PBML

logo

The Prague Bulletin of Mathematical Linguistics

NUMBER ??? DECEMBER 2009 1–9

MANY

Open Source Machine Translation System Combination

Loïc Barrault

Abstract

This paper describes a *push-the-button* MT system combination toolkit. The combination is based on the creation of a lattice made on several confusion networks (CN) connected together. This lattice is then decoded with a token-pass decoder to provide the best and/or n-best outputs. Each CN is built using a modified version of the TERp tool. The toolkit is made of several scripts along a core program developed in Java. It is totally configurable and the parameters can be tuned quite easily.

1. Introduction

Machine translation (MT) system combination has taken a great importance these past few years. This is mainly due to the fact that single systems achieved good performance and the possibility of taking the most of their complementarity in a system combination framework is very attractive. Many techniques can be used for system combination. One concerns hypothesis selection using nbest list reranking based on various features as described in (Hildebrand and Vogel, 2009). Another approach is to consider source text and systems outputs as bitext and train a new SMT system on these data (Chen et al., 2009).

In this paper, a system combination based on confusion network (CN) is described. This approach is not new, and numerous publications are available on that subject, see for example, (Rosti, Matsoukas, and Schwartz, 2007); (Shen et al., 2008); (Karakos et al., 2008) and (Leusch, Matusov, and Ney, 2009). Such an approach is presented in Figure 1. The protocol can be decomposed into three steps :

1. 1-best hypotheses from all M systems are aligned in order to build confusion networks.

Please cite this article as: Loïc Barrault, MANY: Open Source Machine Translation System Combination. The Prague Bulletin of Mathematical Linguistics No. ???, 2009, 1–9.

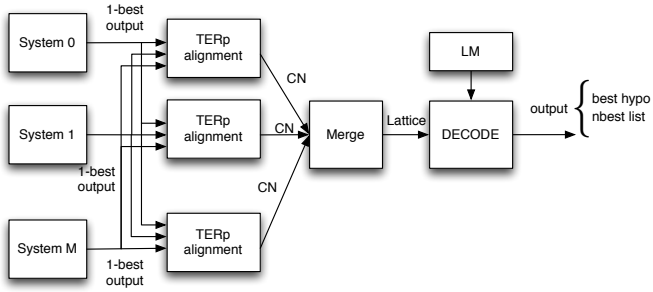

Figure 1. MT system combination.

2. All confusion networks are connected into a single lattice.
3. A language model is used to decode the resulting lattice and the best hypothesis is generated.

Section 2.1 describes the alignment process and in particular the new features added to TERp in order to be enable alignment of an hypothesis against a CN. The decoder is presented in section 3. Some example results obtained at the IWSLT'09 evaluation campaign are given in section 5. Finally, a description of the toolkit is given in section 6.

2. Hypotheses alignment and confusion network generation

The goal of this step is to put the words provided by different systems in competition with each other inside a confusion network (Mangu, Brill, and Stolcke, 1999).

For each segment, the best hypotheses of $M - 1$ systems are aligned against the last one used as backbone. A modified version of the TERp tool (Snover et al., 2006) is used to generate a confusion network (see section 2.1 for details). This is done by incrementally adding the hypotheses to the CN. The hypotheses are added to the backbone beginning with the nearest (in terms of TER) and ending with the more distant one. This differs from the result of (Rosti, Matsoukas, and Schwartz, 2007) where the nearest hypothesis is computed at each step, which is supposed to be better. M confusion networks are generated in this way. Then all the confusion networks are connected into a single lattice by adding a first and last node. The probability of the first arcs (later named priors) must reflect how well such system provide a well structured hypothesis.

2.1. Modified TERp

The modified TERp is based on TERp v0.1 and is written in Java. Some classes have been modified and new ones were created to add some functionalities such as alignment between a sentence and a confusion network. This has been done by modifying the data structure and extending some heuristic to find better alignment.

When using *relaxed* constraints with TERp, the shift heuristics allow a block of words to be moved if it matches (or is a paraphrase) of another block of words somewhere else. Shifts are also allowed when a stem or synonym is found somewhere else.

When considering confusion networks, the same heuristics are applied except that the block of words must match (be a paraphrase, synonyms or stem of) one of the sequence of words represented in the CN. An example of such a case is presented in figure 2. In figure 2, we can notice that the paraphrase *the dinner / supper* allow

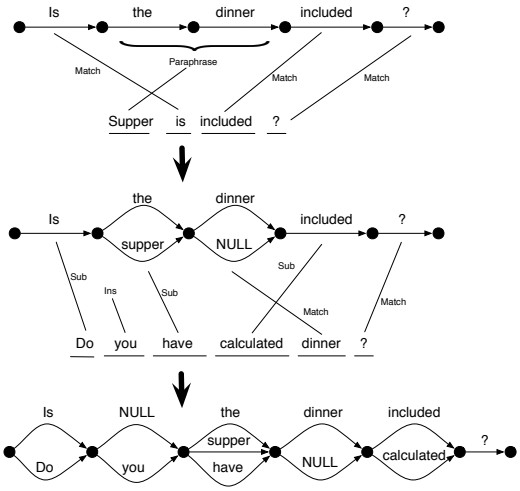

Figure 2. Incremental alignment with TERp resulting in a confusion network.

a switch of block of word. However, the word *supper* is aligned with the word *the* because no rule is used in order to make inside-paraphrase word alignment, yet ! (see section 6.4 for future features).

In addition to the confusion network generation, the possibility of using scores on words has been added, which can be very useful during the decoding. For the moment, these scores must be computed separately from MANY. The idea underneath is to provide an option to include confidence measure at word level, though it can be computed at any level (see for example, (Ueffing and Ney, 2005)). In this version of

the software, the scores are equal to the priors of the systems. However, these values can be modified in the configuration file.

3. Decoding

The decoder is based on the token pass decoding algorithm (see for example (Young, Russell, and Thornton, 1989)). The principle of this decoder is to propagate tokens over the lattice and accumulate various scores into a global score for each hypotheses.

The scores used to evaluate the hypotheses are the following :

- the system score : this replace the score of the translation model. Until now, the words given by all systems have the same probability which are equal to their priors, but any confidence measure can be used at this step.
- the language model (LM) probability.
- a fudge factor to balance the probabilities provided in the lattice with regard to those given by the language model.
- a *null-arc* penalty : this penalty avoids to always go through *null-arcs* encountered in the lattice.
- a length penalty : this score helps to generate correctly sized hypotheses.

The probabilities computed in the decoder can be expressed as follow :

$$\begin{aligned} log(P_W) &= \sum_{n=0}^{Len(W)} [log(P_{ws}(n)) + \alpha P_{lm}(n)] \\ &\quad + Len_{pen}(W) + Null_{pen}(W) \end{aligned} \tag{1}$$

where $Len(W)$ is the length of the hypothesis, $P_{ws}(n)$ is the score of the n^{th} word in the lattice, $P_{lm}(n)$ is its LM probability, α is the fudge factor, $Len_{pen}(W)$ is the length penalty of the word sequence and $Null_{pen}(W)$ is the penalty associated with the number of null-arcs crossed to obtain the hypothesis.

At the beginning, only one token is created at the first node of the lattice. Then this token spread over the consecutive nodes, accumulating the score on the arc it crosses, the language model probability of the word sequence generated so far and null or length penalty if applicable. The number of tokens can increase really quickly to cover the whole lattice, and, in order to keep it tractable, only the N_{max} best tokens are kept (the others are discarded), where N_{max} can be configured in the configuration file. Other methods to restrict the number of tokens (like pruning based on score or other heuristics) can easily be implemented in this software, but this is not done already.

3.1. Technical details about the token pass decoder

This software is based on the Sphinx4 library and is highly configurable. The maximum number of tokens being considered during decoding, the fudge factor, the

null-arc penalty and the length penalty can all be set within the xml configuration file. This is useful for tuning (see the config file generator description in section 6.2).

The probabilities which are manipulated within the decoder are all obtained from the LogMath class which ensure the consistency of the values.

3.2. Language model

There are two ways of loading a LM with this software.

The first solution is to use the LargeTrigramModel class, but as its name tells us, only a 3-gram model can be loaded with this class.

The second and easiest way is to use a language model hosted on a lm-server. This kind of LM can be accessed via the LanguageModelOnServer class which is based on the generic LanguageModel class from the Sphinx4 library. This allow us to load a n-gram LM with n higher than 3, which is not possible with a standard LM class in Sphinx4 yet (it is currently being done).

In addition, the Dictionnary interface has been extended in order to be able to load a simple dictionary containing all the words known by the LM (no need to know the different pronunciations of each words in this case).

As the language model interface is also written in java and is using the Sphinx4 library, one could easily write a new class to load a LM in a proprietary file format.

4. Tuning

There is a lot of parameters which can be tuned in MANY. The edit costs of the modified TERp, the prior costs of each systems in the lattice, the fudge, null-arc penalty and length penalty for the decoder. This can easily been done by generating configuration files (with the help of *genSphinxConfig.pl*, see section 6.3). Parameters for modified TERp, for the decoder and systems weights are currently tuned together. The separate tuning of TERp and decoder parameters is an ongoing work, and I could not say whether it is preferable or not yet.

Any method can then be used to provide new values for these parameters. As an example, we are using Condor (Berghen and Bersini, 2005) to optimize those parameters.

5. Some example results

MANY software has been used for the IWSLT'09 evaluation campaign. Table 1 presents the results obtained with this approach. The *SMT* system is based on MOSES, the *SPE* system corresponds to a rule-based system from SYSTRAN whose outputs have been corrected by a SMT system and the *Hierarchical* is based on Joshua.

In these task, the system combination approach yielded +1.39 BLEU on Ar/En and

Systems	Arabic/English		Chinese/English	
	Dev7	Test09	Dev7	Test09
SMT CSLM	54.75	50.35	41.71	36.04
SPE CSLM	48.13	-	41.23	38.53
Hierarchical	54.00	49.06	39.78	31.89
SMT CSLM + SPE CSLM			42.55	40.14
+ tuning			43.06	39.46
SMT CSLM + Hier.	55.89	50.86		
+ tuning	57.01	51.74		

Table 1. Results of system combination on Dev7 (development) corpus and Test09, the official test corpus of IWSLT'09 evaluation campaign.

+1.7 BLEU on Zh/En. One observation is that tuning parameters did not provided better results for Zh/En.

6. Software description

6.1. Data

The software takes several files as input (which are supposed to be synchronized[1]) containing the 1-best hypothesis of all systems, one sentence per line. These hypotheses can contain foreign words if no translation have been found for them, and they will be considered as unknown words during the decoding step.

6.2. Configuration file

The configuration file is an xml file similar to those used with Sphinx4.

```
<component  name="decoder" type="edu.loic.decoder.TokenPassDecoder">
<property name="dictionary" value="dictionary"/>
<property name="logMath" value="logMath"/>
<property name="logLevel"     value="INFO"/>
<property name="lmonserver" value="lmonserver"/>
<property name="fudge" value="0.2"/> <!-- This value is multiplied by 10 in the software -->
<property name="null_penalty" value="0.3"/>
<property name="length_penalty" value="0.5"/> <!-- This value is multiplied by 10 in the software -->
</component>
```

This part allow us to configure the decoder parameters such and more particularly the fudge factor, the null-arc penalty and the length penalty.

```
<component  name="lmonserver" type="edu.cmu.sphinx.linguist.language.ngram.LanguageModelOnServer">
<property name="lmserverport" value="1234"/>
<property name="lmserverhost" value="machine1"/>
```

[1] *i.e.* each n^{th} line is the translation of the same source sentence

```
<property name="maxDepth" value="4"/>
<property name="logMath" value="logMath"/>
</component>
```

This part configures the LM class which will connect to the lm-server hosted on *machine1* on port "1234". The *maxDepth* field correspond to the depth of the LM loaded on the server.

```
<component name="MANY" type="edu.lium.mt.MANY">
<property name="decoder" value="decoder"/>
<property name="terp" value="terp"/>
<property name="output" value="output.many"/>
<property name="priors" value="4.0e-01 4.0e-01 2.0e-01"/>
<property name="hypotheses" value="hyp0.id hyp1.id hyp2.id" />
<property name="hyps_scores" value="hyp0_sc.id hyp1_sc.id hyp2_sc.id" />
<property name="costs" value="1.0  1.0   1.0   1.0   1.0   0.0   1.0" />
<!--                          del  stem   syn   ins  sub   match shift-->
<property name="terpParams"     value="terp.params"/>
<property name="wordnet"     value="/opt/mt/WordNet-3.0/dict/"/>
<property name="shift_word_stop_list" value="/opt/mt/terp/terp.v1/data/shift_word_stop_list.txt"/>
<property name="paraphrases" value="/opt/mt/terp/terp.v1/data/phrases.db"/>
</component>
```

This part is the core part. It configures the various file to combines, the costs for TERp, the location of WordNet and the paraphrases table (also for TERp). The *priors* can be set here and are used in the lattice.

6.3. Scripts

The main script is called *Many.sh*. Some parameters have to be set inside this script in order to run an system combination experiments. The reader should refer to the *readme* file provided with the software.

Each input sentence (as well as the corresponding word scores) must have an id which is of the following form : [*set*][*doc.*##][*sent*] The shell script *add_id.sh* is in charge of adding such an id to the input data (called in the Many.sh script).

The perl script *genSphinxConfig.pl* is used to generate a new config file with specific values. This is very useful for generating a new config file with parameters estimated by a certain optimization procedure.

6.4. Future features

Several features are planned to be added into MANY. One is the possibility of exploring all shifts which do not decrease the alignment score instead of using heuristics. This has been done by (Rosti et al., 2009) and provided good results (even though the increasing time of processing was not indicated).

Another feature would be the intra-paraphrase word alignment. Like is presented in figure 2, when a paraphrase is found, it appears that the word alignment inside that paraphrase is not always the best. In that example, (*supper* is aligned with *the* instead of *dinner*, which would be better. This could be easily added using a specific alignment model.

As mentioned before, the load of a n-gram (whatever is n) language model has to be added. In some cases, that can be faster than using a LM server.

An alternative to the token pass decoder would be the use of Minimum Bayesian Risk decoder applied on the final lattice (MBR-Lattice) like described in (Tromble et al., 2008)

7. Discussion

One might notice that the performance of a system combination is highly dependent of the input hypotheses (in terms of number of hypotheses, complementarity of the systems which provide them, and of course quality), the parameters of the alignment module and the language model used to decode the lattice. The tuning of all parameters plays consequently a big role in the quality of this kind of approach. As an example, in (Rosti et al., 2009), after the creation of the lattice, three iterations of tuning have been done in order to obtain good results. This kind of tuning procedure is not currently implemented in that software, but it is a very important step which should not be underestimated.

8. Conclusion

This paper presents a machine translation system combination software, MANY, based on the decoding of a lattice made of several confusion networks connected together. The software is written in java and is composed of a modified version of TERp software and a decoder based on Sphinx4 library. This software, which is easily extensible and highly configurable, obtained good results when used during the IWSLT'09 evaluation campaign.

Bibliography

Berghen, Frank Vanden and Hugues Bersini. 2005. CONDOR, a new parallel, constrained extension of powell's UOBYQA algorithm: Experimental results and comparison with the DFO algorithm. *Journal of Computational and Applied Mathematics*, 181:157–175, September.

Chen, Yu, Michael Jellinghaus, Andreas Eisele, Yi Chang, Sabine Hunsicker, Silke Theison, Christian Federmann, and Hans Uszkoreit. 2009. Combining multi-engine translations with moses. In *Workshop on Statistical Machine Translation*, pages 42–46, Athens, Greece, March.

Hildebrand, Almut Silja 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.

Karakos, Damianos, Jason Eisner, Sanjeev Khudanpur, and Markus Dreyer. 2008. Machine translation system combination using itg-based alignments. In *46th Annual Meeting of the*

Association for Computational Linguistics: Human Language Technologies., pages 81–84, Columbus, Ohio, USA, June 16-17.

Leusch, Gregor, 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 61–65, Athens, Greece, March 30-31.

Mangu, L., E. Brill, and A. Stolcke. 1999. Finding consensus among words : Lattice-based word error minimization. In *European Conference on Speech Communication and Technology, Interspeech*, volume I, pages 495–498.

Rosti, A.-V.I., S. Matsoukas, and R. Schwartz. 2007. Improved word-level system combination for machine translation. In *Association for Computational Linguistics*, pages 312–319.

Rosti, A.-V.I., B. Zhang, S. Matsoukas, , and R. Schwartz. 2009. Incremental hypothesis alignment with flexible matching for building confusion networks: Bbn system description for wmt09 system combination task. In *EACL/WMT*, pages 61–65.

Shen, Wade, Brian Delaney, Tim Anderson, and Ray Slyh. 2008. The mit-ll/afrl iwslt-2008 mt system. In *IWSLT*, Hawaii, U.S.A, 69–76.

Snover, Matthew, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Association for Machine Translation in the Americas*.

Tromble, Roy W., Shankar Kumar, Franz Och, and Wolfgang Macherey. 2008. Lattice minimum bayes-risk decoding for statistical machine translation. In *Conference on Empirical Methods in Natural Language Processing*, pages 620–629, Honolulu, Oct.

Ueffing, Nicola and Hermann Ney. 2005. Word-level confidence estimation for machine translation using phrase-based translation models. In *International Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 763–770, Morristown, NJ, USA. Association for Computational Linguistics.

Young, S. J., N. H. Russell, and J. H. S. Thornton. 1989. Token passing : a simple conceptual model for connected speech recognition systems. Technical report, Cambridge University Engineering Department, July.