SRI Submissions to Chinese-English PatentMT NTCIR10 Evaluation

Bing Zhao
SRI International
333 Ravenswood Ave
Menlo Park, CA 94025
bing.zhao@sri.com

Jing Zheng
SRI International
333 Ravenswood Ave
Menlo Park, CA 94025
jing.zheng@sri.com

Wen Wang
SRI International
333 Ravenswood Ave
Menlo Park, CA 94025
wen.wang@sri.com

Nicolas Scheffer
SRI International
333 Ravenswood Ave
Menlo Park, CA 94025
nicolas.scheffer@sri.com

ABSTRACT

The SRI team joined the subtask of Chinese-English Patent machine translation evaluation, and submitted the translation results using a combined output from two types of grammars supported in SRInterp, with two different word segmentations. We investigated the effect of adding sparse features, together with several optimization strategies. Also,for the PatentMT domain, we carried out preliminary experiments on adapting language models. Our results showed positive improvements using these approaches.

Keywords

machine translation, PatentMT, language model adaptation, sparse features, feature selection

1. INTRODUCTION

The SRI team focused on three basic aspects of patentMT in a two-week effort. First is the preprocessing of the patent domain specific data, including word segmentations and named entity tagging; second, we investigated the effects of leveraging sparse features and optimized them for the evaluation task; third, we tried language model adaptation for the patent domain.

There are many long words/terminology terms in the patent data, and word segmentation is one key problem of translating the unknown word or low-frequency terminologies. Two word segmenters are applied in our system, with minimal adjustments of the vocabularies used for the patent data. One is the Stanford word segmenter [9], and the other is the Cambridge word segmenter. With the word-segmented streams, we applied the entity tagging using the text normalization toolkit Datcutr [11]. The formula expressions, new materials' names, and technical terms in the patent data posed great challenges in the preprocessing of the patent data, and the word alignment quality.

Another aspect of the translation in our experiments is to investigate the sparse features. Feature selection is used, which is a two-step process, in our empirical experiments to avoid overfitting, and also to ease the burden of the optimizations. We applied tuning as reranking approach [4], together with an SVM classifier to tune the parameters. This margin-based classifier allows a flexible framework for feature selection and applying priors to the parameters estimations.

We also investigated the effects of language model (LM) adaptation. Due to time constraints, the results did not go into our final submission. A subset of data is retrieved regarding the relevant translations, and a smaller domain-specific LM is then built. It is interpolated with the big background LM, and the weights are learned from held-out data. Overall, we observed a small improvement using the LM adaptation.

2. PREPROCESSING OF THE DATA

The patent data poses special preprocessing challenges to the preprocessing process. Many times, we found that mixed regular expressions can cut sentences in half, and cause serious problems in our system building. Formula, English expressions such as, "FIG", "Fig.", are mixed with Chinese character streams, and short-hand representations of the technical terms. Entity tagging tends to cause serious mistakes in these cases, often breaking things apart, the alignment suffering from non 1-to-1 mapping between entities, and finally losing translations for them. The many UNK words also posed significant challenges to the language model scoring, which cannot figure out the correct probabilities inside of the tagged entities.

We modified the regular expressions inside the Decatur pipeline, to mainly accommodate cases occurring in the patent Chinese data. Due to the nature of the word segmenters, we need to modify the MToken inside the Decatur pipeline to handle the broken segmentations, and try to simulate the same processing the English-alike tokens in the Chinese streams. In the end, the Stanford Chinese word segmenter tends to generates long words, such as Chinese terminologies, resulting a large vocabulary containing 272K words. For the Cambridge word segmenter, the resulting vocabulary is much smaller at 101K words, without breaking the long terminologies into too many small pieces like characters.

Besides the MToekns as in the Decatur pipeline for tagging numbers and dates, for test data we introduced a special token called $eng to mark the English tokens in the source

Table 1: Sparse Feature Types and Examples

Feat Categories	Information	Examples
Lexical	if the word-pair is seen in a lexicon	$f - e$
Fertility	Source word fertility	$f - v0$, $f - v1$, $f - v2$, $f - v3+$
Rule type	Detailed Hiero rule types	F-X1-F-X2 ↔ X1-E-X2
Reorder type	If the target side contains monotone or reordering of non-terminals	WX0WX1W
Target spontaneous words	Predefined English spontaneous words	the, this, such, was...
Bigrams	Bigrams seen in the target side of the phrases	BI_wang_jin-ping
Frequency of rules	Bined frequency if the observed rules	freq1, ..., freqK

language such as "GPS", "LED", and company names such as "Merck". In the ngram queries for LM probabilities, the decoding process will look inside of $eng's content for computing probabilities.

3. TRANSLATION ENGINE

Our systems are based on the SRInterp [12], supporting several PSCFG grammars, including Hiero and string-to-dependency tree. The decoder is chart-based, and standard CKY algorithms are applied to derive translations from a packed forest. Several pruning strategies are supported for speed; consensus decoding and force decoding are also supported. Multiple types and multiple mixtures of language models are also supported. In our submission, we applied Hiero-style grammar, and a string-to-dependency tree grammar (S2D), with a considerations of their speed for short development cycles. In our basic setup, we have about 12 dimensions of features and one single LM. In our submitted system, we chose the sparse feature from the setup, as described in section 4.

With these two different types of grammar, we applied a system combination, to combine the outputs from different grammars and word segmentations. It turns out the two grammars are similar in terms of performance for the development data, and we did not observe significant improvements over the best baseline system.

System combination uses a two-pass alignment algorithm to generate a confusion network [1] from unique 10-best system outputs from each of the systems, and then finds the final outputs based on a set of features, including system ID and number of OOVs in the pass, and a language model estimated from all n-best hypotheses on the test sets. The feature and language model weights were trained using minimum error rate training [5] with a simplex search algorithm. The simplex search algorithm is implemented inside SRILM [8].

4. SPARSE FEATURES

Sparse features are introduced to handle the specific errors in the translation output. In our system, we computed seven categories of sparse features, as listed in Table 1. Some of the features can be treated as dense, such as the category of frequency of the rule, but majority of them are sparse.

The lexicon features are derived from the IBM model-1, in which we check if a word-pair $f \leftrightarrow e$ occurs in the derivation. The fertility features check how many times a word is aligned to 1 word, 2 words, or 3+ words. The reordering features will only check the reordering between nonterminals, which is basically a simple count of the reorderings in the derivation tree; the rule type is a more detailed description of the PSCFG rules used, and 38 types of rules are defined in our system. Several examples together with the weights are given in Table 2.

We start with a list of $2,739,369$ sparse features, and run optimizations and selections iteratively to avoid overfitting.

4.1 Optimization

We chose the tuning-as-reranking algorithm [4] for optimizing the sparse features. We use the margin-based classifier SVM to leverage the priors for the weights to be learned. The weights for each feature can be used for a second step of feature selection. The weights closer to zero, can be safely discarded in the next iteration for optimization. Empirically, the weights learned from SVM are also more interpretable than the weights learned from the maxent classifier. As shown in Table 2, the optimization preferences are illustrated by the weights learned. The positive weights are preferred, and the negative weights indicate that the system dislikes observations of the feature. For instance, the system likes to see more of the English word "some", but not the word "was"; it likes the rule type "F-X-F-X-F->X-E-X-E-X", which seems to have more lexical items to bound the nonterminals, and it likes less the rule type of "F-X-F-X-F -> X-E-X", which seems to have unbounded nonterminals X in the target side. These unbounded nonterminals might indicate that the rules were extracted from more ambiguous word alignments, and might be less trustworthy in applying to the derivations.

To avoid overfitting, we also try to build tuning data from various sources to have a representative tuning set, and reduce the risks of overfitting. More details to follow in the full version of the paper.

4.2 Feature Selection

Overfitting is not always avoidable for tuning sparse features. Besides building relevant tuning set, we carried out two simple feature selection strategies. First, we start from the full list of the sparse features, which are around 2.7 million, and use PRO with SVM to tune all the parameters toward BLEU. All the features receiving a weight close to zero will then be removed from the list, and the remaining features will be used for the next optimization iteration. Iteratively, we keep only the remaining top 300 weighted features, which are spread among all the seven categories. In this process, the top 300 weighted features are selected from the optimization process together with the baseline basic dense features; we further optimize the selected sparse features alone by fixing the weights learned for the dense features. This seems to be beneficial and more stable for the unseen testset, as shown in Table 3.

Table 2: Features and their weights

Features	Weights	Features	Weights	Features	Weights
WX0WX1W	0.1655	F-X-F-X-F->X-E-X-E	0.2022	some	0.1973
WX1WX0W	0.1559	F-X-F-X -> E-X-E-X-E	0.2312	an	0.099
X0WX1	0.0744	X-F-X -> X-E-X-E	0.4769	such	-0.0514
X1X0W	-0.0735	F-X-F-X-F -> X-E-X	-0.0618	was	-0.0554

Table 3: Experiments on

Setup	BLEU
Baseline	31.4
all sparse features	30.4
300 sparse features	31.6
updated optimization	32.8

Table 4: Data

Word Segmenter	Chinese Tokens	English Tokens
Stanford	38,335,422	44, 289,651
Cambridge	41,151,267	44, 289,651

Table 5: Comparision between different word segmenters and grammars

Word Segmenter	Hiero	Str-to-DepTree
Stanford	33.3	32.1
Cambridge	33.6	32.4

Table 6: System combination over different word segmenters and grammars

Setups	Bleu
Baseline	34.4
Syscomb	34.6

5. EXPERIMENTS

In our experiments, we mainly use lower-cased BLEU [6] for tuning and testing; in our final submission, a truecaser that is also trained for the patent domain is applied to the final output. All scores reported are lower-cased scores except those mentioned explicitly for mix-cased cases. Our development data contains 2000 sentences, and the test set contains 2000 sentences. We further split the test set into two smaller parts to have one development set for LM adaptation. The training, test conditions, and our final results are listed in the evaluation overview paper [3]. As the monolingual English data has a size of almost 14 billion running tokens, a 7-gram LM is learned, and then shrunk using a bloom-filter LM; the final LM has a size of 6.2 GB on the disk for final evaluation systems. As this LM is large, we also trained an LM from the 45M words of the parallel data, and use the smaller LM for tuning parameters of our MT engine. The large LM is used only for our final system submission, with borrowed parameters from the smaller LM setup.

To speed up the decoding, we applied the pruning of the grammars similar to the work in [10]. We cache language model ngram queries in a trie, and look inside of the content of the English entities ($eng{}) during the decoding.

5.1 Data

We used all of the one million parallel training sentences, and applied two pipelines of preprocessing using Stanford word segmenter and the Cambridge word segmenter.

The Stanford word segmenter tends to glue pieces into long words. The Cambridge one, which is HMM based, usually breaks long words into shorter ones. As shown in Table 4, using Stanford word segmenter, the number of tokens is quite far from the number of tokens in English, indicating that there could be a systematic gap to fill in for the alignment model. On the other hand, the Cambridge word segmenter seems to generate a reasonable number of tokens, and is close to the number of tokens in the target side.

For two different word segmentations, we run the evaluations on our test set. in Table 5, we can see that the two word segmenters, however, did not differ much in the final translation quality, even though they are much different in terms of the vocabulary sizes and the gap between the number of tokens in the parallel training data.

5.2 Translation Models

We built two sets of PSCFG grammars for our evaluation submission: the standard Hiero grammar [2], and the string-to-dependency tree grammar [7]. As shown in Table 5, the standard Hiero grammar is about 1 BLEU point better than the string-to-dependency tree grammar (S2D). A closer check showed that the translation output from the S2D is shorter than the Hiero output in general for patent data. We suspect that the dependency trees we collected for patent data might favor shorter and shallow structures.

5.3 System Combination

Due to the nature of the similar preprocessing pipelines and translation models, our system combination, even though it is based on multiple segmentation and grammars, gains only a small margin over our best single system, which is a Hiero grammar using the Cambridge word segmenter. Various systems might bring better performance to the final system combinations.

5.4 Language Model Adaptation

As patent data is domain specific, we tried to integrate language model adaptation into our pipeline, to further boost performance. For this experiment, we used slightly different tuning and test sets. Due to time constraints, the results did not go into our final submissions. Table 7 showed the improvements on top of our best baseline single engine, with the Cambridge word segmenter, but using a small language model trained with 45M words. We split the 2000 sentences development data into a 1000-sentence tune-set and 1000-sentence test set. We investigated two approaches for language model adaptation.

SRC-adapt-1passdecode: use the source sentences in one document to retrieve the indexed source side of parallel text,

Table 7: Language Model Adaptation Results in terms of BLEU and TER We observed better TER improvements than BLEU over the baseline using two adaptation strategies.

Setups	Tune	Test
Baseline	33.27/50.37	32.18/51.46
TGT-adapt-redecode	33.75/49.84	32.42/50.94
SRC-adapt-1passdecode	33.88/49.62	32.54/50.67

Table 8: Final System Performances on Dev and Test data

Setup	Dev	Test
Dense Features	30.1	31.9
Sparse Features	37.1	31.7
feature selection-1	35.3	32.4
feature selection-2	35.3	32.7

train 7-gram adapted LM using corresponding target side sentences for each test document, and linearly interpolate with the 7-gram BGLM (i.e., decoding LM trained on the parallel text) and re-decode.

TGT-adapt-1passdecode: run 1pass decoding with BGLM, use the translation hyps in one document to retrieve the indexed target-side sentences, train 7-gram adapted LM for each document and linearly interpolate with the 7-gram background LM, and run 1pass decoding.

For tuning the parameters, we keep the same LM weight, and tune on the tune set the threshold of retrieved data, and the interpolation weight between adapted LM and background LM. We observed better TER score improvement than BLEU scores.

5.5 Results

We carried out two feature selections on building our final systems. The first one is to select the top 300 weighted features. The second one is to slightly relax the feature eliminations in the optimization process, and try to optimize the sparse features separately by fixing the weights of the dense features; This allows us to use a slightly larger feature set, which contains about 923 features. In the final submissions, we used the second feature selection for single engines, which seems to perform slightly better than the first one on the unseen test set.

On the unseen testset, we did not observe the significant improvement we obtained on the development set, and the feature selection based on the development set might be still loose to avoid the overfitting. By constructing a relevant tuning and development set, we might get better translations by using sparse features.

6. DISCUSSIONS AND CONCLUSIONS

Preprocessing is one of the hurdle in our evaluation system, and we observed serious problems in the final translation output. Simple ngrams LM were applied in our evaluation system, and we plan to investigate more structured language models to leverage more of the document context.

Feature selection is another issue in our development. Overfitting is observed in our tuning and devset, and the our feature selection is done in an iterative selection-optimization process, that cannot guarantee a converged set of features. A better feature selection algorithm may identify the relevant set of features and ease the burden of optimizations.

7. REFERENCES

[1] N. F. Ayan, J. Zheng, and W. Wang. Improving alignments for better confusion networks for combining machine translation systems. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 33–40, Manchester, UK, August 2008. Coling 2008 Organizing Committee.

[2] D. Chiang. Hierarchical phrase-based translation. In *Computational Linguistics*, volume 33(2), 2007.

[3] I. Goto, K. p. Chow, B. Lu, E. Sumita, and B. K. Tsou. Overview of the patent machine translation task at the ntcir-10 workshop. In *Proceedings of the 2013 NTCIR Patent MT workshop*, Japan, June 2013.

[4] M. Hopkins and J. May. Tuning as ranking. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1352–1362, Edinburgh, Scotland, UK., July 2011. Association for Computational Linguistics.

[5] F. J. Och. Minimum error rate training for statistical machine translation. In *Proc. of the 41st Annual Meeting of the Association for Computational Linguistics*, Japan, Sapporo, July 2003.

[6] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proc. of the ACL-02)*, pages 311–318, Philadelphia, PA, July 2002.

[7] L. Shen, J. Xu, and R. Weischedel. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proceedings of ACL*, 2008.

[8] A. Stolcke. SRILM – An extensible language modeling toolkit. In *Proc. Intl. Conf. on Spoken Language Processing*, volume 2, pages 901–904, Denver, 2002.

[9] H. Tseng, P. Chang, G. Andrew, D. Jurafsky, and C. Manning. A conditional random field word segmenter. In *Fourth SIGHAN Workshop on Chinese Language Processing*, 2005.

[10] R. Zens, D. Stanton, and P. Xu. A systematic comparison of phrase table pruning techniques. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 972–983, Jeju Island, Korea, July 2012. Association for Computational Linguistics.

[11] B. Zhang and J. G. Kahn. Evaluation of decatur text normalizer for language model training. In *Technical Report, University of Washington*, 2008.

[12] J. Zheng. Srinterp: Sri's scalable multipurpose smt engine. Technical report, SRI International, 2007.