A Machine-Learning Framework for Hybrid Machine Translation

Christian Federmann

Language Technology Lab,
German Research Center for Artificial Intelligence,
Stuhlsatzenhausweg 3, D-66123 Saarbrücken, Germany
cfedermann@dfki.de

Abstract. We present a Machine-Learning-based framework for hybrid Machine Translation. Our approach combines translation output from several black-box source systems. We define an extensible, total order on translation output and use this to decompose the n-best translations into pairwise system comparisons. Using joint, binarised feature vectors we train an SVM-based classifier and show how its classification output can be used to generate hybrid translations on the sentence level. Evaluations using automated metrics shows promising results. An interesting finding in our experiments is the fact that our approach allows to leverage good translations from otherwise bad systems as the combination decision is taken on the sentence instead of the corpus level. We conclude by summarising our findings and by giving an outlook to future work, e.g., on probabilistic classification or the integration of manual judgements.

Keywords: Hybrid Machine Translation, System Combination, Machine Learning, Support Vector Machines, Feature-Based Classification.

1 Introduction

Research on the automatic translation of written texts or *machine translation* (MT) has resulted in many different MT paradigms, each having individual strengths and weaknesses. Amongst others, there are:

a) *statistical machine translation* (SMT) which aims at learning translation probabilities from large amounts of parallel data, working on non-linguistic phrases;
b) *rule-based machine translation* (RBMT) which relies on hand-crafted parsers and grammars that transform a given input sentence into a foreign language translation output; and
c) *hybrid machine translation* approaches which focus on creating translations from several source systems, based on the assumption that the different MT paradigms' individual strengths and shortcomings are often complementary which implies that a clever combination of their translations would yield an overall better translation output.

Regardless of the actual methodology of a given MT system, the production of the translations usually involves a lot of heterogeneous features. These range from simple language model scores, parser or phrase table probabilities, and confidence estimates to hierarchical parse trees or even full parse forests. This makes it very difficult to *intuitively* understand the inner workings of the MT engine in question; it is hence clear that research on the optimal combination of different machine translation systems into better, hybrid MT systems is of utmost importance to the field. To overcome the problem of incomprehensible feature values, we propose a method that applies Machine Learning (ML) tools, leaving the exact interpretation and weighting of features to the ML algorithms.

The remainder of this paper is structured as follows. After having introduced the matter of investigation in this section, we present related work in Section 2 before defining and explaining in detail our Machine-Learning-based framework for hybrid MT in Section 3. We first give an overview on the basic approach in Section 3.1 and then discuss its most important components: the extensible, total order on translations is defined in Section 3.2 while the notion of joint, binarised feature vectors for Machine Learning is introduced in Section 3.3. In Section 4 we present the experiments we have conducted in order to measure the proposed method's performance; results from this assessment are presented in Section 5. We conclude by summarising our findings and by discussing future research questions in Section 6.

2 Related Work

Hybrid translation approaches and system combination methods have received a lot of research attention over the last decade. There is general consensus that it is possible to combine translation output from different systems reaching an improvement over the individual baseline systems, e.g., [8,15,20].

Confusion Networks can be used for system combination [4,7,16]. One of the MT systems is chosen to become the *backbone* or *skeleton* of the hybrid translation, while other translations are connected via word alignment techniques such as GIZA++ [17]. Together, the systems then form a network with different paths through the network resulting in different translations. An open-source system combination toolkit like this is described in [2].

As the combination of translation output using phrase-based methods may not preserve the syntactic structure of the translation backbone, there also are methods which perform *Sentence-based Combination*, trying to select the best of several black-box translations for a given source text. This is similar to *Re-ranking Approaches* in SMT. See [1,9,20].

Finally, there are *Machine-Learning-based Combination* methods which train classifiers using, e.g., Support Vector Machines [22] to determine if a translation output is good or bad. Recent work such as [11,12] applies Machine Learning tools to estimate translation quality and re-rank a given set of candidate translations on the sentence level. Of course, there also exist various combinations of the aformentioned methods, e.g., [1,18].

3 Methodology

3.1 Classification-Based Hybrid Machine Translation

In this section, we describe a Machine-Learning-based framework for hybrid machine translation. Given a set of n translations from several, black-box systems and a tuning set including reference text, we perform the following steps to produce a hybrid translation for some given test set:

1. Compute a system ranking on the tuning set using some order relation based on quality assessment of the translations with automatic metrics. This can be extended to also include results from manual evaluation;
2. Decompose the aforementioned system ranking into a set of pairwise comparisons for any two pairs of systems A, B. As we do not allow for ties in our system comparisons, the two possible values $A > B$, $A < B$ also represent our *Machine-Learning classes* $+1/-1$, respectively;
3. Annotate the translations with feature values derived from natural language processing tools such as *language models*, *part-of-speech taggers*, or *parsers*;
4. Create a data set for training an SVM-based classifier that can estimate which of two systems A, B is *better* according to the available features[1];
5. Train an SVM-based classifier model using, e.g., `libSVM`, see [3];

Steps 1–5 represent the *training phase* in our framework. They require the availability of a tuning set including references to allow the definition of the order relation which subsequently defines the training instances for the SVM-based classifier. After training, we can use the classifier as follows:

6. Apply the resulting classification onto the candidate translations from the given test set. This will produce pairwise estimates $+1/-1$ for each possible combination of systems A, B;
7. Perform *round-robin playoff* elimination to determine the single-best system from the set of candidate translations on a per sentence level[2];
8. Synthesise the final, hybrid translation output.

Steps $6-8$ represent the *decoding phase* in which the trained classifier is applied to a set of *unseen* translations without any reference text available. By computing pairwise *winners* for each possible pair of systems and each individual sentence of the test set, we determine the single-best system on the sentence level. This is similar to SVMRank[3] and it will be an interesting extension of the work described in this paper comparing the performance of the two approaches.

[1] We had used decision trees learnt on annotated data in previous work with moderate success. For the experiments reported in this paper, we hence focused on SVM-based classification instead in order to learn about its performance in our problem setting.

[2] This may introduce problems such as loss of contextual information. For the work described in this paper, we ignore any such problems leaving them to future work.

[3] See `http://cs.cornell.edu/people/tj/svm_light/svm_rank.html`

3.2 An Extensible, Total Order on Translations

In order to rank the given source translations, we first need to define an *ordering relation* over the set of translation outputs. For this, we apply three renowned MT evaluation metrics which are the *de-facto standards* for automated assessment of machine translation quality. We consider:

1. The Meteor score, both on the sentence and on the corpus level, see [5];
2. The NIST n-gram co-occurence score on the corpus level, see [6]; and
3. The BLEU score which is the most widely used evaluation metric, see [19].

While both the BLEU and the NIST scores are designed to have a high correlation with judgements from manual evaluation on the corpus level (denoted by suffix $_C$), the Meteor metric can also be used to meaningfully compare translation output on the level of individual sentences (denoted by suffix $_S$). We make use of this property when defining our order $ord(A, B)$ on translations, as shown in Equations 1–5:

$$ord_{BLEU_C}(A,B) \overset{\text{def}}{=} \begin{cases} 1 & \text{if } A_{BLEU_C} > B_{BLEU_C} \\ -1 & \text{if } A_{BLEU_C} < B_{BLEU_C} \\ 0 & \text{else} \end{cases} \tag{1}$$

$$ord_{NIST_C}(A,B) \overset{\text{def}}{=} \begin{cases} 1 & \text{if } A_{NIST_C} > B_{NIST_C} \\ -1 & \text{if } A_{NIST_C} < B_{NIST_C} \\ ord_{BLEU_C}(A,B) & \text{else} \end{cases} \tag{2}$$

$$ord_{Meteor_C}(A,B) \overset{\text{def}}{=} \begin{cases} 1 & \text{if } A_{Meteor_C} > B_{Meteor_C} \\ -1 & \text{if } A_{Meteor_C} < B_{Meteor_C} \\ ord_{NIST_C}(A,B) & \text{else} \end{cases} \tag{3}$$

$$ord_{Meteor_S}(A,B) \overset{\text{def}}{=} \begin{cases} 1 & \text{if } A_{Meteor_S} > B_{Meteor_S} \\ -1 & \text{if } A_{Meteor_S} < B_{Meteor_S} \\ ord_{Meteor_C}(A,B) & \text{else} \end{cases} \tag{4}$$

$$ord(A,B) \overset{\text{def}}{=} ord_{Meteor_S}(A,B) \tag{5}$$

Note that the we called our order $ord(A, B)$ an *extensible, total order*. It can easily be extended to include, e.g., results from manual evaluation of translation output. In fact, this would be a very welcome addition as it would allow to bring in knowledge from domain experts. However, as a manual annotation campaign for n systems is both very time-consuming and expensive, we leave the integration of manual judgements into our ordering relation to future work.

3.3 Machine Learning Using Joint, Binarised Feature Vectors

As previously mentioned in Section 2, many Machine-Learning-based approaches for system combination use classifiers to estimate the quality or *confidence* in an individual translation output and compare it to other translations afterwards. This means that the feature vector for a given translation A is computed solely on information available in A, not considering any other translation B. Formally, this can be expressed by defining $vector_{single}(A) \in \mathbb{R}^n$ as follows:

$$vec_{single}(A) \overset{\text{def}}{=} \begin{pmatrix} feat_1(A) \\ feat_2(A) \\ \vdots \\ feat_n(A) \end{pmatrix} \tag{6}$$

We take a different approach here and compute feature vectors for all possible, *pairwise comparisons* of translations A, B, storing *binary feature values* to model if a given feature value $feat_x(A)$ for system A is better or worse than the corresponding feature value $feat_x(B)$ for the competing system B. Effectively, this means that we compare translations *directly* when constructing the set of feature vectors required for training our ML classifier. Equation 7 shows our definition of a *joint, binarised* feature $vector_{joint}(A, B) \in \mathbb{B}^n$:

$$vec_{joint}(A, B) \overset{\text{def}}{=} \begin{pmatrix} feat_1(A) > feat_1(B) \\ feat_2(A) > feat_2(B) \\ \vdots \\ feat_n(A) > feat_n(B) \end{pmatrix} \tag{7}$$

The reason to store binary features values $feat_x \in \mathbb{B}$ lies in the fact that these can be processed more efficiently during SVM training. Also, previous experiments have shown that using the actual feature values $feat_x \in \mathbb{R}$ does not give any additional benefit so that we decided to switch to binary notation instead. We have also run experiments using only joint feature vectors by concatenating feature values for systems A, B and using the corresponding feature vectors of length $2 * n$ for classifier training. As the comparison $A > B$ between systems could not be taken using the resulting classifier we started investigating the binarised model.

Note that the order in which features for translations A, B are compared does not strictly matter. For the sake of consistency, we have decided to compare feature values using simple $A > B$ operations, leaving the actual interpretation of these values or their polarity to the Machine Learning toolkit. This assumes that feature values are ordered in some way; which holds for the selected set of features. In future work, we plan to extend this introducing *feature-specific comparison* operators.

3.4 Feature Set for Training a Binary Classifier

We create the data set for classifier training using a selection of *features*. While there are many features which could be added to this feature set, we restricted ourselves to the following choice, leaving changes to future work:

1. Float values $\in [0.0, 1.0]$
   - ratio of target/source tokens;
   - ratio of target/source parse tree nodes;
   - ratio of target/source parse tree depth;
2. Integer values $\in \{0, 1, \ldots, \}$
   - number of target tokens;
   - number of target parse tree nodes;
   - number of target parse tree depth;
3. *log* probabilities
   - n-gram score for order $n \in \{1, \ldots, 5\}$;
   - inverse phrase translation probability $\rho(f|e)$
4. perplexity for order $n \in \{1, \ldots, 5\}$.

The selected features represent a combination of (shallow) parsing and language model scoring and are derived from the set of features that are most often used in the Machine-Learning-based system combination literature [1,9,11,12,18].

3.5 Creating Hybrid Translations Using an SVM Classifier

Given an SVM classifier trained on joint, binary feature vectors as previously described, we can now create hybrid translation output. The basic algorithm is depicted in Figure 1. It estimates the single-best translation for each sentence in the test set, based on the $+1/-1$ output of the classifier.

For each sentence, we create a dictionary that stores for some system X the set of systems which were outperformed by X according to our classifier. To do so, we consider each pairwise comparison of systems A, B and compute the corresponding feature vector which is then classified by the SVM. Only systems winning at least once in these pairwise comparisons end up as keys in our dictionary. The cardinality of the set of outperformed systems implicitly represents the number of wins for a system X.

Finally, we compute the single-best translation for a sentence by sorting the `system_wins` dictionary so that systems with a larger number of wins come first. There are three cases to consider:

1. If there is only one top-ranked system, this becomes the winning translation for the current sentence;
2. If two systems are top-ranked, the winner depends on the comparison of these. As we do not allow for ties in our comparisons, this is guaranteed to determine a single winner;
3. If more than two systems are top-ranked, we check if one of the systems outperforms the others. This may not yield a unique winner, in which case we fall back to scoring the systems with $ord_{Meteor_C}(A, B)$, effectively using the corpus level system rankings obtained on the tuning set to reach a final decision on the best translation for the current sentence.

```
for s_id in 1..len(sentences):
  system_wins = {}
  for (A, B) in system_pairs:
    joint_feature_vector = compute_feature_vector(A, B, s_id)
    classification_result = classify(joint_feature_vector)
    if classification_result == "+1":
      system_wins[A].append(B)
    else:
      system_wins[B].append(A)
  compute_best_system(system_wins)
```

Fig. 1. Pseudo-code illustrating how an SVM classifier can be used to determine the single-best translation using round robin playoff elimination. This operates on the sentence level, `compute_best_system()` finally computes the system with most "wins" over the competing systems. If two systems A, B have scored the same number of wins, the algorithm falls back to the comparison of these two systems. As we do not allow for ties in our system comparisons, the algorithm is guaranteed to terminate and will always return the—*according to the classifier used*—single-best system for a sentence.

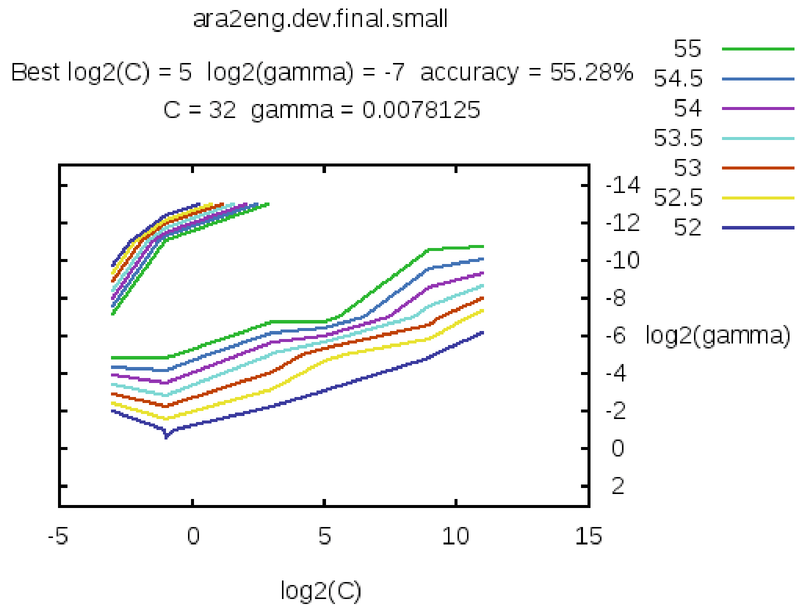
4 Experiments

In order to assess the performance of the proposed approach, we conduct several experiments and measure the translation quality of the resulting hybrid output. Note that in the data sets used for experimentation individual system names are anonymised as the translation output is part of a shared translation task.

We train SVM classifier models for two language pairs: Arabic→English and Chinese→English. For the first pair we work on translation output generated by $n = 10$ different systems, for the latter pair there are $n = 15$ systems to consider. The source text originates from the news domain.

As training data, we receive a tuning set including reference text as well as a test set without reference. We apply our order relation on the given translations to determine a system ranking on the sentence level. Using this information, we then compute pairwise system comparisons as SVM class labels and annotate individual translations with parser output and language model scores. We use the Stanford Parser [10,13,14] to process the source text and the corresponding translations. For language model scoring, we use the SRILM toolkit [21] training a 5-gram target language model for English. We do not consider source language language models in this work.

Figure 2 shows the optimisation grids we obtained during SVM tuning, using 5-fold cross validation an interval width *step* $= 1$, as implemented in *grid.py* from `libSVM`. They show which settings for C and γ result in the best prediction rate. Note how the graphs are similar regarding the *optimal area*. We train our final SVM classifiers with parameters from this area, giving preference to smaller values for both C and γ to reduce computational cost and thus training time.



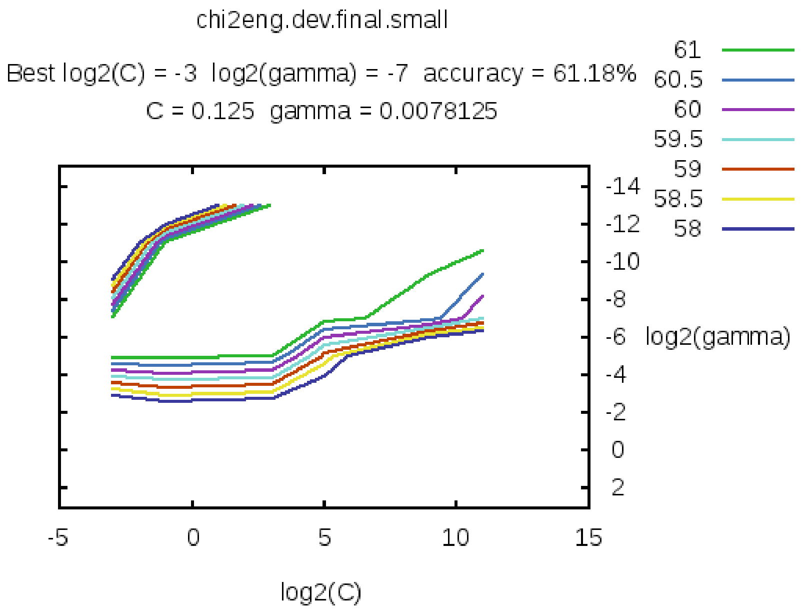


Fig. 2. Optimisation grids of SVM parameters C and γ for language pairs Arabic→English and Chinese→English. Note the similarity of the grids, indicating that our feature vectors are not depending on language-specific characteristics.

5 Evaluation

5.1 Automatic Metrics' Scores

We evaluated the translation quality of our hybrid MT system (referred to as *SVM-combo* in tables and figures) using BLEU and NIST, both of which are predominantly used in MT evaluation. Given the relatively small feature set and the sub-optimal prediction rate of the SVM classifier we achieve promising results for both language pairs.

For Arabic→English we achieved a NIST score of 10.3584 and a BLEU score of 0.4523. The single-best NIST score for the same language pair was 11.5046, for BLEU it was 0.4951 For Chinese→English we achieved a NIST and BLEU scores of 7.7636 and 0.2663, respectively, with single-best scores reported as 9.2705 and 0.3372. To correctly interpret these scores, it is important to note that our main focus lies on performance wrt. Meteor, as described in Section 3.2; here, the method performed better, with a result of 0.327 (best score 0.330) for Arabic→English and 0.307 (best score: 0.318) for Chinese→English.

5.2 System Contribution

Another interesting aspect related to the quality of the proposed method is the *system contribution* of the individual source systems. In order to better understand how much each of the systems added to the hybrid translation output, we compare the expected and the actual contribution in the translation result.

Expected contribution is computed as the average of the ranks assigned to each system by the metrics used in our order relation $ord(A, B)$, namely BLEU, NIST, and Meteor. Actual contribution is computed from the relative frequency of translations a system added to the hybrid translation output. We measure the difference between expected and actual contribution as Δ; positive values $+x$ mean that a system contributed more to the hybrid translation than we had expected, negative values $-y$ denote the opposite. Table 1 shows the comparison of system contribution for Arabic→English, table 2 for Chinese→English.

Table 1. System contribution for language pair Arabic→English

	System									
Rank	s01	s03	s05	s06	s07	s08	s11	s12	s16	s18
BLEU	3	10	7	2	1	6	8	9	4	5
NIST	2	10	6	1	3	5	7	9	4	8
Meteor	4	10	7	1	2	5	9	8	3	6
Expected	3	10	7	1	2	5	8	9	4	6
Combo	3	6	4	2	1	7	10	9	8	5
Δ		+4	+3	−1	+1	−2	+2		−4	+1

Table 2. System contribution for language pair Chinese→English

	System														
Rank	s01	s02	s04	s06	s07	s08	s09	s10	s12	s13	s14	s15	s16	s17	s18
BLEU	2	14	15	1	3	8	13	11	5	12	7	10	9	6	4
NIST	3	15	14	1	2	9	13	10	5	11	7	12	8	6	4
Meteor	4	15	14	1	2	9	13	11	6	12	7	10	8	5	3
Expected	3	15	14	1	2	9	13	10	5	12	7	10	8	6	4
Combo	1	14	7	9	11	8	13	6	2	3	4	14	12	10	5
Δ	+2	+1	+7	−8	−9	+1		+4	+3	+9	+3	−4	−4	−4	−1

6 Conclusion

6.1 Summary of Findings

We have described a Machine-Learning-based framework for hybrid MT. We defined a total order on translation output that can be applied during feature vector generation. Our method differs from previous work as we consider joint, binarised feature vectors instead of separate feature vectors for each of the source systems. We proposed an algorithm to make use of an SVM-based classifier trained on these feature vectors for the creation of hybrid translations.

In our experiments for language pairs Arabic→English and Chinese→English, we could observe promising results according to Meteor scores. We also analysed the expected contribution of the source systems to the hybrid translation output and found that our classifier was able to make use of good sentence translations from systems which performed bad on the corpus level. We observed interesting differences between expected and actual contribution ranks.

6.2 Outlook on Future Work

The total order on translation defined in Section 3 can be extended to make use of results from manual judgements regarding the quality of candidate translations. It is not yet clear how the SVM model's prediction rate and the quality of the final translation are interrelated or how changes of the former would alter the latter, leaving room for future research work. We also intend to invest effort into a large, manual evaluation campaign to examine in more detail what translations are selected by our combination method and if these are actually the best (or at least a good) translation considering the given source sentence.

Acknowledgments. This work has been funded under the Seventh Framework Programme for Research and Technological Development of the European Commission through the T4ME contract (grant agreement no.: 249119). The author would like to thank Sabine Hunsicker and Yu Chen for their support with the experiments, and Hans-Ulrich Krieger as well as Geert-Jan Kruijff for their helpful comments. We are grateful to the anonymous reviewers for their valuable feedback.

References

1. Avramidis, E.: DFKI System Combination with Sentence Ranking at ML4HMT-2011. In: Proceedings of the International Workshop on Using Linguistic Information for Hybrid Machine Translation (LIHMT 2011) and of the Shared Task on Applying Machine Learning Techniques to Optimise the Division of Labour in Hybrid Machine Translation (ML4HMT). META-NET, Barcelona (2011)
2. Barrault, L.: Many: Open source machine translation system combination. Prague Bulletin of Mathematical Linguistics, Special Issue on Open Source Tools for Machine Translation 1(93), 145–155 (2010), `http://www-lium.univ-lemans.fr/sites/default/files/Barrault-MANY2010.pdf`
3. Chang, C.C., Lin, C.J.: LIBSVM: A Library for Support Vector Machines. ACM Transactions on Intelligent Systems and Technology 2, 27:1–27:27 (2011), Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`
4. Chen, Y., Eisele, A., Federmann, C., Hasler, E., Jellinghaus, M., Theison, S.: Multi-engine machine translation with an open-source SMT decoder. In: Proceedings of the Second Workshop on Statistical Machine Translation, pp. 193–196. Association for Computational Linguistics, Prague (2007), `http://www.aclweb.org/anthology/W/W07/W07-0726`
5. Denkowski, M., Lavie, A.: Meteor 1.3: Automatic Metric for Reliable Optimization and Evaluation of Machine Translation Systems. In: Proceedings of the Sixth Workshop on Statistical Machine Translation, pp. 85–91. Association for Computational Linguistics, Edinburgh (2011), `http://www.aclweb.org/anthology-new/W/W11/W11-2107`
6. Doddington, G.: Automatic Evaluation of Machine Translation Quality Using n-gram Co-occurrence Statistics. In: Proceedings of the Second International Conference on Human Language Technology Research, HLT 2002, pp. 138–145. Morgan Kaufmann Publishers Inc., San Francisco (2002), `http://www.itl.nist.gov/iad/mig/tests/mt/doc/ngram-study.pdf`
7. Eisele, A., Federmann, C., Saint-Amand, H., Jellinghaus, M., Herrmann, T., Chen, Y.: Using Moses to integrate multiple rule-based machine translation engines into a hybrid system. In: Proceedings of the Third Workshop on Statistical Machine Translation, pp. 179–182. Association for Computational Linguistics, Columbus (2008), `http://www.aclweb.org/anthology/W/W08/W08-0328`
8. Frederking, R., Nirenburg, S.: Three Heads are Better Than One. In: Proceedings of the Fourth Conference on Applied Natural Language Processing, ANLC 1994, pp. 95–100. Association for Computational Linguistics, Stroudsburg (1994), `http://ww2.cs.mu.oz.au/acl/A/A94/A94-1016.pdf`
9. Gamon, M., Aue, A., Smets, M.: Sentence-level MT Evaluation Without Reference Translations: Beyond Language Modeling. In: Proceedings of the 10th EAMT Conference "Practical Applications of Machine Translation", pp. 103–111. European Association for Machine Translation (May 2005), `http://research.microsoft.com/research/pubs/view.aspx?pubid=1426`
10. Green, S., Manning, C.D.: Better Arabic Parsing: Baselines, Evaluations, and Analysis. In: Proceedings of the 23rd International Conference on Computational Linguistics, COLING 2010, pp. 394–402. Association for Computational Linguistics, Stroudsburg (2010), `http://dl.acm.org/citation.cfm?id=1873826`
11. He, Y., Ma, Y., van Genabith, J., Way, A.: Bridging SMT and TM with Translation Recommendation. In: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, ACL 2010, pp. 622–630. Association for Computational Linguistics, Stroudsburg (2010), `http://aclweb.org/anthology-new/P/P10/P10-1064.pdf`

12. He, Y., Ma, Y., Way, A., van Genabith, J.: Integrating N-best SMT Outputs into a TM System. In: Proceedings of the 23rd International Conference on Computational Linguistics, COLING 2010, pp. 374–382. Association for Computational Linguistics, Stroudsburg (2010), `http://doras.dcu.ie/15799/1/Integrating_N-best_SMT_Outputs_into_a_TM_System.pdf`
13. Klein, D., Manning, C.: Accurate Unlexicalized Parsing. In: Proceedings of the 41st Meeting of the Association for Computational Linguistics, ACL 2003, vol. 1, pp. 423–430. Association for Computational Linguistics, Stroudsburg (2003), `http://acl.ldc.upenn.edu/P/P03/P03-1054.pdf`
14. Levy, R., Manning, C.: Is it harder to parse Chinese, or the Chinese Treebank? In: Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics, ACL 2003, pp. 439–446. Association for Computational Linguistics, Stroudsburg (2003), `http://www.aclweb.org/anthology/P03-1056`
15. Macherey, W., Och, F.J.: An Empirical Study on Computing Consensus Translations from Multiple Machine Translation Systems. In: Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL), pp. 986–995. Association for Computational Linguistics, Prague (2007), `http://www.aclweb.org/anthology/D/D07/D07-1105`
16. Matusov, E., Ueffing, N., Ney, H.: Computing Consensus Translation from Multiple Machine Translation Systems Using Enhanced Hypotheses Alignment. In: Conference of the European Chapter of the Association for Computational Linguistics, pp. 33–40. Association for Computational Linguistics, Stroudsburg (2006), `http://acl.ldc.upenn.edu/E/E06/E06-1005.pdf`
17. Och, F.J., Ney, H.: A Systematic Comparison of Various Statistical Alignment Models. Computational Linguistics 29(1), 19–51 (2003), `http://acl.ldc.upenn.edu/J/J03/J03-1002.pdf`
18. Okita, T., van Genabith, J.: DCU Confusion Network-based System Combination for ML4HMT. In: Proceedings of the International Workshop on Using Linguistic Information for Hybrid Machine Translation (LIHMT 2011) and of the Shared Task on Applying Machine Learning Techniques to Optimise the Division of Labour in Hybrid Machine Translation (ML4HMT). META-NET, Barcelona (2011)
19. Papineni, K., Roukos, S., Ward, T., Zhu, W.J.: BLEU: A Method for Automatic Evaluation of Machine Translation. In: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, ACL 2002, pp. 311–318. Association for Computational Linguistics, Stroudsburg (2002), `http://acl.ldc.upenn.edu/P/P02/P02-1040.pdf`
20. Rosti, A.V., Ayan, N.F., Xiang, B., Matsoukas, S., Schwartz, R., Dorr, B.: Combining Outputs from Multiple Machine Translation Systems. In: Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference, pp. 228–235. Association for Computational Linguistics, Rochester (2007), `http://www.aclweb.org/anthology/N/N07/N07-1029`
21. Stolcke, A.: SRILM - An Extensible Language Modeling Toolkit. In: Proceedings of the International Conference on Spoken Language Processing, pp. 257–286 (November 2002)
22. Vapnik, V.N.: The Nature of Statistical Learning Theory. Springer-Verlag New York, Inc., New York (1995)