NIST Open Machine Translation 2008 Evaluation: Stanford University's System Description

Michel Galley, **Pi-Chuan Chang**, **Daniel Cer**, **Jenny R. Finkel**, and **Christopher D. Manning**
Computer Science and Linguistics Departments
Stanford University

1 Site affiliation

Stanford University

2 Contact information

Michel Galley
Dept. of Computer Science, Gates 2A
Stanford, CA 94305-9010
USA
mgalley@cs.stanford.edu

3 Submission

SU_chinese_constrained_primary

4 Primary system specifications

This document describes Stanford University's first entry into a NIST MT evaluation. Our entry to the 2008 evaluation mainly focused on establishing a competent baseline with a phrase-based system similar to (Och and Ney, 2004; Koehn et al., 2007). In a three-week effort prior to the evaluation, our attention focused on scaling up our system to exploit nearly all Chinese-English parallel data permissible under the constrained track, incorporating competitive language models into the decoder using Gigaword and Google *n*-grams, evaluating Chinese word segmentation models, and incorporating a document classifier as a pre-processing stage to the decoder.

4.1 Core MT engine algorithmic approach

Our translation system contains four main components: a phrase-based translation system, a Chinese word segmenter, a text categorizer, and a truecaser. The purpose of the text categorizer is to classify each test document as either newswire or web, which allows us to run our phrase-based decoder with a set of log-linear parameters tuned on the predicted genre. The three latter components are presented in Section 4.2.

4.1.1 Parallel data

Our translation models were trained using almost all of the constrained track Chinese-English data, leaving aside only the Multiple Translation Chinese corpus (LDC2006T04), OntoNotes (LDC2007T21), test data from previous NIST MT evaluations (LDC2006E43, LDC2006E38, LDC2007E59), and the ISI Chinese-English automatically extracted parallel texts (LDC2007T09). We only exploited a four-million word subset of the public version of the FBIS corpus (LDC2003E14). In total, the parallel training corpus contains 237.6 million English words and 215.4 million Chinese words.

Our DARPA GALE collaborators at IBM Research provided us pre-processed and sentence-aligned parallel data. We further removed sentence pairs deemed inappropriate for training, in particular sentence pairs with fertility larger than nine, sentences longer than 100 words, and sentences with errorful UTF-8 encoding. All words in both source and target language texts were downcased.

We used the test set of the 2006 evaluation (MT06, LDC2007E59) for parameter tuning, and the test set of 2005 (MT05, LDC2006E38) as our development test set.

4.1.2 Approach

Our phrase-based system employs a log-linear approach common to many state-of-the-art statistical machine translation (SMT) systems (Och and Ney, 2004). Given an input Chinese sentence **f**, which is to be translated into an English sentence **e**, the decoder searches for the most probable translation $\hat{\mathbf{e}}$ according to the following decision rule:

$$\hat{\mathbf{e}} = \arg\max_{\mathbf{e}}\{P(\mathbf{e}|\mathbf{f})\} = \arg\max_{\mathbf{e}}\{\sum_{m=1}^{M}\lambda_m h_m(\mathbf{f},\mathbf{e})\} \quad (1)$$

$h_m(\mathbf{f},\mathbf{e})$ are M arbitrary feature functions over sentence pairs, such as translation probabilities. The search is performed with the Moses decoder (Koehn et al., 2007).

While our general research direction is to apply feature functions based on deep linguistic analysis, the work for this submission concentrated on features yielding a competent baseline phrase-based SMT system, given the relatively short time at our disposal before the evaluation. We finally incorporated the following 15 feature functions:

- **Two phrase translation probabilities** $P_{ml}(\overline{e}|\overline{f})$ and $P_{ml}(\overline{f}|\overline{e})$, computed using the (unsmoothed) relative frequency estimate

  $$P_{ml}(\overline{e}|\overline{f}) = \text{count}(\overline{e},\overline{f})/\big(\sum_{\overline{e'}}\text{count}(\overline{e'},\overline{f})\big), \quad (2)$$

  where $\overline{f}$ and $\overline{e}$ constitute a pair of aligned phrases.

- **Two lexical translation probabilities** $P_{lex}(\overline{e}|\overline{f},\mathbf{a})$ and $P_{lex}(\overline{f}|\overline{e},\mathbf{a})$, similar to those presented in (Koehn et al., 2003):

  $$P_{lex}(\overline{e}|\overline{f},\mathbf{a}) = \prod_{i=1}^{n}\frac{1}{|\{i|(i,j)\in\mathbf{a}\}|}\sum_{(i,j)\in\mathbf{a}}p(\overline{e}_i|\overline{f}_j), \quad (3)$$

  where n is the length of the phrase $\overline{e}$, and **a** is the internal word alignment between $\overline{e}$ and $\overline{f}$.[1]

- **Six lexicalized phrase re-ordering probabilities**, which distinguish three types of reorderings (monotone, swap, and discontinuous) and model both left-to-right and right-to-left re-orderings, and thus define six features functions for each phrase pair. We applied Laplace smoothing to lexicalized re-ordering probabilities, with $\lambda = 0.5$.

- **Two language models**, from Gigaword and Google *n*-grams.

- **Word penalty** as in (Koehn et al., 2007).

- **Phrase penalty** as in (Koehn et al., 2007).

- **Linear reordering penalty** as defined in (Koehn et al., 2007).

The weights of these feature functions were set using minimum error rate training (MERT) (Och, 2003). We divided our tuning set (MT06) into two sets—newswire stories and newsgroup messages—and discarded all other genres (broadcast news). We ran MERT on each set, and used newswire MERT parameters for decoding documents classified as 'newswire', and used newsgroup MERT parameters for decoding documents classified as 'web'. In each case, we ran MERT twice: once with only one language model (Gigaword and parallel data), and second time with the addition of a Google language model (see Section 4.1.4). The effect of the Google language model is discussed in the results section (Section 4.5).

4.1.3 Phrase tables

This section describes the computation of phrase translation and lexicalized re-ordering probabilities, which we computed for all observed phrases no longer than seven words on each side. First, we ran GIZA++ (Och and Ney, 2003) to produce word alignments for the entire data set. We ran five iterations of IBM Model 1 (Brown et al., 1993), five iterations of the homogeneous HMM model described in (Vogel et al., 1996), and three iterations of IBM Model 4. Note that training with IBM Model 3 was entirely skipped.

We built our own implementation of phrase-extract (Och, 2002), which, as opposed to Moses, builds phrase tables directly tailored to specific development and test sets. This considerably reduces the burden of computing normalization counts, since

[1] Distinct instances of a given phrase pair $(\overline{e},\overline{f})$ may be observed with different internal alignments. Similarly to Moses, and in contrast to (Koehn et al., 2003), we select in such a case the most frequent alignment. Since our feature extraction implementation differs from the one of Moses in the way it breaks ties between alignment counts, about 0.3% of our phrases have lexical translation probabilities that differ from the ones computed by Moses, but this does not impact MT performance.

our phrase extraction system can generally fit all relevant phrase pairs into memory (as opposed to, e.g., Moses, which sorts large collections of phrases on disk to compute normalization counts). This enabled us to quickly experiment with many phrase extraction heuristics. On a 41 million English word subset of the parallel data, we found that the alignment symmetrization that worked best with our system is the grow-diag heuristic (Koehn et al., 2007). We pruned phrase tables produced with this heuristic by deleting all phrases that do not satisfy $P_{ml}(\overline{e}|\overline{f}) \geq .0001$. This filtering typically yields phrase tables 2 to 3 times smaller, with generally little impact on MT performance (0.2% BLEU reduction at worst). Since all our language models are filtered against the target side of our phrase tables, this deletion of very unlikely translations allowed us to considerably reduce *n*-gram count thresholds—i.e., the number of times each *n*-gram must be observed to be included in the language model—and to incidentally capitalize more on *n*-grams that are likely to be seen at decoding time.

4.1.4 Language models

Our language models were trained using Google *n*-grams (LDC2006T13) and the Xinhua News and Agence France-Presse (AFP) sections of English Gigaword, third edition (LDC2007T07). Since the test periods of our development sets (MT05 and MT06) overlap with the Gigaword corpus, we manually removed stories released between November 2004 and February 2005, and between January and March 2006. The tokenization of both Gigaword and Google *n*-grams was roughly matched to the tokenization of the target side of the parallel texts.

Our system for this submission incorporates two language models built using the SRI language modeling toolkit (SRILM) (Stolcke, 2002). The first model was trained using stories from Xinhua News and AFP, as well as the entire target-language side of the parallel data (Section 4.1.1), which represent a total of about 970 million English tokens, including punctuations. We built a back-off 5-gram language model smoothed with the modified Kneser-Ney algorithm (Chen and Goodman, 1996). Due to memory constraints (16GB of RAM), we discarded all 4-grams that occurred only once and all 5-grams that occurred only once or twice.

We also experimented with the four remaining sections of the Gigaword corpus (Associated Press Worldstream, Central News Agency, Los Angeles Times, and New York Times), and built mixture language models combining the different sources (mixture parameters were tuned to either maximize BLEU or minimize perplexity). Despite our observation that these sections helped significantly reduce the perplexity of the English references of our tuning set (6.7% relative perplexity reduction), these extra sources did not yield any significant improvement on the development test set (MT05) in terms of BLEU scores, and so we didn't use them.

We built a second language model using Google *n*-grams. Since the Google collection does not contain *n*-grams with counts lower than 40, it is impractical to utilize smoothing techniques (such as Good-Turning or Kneser-Ney) that rely on "counts-of-counts" statistics to estimate the probability of rare events. We relied instead on Jelinek-Mercer smoothing (Bahl et al., 1983) (known as a "count-based" language model in SRILM), which implements a mixture of count-based maximum-likelihood estimators. In our experiments, the *n*-grams of each order were partitioned by counts into 15 buckets (each bearing a unique interpolation weight), and maximum-likelihood estimates typically converged after 3 to 5 iterations of expectation-maximization (EM) (Dempster et al., 1977). Since SRILM falls short of explicitly enumerating all *n*-grams of count-based language models—inasmuch as such models only contain a few distinct interpolation weights—we converted our count-based language model into the kind of back-off language model expected by our decoder (an ARPA file).[2] Since building a back-off language model requires loading all *n*-grams at once into memory, we limited our use to *n*-grams up to order 3 and removed trigrams that appeared less than 300 times in the Google collection.

[2] One way to achieve this is to create an intermediate back-off ARPA language model containing all *n*-grams of interest (e.g., those that may be applicable at decoding time), then rescore this model with our count-based language model using `ngram -rescore-ngram` in SRILM.

Lexicon-based Features	
(1.1)	$L_{Begin}(C_n), n \in [-2,1]$
(1.2)	$L_{Mid}(C_n), n \in [-2,1]$
(1.3)	$L_{End}(C_n), n \in [-2,1]$
(1.4)	$L_{End}(C_{-1}) + L_{End}(C_0) + L_{End}(C_1)$
(1.5)	$L_{End}(C_{-2}) + L_{End}(C_{-1})$ $+L_{Begin}(C_0) + L_{Mid}(C_0)$
(1.6)	$L_{End}(C_{-2}) + L_{End}(C_{-1})$ $+L_{Begin}(C_{-1})$ $+L_{Begin}(C_0) + L_{Mid}(C_0)$
Linguistic Features	
(2.1)	$C_n, n \in [-2,1]$
(2.2)	$C_{n-1}C_n, n \in [-1,1]$
(2.3)	$C_{n-2}C_n, n \in [1,2]$
(2.4)	$Single(C_n), n \in [-2,1]$
(2.5)	$UnknownBigram(C_{-1}C_0)$
(2.6)	$ProductiveAffixes(C_{-1}, C_0)$
(2.7)	$Reduplication(C_{-1}, C_n), n \in [0,1]$

Table 1: Features for the Chinese segmenter.

4.2 Critical additional features and tools used

4.2.1 Chinese word segmenter

In Section 4.1.1 we mentioned that we used the parallel data provided by IBM Research. However, instead of use the original segmentation, we trained a segmenter with features for better segmentation consistency. Our segmenter is using a CRF model (Lafferty et al., 2001), and we treated Chinese word segmentation as a binary decision task where each character is labeled either as the first character of a word or not (Peng et al., 2004).

The features we are using are listed in Table 1. The segmenter was trained on all of Chinese Treebank (LDC2005T01). In order to evaluate the performance of the segmenter, we also trained on the SIGHAN Bakeoff 2006 training data (the UPUC data) and evaluate on the test data. The overall F measure was 0.940. The OOV recall rate was 0.729, and the IV recall rate is 0.970, which is very close to the best result of SIGHAN Bakeoff 2006.

4.2.2 Text categorizer

The text categorizer is a linear regression classifier trained to distinguish three genres: newswire, newsgroups, and weblogs. At test time, the newsgroup and weblog categories were merged into one 'web' class (training directly a two-way classifier turned out to particularly less effective). Data sources for training included: 600 documents randomly selected from Xinhua News stories of 2006, and all newsgroup and weblog documents in LDC2006E34, LDC2006E24, LDC2006E92, LDC2006E85, and LDC2005E83 (GALE data). Features included average sentence length and all words of the document. We tried several other features, which did not help.

4.2.3 Truecaser

Our truecaser is a CRF classifier with four classes: all lowercase (LC), first letter uppercase (UC), all letters uppercase (CA), and mixed case word (MC) (cf. (Lita et al., 2003)). For building the CRF classifier, we used a subset of the features used in the Stanford Named Entity Recognizer (Finkel et al., 2005). Our truecaser was trained on LDC2005E83, LDC2006E24, LDC2003E14, and part of the Xinhua data (LDC2007T07).

After running the truecaser, we applied 4 different post-processing steps. First, we disambiguate the mixed case words by looking up a list we extracted from a larger set of training data. Second, since the training data we are using actually put many city names into all uppercase (CA), we post-processed those cases to make them in the UC category. Third, we made the first non-punctuation word of every sentence in the UC category. Fourth, we capitalized the first sentence of newswire data (categorized by our categorization tool. We tested the truecaser on the four references of MT06 newswire and newsgroup data. The total per-word accuracy is 96.13%. To further analyze the truecaser, we look at accuracy for each class: 99.73% for MC, 98.67% for LC, 82.37% for CA, 81.79% for UC. We can see from the results that the worst performing category is UC, in which 18.02% were mistaken as LC. This is the place where out truecaser needs to be improved the most. Finally the spacing between tokens was normalized to resemble standard English spacing rules.

4.3 Significant data pre/post-processing

We use the Chinese-English Name Entity Lists v1.0 (LDC2005T34) for Chinese person name transliteration. First, we made a list of Chinese family name characters by taking the Chinese names in Who-Is-Who-China name list, excluding entries that have punctuations, and taking the first characters that ap-

		BLEU[%]	
Set	Google LM	newswire	web
MT06	no	31.58	26.75
MT06	yes	31.48	27.68
MT05	no	32.90	-
MT05	yes	32.92	-
MT08	no	28.43	19.53
MT08	yes	29.24	20.39

Table 2: Machine translation performance by genre (true genre, not the genre predicted by our classifier). The "Google LM" column indicates whether or not the Google language model was incorporated into the decoder. BLEU scores were computed with the script `mteval-v11b.pl` available from NIST.

			uncased
Set	Google LM	BLEU[%]	BLEU[%]
MT05	no	32.90	35.01
MT05	yes	32.92	34.96
MT08	no	24.63	26.30
MT08	yes	25.47	27.23

Table 3: Test set machine translation performance.

pear more than 63 times in the name lists. We obtained a list of 87 common Chinese first names by doing this. With this list, we then get a list of 24,382 Chinese names from the Who-Is-Who-China names with those common family names. We extract the most common transliteration of the characters present in this list.

When dealing with unknown words, if they were of length 2 or 3, started with a character in our common Chinese family name list, and if we have transliteration mapping for every character of that unknown word, we will look up our mapping to transliterate it.

4.4 Other data used

We used no data outside the prescribed LDC data.

4.5 Results

Results by genre for the tuning set (MT06), development test set (MT05), and evaluation test set (MT08) are displayed in Table 2. Overall test set performance is displayed in Table 3. As mentioned previously, we ran MERT on MT06 to generate two sets of weights for each genre, one with the Google LM and one without. Models tuned on web data tend to favor the Google LM more than for newswire data, as shown in Table 4. We then relied on our text categorizer to classify each document of MT06 and MT08 as either 'newswire' or 'web', and then used MERT parameters tuned for the predicted genre. Text categorization performance is shown in Table 5.

Feature name	newswire	web
$P_{ml}(f\|e)$	.0333	.0060
$P_{ml}(e\|f)$	.0256	.0727
$P_{lex}(f\|e)$	.0343	.0818
$P_{lex}(e\|f)$	.0275	.0169
primary LM	.0641	.0568
Google LM	.0156	.0289
linear re-ordering	.0192	.0556
forward monotone	.0679	.0834
forward swap	.0773	−.0081
forward discontinuous	.0942	.0794
backward monotone	.1004	.0416
backward swap	.0299	.0996
backward discontinuous	.1005	.0390
word penalty	−.2083	−.2496
phrase penalty	.1018	.0808

Table 4: MERT parameters for newswire and web genres.

MT08	newswire (true)	web (true)
newswire (predicted)	67	4
web (predicted)	9	29

Table 5: Text categorization confusion matrix. Classification accuracy on MT08 is 88.1%.

We can see from Table 2 that the Google LM provides a significant improvement on web data (0.93 and 0.86 BLEU points on MT06 and MT08, respectively). While the Google LM did not provide any significant improvement on the newswire sections of MT05 and MT06, we nevertheless decided to incorporate this language model into our primary newswire system. We hypothesized that one reason the Google LM did little to improve the Gigaword LM on newswire data was that documents of Gigaword used for training are close to the MT05 and MT06 test periods (some training documents were released just one month prior to or after the two test periods), and that the situation would be different on true test data (MT08). This decision turned out to be good, since MERT parameters with the Google LM (primary submission) outperform MERT parameters without Google LM by 0.84 BLEU point on newswire.

5 Key difference in contrastive systems

No contrastive system was submitted.

Acknowledgements

We thank our collaborators at IBM Research, in particular Niyu Ge, for pre-processing the parallel data used in this work. We thank the developers of Moses, and in particular Philip Koehn, for their generosity in making Moses open source and hence enabling much MT research. This paper is based on work funded in part by the Defense Advanced Research Projects Agency through IBM. The content does not necessarily reflect the views of the U.S. Government, and no official endorsement should be inferred.

References

Lalit R. Bahl, Frederick Jelinek, and Robert L. Mercer. 1983. A maximum likelihood approach to continuous speech recognition. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 5(2):179–190.

Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: parameter estimation. *Comput. Linguist.*, 19(2):263–311.

Stanley F. Chen and Joshua Goodman. 1996. An empirical study of smoothing techniques for language modeling. In *Proceedings of the Thirty-Fourth Annual Meeting of the Association for Computational Linguistics*, pages 310–318.

A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38.

Jenny Rose Finkel, Trond Grenager, and Christopher D. Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *ACL*.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *NAACL '03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*. Association for Computational Linguistics.

Philipp Koehn, Hieu Hoang, Alexandra Birch Mayne, Christopher Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Annual Meeting of the Association for Computation Linguistics (ACL), Demonstration Session*.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. 18th International Conf. on Machine Learning*.

Lucian Vlad Lita, Abe Ittycheriah, Salim Roukos, and Nanda Kambhatla. 2003. tRuEcasIng. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 152–159.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Franz Josef Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4):417–449.

Franz Josef Och. 2002. *Statistical Machine Translation: From Single-Word Models to Alignment Templates*. Ph.D. thesis, RWTH Aachen.

Franz Josef Och. 2003. Minimum error rate training for statistical machine translation. In *ACL 2003: Proc. of the 41st Annual Meeting of the Association for Computational Linguistics*.

Fuchun Peng, Fangfang Feng, and Andrew McCallum. 2004. Chinese segmentation and new word detection using conditional random fields. In *Proceedings of Coling 2004*.

Andreas Stolcke. 2002. SRILM – an extensible language modeling toolkit. In *Proc. Intl. Conf. on Spoken Language Processing (ICSLP–2002)*.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. 1996. HMM-based word alignment in statistical translation. In *Proceedings of the 16th conference on Computational linguistics*, pages 836–841.