Optimization of Log-linear Machine Translation Model Parameters Using SVMs

Jesús González-Rubio[1], Daniel Ortiz-Martínez[1] and Francisco Casacuberta[2]

[1] Instituto Tecnológico de Informática
Universidad Politécnica de Valencia, Camino de Vera S/N, Valencia, Spain

[2] Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia, Camino de Vera S/N, Valencia, Spain

Abstract. The state-of-the art in statistical machine translation is based on a log-linear combination of different models. In this approach, the coefficients of the combination are computed by using the MERT algorithm with a validation data set. This algorithm presents high computational costs. As an alternative, we propose a novel technique based on Support Vector Machines to calculate these coefficients using a loss function to be minimized. We report the experiments on a Italian-English translation task showing encouraging results.

1 Introduction

Machine Translation (MT) is a research field of great importance in the European Community, where language plurality implies both a very important cultural richness and not negligible obstacle towards building a unified Europe. Because of this, a growing interest on MT has been shown both by politicians and research groups, which become more and more specialised in this field. In addition, Statistical Machine Translation (SMT) systems have proved in the last years to be an important alternative to rule-based MT systems, even outperforming commercial MT systems in the tasks they have been trained on. Moreover, the development effort behind a rule-based MT system and an SMT system is dramatically different, the latter being able to adapt to new language pairs with little or no human effort, whenever suitable corpora are available.

SMT is a pattern-recognition approach to MT. The grounds of modern SMT were established in [1], where the problem of MT was defined as following: given a sentence f from a certain source language, an adequate sentence $\hat{e}$ that maximises the posterior probability is to be found.

$$\hat{e} = \underset{e}{\operatorname{argmax}}\, Pr(e)Pr(f|e) \ . \tag{1}$$

At the origins of SMT, only word-based translation models $Pr(f|e)$ and target language models $Pr(e)$ were used, but since their introduction in SMT by Och and Ney [2], log-linear models have been a standard way to combine sub-models in MT systems. A log-linear model implies the following decision rule:

$$\hat{e} = \underset{e}{\operatorname{argmax}} \left\{ \sum_i \omega_i \theta_i(f, e) \right\} \ , \tag{2}$$

González-Rubio J., Ortiz-Martínez D. and Casacuberta F. (2008).
Optimization of Log-linear Machine Translation Model Parameters Using SVMs.
In *Proceedings of the 8th International Workshop on Pattern Recognition in Information Systems*, pages 48-56

where θ_i are features of the hypothesis e and ω_i are weights associated with those features.

Selecting appropriate weights ω_i is essential in order to obtain good translation performance. In [3] the Minimum Error Rate Training (MERT) was introduced. The MERT technique allows to find the values of the weights that minimize a given error rate measure. This has become much more standard than optimizing the conditional probability of the training data given the model (i.e., a maximum likelihood criterion), as was common previously. In [3] was also stated that system performance is best when parameters are optimized using the same objective function that will be used for evaluation; BLEU [4], which computes the precision of unigrams, bigrams, trigrams and 4-grams[3] with respect to a reference translation, remains common for both purposes and is often retained for parameter optimization even when alternative evaluation measures [5, 6] are used.

The MERT technique relies on data sets in which source language sentences are paired with (sets of) reference translations. This technique applies an iterative and (locally) convergent strategy to find a set of weights which optimizes the BLEU score; a n-best list of translations provided by the decoder is exploited for this purpose after each translation step. At each iteration of the MERT procedure, the whole corpus is translated, and this process continues until convergence is reached.

The main disadvantage of the MERT procedure consists in its high time complexity. Such time complexity is due to the above mentioned iterative nature of the MERT procedure.

1.1 Support Vector Machines

Support Vector Machines (SVMs) are a learning method introduced by Vapnik in [7] and [8]. SVMs are a set of related supervised learning methods used for classification and regression. They belong to a family of generalized linear classifiers. A special property of SVMs is that they simultaneously minimize the empirical classification error and maximize the geometric margin, hence they are also known as maximum margin classifiers.

SVMs are well-founded in terms of computational learning theory and very open to theoretical understanding and analysis. In [9] a generalization of the multiclass SVM learning [10, 11] was introduced. Such a formulation involves features extracted jointly from inputs and outputs. The naive approach of treating each structure as a separate class is often unfeasible, since it leads to a multiclass problem with a very large number of classes. This problem is overcome by specifying discriminant functions that exploit the structure and dependencies within the outputs.

SVM^{struct}[4] [9] is a SVM algorithm for predicting multivariate or structured outputs. It performs supervised learning by approximating a mapping

$$H : \mathcal{X} \rightarrow \mathcal{Y} \,, \tag{3}$$

[3] A n-grams is a sequence of n consecutive words.

[4] http://svmlight.joachims.org/svm_struct.html

using labeled training examples $(x_1, y_1), \ldots, (x_n, y_n)$. However, unlike regular SVMs which consider only uni-variate predictions like in classification, SVM^{struct} can predict complex objects like trees, sequences, or sets. Examples of problems with complex outputs are natural language parsing, sequence alignment in protein homology detection, and Markov models for part-of-speech tagging. The SVM^{struct} algorithm can also be used for linear-time training of binary and multiclass SVMs under the linear kernel [12].

The 1-slack cutting-plane algorithm implemented in SVM^{struct} V3.00 uses a new but equivalent formulation of the structural SVM quadratic program which allows a cutting-plane algorithm that has time complexity linear in the number of training examples. The n-slack algorithm of SVM^{struct} V2.50 is described in [13, 9]. The SVM^{struct} implementation is based on the SVM^{light} quadratic optimizer [14].

SVM^{struct} can be thought of as an API for implementing different kinds of complex prediction algorithms, e.g. Multiclass classification [9], Label sequence learning [9], Natural language parsing [9] and Protein Sequence Alignment [15].

1.2 Motivation

The aim of this work is to replace the slow iterative MERT procedure by a new non-iterative algorithm based on the SVM^{struct} algorithm. The proposed algorithm is able to perform the log-linear model parameter optimization with a linear time complexity.

2 Structured SVMs for Log-linear Model Parameter Optimization

This paper introduces a new proposal to perform the optimization of parameters of a log-linear translation model using the SVM^{struct} algorithm.

A log-linear model implies the following decision rule:

$$\hat{e} = \underset{e}{\text{argmax}} \left\{ \sum_i \omega_i \theta_i(f, e) \right\} \ , \tag{4}$$

where θ_i are features of the hypothesis e and ω_i are weights associated with those features. The problem consists on selecting the appropriate vector of weights $\boldsymbol{\omega}$ so an objective function is optimized. SVMs are used to accomplish this optimization.

The vector $\boldsymbol{\omega}$ has a crucial influence on the quality of the translations. In the following, we aim to learn $\boldsymbol{\omega}$ from a set $\mathcal{T}$ of training examples:

$$\mathcal{T} = ((f_1, e_1), (f_2, e_2), \ldots, (f_n, e_n)) \ , \tag{5}$$

where (f_i, e_i) are sentence pairs.

This training set is assumed to be generated independently and identically distributed according to some unknown distribution $\mathcal{P}(F, E)$. A MT algorithm can be seen as a function:

$$h_{\boldsymbol{\omega}}(f) = \underset{e \in E}{\text{argmax}} \left\{ \boldsymbol{\omega} \cdot \Psi(f, e) \right\} \ , \tag{6}$$

which maps a given source sentence f to a target sentence e. Our goal is to find a parameter vector $\boldsymbol{\omega}$ so that the predicted translation $h_{\boldsymbol{\omega}}(f)$ matches the correct translation on new test data as well as possible. In particular, we want to find the values of $\boldsymbol{\omega}$ that minimizes the expected loss (also called risk) for the data distribution $\mathcal{P}(F, E)$:

$$R_{\mathcal{P}}(h_{\boldsymbol{\omega}}) = \int \Delta(e, h_{\boldsymbol{\omega}}(f)) d\mathcal{P}(F, E) \ , \tag{7}$$

where $\Delta(e, e')$ is a user defined (non-negative) loss function that quantifies how 'bad' it is to predict e' when e is the correct translation. For example, one may choose $\Delta(e, e')$ to be equal to 1 minus the BLEU score for e'.

Following the principle of (Structural) Empirical Risk Minimization [16], finding a value of $\boldsymbol{\omega}$ that predicts well on new data can be achieved by minimizing the empirical loss (i.e the training error) on the training set $\mathcal{T}$.

$$R_{\mathcal{T}}(h_{\boldsymbol{\omega}}) = \sum_{i=1}^{n} \Delta(e_i, h_{\boldsymbol{\omega}}(f_i)) \ . \tag{8}$$

This minimization lead to the computational problem of finding the value of $\boldsymbol{\omega}$ which minimizes $R_{\mathcal{T}}(h_{\boldsymbol{\omega}})$. This vector $\boldsymbol{\omega}$ is the vector of optimized weights for the log-linear combination of models.

The problem of finding the value of $\boldsymbol{\omega}$ that minimizes the empirical loss $R_{\mathcal{T}}(h_{\boldsymbol{\omega}})$ of the translation algorithm was formulated as a minimization problem [9]:

$$\begin{aligned} \min_{\boldsymbol{\omega}, \xi} \left\{ \frac{1}{2} ||\boldsymbol{\omega}||^2 + \frac{C}{n} \sum_{i=1}^{n} \xi_i \right\} \quad s.t \ \forall i \ : \ \xi_i \geq 0 \ , \\ \forall i, \ \forall e \in E \ : \ \boldsymbol{\omega} \cdot \delta\Psi_i(e) \geq \Delta(e_i, e) - \xi_i \ , \end{aligned} \tag{9}$$

where $\delta\Psi_i(e) \ = \ \Psi(e_i, f_i) - \Psi(e, f_i)$.

The objective is the conventional regularized risk used in SVMs. The constraints in Equation 9 state that the score $\boldsymbol{\omega} \cdot \Psi(e_i, f_i)$ of the correct translation e_i must be greater than the score $\boldsymbol{\omega} \cdot \Psi(e, f_i)$ of any other alternative translation e.

This formulation includes a loss function $\Delta(e_i, e)$ that scales the desired difference in score. Intuitively, the larger the loss of an alternative translation e, the further should the score be away for that of the correct translation e_i. ξ_i is a slack variable shared among constraints from the same example, since in general the constraint system is not feasible. In [17] is proved that this formulation minimizes training loss, while the SVM-style regularization with the norm $\boldsymbol{\omega}$ in the objective provides protection against over-fitting for high-dimensional $\boldsymbol{\omega}$. The parameter C allows the user to control the trade-off between training error and regularization.

The general training algorithm [9] can be seen in Figure 1. This algorithm requires the implementation of the feature mapping function $\Psi(f, e)$, the loss function $\Delta(e_i, e)$ and the maximization given in the 6^{th} line of the algorithm in order to be adapted to a specific task.

Following sections explain how to adapt the SVM^{struct} algorithm to perform the log-linear model parameter optimization.

1: Input: $(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_n, \mathbf{y}_n)$, C, ϵ
2: $S_i \leftarrow \emptyset$ for all $i = 1, \ldots, n$
3: **repeat**
4: **for** $i = 1, \ldots, n$ **do**
5: set up cost function
$\text{SVM}_1^{\triangle s}$: $H(\mathbf{y}) \equiv (1 - \langle \delta\Psi_i(\mathbf{y}), \mathbf{w} \rangle) \triangle(\mathbf{y}_i, \mathbf{y})$
$\text{SVM}_2^{\triangle s}$: $H(\mathbf{y}) \equiv (1 - \langle \delta\Psi_i(\mathbf{y}), \mathbf{w} \rangle) \sqrt{\triangle(\mathbf{y}_i, \mathbf{y})}$
$\text{SVM}_1^{\triangle m}$: $H(\mathbf{y}) \equiv \triangle(\mathbf{y}_i, \mathbf{y}) - \langle \delta\Psi_i(\mathbf{y}), \mathbf{w} \rangle$
$\text{SVM}_2^{\triangle m}$: $H(\mathbf{y}) \equiv \sqrt{\triangle(\mathbf{y}_i, \mathbf{y})} - \langle \delta\Psi_i(\mathbf{y}), \mathbf{w} \rangle$
where $\mathbf{w} \equiv \sum_j \sum_{\mathbf{y}' \in S_j} \alpha_{j\mathbf{y}'} \delta\Psi_j(\mathbf{y}')$.
6: compute $\hat{\mathbf{y}} = \arg\max_{\mathbf{y} \in Y} H(\mathbf{y})$
7: compute $\xi_i = \max\{0, \max_{\mathbf{y} \in S_i} H(\mathbf{y})\}$
8: **if** $H(\hat{\mathbf{y}}) > \xi_i + \epsilon$ **then**
9: $S_i \leftarrow S_i \cup \{\hat{\mathbf{y}}\}$
10: $\alpha_S \leftarrow$ optimize dual over S, $S = \cup_i S_i$.
11: **end if**
12: **end for**
13: **until** no S_i has changed during iteration

Fig. 1. General training algorithm for structured SVMs.

2.1 Feature Mapping

The feature mapping function is a combined feature representation of the inputs and outputs. In our case, the mapping function takes a pair of input/output sentences and returns a vector with the scores of each of the models in the log-linear combination for this pair of sentences.

2.2 Loss Function

The MERT algorithm performs an optimization of the log-linear parameters in order to obtain the translation which maximizes the BLEU [4] score. Specifically, the BLEU score measures the precision of unigrams, bigrams, trigrams, and 4-grams between two sentences. Since the BLEU measure is a score instead of an error rate, the following loss function is used:

$$\Delta(e_i, e) \;=\; 1 - BLEU(e_i, e)\;. \tag{10}$$

As said in this section, the training algorithm (Figure 1) minimizes the training loss, so BLEU will be maximized.

Other measures, as for example, TER (Translation Edit Rate) [6] or WER (Word Error Rate), can be used as well.

2.3 Maximization

While the modeling of the feature mapping and the loss function is more or less straightforward, solving the maximization problem typically requires exploiting the structure of the output values.

In our case, the maximization is stated as follows:

$$\hat{e} = \underset{e \in E}{\text{argmax}}\, H(e)\ . \tag{11}$$

Among all possible target sentences E, we have to be able to choose that one which maximizes $H(e)$. The set of all possible target sentences is infinite so we approximated this maximization by using n-best lists.

3 Implementation Details

This section describes the implementation details of the proposed optimization algorithm. In our implementation, publicly-available well-known software in the field of SMT has been used.

To calculate the feature function $\Psi(e, f)$, the score of each model in the log-linear combination for the pair of sentences has to be computed. To calculate these scores we have used an extension of the THOT toolkit [18], which is a toolkit for SMT to train phrase-based models. The above mentioned extension of the THOT toolkit allows to obtain the alignment for a pair of sentences which maximizes the probability given by the log-linear model. It uses the current vector of weights $\boldsymbol{\omega}$ (see section 2) to calculate this alignment and returns the score of each model for this pair of sentences given this alignment.

Regarding the maximization problem described in section 2.3, given a source sentence e_i we have used the MOSES toolkit [19] to calculate a n-best list of translations according to the current vector of weights $\boldsymbol{\omega}$. Then these n-best hypothesis are re-scored according to $H(e)$, and the one with the maximum score is returned as the required target sentence.

The THOT toolkit and the MOSES toolkit use slightly different translation tables. Specifically, the MOSES toolkit allows to work with one or more score components for each phrase pair while the THOT toolkit only allows to work with one. By this reason, it is necessary to keep two translation tables, one for the MOSES toolkit where the score for each component appears separately and one for the THOT toolkit where all the components are gathered in only one value.

4 Experiments

We have carried out an experimentation in order to verify the effectiveness of our proposal. In our experiments we have compared the performance of both the MERT procedure and our proposed technique. All the experiments have been carried out with the FUB corpus.

4.1 The FUB Corpus

The FUB corpus [20], is a bilingual Italian–English corpus with a restricted semantic domain. The application is the translation of queries, requests and complaints that a tourist may make at the front desk of a hotel, for example, asking for a booked room, requesting a service of the hotel, etc. The statistics of the corpus are shown in Table 1.

Table 1. FUB corpus statistics.

	Training		Development		Test	
Language	Italian	English	Italian	English	Italian	English
#Sentences	2900		138		300	
#Words	51902	62335	2609	3119	6121	7243
Vocabulary size	2480	1671	534	443	715	547
#Out of vocabulary	—		55	31	129	84
Perplexity (trigram)	—		19.9	10.6	19.6	10.2

4.2 Results

The experimentation consists on training a MT model with the MOSES toolkit using the Training set. Then the Development set is used to optimize the parameters of the trained log-linear model. The MERT procedure and our algorithm are used to perform the optimization. Finally the translation results of each of them with the Test set are compared.

As first step, a log-linear model is trained using the MOSES toolkit. This log-linear combination is composed of eight models: the distortion (reordering) model, the target language model, the translation model which is also composed of five sub-models and the word penalty model.

As said in Section 3, the THOT toolkit works with translation tables with only one score for each phrase pair. So a new translation table has to be built. The score of a phrase pair in that table is the weighted average (the MOSES default weights are used) of the five scores in the MOSES translation table for that phrase pair. Once the translation scores have been gathered, a log-linear combination of four models is obtained. The new table with the gathered scores is used to perform the optimization of parameters.

To optimize the parameters the MERT procedure is used with its default options values. It uses a 100-best list of translations.

Our proposal uses the extension of the THOT toolkit to perform the feature mapping. The maximization described in Section 2.3 is carried out using 10-best lists of translations. The 10-best translations list is re-scored using the following equation:

$$H(e) = \Delta(e_i, e) - \langle \delta\Psi_i(e) \cdot \boldsymbol{\omega} \rangle \ . \tag{12}$$

This $H(e)$ function corresponds to the margin re-scaling ($SVM_1^{\Delta m}$) on Figure 1 [9].

SVM^{struct} allows to modify a great amount of parameters relative to the SVMs optimization process. Different combinations of values of some parameters have been tested to choose those values with better performance.

Table 2 shows the BLEU scores for the different models after translating the Test set. Baseline corresponds with the log-linear model before any parameter optimization, MERT corresponds with the model after being optimized using the MERT procedure and SVMs corresponds with a model which parameters had been optimized using our proposal.

The results on Table 2 show that our proposal is able to outperform the MERT procedure. But, if we optimize the parameters using the MERT procedure and the original

Table 2. BLEU score results summary.

Baseline	MERT	SVMs
64.46	64.93	**65.38**

table (the one with eight scores per phrase pair), the BLEU score raises to 65.89. In this case, the MERT procedure is able to optimize the weights for each of the sub-models in the translation model independently. So, the relative significance of each of this sub-models can vary. Our proposal optimizes the weight of the gathered translation model, so the relative importance of each of the sub-models do not change respect to the non-optimized model.

5 Conclusions

This work have introduced a new method to optimize the parameters of a log-linear translation model using SVMs. Our proposal is based on the SVM^{struct} algorithm which is an SVM optimization algorithm for multivariate or structured outputs. The obtained results are very promising: using only a 10-best translations list, we outperform the MERT procedure when using equal number of components in the log-linear combination.

As future work, our main goal is to compare our proposal with a standard implementation of the MERT procedure in terms of time complexity; to achieve such a goal it is necessary to integrate the functionalities of the THOT, MOSES and SVM^{struct} toolkits, so the efficiency of the algorithm will be dramatically increased. In addition, we also plan to accomplish experiments with larger corpora, to use other measures as WER or TER as loss function, to use word graphs instead of n-best lists to perform the maximization and finally to find the best way to go through the differences between the THOT toolkit and the MOSES toolkit.

Acknowledgements

This work has been partially supported by the Spanish *Ministerio de Educación y Ciencia* (MEC) under FPU scholarship AP2006-00691, the Spanish MEC under grant Consolider Ingenio 2010 CSD2007-00018 and the EC (FEDER) and the Spanish MEC under grant TIN2006-15694-CO2-01.

References

1. Brown, P.F., Pietra, S.A.D., Pietra, V.J.D., Mercer, R.L.: The mathematics of machine translation. In: Computational Linguistics. Volume 19. (1993) 263–311
2. Och, F., Ney, H.: Discriminative training and maximum entropy models for statistical machine translation (2002)
3. Och, F.J.: Minimum error rate training in statistical machine translation. In: Proc. of the Association for Computational Linguistics, Sapporo, Japan (2003)

4. Papineni, K., Kishore, A., Roukos, S., Ward, T., Zhu, W.: Bleu: A method for automatic evaluation of machine translation. In: Technical Report RC22176 (W0109-022), IBM Research Division, Thomas J. Watson Research Center, Yorktown Heights, NY (2001)
5. Banerjee, S., Lavie, A.: METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In: Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization, Ann Arbor, Michigan, Association for Computational Linguistics (2005) 65–72
6. Snover, M., Dorr, B., Schwartz, R., Micciulla, L., Makhoul, J.: A study of translation edit rate with targeted human annotation. In: Proceedings of AMTA. (2006)
7. Cortes, C., Vapnik, V.: Support-vector networks. Mach. Learn. **20** (1995) 273–297
8. Vapnik, V.N.: The nature of statistical learning theory. Springer-Verlag New York, Inc., New York, NY, USA (1995)
9. Tsochantaridis, I., Hofmann, T., Joachims, T., Altun, Y.: Support vector machine learning for interdependent and structured output spaces. In: ICML '04: Proceedings of the twenty-first international conference on Machine learning, New York, NY, USA, ACM (2004) 104
10. Weston, J., Watkins, C.: Multi-class support vector machines (1998)
11. Crammer, K., Singer, Y.: On the algorithmic implementation of multiclass kernel-based vector machines. J. Mach. Learn. Res. **2** (2002) 265–292
12. Joachims, T.: Training linear svms in linear time. In: KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, New York, NY, USA, ACM (2006) 217–226
13. Joachims, T.: Learning to align sequences: A maximum-margin approach (2003)
14. Joachims, T.: Making large-scale support vector machine learning practical. In B. Schólkopf, C. Burges, A.S., ed.: Advances in Kernel Methods: Support Vector Machines. MIT Press, Cambridge, MA (1998)
15. Yu, C.N.J., Joachims, T., Elber, R., Pillardy, J.: Support vector training of protein alignment models. In: RECOMB. (2007) 253–267
16. Vapnik, V.N.: Statistical Learning Theory. Wiley-Interscience (1998)
17. Tsochantaridis, I., Joachims, T., Hofmann, T., Altun, Y.: Large margin methods for structured and interdependent output variables. Journal of Machine Learning Research **6** (2005) 1453–1484
18. Ortiz-Martínez, D., García-Varea, I., Casacuberta, F.: Thot: a toolkit to train phrase-based statistical translation models. In: Tenth Machine Translation Summit. AAMT, Phuket, Thailand (2005)
19. Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., Herbst, E.: Moses: Open source toolkit for statistical machine translation. Annual Meeting of the Association for Computational Linguistics (ACL), demonstration session (2007)
20. Casacuberta, F., Ney, H., Och, F.J., Vidal, E., Vilar, J.M., Barrachina, S., Garcia-Varea, I., D. Llorens, C.M., Molau, S., Nevado, F., Pastor, M., Pico, D., Sanchis., A.: Some approaches to statistical and finite-state speech-to-speech translation. Computer Speech and Language **18** (2004) 25–47