System Combination Using Joint, Binarised Feature Vectors

Christian FEDERMANN[1]
(1) DFKI GmbH, Language Technology Lab,
Stuhlsatzenhausweg 3, D-66123 Saarbrücken, GERMANY
cfedermann@dfki.de

Abstract

We describe a method for system combination based on joint, binarised feature vectors. Our method can be used to combine several black-box source systems. We first define a total order on given translation output which can be used to partition an *n*-best list of translations into a set of pairwise system comparisons. Using this data, we explain how an SVM-based classification model can be trained and how this classifier can be applied to combine translation output on the sentence level. We describe our experiments for the ML4HMT-12 shared task and conclude by giving a summary of our findings and by discussing future extensions and experiments using the proposed approach.

Keywords: Machine Translation, System Combination, Machine Learning.

Second ML4HMT Workshop, pages 77–84,
COLING 2012, Mumbai, December 2012.

1 Introduction

Research efforts on machine translation (MT) have resulted in many different methods and MT paradigms, each having individual strengths and weaknesses. There exist approaches following linguistic theory as well as data-driven methods relying on statistical processing with only little linguistic knowledge involved. In recent years, there has also been a lot of research on the automatic combination of machine translation output, resulting in so-called hybrid MT engines.

Regardless of the actual method implemented in a given machine translation system, creating translation output usually requires several, often heterogeneous, features. These can be 1) simple scores, e.g., for language model scores, parser or phrase table probabilities; 2) more complex data such as hierarchical parse trees or word alignment links; or 3) even full parse forests or n-best lists.

Given this wide range of heterogenous features and their diversity, it is very difficult to get an *intuitive* understanding of the inner workings of the MT engine in question; thus, further research work on the combination of machine translation systems into better, hybrid MT systems seems to be of high importance to the field. To overcome the aforementioned problem of incomprehensible feature values, we propose a method based on Machine Learning (ML) tools, leaving the exact interpretation and weighting of features to the ML algorithms.

The remainder of this paper is structured in the following way. After having set the topic in this section, we briefly describe relevant related work in Section 2 before defining and explaining our Machine-Learning-based hybrid MT framework in Section 3. We first give an overview on the basic approach in Subsection 3.1 and then discuss the two most important components: the order on translations is defined in Subsection 3.2 while the notion of joint, binarised feature vectors for ML is introduced in Subsection 3.3. We discuss the experiments conducted for the ML4HMT-12 shared task in Section 4 and then conclude by giving a summary of our findings and by discussing upcoming research in Section 5.

2 Related Work

Hybrid machine translation methods and system combination approaches have been receiving a lot of research attention over the last decade. Several papers, including seminal work from (Frederking and Nirenburg, 1994), support the general belief that it is possible to combine translation output from different systems achieving an improvement over the individual baseline systems.

System combination on the phrasal level can be realised using so-called *Confusion Networks*. Previous work on this approach are described in more detail in (Federmann et al., 2009; Federmann and Hunsicker, 2011). The algorithm chooses one of the given MT systems as *backbone* or *skeleton* of the hybrid translation, while all other translations are connected using word alignment techniques. Together, the systems then form a network with different paths through the network resulting in different translations.

Next to phrasal combination methods, there also are approaches that focus on preserving the syntactic structure of the translation backbone, and hence perform *Sentence-based Combination*. Here, several given black-box translations are re-scored or re-ranked in order to determine the best translation for a given sentence in the source text. This is similar to *Re-ranking Approaches* in SMT. See related work from (Avramidis, 2011), (Gamon et al., 2005), or (Rosti et al., 2007).

Finally, there are *Machine-Learning-based Combination* approaches which train classifiers to assess the quality of given translation output. Recent work (He et al., 2010) applies such Machine Learning tools to estimate translation quality and re-rank a set of translations on the sentence level.

3 Methodology

3.1 Classification-Based Hybrid MT

In this Section, we describe our architecture for hybrid machine translation. It is based on classifiers trained using state-of-the-art Machine-Learning tools. Given a set of n translations from several systems that are treated as "black boxes" and a development set including corresponding reference, we perform the following processing steps to generate a hybrid translation for some given test set:

1. Compute a total order of individual system output on the development set using some order relation based on quality assessment of the translations with automatic metrics. This can also be defined to include results from, e.g., 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 NLP 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 given systems A, B is *better* according to the available features;
5. Train an SVM-based classifier model using, e.g., `libSVM`, see (Chang and Lin, 2011);

Steps 1–5 represent the *training phase* in our framework. The availability of a development set including references is required as this is needed to allow the definition of an ordering 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 pair of systems A, B;
7. Perform *round-robin system elimination* to determine the single best system from the set of candidate translations on a per sentence level;
8. Using this data, 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. The methodology is explained in more detail in (Federmann, 2012b,c).

3.2 A 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 consider manual judgements wherever available and, as a fallback, also apply three renowned evaluation metrics which are the *de-facto standards* for automated assessment of machine translation quality:

1. The Meteor score, on the sentence and on the corpus level, see (Denkowski and Lavie, 2011);
2. The NIST n-gram cooccurence score on the corpus level, see (Doddington, 2002); and
3. The BLEU score which is the most widely used evaluation metric, see (Papineni et al., 2002).

While both the BLEU and the NIST scores are designed to have a high correlation with judgements from manual evaluation on the corpus level, the Meteor metric can also be used to meaningfully compare translation output on the level of individual sentences. We make use of this property when defining our order $ord(A, B)$ on translations, as described in (Federmann, 2012c).

3.3 Using Joint, Binarised Feature Vectors

Many Machine-Learning-based approaches for system combination use classifiers to estimate the quality of 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 from features of A, not considering any other translation B as additional source of information, or formally:

$$vec_{single}(A) \stackrel{\text{def}}{=} \begin{pmatrix} f_1(A) \\ \vdots \\ f_n(A) \end{pmatrix} \in \mathbb{R}^n \tag{1}$$

We aim to explicitly model *pairwise feature comparisons* of translations A, B, storing *binary values* to model if a given feature value $f_x(A)$ for system A is better or worse than corresponding feature value $f_x(B)$ for the competing system B. Effectively, this means that, in our approach, we compare translations *directly* when constructing the set of training instances. Equation 2 shows the formal definition of a so-called *joint, binarised* feature vector:

$$vec_{binarised}(A, B) \stackrel{\text{def}}{=} \begin{pmatrix} f_1(A) > f_1(B) \\ \vdots \\ f_n(A) > f_n(B) \end{pmatrix} \in \mathbb{B}^n \tag{2}$$

The reason to store binary features values $f_x \in \mathbb{B}$ lies in the fact that these can be processed more efficiently during SVM training. Also, previous experiments (Federmann, 2012a; Hunsicker et al., 2012) have shown that the usage of actual feature values $f_x \in \mathbb{R}$ in the feature vector does not give any additional benefit so that we decided to switch to binary notation instead[1]. 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.

3.4 Creating Translations Using a Classifier

Given an SVM classifier trained on joint, binary feature vectors as previously described, we can now create hybrid translation output. A schematic overview is depicted in Figure 1. We compute the best translation for each sentence in the test set, based on the $+1/-1$ output of the classifier for a total of $\frac{n(n-1)}{2}$ unique comparisons.

For each sentence, we create a lookup table 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, for each of these, compute the corresponding feature vector which is then

[1] Also note that by using, e.g., *combined feature vectors*, which are comprised of feature values $f_{1-n}(A)$ followed by features $f_{1-n}(B)$, the amount of training data required for meaningful training of a machine learning classifier would need to be increased.

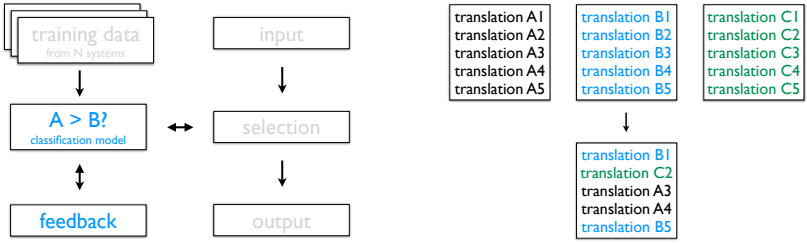

Figure 1: Schematic overview illustrating how an SVM classifier can be used to determine the single best translation using round robin playoff elimination. This operates on the sentence level.

classified by the SVM classifier. Only systems winning at least once during these comparisons end up as keys in our table. The cardinality of the resulting set of outperformed systems implicitly represents the number of wins for a system X. There are three cases to consider:

1. There is exactly one top-ranked system which becomes the translation for the current sentence;
2. Two systems are top-ranked, so the decision 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. In rare cases, this may not yield a winner and we have to fall back to a pre-defined winner, usually the best system from training.

4 Experiments

We worked on a submission for language pair Spanish→English. For this language pair, translation output from four different translation engines was made available by the organisers of the shared task. For each of the systems both translation output and system-specific annotations could be used. As our method relies on *comparable features*, we decided to extract features for all candidate systems ourselves, hence constraining ourselves to only using the given translation output.

We created the data set for classifier training using the following selection of linguistic features:

- number of target tokens, parse tree nodes, and parse tree depth;
- ratio of target/source tokens, parse tree nodes, and parse tree depth;
- n-gram score for n-gram order $n \in \{1,\ldots,5\}$;
- perplexity for n-gram order $n \in \{1,\ldots,5\}$.

These 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.

We use the Stanford Parser (Klein and Manning, 2003) to process the source text and the corresponding translations. For language model scoring, we use the SRILM toolkit (Stolcke, 2002) training a 5-gram language model for English. In this work, we do not consider any source language models.

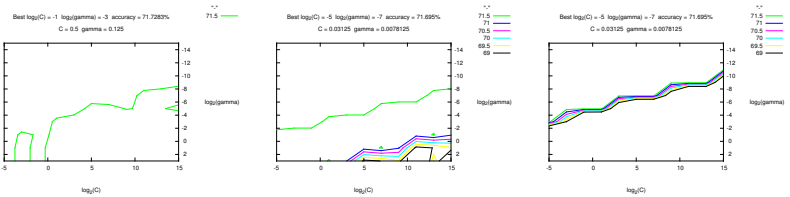

Figure 2: Optimisation grids for linear (left), polynomial (middle), and sigmoid (right) kernels. Note how the decision boundary of the optimal area manifests itself from left to right.

Figure 2 shows the optimisation grids for linear (left), polynomial (middle), and sigmoid (right) kernels. Note how the decision boundary of the optimal area manifests itself from left to right. We ended up using a sigmoid kernel ($C = 2$, $\gamma = 0.015625$) and observed a prediction rate of 68.9608% on the training instances.

5 Conclusion

We have described our submission to the ML4HMT-12 shared task which is based on a Machine-Learning-based framework for hybrid Machine Translation. Using so-called joint, binarised feature vectors, we implemented an algorithm that applies an SVM-based classifier to generate hybrid translations for the language pair Spanish→English. Combination is done on the sentence level.

Upcoming work will involve the refinement of our set of linguistic features and subsequent training and tuning of a Machine Learning classifier to improve the proposed method on additional data. We have already achieved promising baseline results in this respect and look forward to further test our approach, e.g., for ML4HMT-12's second language pair Chinese→English.

The total order on translation output described in Section 3 can be altered to also consider results from manual judgements regarding translation quality. This has not yet been used for our submission, but it would be an interesting extension of this paper.

Acknowledgments

This work has been supported by the Seventh Framework Programme for Research and Technological Development of the European Commission through the T4ME contract (grant agreement: 249119).

References

Avramidis, E. (2011). DFKI System Combination with Sentence Ranking at ML4HMT-2011. In *Proceedings of the International Workshop on Applying Machine Learning Techniques to Optimise the Division of Labour in Hybrid Machine Translation (ML4HMT)*, Barcelona, Spain. META-NET.

Chang, C.-C. and Lin, C.-J. (2011). LIBSVM: A Library for Support Vector Machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27.

Denkowski, M. and Lavie, A. (2011). Meteor 1.3: Automatic Metric for Reliable Optimization and Evaluation of Machine Translation Systems. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 85–91, Edinburgh, Scotland. ACL.

Doddington, G. (2002). 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 '02, pages 138–145, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Federmann, C. (2012a). Can Machine Learning Algorithms Improve Phrase Selection in Hybrid Machine Translation? In *Proceedings of the Joint Workshop on Hybrid Approaches to Machine Translation (HyTra)*, pages 113–118, Avignon, France. European Chapter of the ACL (EACL).

Federmann, C. (2012b). Hybrid Machine Translation Using Joint, Binarised Feature Vectors. In *Proceedings of the Tenth Biennial Conference of the Association for Machine Translation in the Americas (AMTA 2012)*, pages 113–118, San Diego, USA. AMTA.

Federmann, C. (2012c). A Machine-Learning Framework for Hybrid Machine Translation. In *Proceedings of the 35th Annual German Conference on Artificial Intelligence (KI-2012)*, pages 37–48, Saarbrücken, Germany. Springer, Heidelberg.

Federmann, C. and Hunsicker, S. (2011). Stochastic Parse Tree Selection for an Existing RBMT System. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 351–357, Edinburgh, Scotland. ACL.

Federmann, C., Theison, S., Eisele, A., Uszkoreit, H., Chen, Y., Jellinghaus, M., and Hunsicker, S. (2009). Translation Combination using Factored Word Substitution. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 70–74, Athens, Greece. ACL.

Frederking, R. and Nirenburg, S. (1994). Three Heads are Better Than One. In *Proceedings of the Fourth Conference on Applied Natural Language Processing*, ANLC '94, pages 95–100, Stroudsburg, PA, USA. ACL.

Gamon, M., Aue, A., and Smets, M. (2005). Sentence-level MT Evaluation Without Reference Translations: Beyond Language Modeling. In *Proceedings of the 10th EAMT Conference "Practical applications of machine translation"*, pages 103–111. EAMT.

He, Y., Ma, Y., van Genabith, J., and Way, A. (2010). Bridging SMT and TM with Translation Recommendation. In *Proceedings of the 48th Annual Meeting of the ACL*, ACL '10, pages 622–630, Stroudsburg, PA, USA. ACL.

Hunsicker, S., Yu, C., and Federmann, C. (2012). Machine Learning for Hybrid Machine Translation. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 312–316, Montréal, Canada. ACL.

Klein, D. and Manning, C. (2003). Accurate Unlexicalized Parsing. In *Proceedings of the 41st Meeting of the Association for Computational Linguistics*, volume 1 of *ACL '03*, pages 423–430, Stroudsburg, PA, USA. ACL.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). BLEU: A Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, ACL '02, pages 311–318, Stroudsburg, PA, USA. ACL.

Rosti, A.-V., Ayan, N. F., Xiang, B., Matsoukas, S., Schwartz, R., and Dorr, B. (2007). Combining Outputs from Multiple Machine Translation Systems. In *HLT 2007: The Conference of the North American Chapter of the ACL*, pages 228–235, Rochester, New York. ACL.

Stolcke, A. (2002). SRILM - An Extensible Language Modeling Toolkit. In *Proceedings of the International Conference on Spoken Language Processing*, pages 257–286.