Language Models and Reranking for Machine Translation

Marian Olteanu, Pasin Suriyentrakorn and Dan Moldovan
Language Computer Corp.
Richardson, TX 75080
{marian,psuri,moldovan}@languagecomputer.com

Abstract

Complex Language Models cannot be easily integrated in the first pass decoding of a Statistical Machine Translation system – the decoder queries the LM a very large number of times; the search process in the decoding builds the hypotheses incrementally and cannot make use of LMs that analyze the whole sentence. We present in this paper the Language Computer's system for WMT06 that employs LM-powered reranking on hypotheses generated by phrase-based SMT systems

1 Introduction

Statistical machine translation (SMT) systems combine a number of translation models with one or more language models. Adding complex language models in the incremental process of decoding is a very challenging task. Some language models can only score sentences as a whole. Also, SMT decoders generate during the search process a very large number of partial hypotheses and query the language model/models [1].

The solution to these problems is either to use multiple iterations for decoding, to make use of the complex LMs only for complete hypotheses in the search space or to generate n-best lists and to rescore the hypotheses using also the additional LMs. For the WMT 2006 shared task we opted for the reranking solution. This paper describes our solution and results.

2 System Description

We developed for the WMT 2006 shared task a system that is trained on a (a) word-aligned bilingual corpus, (b) a large monolingual (English) corpus and (c) an English treebank and it is capable of translating from a source language (German, Spanish and French) into English.

Our system embeds `Phramer`[2] (used for minimum error rate training, decoding, decoding tools), `Pharaoh` (Koehn, 2004) (decoding), `Carmel` [3] (helper for `Pharaoh` in n-best generation), Charniak's parser (Charniak, 2001) (language model) and SRILM[4] (n-gram LM construction).

2.1 Translation table construction

We developed a component that builds a translation table from a word-aligned parallel corpus. The component generates the translation table according to the process described in the Pharaoh training manual[5]. It generates a vector of 5 numeric values for each phrase pair:

- phrase translation probability:

$$\phi(\bar{f}|\bar{e}) = \frac{count(\bar{f},\bar{e})}{count(\bar{e})}, \phi(\bar{e}|\bar{f}) = \frac{count(\bar{f},\bar{e})}{count(\bar{f})}$$

[1] During the translation of the first 10 sentences of the *devtest2006.de* dataset using Phramer and the configuration described in Section 3, the 3-gram LM was queried 27 million times (3 million distinct queries).

[2] http://www.phramer.org/ – Java-based open-source phrase based SMT system

[3] http://www.isi.edu/licensed-sw/carmel/

[4] http://www.speech.sri.com/projects/srilm/

[5] http://www.iccs.inf.ed.ac.uk/~pkoehn/training.tgz

Proceedings of the Workshop on Statistical Machine Translation, pages 150–153,
New York City, June 2006. ©2006 Association for Computational Linguistics

- lexical weighting (Koehn et al., 2003):

$$lex(\bar{f}|\bar{e},a) = \prod_{i=1}^{n} \frac{1}{|\{j|(i,j) \in a\}|} \sum_{\forall(i,j)\in a} w(f_i|e_j)$$

$$lex(\bar{e}|\bar{f},a) = \prod_{j=1}^{m} \frac{1}{|\{i|(i,j) \in a\}|} \sum_{\forall(i,j)\in a} w(e_j|f_i)$$

- phrase penalty: $\tau(\bar{f}|\bar{e}) = e; log(\tau(\bar{f}|\bar{e})) = 1$

2.2 Decoding

We used the `Pharaoh` decoder for both the Minimum Error Rate Training (Och, 2003) and test dataset decoding. Although `Phramer` provides decoding functionality equivalent to `Pharaoh`'s, we preferred to use `Pharaoh` for this task because it is much faster than `Phramer` – between 2 and 15 times faster, depending on the configuration – and preliminary tests showed that there is no noticeable difference between the output of these two in terms of BLEU (Papineni et al., 2002) score.

The log-linear model uses 8 features: one distortion feature, one basic LM feature, 5 features from the translation table and one sentence length feature.

2.3 Minimum Error Rate Training

To determine the best coefficients of the log-linear model ($\overline{\lambda}$) for both the initial stage decoding and the second stage reranking, we used the *unsmoothed Minimum Error Rate Training* (MERT) component present in the `Phramer` package. The MERT component is highly efficient; the time required to search a set of 200,000 hypotheses is less than 30 seconds per iteration (search from a previous/random $\overline{\lambda}$ to a local maximum) on a 3GHz P4 machine. We also used the *distributed decoding* component from `Phramer` to speed up the search process.

We generated the n-best lists required for MERT using the `Carmel` toolkit. `Pharaoh` outputs a lattice for each input sentence, from which `Carmel` extracts a specific number of hypotheses. We used the *europarl.en.srilm* language model for decoding the n-best lists.

The weighting vector is calculated individually for each subtask (pair of source and target languages).

No. of sentences	96.7 M
No. of tokens	2.3 B
Vocabulary size	1.6 M
Distinct grams	1 B

Table 1: English Gigaword LM statistics

2.4 Language Models for reranking

We employed both syntactic language models and n-gram based language models extracted from very large corpora for improving the quality of the translation through reranking of the n-best list. These language models add a total of 13 new features to the log-linear model.

2.4.1 English Gigaword

We created large-scale n-gram language models using English Gigaword Second Edition[6] (EGW).

We split the corpus into sentences, tokenized the corpus, lower-cased the sentences, replaced every digit with "9" to cluster different numbers into the same unigram entry, filtered noisy sentences and we collected n-gram counts (up to 4-grams). Table 1 presents the statistics related to this process.

We pruned the unigrams that appeared less than 15 times in the corpus and all the n-grams that contain the pruned unigrams. We also pruned 3-grams and 4-grams that appear only once in the corpus. Based on these counts, we calculated 4 features for each sentence: the logarithm of the probability of the sentence based on unigrams, on bigrams, on 3-grams and on 4-grams. The probabilities of each word in the analyzed translation hypotheses were bounded by 10^{-5} (to avoid overall zero probability of a sentence caused by zero-counts).

Based on the unpruned counts, we calculated 8 additional features: how many of the n-grams in the the hypothesis appear in the EGW corpus and also how many of the n-grams in the hypotheses don't appear in the Gigaword corpus ($n = 1..4$). The two types of counts will have different behavior only when they are used to discriminate between two hypotheses with different length.

The number of n-grams in each of the two cases is presented in Table 2.

[6] http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2005T12

	sentence probability model	n-gram hit/miss model
1-grams	310 K	310 K
2-grams	45 M	45 M
3-grams	123 M	283 M
4-grams	235 M	675 M

Table 2: Number of n-gram entries in the EGW LM

2.4.2 Charniak parsing

We used Charniak's parser as an additional LM (Charniak, 2001) in reranking. The parser provides one feature for our model – the log-grammar-probability of the sentence.

We retrained the parser on lowercased Penn Treebank II (Marcus et al., 1993), to match the lowercased output of the MT decoder.

Considering the huge number of hypotheses that needed to be parsed for this task, we set it to parse very fast (using the command-line parameter *-T10*[7]).

2.5 Reranking and voting

A $\overline{\lambda}$ weights vector trained over the 8 basic features ($\overline{\lambda_1}$) is used to decode a n-best list. Then, a λ vector trained over all 21 features ($\overline{\lambda_2}$) is used to rerank the n-best list, potentially generating a new first-best hypothesis.

To improve the results, we generated during training a set of distinct $\overline{\lambda_2}$ weight vectors (4-10 different weight vectors). Each $\overline{\lambda_2}$ picks a preferred hypothesis. The final hypothesis is chosen using a voting mechanism. The computational cost of the voting process is very low - each of the $\overline{\lambda_2}$ is applied on the same set of hypotheses - generated by a single $\overline{\lambda_1}$.

2.6 Preprocessing

The vocabulary of languages like English, French and Spanish is relatively small. Most of the new words that appear in a text and didn't appear in a predefined large text (i.e.: translation table) are abbreviations and proper nouns, that usually don't change their form when they are translated into another language. Thus `Pharaoh` and `Phramer` deal with out-of-vocabulary (OOV) words – words that don't appear in the translation table – by copying them into the output translation. German is a compounding language, thus the German vocabulary is virtually infinite. In order to avoid OOV issues for new text, we applied a heuristic to improve the probability of properly translating compound words that are not present in the translation table. We extracted the German vocabulary from the translation table. Then, for each word in a text to be translated (development set or test set), we checked if it is present in the translation dictionary. If it was not present, we checked if it can be obtained by concatenating two words in the dictionary. If we found at least one variant of splitting the unknown word, we altered the text by dividing the word into the corresponding pieces. If there are multiple ways of splitting, we randomly took one. The minimum length for the generated word is 3 letters.

In order to minimize the risk of inserting words that are not in the reference translation into the output translation, we applied a OOV pruning algorithm (Koehn et al., 2005) – we removed every word in the text to be translated that we know we cannot translate (doesn't appear either in the foreign part of the parallel corpus used for training) or in what we expect to be present in an English text (doesn't appear in the English Gigaword corpus). This method was applied to all the input text that was automatically translated – development and test; German, French and Spanish.

For the German-to-English translation, the compound word splitting algorithm was applied before the unknown word removal process.

3 Experimental Setup

We generated the translation tables for each pair of languages using the alignment provided for this shared task.

We split the *dev2006* files into two halves. The first half was used to determine $\overline{\lambda_1}$. Using $\overline{\lambda_1}$, we created a 500-best list for each sentence in the second half. We calculated the value of the enhanced features (EGW and Charniak) for each of these hypotheses. Over this set of almost 500 K hypotheses, we computed 10 different $\overline{\lambda_2}$ using MERT. The search process was seeded using $\overline{\lambda_1}$ padded with 0 for the new 13 features. We sorted the $\overline{\lambda_2}$s by the BLEU score estimated by the MERT algorithm. We pruned manually the $\overline{\lambda_2}$s that diverge too much from the overall set of $\overline{\lambda_2}$s (based on the observation that

[7]Time factor. Higher is better. Default: 210

	500-best oracle	$\overline{\lambda_1}$	best $\overline{\lambda_2}$	voting $\overline{\lambda_2}$	WPT05 best
DE-EN					
– no split		25.70			
– split	33.63	25.81	26.29	26.28	24.77
FR-EN	37.33	30.90	31.21	31.21	30.27
ES-EN	38.06	31.13	31.15	31.22	30.95

Table 3: BLEU scores on the *devtest2006* datasets. Comparison with WPT05 results

	500-best oracle	$\overline{\lambda_1}$	voting $\overline{\lambda_2}$
DE-EN (split)	30.93	23.03	**23.55**
FR-EN	34.71	27.83	**28.00**
ES-EN	37.68	29.97	**30.12**

Table 4: BLEU scores on the *test2006* datasets. Submitted results are bolded.

these weights are overfitting). We picked from the remaining set the best $\overline{\lambda_2}$ and a preferred subset of $\overline{\lambda_2}$s to be used in voting.

The $\overline{\lambda_1}$ was also used to decode a 500-best list for each sentence in the *devtest2006* and *test2006* sets. After computing value of the enhanced features for each of these hypotheses, we applied the reranking algorithm to pick a new first-best hypothesis – the output of our system.

We used the following parameters for decoding: *-dl 5 -b 0.0001 -ttable-limit 30 -s 200* for French and Spanish and *-dl 9 -b 0.00001 -ttable-limit 30 -s 200* for German.

4 Results

Table 3 presents the detailed results of our system on the *devtest2006* datasets and comparison with WMT 2006 best results [8]. The final results, on the test set of the shared task, are reported in Table 4.

5 Conclusions

By analyzing the results, we observe that a very powerful component of our system is the MERT component of `Phramer`. It provided a very high baseline for the *devtest2006* sets (WPT05 test sets).

The additional language models seem to consistently improve the results, although the increase is not very significant on FR-EN and ES-EN subtasks. The cause might be the specifics of the data involved in this shared task – mostly European Parliament proceedings, which is different than the domain of both Treebank and English Gigaword – newswire. The enhanced LMs compete with the default LM (which is also part of the model) that is trained on European Parliament data.

The word splitting heuristics offers also a small improvement for the performance on DE-EN subtask.

Voting seems to slightly improve the results in some cases (ES-EN subtask). We believe that the voting implementation reduces λ weights overfitting, by combining the output of multiple local maxima of the development set. The size of the development set used to generate $\overline{\lambda_1}$ and $\overline{\lambda_2}$ (1000 sentences) compensates the tendency of the unsmoothed MERT algorithm to overfit (Och, 2003) by providing a high ratio between number of variables and number of parameters to be estimated.

[8] http://www.statmt.org/wpt05/mt-shared-task/

References

Eugene Charniak. 2001. Immediate-head parsing for language models. In *Proceedings of 39th Annual Meeting of the Association for Computational Linguistics*, pages 124–131.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of HLT/NAACL 2003*, Edmonton, Canada.

Philipp Koehn, Amittai Axelrod, Alexandra Birch Mayne, Chris Callison-Burch, Miles Osborne, David Talbot, and Michael White. 2005. Edinburgh system description for the 2005 NIST MT Evaluation.

Philipp Koehn. 2004. Pharaoh: A beam search decoder for phrase-based statistical machine translation models. In *Proceedings of AMTA*.

Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In Erhard Hinrichs and Dan Roth, editors, *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 160–167.

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