CMU Multi-Engine Machine Translation for WMT 2010

Kenneth Heafield
Carnegie Mellon University
Pittsburgh, PA, USA.
heafield@cs.cmu.edu

Alon Lavie
Carnegie Mellon University
Pittsburgh, PA, USA.
alavie@cs.cmu.edu

Abstract

This paper describes our submission, cmu-heafield-combo, to the WMT 2010 machine translation system combination task. Using constrained resources, we participated in all nine language pairs, namely translating English to and from Czech, French, German, and Spanish as well as combining English translations from multiple languages. Combination proceeds by aligning all pairs of system outputs then navigating the aligned outputs from left to right where each path is a candidate combination. Candidate combinations are scored by their length, agreement with the underlying systems, and a language model. On tuning data, improvement in BLEU over the best system depends on the language pair and ranges from 0.89% to 5.57% with mean 2.37%.

1 Introduction

System combination merges the output of several machine translation systems into a single improved output. Our system combination scheme, submitted to the Workshop on Statistical Machine Translation (WMT) 2010 as cmu-heafield-combo, is an improvement over our previous system (Heafield et al., 2009), called cmu-combo in WMT 2009. The scheme consists of aligning 1-best outputs from each system using the METEOR (Denkowski and Lavie, 2010) aligner, identifying candidate combinations by forming left-to-right paths through the aligned system outputs, and scoring these candidates using a battery of features. Improvements this year include unigram paraphrase alignment, support for all target languages, new features, language modeling without pruning, and more parameter optimization. This paper describes our scheme with emphasis on improved areas.

2 Related Work

Confusion networks (Rosti et al., 2008) are the most popular form of system combination. In this approach, a single system output acts as a backbone to which the other outputs are aligned. This backbone determines word order while other outputs vote for substitution, deletion, and insertion operations. Essentially, the backbone is edited to produce a combined output which largely preserves word order. Our approach differs in that we allow paths to switch between sentences, effectively permitting the backbone to switch at every word.

Other system combination techniques typically use TER (Snover et al., 2006) or ITGs (Karakos et al., 2008) to align system outputs, meaning they depend solely on positional information to find approximate matches; we explicitly use stem, synonym, and paraphrase data to find alignments. Our use of paraphrases is similar to Leusch et al. (2009), though they learn a monolingual phrase table while we apply cross-lingual pivoting (Bannard and Callison-Burch, 2005).

3 Alignment

System outputs are aligned at the token level using a variant of the METEOR (Denkowski and Lavie, 2010) aligner. This identifies, in decreasing order of priority: exact, stem, synonym, and unigram paraphrase matches. Stems (Porter, 2001) are available for all languages except Czech, though this is planned for future work and expected to produce significant improvement. Synonyms come from WordNet (Fellbaum, 1998) and are only available in English. Unigram paraphrases are automatically generated using phrase table pivoting (Bannard and Callison-Burch, 2005). The phrase tables are trained using parallel data from Europarl (fr-en, es-en, and de-en), news commentary (fr-en, es-en, de-en, and cz-en), United Na-

Proceedings of the Joint 5th Workshop on Statistical Machine Translation and MetricsMATR, pages 93–98,
Uppsala, Sweden, 11-16 July 2010.

tions (fr-en and es-en), and CzEng (cz-en) (Bojar and Žabokrtský, 2009) sections 0–8. The German and Spanish tables also use the German-Spanish Europarl corpus released for WMT08 (Callison-Burch et al., 2008). Currently, the generated paraphrases are filtered to solely unigram matches; full use of this table is planned for future work. When alignment is ambiguous (i.e. "that" appears twice in a system output), an alignment is chosen to minimize crossing with other alignments. Figure 1 shows an example alignment. Compared to our previous system, this replaces heuristic "artificial" alignments with automatically learned unigram paraphrases.

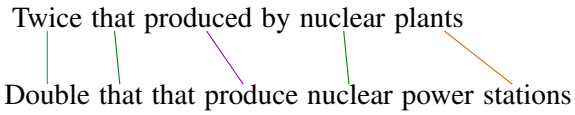


Figure 1: Alignment generated by METEOR showing exact (that–that and nuclear–nuclear), stem (produced–produce), synonym (twice–double), and unigram paraphrase (plants–stations) alignments.

4 Search Space

A candidate combination consists of a string of tokens (words and punctuation) output by the underlying systems. Unconstrained, the string could repeat tokens and assemble them in any order. We therefore have several constraints:

Sentence The string starts with the beginning of sentence token and finishes with the end of sentence token. These tokens implicitly appear in each system's output.

Repetition A token may be used at most once. Tokens that METEOR aligned are alternatives and cannot both be used.

Weak Monotonicity This prevents the scheme from reordering too much. Specifically, the path cannot jump backwards more than r tokens, where positions are measured relative to the beginning of sentence. It cannot make a series of smaller jumps that add up to more than r either. Equivalently, once a token in the ith position of some system output is used, all tokens before the $i - r$th position in their respective system outputs become unusable. The value of r is a hyperparameter considered in Section 6.

Completeness Tokens may not be skipped unless the sentence ends or another constraint would be violated. Specifically, when a token from some system is used, it must be the first (leftmost in the system output) available token from that system. For example, the first decoded token must be the first token output by some system.

Together, these define the search space. The candidate starts at the beginning of sentence by choosing the first token from any system. Then it can either continue with the next token from the same system or switch to another one. When it switches to another system, it does so to the first available token from the new system. The repetition constraint requires that the token does not repeat content. The weak monotonicity constraint ensures that the jump to the new system goes at most r words back. The process repeats until the end of sentence token is encountered.

The previous version (Heafield et al., 2009) also had a hard phrase constraint and heuristics to define a phrase; this has been replaced with new match features.

Search is performed using beam search where the beam contains partial candidates of the same length, each of which starts with the beginning of sentence token. In our experiments, the beam size is 500. When two partial candidates will extend in the same way (namely, the set of available tokens is the same) and have the same feature state (i.e. language model history), they are recombined. The recombined partial candidate subsequently acts like its highest scoring element, until k-best list extraction when it is lazily unpacked.

5 Scoring Features

Candidates are scored using three feature classes:

Length Number of tokens in the candidate. This compensates, to first order, for the impact of length on other features.

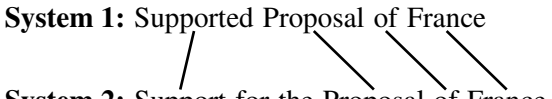
Match For each system s and small n, feature $m_{s,n}$ is the number of n-grams in the candidate matching the sentence output by system s. This is detailed in Section 5.1.

Language Model Log probability from a n-gram language model and backoff statistics. Section 5.2 details our training data and backoff features.

Features are combined into a score using a linear model. Equivalently, the score is the dot product of a weight vector with the vector of our feature values. The weight vector is a parameter optimized in Section 6.

5.1 Match Features

The n-gram match features reward agreement between the candidate combination and underlying system outputs. For example, feature $m_{1,1}$ counts tokens in the candidate that also appear in system 1's output for the sentence being combined. Feature $m_{1,2}$ counts bigrams appearing in both the candidate and the translation suggested by system 1. Figure 2 shows example feature values.



Candidate: Support for Proposal of France

	Unigram	Bigram	Trigram
System 1	4	2	1
System 2	5	3	1

Figure 2: Example match feature values with two systems and matches up to length three. Here, "Supported" counts because it aligns with "Support".

The match features count n-gram matches between the candidate and each system. These matches are defined in terms of alignments. A token matches the system that supplied it as well as the systems to which it aligns. This can be seen in Figure 2 where System 1's unigram match count includes "Supported" even though the candidate chose "Support". Longer matches are defined similarly: a bigram match consists of two consecutive alignments without reordering. Since METEOR generates several types of alignments as shown in Figure 1, we wonder whether all alignment types should count as matches. If we count all types of alignment, then the match features are blind to lexical choice, leaving only the language model to discriminate. If only exact alignments count, then less systems are able to vote on a word order decision mediated by the bigram and trigram features. We find that both versions have their advantages, and therefore include two sets of match features: one that counts only exact alignments and another that counts all alignments. We also tried copies of the match features at the stem and synonym level but found these impose additional tuning cost with no measurable improvement in quality.

Since systems have different strengths and weaknesses, we avoid assigning a single system confidence (Rosti et al., 2008) or counting n-gram matches with uniform system confidence (Hildebrand and Vogel, 2009). The weight on match feature $m_{s,n}$ corresponds to our confidence in n-grams from system s. These weights are fully tunable. However, there is another hyperparameter: the maximum length of n-gram considered; we typically use 2 or 3 with little gain seen above this.

5.2 Language Model

We built language models for each of the five target languages with the aim of using all constrained data. For each language, we used the provided Europarl (Koehn, 2005) except for Czech, News Commentary, and News monolingual corpora. In addition, we used:

Czech CzEng (Bojar and Žabokrtský, 2009) sections 0–7

English Gigaword Fourth Edition (Parker et al., 2009), Giga-FrEn, and CzEng (Bojar and Žabokrtský, 2009) sections 0–7

French Gigaword Second Edition (Mendonca et al., 2009a), Giga-FrEn

Spanish Gigaword Second Edition (Mendonca et al., 2009b)

Paragraphs in the Gigaword corpora were split into sentences using the script provided with Europarl (Koehn, 2005); parenthesized formatting notes were removed from the NYT portion. We discarded Giga-FrEn lines containing invalid UTF8, control characters, or less than 90% Latin characters or punctuation. Czech training data and system outputs were preprocessed using TectoMT (Žabokrtský and Bojar, 2008) following the CzEng 0.9 pipeline (Bojar and Žabokrtský, 2009). English training data and system outputs were tokenized with the IBM tokenizer. French, German, and Spanish used the provided tokenizer.

Czech words were truecased based on automatically identified lemmas marking names; for other languages, training data was lowercased and systems voted, with uniform weight, on capitalization of each character in the final output.

With the exception of Czech (for which we used an existing model), all models were built with no lossy pruning whatsoever, including our English model with 5.8 billion tokens (i.e. after IBM tokenization). Using the stock SRILM (Stolcke, 2002) toolkit with modified Kneser-Ney smoothing, the only step that takes unbounded memory is final model estimation from n-gram counts. Since key parameters have already been estimated at this stage, this final step requires only counts for the desired n-grams and all of their single token extensions. We can therefore filter the n-grams on all but the last token. Our scheme will only query an n-gram if all of the tokens appear in the union of system outputs for some sentence; this strict filtering criterion is further described and released as open source in Heafield and Lavie (2010). The same technique applies to machine translation systems, with phrase table expansion taking the place of system outputs.

For each language, we built one model by appending all data. Another model interpolates smaller models built on the individual sources where each Gigaword provider counts as a distinct source. Interpolation weights were learned on the WMT 2009 references. For English, we also tried an existing model built solely on Gigaword using interpolation. The choice of model is a hyperparameter we consider in Section 6.

In the combination scheme, we use the log language model probability as a feature. Another feature reports the length of the n-gram matched by the model; this exposes limited tunable control over backoff behavior. For Czech, the model was built with a closed vocabulary; when an out-of-vocabulary (OOV) word is encountered, it is skipped for purposes of log probability and a third feature counts how often this happens. This amounts to making the OOV probability a tunable parameter.

6 Parameter Optimization

6.1 Feature Weights

Feature weights are tuned using Minimum Error Rate Training (MERT) (Och, 2003) on the 455 provided references. Our largest submission, xx-en primary, combines 17 systems with five match features each plus three other features for a total of 88 features. This immediately raises two concerns. First, there is overfitting and we expect to see a loss in the test results, although our experience in the NIST Open MT evaluation is that the amount of overfitting does not significantly increase at this number of parameters. Second, MERT is poor at fitting this many feature weights. We present one modification to MERT that addresses part of this problem, leaving other tuning methods as future work.

MERT is prone to local maxima, so we apply a simple form of simulated annealing. As usual, the zeroth iteration decodes with some initial feature weights. Afterward, the weights $\{\lambda_f\}$ learned from iteration $0 \leq j < 10$ are perturbed to produce new feature weights

$$\mu_f \sim U\left[\frac{j}{10}\lambda_f, \left(2 - \frac{j}{10}\right)\lambda_f\right]$$

where U is the uniform distribution. This sampling is done on a per-sentence basis, so the first sentence is decoded with different weights than the second sentence. The amount of random perturbation decreases linearly each iteration until the 10th and subsequent iterations whose learned weights are not perturbed. We emphasize that the point is to introduce randomness in sentences decoded during MERT, and therefore considered during parameter tuning, and not on the specific formula presented in this system description. In practice, this technique increases the number of iterations and decreases the difference in tuning scores following MERT. In our experiments, weights are tuned towards uncased BLEU (Papineni et al., 2002) or the combined metric TER-BLEU (Snover et al., 2006).

6.2 Hyperparameters

In total, we tried 1167 hyperparameter configurations, limited by CPU time during the evaluation period. For each of these configurations, the feature weights were fully trained with MERT and scored on the same tuning set, which we used to select the submitted combinations. Because these configurations represent a small fraction of the hyperparameter space, we focused on values that work well based on prior experience and tuning scores as they became available:

Set of systems Top systems by BLEU. The number of top systems included ranged from 3 to

Pair	Entry	#Sys	r	Match	LM	Objective	ΔBLEU	ΔTER	ΔMETE
cz-en	main	5	4	2	Append	BLEU	2.38	0.99	1.50
de-en	main	6	4	2	Append	TER-BLEU	2.63	-2.38	1.36
	contrast	7	3	2	Append	BLEU	2.60	-2.62	1.09
es-en	main	7	5	3	Append	BLEU	1.22	-0.74	0.70
	contrast	5	6	2	Gigaword	BLEU	1.08	-0.80	0.97
fr-en	main	9	5	3	Append	BLEU	2.28	-2.26	0.78
	contrast	8	5	3	Append	BLEU	2.19	-1.81	0.63
xx-en	main	17	5	3	Append	BLEU	5.57	-5.60	4.33
	contrast	16	5	3	Append	BLEU	5.45	-5.38	4.22
en-cz	main	7	5	3	Append	TER-BLEU	0.74	-0.26	0.68
en-de	main	6	6	2	Interpolate	BLEU	1.26	0.16	1.14
	contrast	5	4	2	Interpolate	BLEU	1.26	0.30	1.00
en-es	main	8	5	3	Interpolate	BLEU	2.38	-2.20	0.96
	contrast	6	7	2	Append	BLEU	2.40	-1.85	1.02
en-fr	main	6	7	2	Append	BLEU	2.64	-0.50	1.55

Table 1: Submitted combinations chosen from among 1167 hyperparameter settings by tuning data scores. Uncased BLEU, uncased TER, and METEOR 1.0 with adequacy-fluency parameters are shown relative to top system by BLEU. Improvement is seen in all pairs on all metrics except for TER on cz-en and en-de where the top systems are 5% and 2% shorter than the references, respectively. TER has a well known preference for shorter hypotheses. The #Sys column indicates the number of systems combined, using the top scoring systems by BLEU. The Match column indicates the maximum n-gram length considered for matching on all alignments; we separately counted unigram and bigram exact matches. In some cases, we made a contrastive submission where metrics disagreed or length behavior differed near the top; contrastive submissions are not our 2009 scheme.

all of them, except on xx-en where we combined up to 17.

Jump limit Mostly $r = 5$, with some experiments ranging from 3 to 7.

Match features Usually unigram and bigram features, sometimes trigrams as well.

Language model Balanced between the appended and interpolated models, with the occasional baseline Gigaword model for English.

Tuning objective Usually BLEU for speed reasons; occasional TER-BLEU with typical values for other hyperparameters.

7 Conclusion

Table 1 shows the submitted combinations and their performance. Our submissions this year improve over last year (Heafield et al., 2009) in overall performance and support for multiple languages. The improvement in performance we primarily attribute to the new match features, which account for most of the gain and allowed us to include lower quality systems. We also trained language models without pruning, replaced heuristic alignments with unigram paraphrases, tweaked the other features, and improved the parameter optimization process. We hope that the improvements seen on tuning scores generalize to significantly improved test scores, especially human evaluation.

Acknowledgments

Ondřej Bojar made the Czech language model and preprocessed Czech system outputs. Michael Denkowski provided the paraphrase tables and wrote the version of METEOR used. This work was supported in part by the DARPA GALE program and by a NSF Graduate Research Fellowship.

References

Colin Bannard and Chris Callison-Burch. 2005. Paraphrasing with bilingual parallel corpora. In *Proceedings ACL.*

Ondřej Bojar and Zdeněk Žabokrtský. 2009. CzEng

0.9, building a large Czech-English automatic parallel treebank. *The Prague Bulletin of Mathematical Linguistics*, (92):63–83.

Chris Callison-Burch, Cameron Fordyce, Philipp Koehn, Christof Monz, and Josh Schroeder. 2008. Further meta-evaluation of machine translation. In *Proceedings of the Third Workshop on Statistical Machine Translation*, pages 70–106, Columbus, Ohio, June. Association for Computational Linguistics.

Michael Denkowski and Alon Lavie. 2010. Extending the METEOR machine translation metric to the phrase level. In *Proceedings NAACL 2010*, Los Angeles, CA, June.

Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. MIT Press.

Kenneth Heafield and Alon Lavie. 2010. Combining machine translation output with open source: The Carnegie Mellon multi-engine machine translation scheme. In *The Prague Bulletin of Mathematical Linguistics*, number 93, pages 27–36, Dublin.

Kenneth Heafield, Greg Hanneman, and Alon Lavie. 2009. Machine translation system combination with flexible word ordering. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 56–60, Athens, Greece, March. Association for Computational Linguistics.

Almut Silja Hildebrand and Stephan Vogel. 2009. CMU system combination for WMT'09. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 47–50, Athens, Greece, March. Association for Computational Linguistics.

Damianos Karakos, Jason Eisner, Sanjeev Khudanpur, and Markus Dreyer. 2008. Machine translation system combination using ITG-based alignments. In *Proceedings ACL-08: HLT, Short Papers (Companion Volume)*, pages 81–84.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Proceedings of MT Summit*.

Gregor Leusch, Evgeny Matusov, and Hermann Ney. 2009. The RWTH system combination system for WMT 2009. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 51–55, Athens, Greece, March. Association for Computational Linguistics.

Angelo Mendonca, David Graff, and Denise DiPersio. 2009a. French gigaword second edition. LDC2009T28.

Angelo Mendonca, David Graff, and Denise DiPersio. 2009b. Spanish gigaword second edition. LDC2009T21.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *ACL '03: Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, pages 160–167, Morristown, NJ, USA. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 311–318, Philadelphia, PA, July.

Robert Parker, David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2009. English gigaword fourth edition. LDC2009T13.

Martin Porter. 2001. Snowball: A language for stemming algorithms. http://snowball.tartarus.org/.

Antti-Veikko I. Rosti, Bing Zhang, Spyros Matsoukas, and Richard Schwartz. 2008. Incremental hypothesis alignment for building confusion networks with application to machine translation system combination. In *Proceedings Third Workshop on Statistical Machine Translation*, pages 183–186.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings Seventh Conference of the Association for Machine Translation in the Americas*, pages 223–231, Cambridge, MA, August.

Andreas Stolcke. 2002. SRILM - an extensible language modeling toolkit. In *Proceedings of the Seventh International Conference on Spoken Language Processing*, pages 901–904.

Zdeněk Žabokrtský and Ondřej Bojar. 2008. TectoMT, Developer's Guide. Technical Report TR-2008-39, Institute of Formal and Applied Linguistics, Faculty of Mathematics and Physics, Charles University in Prague, December.