Multi-Engine Machine Translation as a Lifelong Machine Learning Problem

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

Abstract

We describe an approach for multi-engine machine translation that uses machine learning methods to train one or several classifiers for a given set of candidate translations. Contrary to existing approaches in quality estimation which only consider a single translation at a time, we explicitly model pairwise comparison with our feature vectors. We discuss several challenges our method is facing and discuss how lifelong machine learning could be applied to resolve these. We also show how the proposed architecture can be extended to allow human feedback to be included into the training process, improving the system's selection process over time.

Introduction

This paper describes an approach for multi-engine machine translation (MT) that aims at training one or several machine learning (ML) classifiers which can be used to perform pairwise comparison of given candidate translations. Using these classifiers, we then compute a combined translation by selecting the best individual translations on the sentence level. After we have described the basic methodology of our MT approach, we identify challenges that are suitable for lifelong machine learning and discuss how these challenges could be tackled.

Machine translation is a challenging and complex task which has triggered research that resulted in several methods and paradigms, each of which has its individual strengths and weaknesses. While MT research had originally started with rule-based systems which aim at creating linguistic models of both source and target language, recently focus has shifted to more data-driven methods which instead aim at learning translation probabilities from massive amounts of parallel data. Next to research on the individual translation techniques there has also been work on hybrid MT, system combination, or multi-engine machine translation systems, based on the underlying assumptions that:

– Different paradigms have individual but *complementary* strengths and shortcomings;
– Clever combination of translation output from several engines should result in an improved translation.

The research described in this paper is conducted as part of the *T4ME* project where we want to "*provide a systematic investigation and exploration of the space of possible choices in hybrid machine translation, in order to provide optimal support for hybrid MT design, using sophisticated machine learning technologies*".

The remainder of this paper is structured as follows: after having introduced the matter of investigation in this section, we provide a quick overview on related research work. We then explain the system architecture and the key elements of the proposed methodology. In the following section we present several challenges our approach is faced with and discuss how lifelong machine learning methods could be used to overcome these issues. We also discuss how human feedback can be integrated into classifier training to improve translation quality over time. We conclude with a summary of our findings and by giving an outlook to future work in the final section.

Related Work

Multi-engine approaches and system combination methods have have received a lot of research attention over the last decade. Several papers, including seminal work from (Frederking and Nirenburg 1994), support the general assumption that it is possible to create combined translation output from different systems while achieving an improvement, in terms of translation quality, over the individual baseline systems. See, e.g., (Macherey and Och 2007) or (Rosti et al. 2007).

System combination on the phrasal level can be realised using so-called *confusion networks*. Systems following this approach are described in more detail in (Chen et al. 2007), (Eisele et al. 2008), and (Matusov, Ueffing, and Ney 2006). Here, 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 tools such as GIZA++ (Och and Ney 2003). The systems then form a connected graph in which different paths through the network encode different translations.

Next to phrasal combination approaches, there also are methods 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 which of these is the best translation for a given sentence in the source

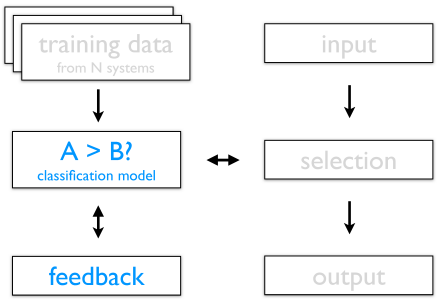

Figure 1: Schematic overview of our system architecture, highlighting where human feedback and/or lifelong machine learning methods can be added.

text. See related work from (Hildebrand and Vogel 2008), (Gamon, Aue, and Smets 2005), or (Rosti et al. 2007) for more information. The multi-engine combination methodology we present in this paper is explained in more detail in (Federmann 2012b; 2012c).

Of course, there also exist various combinations of the aforementioned methods for hybrid machine translation. As part of our research in the *T4ME* project, we have organised a workshop series on "Applying Machine Learning Techniques to Optimise the Division of Labour in Hybrid MT" (ML4HMT[1]); its results are described in (Federmann 2011).

Methodology

Our system combination method is based on classifiers trained using state-of-the-art machine learning tools. Given a set of n candidate translations from a set of N systems that are treated as "black boxes" and a development set including reference text, we perform the following processing steps to generate a combined translation for some given test set:

1. Compute a total order of translations on the development set using some order relation based on quality assessment with automatic metrics. This can also be defined such that it includes results from, e.g., manual evaluation;
2. Decompose the aforementioned system ranking into a set of pairwise comparisons for all possible pairs of systems A, B. As we do not allow for ties in our comparisons, the two possible values $A > B$, $A < B$ also represent our *machine learning classes* $+1/-1$, respectively;
3. Annotate the translations with linguistic 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);

[1] See `http://www.dfki.described/` for more information on the workshop and associated shared task.

Steps 1–5 represent the *training phase* in our approach. We require the availability of a development set including the corresponding reference text to generate training instances for the classification model. After training, we can use the resulting classifier as follows:

6. Produce estimates $+1/-1$ for each pair of systems A, B;
7. Perform a *round-robin tournament* to determine the single best system from the set of available translations on the sentence level.
8. Synthesise the final, combined 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 sentence of the test set, we determine the single best system on the sentence level, eventually leading to the final translation.

Ranking Candidate Translations

In order to rank the given candidate translations, we first have to define an *ordering relation* over the space of given translations. For this, we apply several evaluation metrics which are the *de-facto standards* for automated assessment of machine translation quality. We consider:

1. The Meteor score (Denkowski and Lavie 2011);
2. The NIST score (Doddington 2002); and
3. The BLEU score (Papineni et al. 2002).

While both BLEU and NIST scores are designed to have a high correlation with results from manual evaluation on the corpus level, the Meteor metric can also be used to compare translations on the level of individual sentences. We use this property when defining our order $ord(A, B)$ on translations, as shown in equations 1 and 2.

$$ord(A, B) \stackrel{\text{def}}{=} ord_X(A, B) \tag{1}$$

where $X \in \{Meteor_S, Meteor_C, NIST_C, BLEU_C, \perp\}$ and suffix $_S$ denotes a *sentence-level* quality metric score while suffix $_C$ represents a *corpus-level* score. $\perp$ denotes the "empty" metric which is the "minimal" element in the set of available metrics.

$$ord_X(A, B) \stackrel{\text{def}}{=} \begin{cases} +1 & \text{if} X_A > X_B \\ -1 & \text{if} X_A < X_B \\ ord_{X'}(A, B) & \text{otherwise}, X > X' \end{cases} \tag{2}$$

If candidates A, B are indistinguishable by current metric X, we *recursively* delegate the decision to metric X' where $X' < X$ denotes the "next-best" metric in our (ordered) set of available metrics. By definition,

$$ord_\perp(A, B) = 0 \tag{3}$$

which means that candidate translations A, B are of equal translation quality according to the quality metrics used. Pairs of systems with $ord(A, B) = 0$ can either be removed from the set of instances that would be used for training the machine learning classifier or we can fall back to using a *predefined* robustness fix deciding which of the two systems is supposed to be better.

Learning Translation Comparison

As previously mentioned, many approaches for system combination or quality estimation use classifiers to estimate the quality of translation output on the level of individual translations; only comparing the result to other candidate translations afterwards. This means that the feature vector for a given translation A is computed solely on information available inside A, not considering any other translation B. Formally, we define $vec_{single}(A) \in \mathbb{R}^n$ as follows:

$$vec_{single}(A) \stackrel{\text{def}}{=} \begin{pmatrix} f_1(A) \\ \vdots \\ f_n(A) \end{pmatrix} \quad (4)$$

In previous experiments we followed a different strategy and tried to explicitly model comparison with our feature vectors. We introduced the notion of *joint* feature vectors by combining feature values for translations A, B into a single, *joint* feature vector of size $2n$, formally defined as $vec_{joint}(A, B) \in \mathbb{R}^{2n}$:

$$vec_{joint}(A, B) \stackrel{\text{def}}{=} \begin{pmatrix} f_1(A) \\ \vdots \\ f_n(A) \\ f_1(B) \\ \vdots \\ f_n(B) \end{pmatrix} \quad (5)$$

Results from experimentation with joint feature vectors were not conclusive so we decided to switch to another approach that computes feature vectors for all *pairwise comparisons* of translations A, B, storing *binary feature values* to model if a feature value $f_x(A)$ for system A is better or worse than the corresponding feature value $f_x(B)$ for the competing system B. This also makes more sense as the comparison of feature values for the two systems A, B is modeled in an explicit way. Equation 6 shows the definition of a *joint, binarised* feature vector $vec_{binarised}(A, B) \in \mathbb{B}^n$:

$$vec_{binarised}(A, B) \stackrel{\text{def}}{=} \begin{pmatrix} f_1(A) > f_1(B) \\ \vdots \\ f_n(A) > f_n(B) \end{pmatrix} \quad (6)$$

The reason to store binary features values $f_x \in \mathbb{B}$ lies in the fact that these can be handled in a more efficient way during SVM training. Note that the order in which features for translations A, B are compared does not strictly matter. For the sake of consistency, we 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.

Using feature vectors defined like this, we have built a prototype of the proposed method and applied it to the OpenMT12 shared task, achieving promising results when considering the small set of features used. The authors have also participated in another shared task[2] where the proposed approach achieved best performance in terms of Meteor score, winning the competition.

[2]Name, reference and URL to be added in the accepted version of this paper to avoid identification of paper authors.

Machine Learning Challenges

After having briefly described our approach, we now want to focus on several problems with the basic implementation and discuss how lifelong machine learning could be applied to resolve these issues and, in turn, help to improve resulting translation quality. For each of the following sections, we first identify the challenge within our application and then discuss how lifelong machine learning would be helpful in solving them.

Improving Classification Quality

The prediction quality of our classification model is of key importance for the success of our combination method. Any improvement wrt. the classification process will result in a more accurate prediction of pairwise system comparison and, thus, increase the quality of the candidate translation selection. Research on better (or more refined) ways of pairwise classification therefore is beneficial for our method. Lifelong machine learning researchers could make use of the original datasets[3] for some of our combination experiments and work on improved classification performance. Then, the resulting classifiers could be integrated into our multi-engine machine translation system and evaluated in terms of their translation quality using either automated metrics or human judgment. Efforts to improve classification quality should focus on a fixed set of features and candidate translations, to allow specific tuning of the classifier model.

Exploring Different Machine Learning Methods

So far, we have applied SVM-based classification only. Our combination approach is however not bound to SVM-based models; in fact, any learning framework could be used.[4] Given the wide range of existing algorithms and tools for machine learning, it would be a natural extension of our work to investigate how those different algorithms could be applied and what impact they would have on the classifier's prediction rate. As the overall performance of our method is strongly dependent on the quality of the classification model, it would be an interesting challenge to see how other ML techniques performed compared to the SVM baseline. If some other method would achieve an improved prediction rate, it could be integrated into our system implementation and then be evaluated, similarly to the previous section on improved (SVM-based) classification quality. Datasets for such experiments can be made available to ML researchers. Again, the set of linguistic features and the given candidate translations should be fixed to allow a fair comparison to the baseline system's performance.

From Global to Local Classification

Our basic implementation of the combination framework uses a single classifier for all possible, pairwise comparisons of systems A, B. For N given systems, we have to estimate

[3]See the description of "Datasets for Lifelong Machine Learning Research" on the following page.

[4]The decision to use SVM-based models rather having to do with the fact that `libSVM` makes it easy to work with them.

"*Is system A better than system B?*" for a total of $\frac{N(N-1)}{2}$ comparison operations. It can be argued that using a single classification model for all these comparisons is inferior to the usage of *more specialised* models. As an intermediate solution, we could train one classification model for each system, ending up with N models in total. By focusing on a single system's qualitative differences compared to a set of other systems, it might be possible to make better use of the available feature values, hopefully achieving an improved prediction rate. In the extreme case, we would then train one classifier for each possible, pairwise comparison. Using the aforementioned sets of linguistic features and candidate translations, resulting classifiers could be integrated into our system combination method[5] and then be compared to the baseline system which only uses a *global* classifier.

Improved Feature Selection

Similar to the usage of dedicated classification models for pairwise system comparisons, it seems reasonable that the usefulness of individual feature values is varying. This may especially be true for cases in which the given translations originate from fundamentally different MT paradigms. Hence, it makes sense to also evaluate the effectiveness of feature values during classifier training, by itself an area of research in machine learning. Together with individual classification models, we expect improvements to the overall prediction rate of the (joint) classification model and thus a better translation quality of our system. ML researchers could also investigate the effects of new linguistic features. In summary, it seems that any research effort invested into the creation of specialised models for system comparison will pay off with an increased level of performance of the combination system. While the addition of new features would prevent a fair comparison to the baseline in terms of the classification model, the comparison on the level of translation quality would still be reasonable.

Incremental Improvements over Time

Our classification-based system combination model can be extended and, hopefully, be improved over time by replacing the classifier(s) that are used during sentence selection. As long as the defined set of linguistic features is stable, it is possible to use additional training data to refine and improve the models. Multi-model classification could be applied in cases where different feature sets are used. This makes the proposed multi-engine MT method an interesting problem for lifelong machine learning research.

Datasets for Lifelong Machine Learning Research

We intend to release several datasets related to our research which is conducted as part of the *T4ME* project.[6] Using these data researchers from the field of lifelong machine learning could further investigate the machine learning problems our method is facing and, thus, help to improve it.

[5]Using more than one classifier is easily possible and requires only minor changes to be made to the source code.

[6]These will become available from the author's GitHub page at `https://www.github.com/cfedermann`.

Integration of Human Feedback

Note that the our ordering relation $ord(A, B)$ is *extensible*. It can easily be extended to include, e.g., the results from manual evaluation of translation output. In fact, this would be a helpful addition as it would allow to bring in knowledge from domain experts. We want to briefly discuss how such an extension could be achieved. Figure 1 illustrates both the training and decoding phases of our combination approach.

During *training* we generate a classification model that is later used in the *decoding* phase to produce the combined translation output. We aim at integrating human feedback in an incremental manner which enables us to continually refine and improve our classification model. There are two straightforward ways of integrating human feedback into our approach:

– Annotators could be asked to do pairwise comparison of candidate translations on the training set and thus extend the set of available metrics in Equation 1 with a new $Human$ metric which would be used with preference, i.e., extending the set of available quality metrics to:

$$\{Human, Meteor_S, Meteor_C, NIST_C, BLEU_C, \perp\} \quad (7)$$

The new metric would allow to create a refined set of training instances and thus enable the computation of an improved classification model.

– Another way of adding manual judgments into our method would be the addition of a new feature that would encode how often a human annotator had classified the respective translation as "good", or how often the translation had been assigned some rank, e.g., from 1–5.

This feature value could encode judgments from several annotators, either by explicitly storing individual ratings or by averaging over the set of annotators.

Both extensions would enable the classifier to learn, over several iterations, which of its training instances are most useful, hopefully also improving final translation quality. Annotation could be collected using evaluation tools such as *Appraise* (Federmann 2012a) which is an open-source tool for MT evaluation, freely available from GitHub.

Conclusion

We have presented a machine learning approach for multi-engine machine translation. Our approach explicitly models comparison of two candidate translations in the design of its feature vectors and applies support vector machine training to get a machine learning classifier able to perform candidate translation on the sentence level.

We have presented several challenges which our system combination method is facing and discussed how lifelong machine learning methods could help to overcome these. Due to the fact that our sentence selection approach is based on pairwise classification—which itself is a very prominent application of machine learning—there seem to be several research strategies to further improve combination quality.

We intend to release several datasets so that researchers can apply lifelong machine learning methods to our problem scenario, hopefully improving the approach over time.

Acknowledgements

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). It has also been supported by the QTLaunchPad project (grant agreement no.: 296347).

References

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.

Chen, Y.; Eisele, A.; Federmann, C.; Hasler, E.; Jellinghaus, M.; and Theison, S. 2007. Multi-engine machine translation with an open-source SMT decoder. In *Proceedings of the Second Workshop on Statistical Machine Translation*, 193–196. Prague, Czech Republic: Association for Computational Linguistics.

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*, 85–91. Edinburgh, Scotland: Association for Computational Linguistics.

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, 138–145. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Eisele, A.; Federmann, C.; Saint-Amand, H.; Jellinghaus, M.; Herrmann, T.; and Chen, Y. 2008. Using Moses to integrate multiple rule-based machine translation engines into a hybrid system. In *Proceedings of the Third Workshop on Statistical Machine Translation*, 179–182. Columbus, Ohio: Association for Computational Linguistics.

Federmann, C. 2011. Results from the ML4HMT Shared Task on Applying Machine Learning Techniques to Optimise the Division of Labour in Hybrid Machine Translation. 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)*. Barcelona, Spain: META-NET.

Federmann, C. 2012a. Appraise: An open-source toolkit for manual evaluation of machine translation output. *The Prague Bulletin of Mathematical Linguistics* 98:25–35.

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)*, 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)*, 37–48. Saarbrücken, Germany: Springer, Heidelberg.

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, 95–100. Stroudsburg, PA, USA: Association for Computational Linguistics.

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"*, 103–111. European Association for Machine Translation.

Hildebrand, A. S., and Vogel, S. 2008. Combination of machine translation systems via hypothesis selection from combined n-best lists. In *MT at work: Proceedings of the Eighth Conference of the Association for Machine Translation in the Americas*, 254–261. Waikiki, Hawaii: Association for Machine Translation in the Americas.

Macherey, W., and Och, F. J. 2007. 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)*, 986–995. Prague, Czech Republic: Association for Computational Linguistics.

Matusov, E.; Ueffing, N.; and Ney, H. 2006. Computing Consensus Translation from Multiple Machine Translation Systems Using Enhanced Hypotheses Alignment. In *Conference of the European Chapter of the Association for Computational Linguistics*, 33–40. Stroudsburg, PA, USA: Association for Computational Linguistics.

Och, F. J., and Ney, H. 2003. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics* 29(1):19–51.

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, 311–318. Stroudsburg, PA, USA: Association for Computational Linguistics.

Rosti, A.-V.; Ayan, N. F.; Xiang, B.; Matsoukas, S.; Schwartz, R.; and Dorr, B. 2007. 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*, 228–235. Rochester, New York: Association for Computational Linguistics.