Meta-level Statistical Machine Translation

Sajad Ebrahimi[†,*], **Kourosh Meshgi**[††], **Shahram Khadivi**[†]
and Mohammad Ebrahim Shiri Ahmad Abady[*]
[†]Human Language Technology Lab, Amirkabir University of Technology, Tehran, Iran
[††]Graduate School of Informatics, Kyoto University, Kyoto, Japan
[*]Department of Computer Science, Amirkabir University of Technology, Tehran, Iran
ebrahimi.sajad@aut.ac.ir, meshgi-k@sys.i.kyoto-u.ac.jp
khadivi@aut.ac.ir, shiri@aut.ac.ir

Abstract

We propose a simple and effective method to build a *meta-level* Statistical Machine Translation (SMT), called meta-SMT, for system combination. Our approach is based on the framework of *Stacked Generalization*, also known as *Stacking*, which is an ensemble learning algorithm, widely used in machine learning tasks. First, a collection of *base-level* SMTs is generated for obtaining a meta-level corpus. Then a meta-level SMT is trained on this corpus. In this paper we address the issue of how to adapt stacked generalization to SMT. We evaluate our approach on English-to-Persian machine translation. Experimental results show that our approach leads to significant improvements in translation quality over a phrase-based baseline by about 1.1 BLEU points.

1 Introduction

Currently, there exist a number of commercial and research Machine Translation (MT) systems, which are developed under different paradigms such as rule-based, example based, statistical machine translation, trained using different algorithms, e.g., phrase-based SMT, hierarchical phrase-based SMT, syntax-based SMT with different types and amounts of training data. With the emergence of these various structurally different systems, system combination methods have taken a great importance during the past few years.

There are several techniques for combining multiple SMT systems to achieve higher translation quality, e.g. sentence-level combination (Hildebrand and Vogel, 2008) simply selects "the best" of the provided translations and phrase-level combination (Matusov et al., 2006; Rosti et al., 2007) can generate new translations differing from all original translations.

Most of the state-of-the-art SMT system combination methods require multiple SMT systems based on different models. Since it is not easy to have multiple SMT systems, in this work we focus on applying stacking algorithm on a single SMT system rather than multiple SMT systems.

We try to increase the performance of an SMT system by introducing a meta-level SMT which can learn how to decrease or modify translation errors on the translation outputs of original SMT system. This task is also known as automatic post-editing (APE) which is a well-studied topic in machine translation community (Simard et al., 2007a; Béchara et al., 2011). To do this, we use stacked generalization which is an ensemble learning algorithm

Ensemble Learning is a machine learning paradigm where multiple learners are trained to solve the same problem. An *ensemble* is viewed as a collection of learners which are usually called *base learners*. Base learners are usually generated from training data by a *base learning algorithm* which can be decision tree, neural network or other kinds of machine learning algorithms. Most ensemble methods use a single base learning algorithm to produce *homogeneous* base learners, but there are also some methods which use multiple learning algorithms to produce *heterogeneous* learners. The concept of ensembles appeared in classification literature has subsequently been studied in several frameworks, including *stacked generalization* (Wolpert, 1992), *bagging* (Breiman, 1996b), *boosting* (Scharpire, 1990), *model averaging* (Perrone et al., 1993), *forecast combining* (Granger, 1989) and so on.

Previous work have tried to introduce some of these frameworks into SMT (Xiao et al., 2010), none of them adapt stacked generalization to SMT. While stacked generalization has been extensively investigated in machine learning, its adaption to SMT is not a trivial task. In this paper, we show how to make stacked generalization work for a single SMT system.

Stacked generalization is a general method of using a meta-level model to combine base-level models to achieve higher accuracy. However, this algorithm is introduced for combining multiple models, we focus our attention to utilize this algorithm in order to improve only a single SMT system. The basic idea of stacked generalization is to perform cross-validation on the base-level dataset in order to create a meta-level dataset. Then a meta-level model is trained on it. Finally, this system can generate better outputs than original system that is trained on the whole original dataset.

2 Background

Given a source sentence s, the goal of SMT is to find a target sentence t among all possible target strings t_1, that maximizes the probability:

$$t = \arg\max_{t_1} \{\mathrm{pr}(t_1 \mid s)\}$$

Where $\mathrm{pr}(t_1 \mid s)$ is the probability that t_1 is the translation of the given source string s. The target string t_1 is a machine translation for s. In meta-SMT, a monolingual two-side corpus consists of these machine translations along with correct human translations. So, given a machinery output t, the goal of meta-SMT is to find a target sentence $\hat{t}$, that maximizes this probability:

$$\hat{t} = \arg\max_{t_2} \{\mathrm{pr}(t_2 \mid t)\}$$

Where $\mathrm{pr}(t_2 \mid t)$ is the probability that t_2 is the correct final translation of the given machine translated string t and both of t_2 and t are in the same language. The target sentence $\hat{t}$ is a final machine translation for s.

In the next section, we are going to describe stacked generalization for classification tasks and in section 3, we present a general solution to adapting this algorithm to SMT.

2.1 Stacking for Classification

Wolpert (1992) introduced a novel approach for combining multiple classifiers, known as stacked generalization or stacking. The key idea is to learn a meta-level (or level-1) classifier based on the output of base-level (or level-0) classifiers, estimated via cross-validation as follows:

Define $D = \{(x_i, y_i), i = 1,...,K\}$ as a data set, also referred to as level-0 data, where x_i is a feature vector representing the n th instance and y_i is the class value, and $L_1...L_N$ a set of different learning algorithms. During a J-fold cross-validation process, D is randomly split into J disjoint almost equal parts $D_1,...,D_J$. Define D^j and $D \setminus D^j$ to be the test and training sets for j th fold of a J-fold cross-validation. At each j th fold, $j = 1...J$, given the $L_1...L_N$ learning algorithms, we invoke each of them on the data in the training set $D \setminus D^j$ to induce classifiers $C_1(j)...C_N(j)$. Then, these classifiers are applied to the test part D^j. The concatenated predictions of the induced classifiers on each feature vector x_i in D^j, together with the original class value $y_i(x_i)$, form a new MD^j of meta-level vectors.

At the end of the entire cross-validation process, the union $MD = \bigcup_j MD^j, j = 1...J$, constitutes the full meta-level data set, also referred to as level-1 data, which is used for applying a learning algorithm L_M and inducing the meta-level classifier C_M. The learning algorithm that is employed at meta-level could be one of the $L_1...L_N$ or a different one. Finally, the learning algorithms are applied to the entire data set D inducing the final base-level classifiers $C_1...C_N$ to be used at runtime. In order to classify a new instance, the concatenated predictions of all base-level classifiers $C_1...C_N$ form a meta-level vector that is assigned a class value by the meta-level classifier C_M. In the next section we adapt this framework to SMT.

3 Adapting Stacking to SMT

Stacking is composed of two phases and we adapt it to SMT as follows:

First, a typical *SMT paradigm* is trained using *J*-fold cross-validation based on a bilingual cor-

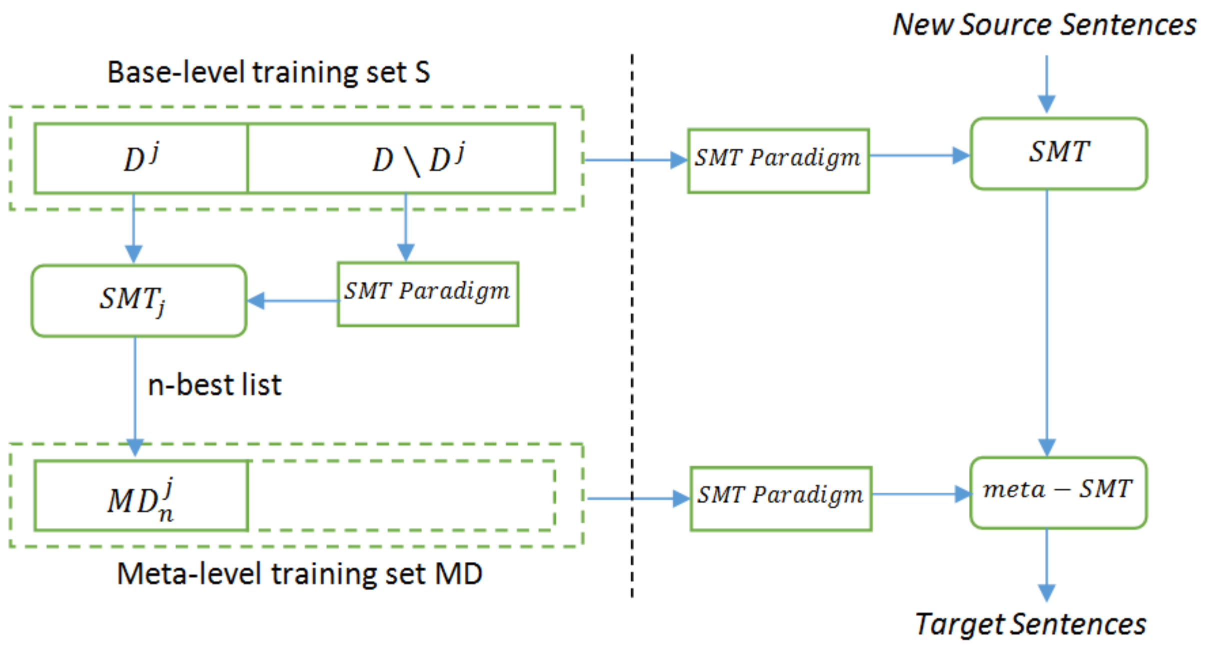

Figure 1. Stacking framework for SMT.

-pus. A popular setting of J is 5 and in this case it is called as *5-fold cross-validation*. We use this setting in our work. At the end of this step, 5 different systems are built based on 5 different training sets, called $SMT_j, j = 1,...,5$. Then, the n-best outputs of these systems are collected to create a new corpus called meta-level corpus MD. For example, if we have a training corpus of N sentences and we use 3-best outputs of each system in cross-validation process, the effective size of the meta-level corpus will be $3 \times N$. In this new corpus, all the generated translations from the source sentences are paired to correct human translations.

Second, this corpus is used with another SMT paradigm -we call it *meta-SMT*- that could be identical to the SMT paradigm we used in cross-validation process or another SMT paradigm, in order to provide the final translation. In this algorithm, any SMT paradigm could be used in base-level and meta-level SMTs such as phrase-based SMT, hierarchical phrase-based SMT and syntax-based SMT. In this work, we utilize a phrase-based SMT for both base-level and meta-level. We use a phrase-based model for meta-level SMT, because we are supposed to improve a single SMT system. In addition, we train another phrase-based SMT based on full training set to produce target 1-best outputs and then use these outputs as input test set for meta-level SMT.

Figure 1 (left side) illustrates the cross-validation methodology, while Figure 1 (right side) illustrates the stacking framework at runtime. Figure 2 also shows the algorithm in details.

3.1 Training base-level SMTs

After splitting the whole training corpus to separate training and test sets during cross-validation process, we train 5 phrase-based SMT systems on the training part and obtain the result of these systems on the corresponding test sets. We need these results for the next step.

3.2 Training meta-level SMTs

We gathered the n-best outputs of base-level SMTs on the corresponding test sets and built a meta-level corpus using these outputs along with correct human translations which was available from our original corpus. Then a meta-level SMT is trained on this corpus. We train our meta-SMT system on 10 meta-level corpus which is progressively created from n-best outputs of base-level systems, $n = 1,...,10$; i.e. each meta-level corpus that is created from n-best list also contains $(1...n-1)$-best list. In the results, we call these systems as meta-SMT (1-best) and meta-SMT (2-best) and so on.

3.3 Tuning meta-level SMTs

We must tune our meta-SMTs in a principled way. Similar to the training step, after splitting the whole tuning corpus to separate tuning and test sets during cross-validation process, we tune

```
Input: Training set $D = \{(e_1, f_1), (e_2, f_2), \dots, (e_d, f_d)\}$;
       Tuning set $T = \{(e'_1, f'_1), (e'_2, f'_2), \dots, (e'_t, f'_t)\}$;
       Test set $S = \{(e''_1, f''_1), (e''_2, f''_2), \dots, (e''_s, f''_s)\}$;
       Base-level and Meta-Level SMT paradigms Bparadigm and Mparadigm, respectively.
Process:
  ▪ $baseline\_SMT = Bparadigm(D)$  % Train baseline SMT on the whole training set.
  ▪ $baseline\_SMT = baseline\_SMT(T)$  % Tune baseline SMT on the whole tuning set.
  ▪ $T_1 = baseline\_SMT(S)$  % Test baseline SMT on the original test set.
  ▪ J-fold cross-validation: Divide the training and tuning sets into $J$ roughly equal parts.
  ▪ $MD = \phi$;  % Generate a new training set.
  ▪ $MT = \phi$;  % Generate a new tuning set.
  for $j = 1, \dots, J$:
      $SMT_j = Bparadigm(D \setminus D^j)$  % Train base-level SMTs by their training parts $D \setminus D^j$.
      $MD_n^j = SMT_j(D^j)$  % Test base-level SMTs on the corresponding test set to obtain n-best list.
      $MD = MD \cup MD_n^j$  % Collect the outputs of base-level SMTs to create meta-level training corpus.
      $SMT_j = SMT_j(T \setminus T^j)$  % Tune base-level SMTs by their tuning parts $T \setminus T^j$.
      $MT^j = SMT_j(T^j)$  % Test base-level SMTs on the corresponding test set of tuning set.
      $MT = MT \cup MT^j$  % Collect the outputs of base-level SMTs to create meta-level tuning corpus.
  end;
$meta-SMT = Mparadigm(MD)$  % Train meta-level SMT by applying it to the new training corpus.
$meta-SMT = meta-SMT(MT)$  % Tune meta-level SMT by applying it to the new tuning corpus.
Output: $T_2 = meta-SMT(T_1)$  % Test meta-SMT on the outputs of baseline SMT as test set.
```

Figure 2. Stacking algorithm adapted to SMT

5 base-level SMT systems on the tuning part and obtain the result of these systems on the corresponding test sets. Finally a meta-level development set is created by gathering these outputs paired with correct human translations to tune meta-level SMTs.

4 Experiments

4.1 Data

The corpus that is used for training and cross-validation process is Verbmobil project corpus which includes some tourists' conversations about time scheduling and appointment settings in German and English (Ney, 2000). Then a large part of English sentences are translated to Persian by human translators to build an English-Persian corpus (Bakhshaei et al., 2010). This dataset includes 23K lines in both sides, 249K and 216K words in Persian and English sides, respectively. We have chosen this corpus because it is small enough to perform cross-validation.

4.2 Experimental Setup

We use GIZA++ (Och and Ney, 2000) to perform the bi-directional word alignment between source and target side of each sentence pair. The final word alignment is generated using the *grow-diag-final-and* symmetrizing strategy. To speed up alignment, all the sentences with more than 80 words are removed. A 3-gram language model is trained on the target side of the bilingual data using the SRILM toolkit (Stolcke, 2002). The translation quality is evaluated in terms of case-insensitive BLEU metric.

We have run a phrase-based statistical machine translation with the Moses decoder (Koehn et al., 2007) to build baseline, base-level and meta-level SMTs.

We use MERT (Och, 2003) to tune the feature weights on the development data.

4.3 Evaluation

We investigate the effectiveness of our approach on improving a phrase-based SMT system. BLEU scores are computed on 250 test sentences

Type of SMT	Test set
baseline SMT	30.47
meta-SMT (1-best)	31.20
meta-SMT (2-best)	31.00
meta-SMT (3-best)	31.37
meta-SMT (4-best)	**31.49**
meta-SMT (5-best)	**31.41**
meta-SMT (6-best)	31.05
meta-SMT (7-best)	31.19
meta-SMT (8-best)	**31.40**
meta-SMT (9-best)	31.30
meta-SMT (10-best)	**31.54**

Table 1: BLEU (%) scores of baseline SMT and meta-SMTs on the Verbmobil test set.

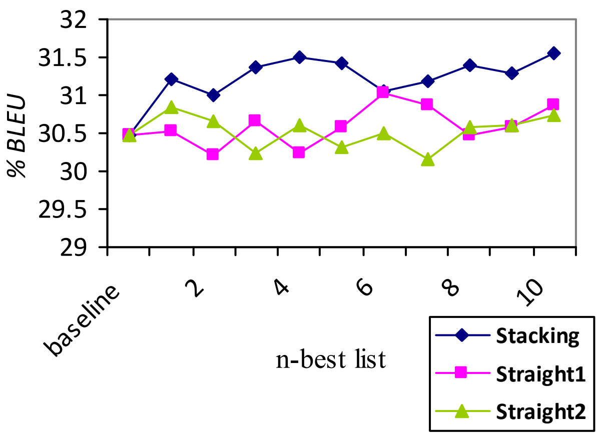

Figure 3. Comparison of Stacking, Straight1 and Straight2.

with four reference translations. Table 1 shows the results of our approach against baseline SMT on the test set. We see that almost all meta-SMTs are achieved over 0.5 BLEU improvement on the test set. The biggest improvement is obtained by meta-SMT (10-best) by 1.07 BLEU improvement. Moreover, we see a similar behavior of our approach on the development set. Considering the results on the development set, meta-SMT (4-best) and meta-SMT (8-best) will be good choices for meta-level SMT.

While these results are very encouraging, we must investigate why this approach is helpful. We find that two factors possibly contribute to these results. first, performing cross-validation on the training set; second, and possibly more importantly, the re-optimization on the system. In order to verify whether the improvements are due to the cross-validation or re-optimizing, we perform two experiments. The first is to test this approach without any cross-validation process, but with the development set obtained from stacking. It means that all source side sentences of the training corpus are translated by using an SMT system that is trained on the bilingual same corpus. This experiment is referred to as Straight1 in the results. The second is to build meta-level SMTs tuned with a development set which is obtained directly from baseline SMT (i.e., without performing cross-validation on it. This experiment is referred to as Straight2 in the results.

A comparison between the results of the three settings (Stacking, Straight1 and Straight2) is shown in Figure 3. The figure shows BLEU curve on the test set, where the X-axis is the number of base-level outputs (n-best) that is used to create meta-level corpus for meta-SMT, and the Y-axis is the BLEU scores of the final meta-SMT calculated from each approach. After analyzing the results, it can be concluded that both factors, i.e., cross-validation and re-optimizing the system with the stacking-based development set, are important to outperform the baseline SMT system. Since use of both factors, consistently lead to the best results.

In all of the experiments, the size of the n-best list varies from 1 up to 10. The main reason for the upper limit is just that the experiments are very time consuming.

We conducted statistical significance tests using *paired bootstrap resampling* proposed by Koehn (2004) to measure the reliability of the conclusion that meta-SMTs are really better than baseline SMT. It is observed that all stacking-based meta-SMTs are really better than the baseline SMT in 99% of the times.

5 Related Work

Stacking is a machine learning (ML) algorithm that is a well-studied topic in the ML community (Wolpert, 1992; Breiman, 1996a), and has been successfully adapted in natural language processing and information retrieval , such as named entity recognition (Wu et al., 2003) and Information extraction (Sigletos et al., 2005).

There are also some researches on applying ensemble learning algorithms into SMT. Xiao et al. (2010) presented a general solution for adaption of bagging and boosting to SMT. The results of their work showed that ensemble learning algorithms are promising in SMT.

Most other researches are in the statistical post-editing (SPE) techniques which have been used successfully to improve the output of Rule-

Based MT (RBMT) systems. Simard et al. (2007a), trained a "mono-lingual" Phrase-based SMT system (the *Portage* system) on the output of an RBMT system for the source side of the training set of the Phrase-based SMT system and the corresponding human translated (manually post-edited) reference. More recently, Béchara et al. (2011) designed a full phrase-based SMT pipeline that included a translation step and a post-editing step. The authors report significant improvements of 2 BLEU points for a French to English translation task, using a novel context aware approach. This method takes into account the source sentences during the post-editing process through a word-to-word alignment between the source words and the target words generated by the translation system.

As far as we are aware, the research presented in this paper is the first attempt to apply stacking algorithm to SMT with the configuration presented.

6 Conclusion and Future Work

We have presented a simple and effective approach to translation error modification by building a meta-level SMT using a meta-level corpus that is created form original corpus by cross validation. Experimental results showed that such a meta-SMT can fix many translation errors that occur in the baseline translations. The proposed method outperforms the baseline SMT on the same test set. We also believe that stacked generalization can be used to combine multiple SMT systems. As a future work, we have planned to develop a technique for combining multiple SMT systems using stacked generalization algorithm.

Moreover, we are running more tests with different language-pairs and larger corpora. We also have planned to use the confusion network methods on the input of meta-level SMTs, so that the meta-SMT can translate a confusion network built based on the n-best output of baseline SMT. As another future work, we will apply our framework under different SMT paradigms such as hierarchical phrase-based SMT and syntax-based SMT.

Acknowledgments

We would like to thank Somayeh Bakhshaei and Abbas Masoumzadeh for their helpful discussions.

References

Almut Silja Hildebrand and Stephan Vogel. 2008. Combination of machine translation systems via hypothesis selection from combined n-best lists. *In Proc. of the 8$_{th}$ AMTA conference*, pages 254-261.

Evegeny Matusov, Nicola Ueffing and Hermann Ney. 2006. Computing consensus translation from multiple machine translation systems using enhanced hypotheses alignment. *In Proc. of EACL 2006*, pages 33-40.

Antti-Veikko Rosti, Spyros Matsoukas and Richard Schwartz. 2007. Improved word-level system combination for machine translation. *In Proc. of the 45th Annual Meeting of the Association for Computational Linguistics*, pages. 312-319.

Michel Simard, Cyril Goutte, and Pierre Isabelle. 2007a. Statistical phrase-based post-editing. *In NAACL-HLT*, pages 508-515

Béchara, H., Y. Ma, and J. van Genabith. 2011. Post-editing for a statistical MT system. *In MT Summit XIII*, pages 308-315

David H. Wolpert. 1992. Stacked generalization. *Neural Networks*, 5(2): 241-259.

Leo Breiman. 1996b. Bagging predictors. *Machine Learning*, 24(2):123-140.

Robert E. Scharpire. 1990. The strength of weak learnability. *Machine Learning*, 5(2):197-227.

Michael P. Perrone and Leaon N. Cooper. 1993. When networks disagree: ensemble methods for hybrid neural networks. *Neural Networks for Speech and Image Processing*. Chapman-Hall, Chapter 10.

Clive W.J Granger. 1989. Combining forecasts-twenty years later. *Journal of Forecasting*, 8(3):167-173.

Tong Xiao, Jingbo Zhu, Muhua Zhu and Huizhen Wang. 2010. Boosting-based system combination for machine translation. *in Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 739–748.

Hermann Ney, Franz J. Och and Stephan Vogel. 2000. Statistical translation of spoken dialogues in the verbmobil system. *In Workshop on Multi-Lingual Speech Communication*, pages 69-74.

Somayeh Bakhshaei, Shahram Khadivi and Noushin Riahi. 2010. Farsi-German statistical machine translation through bridge language. *Telecommunications (IST), 5th International Symposium on*, pages 557-561.

Franz Josef Och and Hermann Ney. 2000. Improved statistical alignment models. *In Proc. of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 440-447.

Andreas Stockle. 2002. SRILM - an extensible language modeling toolkit. *In Proc. of International Conference for Spoken Language Processing*, pages 901-904.

P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin and E. Herbst. 2007. Moses: Open source toolkit for statistical machine translation. *In Proc.of the 45th Annual Meeting of the Association for Computational Linguistics*. Demo and Poster Sessions, pages. 177-180.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. *In Proc. of the 41th Annual Meeting of the Association for Computational Linguistics*, pages 160-167.

Philip Koehn, 2004. Statistical significance tests for machine translation evaluation. *In Proc. of Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 388-395.

Leo Breiman. 1996a. Stacked regressions. *Machine Learning*, 24(1): 49-64.

Dekai Wu, Grace Ngai and Marine Carpuat. A stacked, voted, stacked model for named entity recognition. *In Proc. of CoNLL-2003*, pages 200-203.

Georgios Sigletos, Georgios Paliouras, Constantine D. Spyropoulos and Michali Hatzopoulos. 2005. Combining information extraction systems using voting and stacked generalization. *Journal of Machine Learning Research*, 6: 1751-1782.

Tong Xiao, Jingbo Zhu and Tongran Liu. 2013. Bagging and boosting statistical machine translation systems. *Artificial Intelligence.* vol. 195, pages 496-527.

Kai Ming Ting and Ian H. Witten. 1999. Issues in stacked generalization. *Journal of Artificial Intelligence Research (JAIR)*, 10: 271-289.

Zhi-Hua Zhou. 2012. *Ensemble Methods: Foundations and Algorithms.* Chapman & Hall/CRC

Michel Simard, Nicola Ueffing, Pierre Isabelle, and Roland Kuhn. 2007b. Rule-based translation with statistical phrase-based post-editing. *In Proc. of WMT07*, pages 203-206