	33.67	27.15
4. lex-dist 6	std	34.72	27.83
5. lex-dist 6	hand	34.70	**28.71**
6. lex-dist 6	lattice	**35.06**	28.31

Table 15: Lexicalized distortion results. The baseline includes dynamic mixture LMs (section 5.1), higher-order LMs (section 5.3), larger phrase tables with the insect fix (section 6.3), length-dependent phrase table filtering (section 7.2), and other expanded search options (section 7.2). In the MERT column, *old wts* indicates weights trained on a previous configuration, *std* indicates the standard MERT procedure, *hand* indicates semi-automatic optimization, and *lattice* is lattice MERT (section 7.1).

33.29 for nist06 and 27.27 for nist08. Then 2-feature lexicalized distortion gives gains of 0.38 for nist06 and -0.12 for nist08 over these averages. The 6-feature version gives gains of 1.43 and 0.56 with standard MERT, and is clearly preferable. Using weights from lattice MERT gives highly significant gains of 1.77 and 1.04 over the baseline average.

Future experiments will test the contribution of lexicalized distortion with simpler baseline systems, as well as the effect of the MAP smoothing scheme. $\bigoplus$

6.5 Additional Features

We had plans to try out significance features [11], which were beneficial in the GALE 2008 system, but ran out of time for this.

We also had limited time to try out the phrase-association features described in [5], and experimented only with the top-performing log-likelihood-ratio measure. Unfortunately, this was computed on a phrase table without length-dependent filtering, and so could not be included in the final system, which benefitted from this filtering.

A final additional feature was a simple phrase count, obtained by adding a constant-value column to the phrase table. This was included in the final system, but its results are confounded with other changes to the system, so we have no hard evidence yet that it would be useful on its own. $\bigoplus$

7 Decoding

The major decoding-related strategy implemented for the evaluation was lattice-based MERT. We also experimented with some new variants on search parameters that proved effective.

7.1 Lattice MERT

We implemented the lattice-based MERT procedure described in [13] as an alternative to our standard nbest-list based MERT, which often fails to find a good set of weights, especially for large feature sets [7]. The basic idea in lattice MERT is to replace the nbest list within Och's algorithm [15] with a lattice, which contains many orders of magnitude more hypotheses and therefore offers a better approximation to the true search space (ie, the space of *all* hypotheses) for weight optimization. The key conceptual change this entails is adapting Och's line maximization algorithm to work with sets of lattices rather than sets of nbest lists. Another change in our implementation is that we do not currently merge lattices across successive iterations of Och's procedure as is done with nbest lists. There is therefore no guarantee that the algorithm won't revisit previous "bad" weights (ones where the approximate search space diverges harmfully from the true space), but this is much less likely to happen with lattices, and does not seem to be a significant problem in practice, as lattice MERT tends to avoid such bad weights in the first place.

For many of the experiments described in this report, we held log-linear weights fixed at an optimum that was the best out of a large number of nbest MERT runs with similar configurations. Lattice MERT was unable to do better than this highly evolved optimum in most settings, especially on the critical nist08 test set. However, as shown in table 16, when compared on an equal footing with nbest MERT, the lattice-based approach won convincingly, achieving higher BLEU scores in 12 out of 15 measurements. This trend was particularly marked for configurations with larger numbers of features (where the "hand"-tuned weights were more problematic); in the strongest configuration at the bottom of table 16, lattice MERT did spectacularly well, outperforming nbest by approximately 1 BLEU point on all three devtest corpora. A final observation is that the lattice approach requires only half as many iterations as nbest MERT, so despite somewhat longer per-iteration times with the current implementation, it tended to finish faster. Several of the runs in the table used 10 iterations, but these typically achieved their best results on or before the 7th iteration, which

MERT	num iters	num features	dev1	nist06	nist08
nbest	15	8	**25.60**	32.68	26.00
lattice	6	8	25.52	32.45	26.13
hand	-	8	–	**33.12**	**26.65**
lattice	10	8	25.81	**33.42**	26.96
hand	-	8	–	33.16	**27.57**
nbest	15	14	**27.04**	34.35	27.71
lattice	7	14	26.89	**34.43**	**27.83**
nbest	15	14	27.16	34.72	27.83
lattice	10	14	**27.23**	**35.06**	28.31
hand	-	14	–	34.70	**28.71**
nbest	15	15	26.32	33.80	27.27
lattice	7	15	**26.71**	**34.32**	**27.75**
hand+nbest	15	15	26.52	34.72	27.77
hand+lattice	7	15	**27.57**	**35.70**	**28.60**

Table 16: Results for lattice MERT, compared to nbest MERT, *hand* MERT (weights borrowed judiciously from other configurations), and *hand+* MERT (weights initialized by hand). Each set of results between horizontal lines varies only in the MERT technique used. The first set is the configuration in line 7 of table 3. All other sets use dynamic mixture LMs (section 5.1), higher-order LMs (section 5.3), and larger phrase tables with the insect fix (section 6.3), but vary in other parameters. For the sake of the reader's sanity, we suppress further details and list only the number of features.

seems like a reasonable stopping point.

7.2 Extended Search Options

Toward the very end of preparations for the evaluation, we experimented with various alternative search strategies:

Future scores for forward probabilities

Future score calculations were not implemented for optional "forward" (target given source) phrase probability features when they were originally added to PORTAGE. This was rectified early in 2009 as part of the work for adding "adirectional" (4th column) phrase table features.

Length-based hard phrase table filtering

To speed up search, the phrase table is typically filtered as it is read in by `canoe`, with only the top 30 candidate translations for each source phrase retained.[15] Candidates are ranked for this filtering by applying current weights to forward phrase probabilities (typically the corresponding *backward* weights are used for this, because these seem to be more reliable than the forward weights, which would be a more natural choice). This strategy causes problems for MERT because the phrases available to the decoder change as weights fluctuate across different iterations. Hypotheses added to the nbest lists on previous iterations—and *preferred* by the current weights—may actually be inaccessible to the decoder under these weights, due to filtering.[16] This can cause MERT to converge to a set of weights which give good results on the nbest lists, but poor results with the decoder.

To avoid this undesirable behaviour, we experimented with hard phrase table filtering: fixing the set of translations for each source phrase prior to MERT and subsequent decoding. Motivated by an observation in [2] that larger sets of translations yield dramatically improved coverage of devtest corpora, we were interested in setting the hard threshold to a value greater than 30. Since this can slow translation substantially, we tried allowing more translations only for longer phrases, which in general have more real translations than shorter ones, and are less prone to having long "tails" of noisy translations in the phrase table. We did this by setting the threshold to be $30l$, where l is the number of words in the source phrase. One remaining question is how to set the weights on different phrase tables for hard filtering. This was accomplished by using previous "good" weights, which was an effective but not very automatic procedure. Further work is needed to determine how best to automate this step, as well as to optimize the base threshold of 30, and the functional dependence on length (eg, trying out other schemes besides a simple linear one). $\bigoplus$

[15] More precisely, this filtering happens only before the translation of each source sentence, when a phrase data structure is constructed for that sentence. The distinction is important for speed, because the language model is filtered only once based on the target phrases that are retained when the phrase table is first read in—ie, not those that are used for each individual sentence.

[16] This seems strange, but it happens quite frequently, especially when phrase table weights are negative. In this situation, the hypotheses produced by the decoder with current weights are poor, and the presence of the language model causes the good hypotheses on the nbest list to be ranked higher, even though the phrase scores for those hypotheses are lower.

Expanded-distortion options

To give the lexicalized distortion model more scope to contribute, we experimented with three previously-implemented options that encourage the search procedure to consider hypotheses with more reordering:

- increasing the distortion limit from the standard 7 words to 8;
- always allowing any two contiguous source phrases to be swapped (ie to be aligned in reverse order to two consecutive target phrases) even if the resulting hypothesis violates the current distortion limit; and
- relaxing the interpretation of the distortion limit somewhat to allow more partial hypotheses to be considered (using the `-dist-limit-ext` option to `canoe`)[17]

These options were always used together; we did not consider each on its own. ⨁

Results

Due to the short time available to test the extended search options, we ran only one comparative test for each, as shown in table 17. Results are systematically positive, but the gains are small and not necessarily significant, except for expanded-distortion search, where they are unequivocal (there may have been a slight bias in this case because the weights were tuned for the expanded search rather than the baseline, but this clearly does not account for all of the gain).[18]

For all options, there is slightly more evidence than presented in this table that they help, but it is derived from combining them with other parameters. More experiments are needed in order to provide a clearer picture. ⨁

[17] Let the source coverage of the previous phrase be $[x, y)$, the first non-covered source position be $NCW1$, and the source coverage of next phrase be $[a, b)$. Then `-no-dist-limit-ext` is respected iff $b \leq NCW1 + L$ and $|a - y| \leq L$. `-dist-limit-ext` is respected iff $a \leq NCW1 + L$ and $|a - y| \leq L$.

[18] In an original attempt at running the baseline, we made the interesting mistake of omitting a distortion limit. Even with a lexicalized distortion model, this is *not* a recommended strategy: it caused a drop of around 0.5 BLEU, and required 40G of memory to complete successfully (compared to 8G for the baseline)!

configuration	search option	nist06	nist08
dm6+lbf+ext	no ftm	34.70	28.71
dm6+lbf+ext	ftm	**34.84**	**28.74**
ext	no lbf	33.15	27.45
ext	lbf	**33.16**	**27.57**
dm6+lbf+lmert	no ext	35.00	28.33
dm6+lbf+lmert	ext	**35.70**	**28.60**

Table 17: Results for various extended-search options: *ftm* - future scores with forward phrase probabilities; *lbf* - length-based hard filtering; and *ext* - expanded-distortion search. These are evaluated in each of the pairs of results between horizontal lines. The baseline configuration for all tests is as described in 16, with the additional features noted in the *configuration* column: *dm6* - 6-feature distortion model; and *lmert* - lattice MERT.

8 Rescoring

After decoding, we rescore hypotheses using additional models, as usual. This year we used 1,000-best lists for convenience (in the past we have sometimes used 5,000-best lists). We implemented some new features and used a new feature-selection algorithm that gave significant gains over our best single-pass output.

8.1 Rescoring Features

The models used for rescoring were:

1. The models used in the decoder: 2 language model features, 4 translation model features, 6 distortion model features, and word/phrase penalties. (14 features)

2. Forward and backward lexicon models: IBM1, IBM2, HMM, HMM Viterbi-alignment lexicon scores. (8 features)

3. Features that check that all words are translated, and that no words are inserted, respectively, using IBM1 lexicons. (2 features)

4. IBM1 deletion models [14]. These count all source words whose lexical probability given each target word is below a threshold. (2 features)

5. Posterior probabilities for words (based on both source and target position), phrases (based on both source and target position), n-grams,

configuration	nist06	nist08
baseline	35.70	28.60
+all features	36.01	28.97
+hand-selected features	36.10	28.91
+all features + greedy pruning	36.22	**29.21**
+ hand-selected features + greedy selection	**36.35**	29.11

Table 18: Rescoring results

sentence length [22, 17], and word reordering, all calculated over the nbest lists. (7 features)

6. Numbers of mismatched parentheses and mismatched quotes, within the hypothesis. (2 features)

7. Cache LM over docs defined in docids file. (1 feature)

8. Length-related features: number of characters and ratio of words in the translation over words in the source. (2 features)

9. Ngram frequency [4], sum of n-grams (n=1,2,3,4) frequencies in the N-best lists. (1 feature)

Of the above 25 rescoring features, the word reordering posterior and ngram frequency features are new in PORTAGE.

8.2 Feature Selection

Since some of the features are partially overlapping and highly correlated to each other, we applied a greedy algorithm to select a subset of the features. Two strategies are exploited: a) start from the set of all features, then prune; b) start from a hand-selected set of features (decoder features, forward and backward IBM1 and HMM lexicon features, n-gram and sentence length posterior, n-gram frequency, etc), then expand.

Table 18 shows the results of rescoring the best single-pass configuration (the one from the last line of tables 17 and 16). The two base feature sets (*all* and *hand selected*) both give approximately a 0.3 BLEU gain over the baseline, and the greedy strategies increase this by 0.2-0.3 BLEU. For the evaluation, we used the *all + pruning* method, which does slightly better on the indicative nist08 test set.

9 Truecasing

Work on the final step in the processing chain comprised a comparison between the "naive George" approach and the standard HMM-based truecasing algorithm, experimentation with variants of HMM-based truecasing, and a new title capitalization heuristic.

9.1 Naive George versus HMM

Our standard approach to truecasing is to treat it as a separate postprocessing step that aims to restore normal case conventions to lowercase MT output, independent of the source text. We model the problem using an HMM in which the observed sequence is the MT output and the hidden sequence is the desired truecase version [1]. Hidden-sequence probabilities come from an ngram LM trained on normally-cased text, and output probabilities are p(cased-form|lowercase-form), the relative frequency of cased-form among all forms that map to the given lowercase form in the corpus.[19]

Recent results had indicated that a "naive George" approach gave better truecasing results than the HMM-based approach. For the record, the original, "non-naive George" approach consists of:

1. Train IBM/HMM models on lowercase text as usual.

2. Create an "nc1" version of the training corpus in which sentence-initial capitalization is undone for words which would not be capitalized on their own. This is accomplished by comparing the probability of the first word given the following n-1-gram, $p(w_1|w_2, ..., w_n)$, with the probability of its lowercase version, $p(\text{lc}(w_1)||w_2, ..., w_n)$, and choosing the most likely. Here $p(w|\cdot)$ is estimated over the whole corpus (all sentence positions, not just the first) using standard language modeling techniques.

3. Perform phrase-extraction over the nc1 parallel corpus, mapping words to their lowercase equivalents in order to match the IBM/HMM models for word alignment, but extracting truecase phrases.

4. Decode with the truecase phrase table, using a linear combination of a lowercase and an nc1 LM. Words are dynamically mapped to lowercase before calculating probabilities with the former.

[19] This probability technically should be p(lowercase-form|cased-form), since the HMM's states are cased forms, but because these correct probabilities are always 1, it seems more satisfying to use the incorrect version. The two approaches give similar results.

configuration	nist06	nist08
**baseline	**31.42**	**25.67**
**naive George, baseline weights	30.98	24.66
**naive George, MERT weights	30.25	24.48

Table 19: Results for baseline HMM truecasing compared with "naive George". The configuration for all tests is the same as line 5 in table 3.

5. Restore sentence-initial capitalization to final MT output.

The naive variant, as implemented here[20] consists of simply training and running on nc1 text (ie, without trying to use lowercase IBM/HMM models or LMs).

Table 19 shows a comparison between the standard approach and naive George. The former is so much better than we did not pursue the naive George approach further. It seems likely that using source information in some form will benefit truecasing, but this particular avenue does not look promising at the moment.

9.2 Tuning HMM Truecasing

To tune the HMM truecasing approach, we varied the training corpora used for HMM output probabilities (the map), and for the LM trained on naturally-cased text. We also varied the order for the LM. As shown in table 20, training the map on the parallel corpus and a 4gram LM on the Gigaword corpus gives a slight advantage (averaged over both test corpora) over the baseline approach of training both components on the parallel corpus.

9.3 Title Capitalization

During semi-automatic error analysis of nist08 results, we realized that it would be beneficial to systematically capitalize content words in the titles of newswire articles. We originally relied on the genre tag in the nist data to detect newswire articles, but because of some uncertainty over whether this information was actually legal, we simply applied content-word capitalization to the first line of all documents in the test set, regardless of genre. This had no negative effect on performance, perhaps because many of the webtext documents begin with a title or some sort of capitalized identifier.

[20] Slightly less naive than the version used for previous tests in which the nc1 corpus was produced by systematically lowercasing sentence-initial letters.

map	lm	nist06	nist08
–	–	34.70	28.71
parallel	parallel 4g	32.57	**26.60**
parallel	giga 3g	32.79	26.59
parallel	giga 4g	**32.86**	26.57
parallel	giga 5g	32.75	26.54
parallel	giga 6g	32.84	26.50
parallel small	giga 4g	32.83	26.55
giga	giga 4g	32.71	26.45

Table 20: Different training conditions for HMM-based truecasing. Translations are produced by the system in the first line of table 17 (lowercase BLEU scores are given in the first line here for comparison). The *map* column describes the lowercase/truecase map: *parallel* denotes training on the English half of the parallel corpus; *parallel small* the English parallel corpus minus the UN, Hong Kong Laws, and Hong Kong Hansard; and *giga* the Gigaword corpus. The *lm* column describes the LM: *parallel* denotes training on the parallel corpus, and *giga* on the Gigaword.

method	nist06	nist08
baseline	32.86	26.57
+ title cap	33.01	27.08

Table 21: Effect of title capitalization.

As shown in table 21, this strategy results in a very significant gain over the baseline truecasing approach.

10 Conclusions

To conclude, we first briefly summarize the techniques that were included in this year's system (and not in last year's), along with their approximate BLEU gain (non-cumulative, and typically averaged over several different baselines and test corpora):

- better preprocessing (especially of FBIS): 0.5 (estimated, because results were confounded with corpus switch)
- IR-based relevant corpus as LM mixture component (with max-BLEU weighting): 0.2

- higher-order LMs (5g adapted; 6g Gigaword): 0.1
- lexicalized HMMs [9]: 0.3
- "adapted" lexical probs from HMM trained on in-domain parallel corpus: 0.5
- fixing the "insect": 0.4 (more optimization required)
- Moses-style lexicalized distortion with 6 features, MAP smoothing for rf estimates: 1.3
- lattice MERT [13]: better test results than nbest in 12/15 configurations
- expanded search, with future scores for forward probabilities, length-based phrase-option filtering, distortion limit of 8 (versus 7), relaxed distortion limit (`-dist-limit-ext`), and swapping always allowed: 0.5
- rescoring with IBM- and nbest-based features ([17, 4]: 0.3 BLEU
- greedy feature pruning for rescoring +0.3 BLEU
- truecasing with "title trick": +0.3 BLEU

Here are the techniques that we tested but did not include in this year's system, along with an explanation:

- long-sentence splitting [21]: +0.2 BLEU (positive results weren't known early enough)
- Gigaword mixture adaptation: -0.4 BLEU
- new HMM implementation (fixed end distribution and observed-word conditioning): -0.1 BLEU (was also considerably slower)
- linear phrase-table combination to avoid "holes": +0.3 (gain was very small on nist08 corpus; method somewhat more cumbersome)
- significance features [11]: no time
- phrase-association features [5]: no time
- passive truecasing (train on naturally-cased text): -1.0+

A Maximum A Posteriori (MAP) Estimation

Parameter estimation is a central problem in statistical NLP. Typically, given a model $p(x|\theta)$, we wish to set its parameters θ based on some training data $\mathcal{D}$. A common approach is maximum-likelihood (ML) estimation, which chooses the parameters that maximize the probability of the data according to the model:

$$\hat{\theta} = \operatorname*{argmax}_{\theta} p(\mathcal{D}|\theta),$$

assuming that $\mathcal{D}$ consists of a sequence of independent events x, say. A problem with ML estimation is that it tends to overfit $\mathcal{D}$, since it doesn't take into account any phenomena that don't occur in that data.

MAP estimation explicitly models the *posterior* distribution of the parameters given the data, which is a natural strategy since it is the data and not the parameters that have been observed. It chooses parameters that maximize this distribution:

$$\begin{aligned} \hat{\theta} &= \operatorname*{argmax}_{\theta} p(\theta|\mathcal{D}) \\ &= \operatorname*{argmax}_{\theta} p(\mathcal{D}|\theta)\, p(\theta), \end{aligned}$$

where the two factors on the second line are called the *likelihood* and *prior* distributions. MAP estimation can be seen as ML estimation weighted by the prior, which can be used to express knowledge such as that vocabulary words should not be assigned 0 probability even if they don't occur in a particular training text.

As an aside, MAP estimation is related to Bayesian inference, which does away with discrete parameter estimation by considering all parameter values at each point:

$$\begin{aligned} p(x|\mathcal{D}) &= \int p(x, \theta|\mathcal{D}) d\theta \\ &= \int p(x|\theta)\, p(\theta|\mathcal{D}) d\theta, \end{aligned}$$

where the second factor is the posterior probability that is maximized under MAP. The Bayesian approach has an appealing theoretical motivation (predicting new values x directly from previously observed data), but the resulting formula is usually impossible to calculate analytically, and hence relies on expensive sampling methods. See [3] for more details.

A.1 Using a Dirichlet Prior

In many NLP models for $p(x|\theta)$, x ranges over a set of discrete events such as words or the orientations in section 6.4, and the parameter vector θ contains event probabilities $\theta_i = p(x_i)$, $i = 1...N$. (Typically x is conditioned on a context such as an ngram or phrase pair, but we will leave this conditioning implicit for simplicity, since it doesn't change the development below.) The probability that this model assigns to a set of independent observations $\mathcal{D} = x_1...x_T$ is:

$$p(\mathcal{D}|\theta) = \prod_{j=1}^{T} p(x_j|\theta) = \prod_{i=1}^{N} \theta_i^{f_i}, \tag{1}$$

where f_i is the frequency of x_i in $\mathcal{D}$.[21] It is not hard to show that the maximum-likelihood estimates for this distribution are $\hat{\theta}_i = f_i/T$, $i = 1...N$.

When performing MAP estimation, it is convenient to choose a prior distribution that is *conjugate* to the likelihood function, so that the posterior distribution will have the same functional form as the prior. For the distribution above, the conjugate prior is the Dirichlet:

$$\mathrm{Dir}(\theta|\alpha) = \frac{\Gamma(\alpha_0)}{\Gamma(\alpha_1)\cdots\Gamma(\alpha_N)} \prod_{i=1}^{N} \theta_i^{(\alpha_i - 1)},$$

with parameters $\alpha_1...\alpha_N$, and $\alpha_0 = \sum_{i=1}^{N} \alpha_i$. This assigns a probability to any valid set of multinomial parameters θ, with the usual constraints that $\theta_i \in [0, 1], i = 1...N$, and $\sum_{i=1}^{N} \theta_i = 1$. θ thus ranges over a *simplex* with vertices at the points where each $\theta_i = 1$ and the other parameters are 0. The Dirichlet parameters α control the shape of this distribution, with $\alpha_i < 1.0$ favouring high values of θ_i, $\alpha_i = 1$ giving a uniform distribution, and $\alpha_i > 1$ favouring intermediate values.

Under a Dirichlet prior, the MAP equation becomes:

$$\hat{\theta} = \operatorname*{argmax}_{\theta} p(\mathcal{D}|\theta)\ \mathrm{Dir}(\theta|\alpha) = \operatorname*{argmax}_{\theta} \prod_{i=1}^{N} \theta_i^{(f_i + \alpha_i - 1)},$$

[21] This is very similar to the multinomial distribution, which models the probability that N events will occur with a particular set of frequencies in T trials, rather than the actual sequence of outcomes from the trials, as here. The multinomial therefore includes a factor $T!/\prod_i f_i!$ to count the number of sequences having the event frequencies $f_1...f_N$, each of which has probability $\prod_{i=1}^{N} \theta_i^{f_i}$.

omitting the Dirichlet normalization factor, which doesn't depend on θ. By similarity to (1), this has the solution:

$$\hat{\theta}_i = (f_i + \alpha_i - 1)/(T + \sum_{i=1}^{N} \alpha_i - N),\ i = 1...N.$$

Use of a Dirichlet prior for MAP is thus equivalent to augmenting the event counts in $\mathcal{D}$ by $\alpha_i - 1$, then performing ML estimation. It can be reformulated somewhat more conveniently as:

$$\hat{\theta}_i = (f_i + \alpha\ p_0(x_i))/(T + \alpha),$$

where $p_0(x)$ is a prior having the same form as the original likelihood distribution, and α is a global scalar hyperparameter that controls the relative contribution of the prior and the likelihood (abusing notation somewhat). This formula has the appealing property that it adjusts automatically to the amount of evidence available: large values of f_i will result in estimates that are close to ML, while for small values the prior will dominate.

References

[1] Akakpo Agbago, Roland Kuhn, and George Foster. Truecasing for the portage system. In *Recent Advances in Natural Language Processing (RANLP)*, Borovets, Bulgaria, 2005.

[2] Michael Auli, Adam Lopez, Hieu Hoang, and Philipp Koehn. A systematic analysis of translation model search spaces. In WMT09 [20], pages 224–232.

[3] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[4] B. Chen, R. Cattoni, N. Bertoldi, M. Cettolo, , and M. Federico. ITC-IRST SMT system for IWSLT-2005. In *IWSLT 2005*, Pittsburgh, 2005.

[5] Boxing Chen, George Foster, and Roland Kuhn. Phrase translation model enhanced with association based features. In MTS09 [10]. To appear.

[6] George Foster and Roland Kuhn. Mixture-model adaptation for SMT. In WMT07 [19].

[7] George Foster and Roland Kuhn. Stabilizing minimum error rate training. In WMT09 [20].

[8] George Foster, Roland Kuhn, and Howard Johnson. Phrasetable smoothing for statistical machine translation. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Sydney, Australia, 2006.

[9] Xiaodong He. Using word-dependent transition models in HMM based word alignment for statistical machine translation. In WMT07 [19].

[10] International Association for Machine Translation. *Proceedings of MT Summit XII*, Ottawa, Canada, September 2009.

[11] Howard Johnson, Joel Martin, George Foster, and Roland Kuhn. Improving translation quality by discarding most of the phrasetable. In *Proceedings of the 2007 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Prague, Czech Republic, 2007.

[12] Reinhard Kneser and Hermann Ney. Improved backing-off for m-gram language modeling. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP) 1995*, pages 181–184, Detroit, Michigan, 1995. IEEE.

[13] Wolfgang Macherey, Franz Josef Och, Ignacio Thayer, and Jakob Uszkoreit. Lattice-based minimum error rate training for statistical machine translation. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Honolulu, 2008.

[14] A. Mauser, R. Zens, E. Matusov, S. Hasan, and H. Ney. The RWTH statistical machine translation system for the IWSLT 2006 evaluation. In *IWSLT 2006*, Kyoto, Japan, 2006.

[15] Franz Josef Och. Minimum error rate training for statistical machine translation. In *Proceedings of the 41th Annual Meeting of the Association for Computational Linguistics (ACL)*, Sapporo, July 2003.

[16] Michel Simard and Pierre Isabelle. Phrase-based machine translation in a computer-assisted translation environment. In MTS09 [10]. To appear.

[17] Nicola Ueffing and Hermann Ney. Word-level confidence estimation for machine translation. *Computational Linguistics*, 33(1):9–40, 2007.

[18] WMT. *Proceedings of the NAACL Workshop on Statistical Machine Translation*, New York, June 2006.

[19] WMT. *Proceedings of the ACL Workshop on Statistical Machine Translation*, Prague, June 2007.

[20] WMT. *Proceedings of the 4th Workshop on Statistical Machine Translation*, Athens, March 2009.

[21] Jia Xu, Richard Zens, and Hermann Ney. Partitioning parallel documents using binary segmentation. In WMT06 [18].

[22] R. Zens and H. Ney. N-gram posterior probabilities for statistical machine translation. In WMT06 [18].

[23] Richard Zens and Hermann Ney. Improvements in phrase-based statistical machine translation. In *Proceedings of Human Language Technology Conference / North American Chapter of the ACL*, Boston, May 2004.

[24] Andreas Zollmann, Ashish Venugopal, Franz Och, and Jay Ponte. A systematic comparison of phrase-based, hierarchical and syntax-augmented statistical MT. In *Proceedings of the International Conference on Computational Linguistics (COLING) 2008*, Manchester, August 2008.