A Maximum Entropy/Minimum Divergence Translation Model

George Foster
RALI, Université de Montréal
foster@iro.umontreal.ca

Abstract

I present empirical comparisons between a linear combination of standard statistical language and translation models and an equivalent Maximum Entropy/Minimum Divergence (MEMD) model, using several different methods for automatic feature selection. The MEMD model significantly outperforms the standard model in test corpus perplexity, even though it has far fewer parameters.

1 Introduction

Statistical Machine Translation (SMT) systems use a model of $p(\mathbf{t}|\mathbf{s})$, the probability that a text $\mathbf{s}$ in the source language will translate into a text $\mathbf{t}$ in the target language, to determine the best translation for a given source text. The standard approach to modeling this distribution relies on a "noisy channel" decomposition into a language model $p(\mathbf{t})$ and a translation model $p(\mathbf{s}|\mathbf{t})$, which correspond respectively to prior and likelihood components in a Bayesian formulation:

$$\begin{aligned} p(\mathbf{t}|\mathbf{s}) &= p(\mathbf{t})p(\mathbf{s}|\mathbf{t}) / \sum_{\mathbf{t}} p(\mathbf{t})p(\mathbf{s}|\mathbf{t}) \\ &\propto p(\mathbf{t})p(\mathbf{s}|\mathbf{t}), \end{aligned}$$

where proportionality holds when searching for the optimum target text $\mathbf{t}$ for a given source text $\mathbf{s}$. This equation has been called the "fundamental equation of SMT" (Brown et al., 1993).

In this paper, I investigate an alternate technique for modeling $p(\mathbf{t}|\mathbf{s})$, based on a direct chain-rule expansion of the form:

$$p(\mathbf{t}|\mathbf{s}) = \prod_{i=1}^{|\mathbf{t}|} p(t_i|t_1 \ldots t_{i-1}, \mathbf{s}), \qquad (1)$$

where t_i denotes the ith token in $\mathbf{t}$.[1] The objects to be modeled in this case belong to the family of conditional distributions $p(w|\mathbf{h}, \mathbf{s})$, where w is a target word at a particular position in $\mathbf{t}$, and $\mathbf{h}$ denotes the tokens which precede it in $\mathbf{t}$. The main motivation for this approach is that it simplifies the "decoding" problem of finding the most likely target text according to the model. In particular, if $\mathbf{h}$ is known, the problem of finding the best word at the current position requires only a straightforward search through the target vocabulary, and simple and efficient dynamic-programming based heuristics can be used to extend this to sequences of words. This is very important for applications such as TransType (Foster et al., 1997; Langlais et al., 2000), where the task is to make real-time predictions of the text a human translator will type next, based on the source text under translation and some prefix of the target text that has already been typed.

The main drawback to modeling $p(\mathbf{t}|\mathbf{s})$ in terms of $p(w|\mathbf{h}, \mathbf{s})$ is that the latter distribution is conditioned on two very disparate sources of information which are difficult to combine in a complementary way. One simple strategy is to use a linear combination of

[1] This ignores the issue of normalization over target texts of all possible lengths, which can be easily enforced when desired by using a stop token or a prior distribution over lengths.

language and translation components, of the form:

$$p(w|\mathbf{h},\mathbf{s}) = \lambda p(w|\mathbf{h}) + (1-\lambda)p(w|\mathbf{s}). \quad (2)$$

where $\lambda \in [0,1]$ is a combining weight. However, this is a weak model because it averages over the relative strengths of its components; when $p(w|\mathbf{h})$ is likely to be a more accurate estimate than $p(w|\mathbf{s})$, it is obvious that the model should rely more heavily on $p(w|\mathbf{h})$, and vice versa, rather than using a fixed weight. In theory this could be partially remedied by making λ depend on $\mathbf{h}$ and $\mathbf{s}$, but in practice significant improvements with this technique have proven elusive (Langlais and Foster, 2000). The noisy channel model avoids this problem by making predictions based on $\mathbf{h}$ the responsibility of the language model $p(\mathbf{t})$, and those based on $\mathbf{s}$ the responsibility of the translation model $p(\mathbf{s}|\mathbf{t})$, and combining the two in an optimum way. But this comes at the cost of increased decoding complexity, because the chain rule can no longer be applied as in (1) due to the reversed direction of the translation model. Much recent research in SMT, eg (García-Varea et al., 1998; Niessen et al., 1998; Och et al., 1999; Wang and Waibel, 1998) deals with the decoding problem, either directly or indirectly because of constraints imposed on the form of the translation model.

A statistical technique which has recently become popular for NLP is Maximum Entropy/Minimum Divergence (MEMD) modeling (Berger et al., 1996). One of the main strengths of MEMD is that it allows information from different sources to be combined in a principled and effective way, so it is a natural choice for modeling $p(w|\mathbf{h},\mathbf{s})$. In this paper, I describe a MEMD model for $p(w|\mathbf{h},\mathbf{s})$ and compare its performance to that of an equivalent linear model. I also evaluate several different methods for MEMD feature selection, including a new algorithm due to Printz (1998). To my knowledge, this is the first application of MEMD to building a large-scale translation model, and one of the few direct comparisons between a MEMD model and an almost exactly equivalent linear model.[2]

2 Models

2.1 Linear Model

The baseline model is a linear combination as in (2) of a standard interpolated trigram (Jelinek and Mercer, 1980) for $p(w|\mathbf{h})$ and the IBM model 1 (IBM1) (Brown et al., 1993) for $p(w|\mathbf{s})$. As originally formulated, IBM1 models the distribution $p(\mathbf{t}|\mathbf{s})$, but since target text tokens are predicted independently, it can also be used for $p(w|\mathbf{s})$. The underlying generative process is as follows: 1) pick a token s at random in $\mathbf{s}$, independent of the positions of w and s; 2) choose w according to a word-for-word translation probability $p(w|s)$. Summing over all choices for s gives the complete model:

$$p(w|\mathbf{s}) = \sum_{j=0}^{|\mathbf{s}|} p(w|s_j)/(|\mathbf{s}|+1)$$

where s_j is the jth token in $\mathbf{s}$ for $j > 0$, and s_0 is a special null token prepended to each source sentence to account for target words which have no direct translations. The word-pair parameters $p(w|s)$ can be estimated from a bilingual corpus of aligned sentence pairs using the EM algorithm, as described in (Brown et al., 1993).

2.2 MEMD Model

A MEMD model for $p(w|\mathbf{h},\mathbf{s})$ has the general form:

$$p(w|\mathbf{h},\mathbf{s}) = \frac{q(w|\mathbf{h},\mathbf{s})\exp(\vec{\alpha}\cdot\mathbf{f}(w,\mathbf{h},\mathbf{s}))}{Z(\mathbf{h},\mathbf{s})},$$

where $q(w|\mathbf{h},\mathbf{s})$ is a reference distribution, $\mathbf{f}(w,\mathbf{h},\mathbf{s})$ maps $(w,\mathbf{h},\mathbf{s})$ into an n-dimensional feature vector, $\vec{\alpha}$ is a corresponding vector of feature weights (the parameters of the model), and $Z(\mathbf{h},\mathbf{s}) = \sum_w q(w|\mathbf{h},\mathbf{s})\exp(\vec{\alpha}\cdot\mathbf{f}(w,\mathbf{h}))$ is a normalizing factor.

[2] Rosenfeld (1996) reports a greater perplexity reduction (23% versus 10%) over a baseline trigram language model due the use of ME versus linear word triggers. However, since the models tested apparently differed in other aspects, it is hard to determine how much of this gain can be attributed to the use of ME.

It can be shown (Berger et al., 1996) that the use of this model with maximum likelihood parameter estimation is justified on information-theoretic grounds when q represents some prior knowledge about the true distribution and when the expected values of $\mathbf{f}$ in the training corpus are identical to their true expected values.[3] There is no requirement that the components of $\mathbf{f}$ represent disjoint or statistically independent events. This result motivates the use of MEMD models, but it offers only weak guidance on how to select q or $\mathbf{f}$. In practice, q is usually chosen on the basis of efficiency considerations (when the information it captures would be computationally expensive to represent as components of $\mathbf{f}$), and $\mathbf{f}$ is established using heuristics such as described in the next section. Once q and $\mathbf{f}$ have been chosen, the IIS algorithm (Della Pietra et al., 1995) can be used to find maximum likelihood parameter values.

In the current context, since the aim was to compare equivalent linear and MEMD models, I used an interpolated trigram as the reference distribution q and boolean indicator functions over bilingual word pairs as features (ie, components of $\mathbf{f}$). A pair of source,target words (s, t) has a corresponding feature function:

$$f_{st}(w, \mathbf{h}, \mathbf{s}) = \begin{cases} 1, & s \in \mathbf{s} \text{ and } t = w \\ 0, & \text{else} \end{cases}$$

Using the notational convention that α_{st} is 0 whenever the corresponding feature f_{st} does not exist in the model, the final MEMD model can be written compactly as:

$$p(w|\mathbf{h}, \mathbf{s}) = q(w|\mathbf{h}) \exp(\sum_{s \in \mathbf{s}} \alpha_{sw}) / Z(\mathbf{h}, \mathbf{s}).$$

This model is structurally quite similar to the one defined in the previous section:

$$p(w|\mathbf{h}, \mathbf{s}) = \lambda q(w|\mathbf{h}) + \frac{1-\lambda}{|\mathbf{s}|+1} \sum_{j=0}^{|\mathbf{s}|} p(w|s_j)$$

with the MEMD feature weights α_{sw} playing the role of the IBM1 probabilities $p(w|s)$, and the MEMD model summing over contributions from source sentence words rather than tokens for efficiency. If there are m free parameters in the trigram and n word pairs, the MEMD model will contain $m + n$ free parameters and the linear model will contain $m+n+1-|V_s|+|V_t|-1$[4] free parameters, so if the source and target vocabulary sizes $|V_s|$ and $|V_t|$ are equal the two models will contain precisely the same number of free parameters.

One important practical difference between the two models is the requirement to calculate the MEMD normalizing factor $Z(\mathbf{h}, \mathbf{s})$ for each context in which this model is used. This makes the MEMD model much more computationally expensive than the linear model, so that it is not feasible to have it incorporate all available word-pair features (ie all bilingual pairs of words which cooccur in some aligned sentence pair in the training corpus). Moreover, since the empirical expectations of features are supposed to reflect their true values, having a feature for *every* cooccurring pair in the corpus would be theoretically inadvisable even if it were computationally feasible. Some method of selecting a subset of reliable features is therefore required, as described in the next section.

3 Feature Selection

I experimented with three methods for selecting bilingual word pairs for inclusion in the models. All methods assign scores to individual pairs, so feature subsets of any desired size can be extracted by taking the highest-ranked pairs.

3.1 Mutual Information

The simplest scoring method was mutual information (MI), defined for a pair (s,t) as:

$$I(s;t) = \sum_{x \in \{s,\bar{s}\}} \sum_{y \in \{t,\bar{t}\}} \tilde{p}(x,y) \log \frac{\tilde{p}(x,y)}{\tilde{p}(x)\tilde{p}(y)},$$

[3] Another interpretation, which has been less well publicized in the NLP literature, is that of a single-layer neural net with certain weight constraints and a "softmax" output function (Bishop, 1995).

[4] One free combining weight, one normalization constraint per source word, and $|V_t| - 1$ free parameters from $p(w|s_0)$

where $\tilde{p}(s,t)$ is the probability that a randomly chosen pair of cooccurring source and target tokens in the corpus is (s,t); $\tilde{p}(s,\bar{t})$ is the probability that the source token is s and the target token is not t; etc; and $\tilde{p}(x)$ and $\tilde{p}(y)$ are the left and right marginals of $\tilde{p}(x,y)$. Mutual information measures the degree to which s and t are non-independent, so it is a reasonable choice for scoring pairs.

3.2 MEMD Gains

The second scoring method was an approximation of the MEMD gain for feature f_{st}, defined as the log-likelihood difference between a MEMD model which includes this feature and one which does not:

$$G_{st} = \frac{1}{|\mathcal{T}|} \log \frac{p_{st}(\mathcal{T}|\mathcal{S})}{p(\mathcal{T}|\mathcal{S})}$$

where the training corpus $(\mathcal{S},\mathcal{T})$ consists of a set of (statistically independent) sentence pairs $(\mathbf{s},\mathbf{t})$, and p_{st} is the model which includes f_{st}. Since MEMD models are trained by finding the set of feature weights which maximizes the likelihood of the training corpus, it is natural to rate features according to how much they contribute to this likelihood. A powerful strategy for using gains is to build a model iteratively by adding at each step the feature which gives the highest gain with respect to those already added. Berger et al (1996) describe an efficient algorithm for accomplishing this in which approximations to $p_{st}(\mathcal{T}|\mathcal{S})$ are computed in parallel for all (new) features f_{st} by holding all weights in the existing model fixed and optimizing only over α_{st}. However, this method requires many expensive passes over the corpus to optimize the weights for the set of features under consideration at each step, and it adds only one feature per step, so it is not practical for constructing models containing thousands of features or more.

In a recent paper (Printz, 1998), Printz argues that it is usually sufficient to perform the iteration described in the previous paragraph only once, in other words that features can be ranked simply according to their gain with respect to some initial model. He also gives an algorithm for computing gains using a numerical approximation which requires only a single pass over the training corpus. I adopted Printz' method for computing MEMD gains, using the reference trigram as the initial model.

3.3 IBM1 Gains

The final scoring method involved the gain of each word-pair parameter $p(t|s)$ within IBM1. Instead of taking gains with respect to an initial model as in the previous section, I computed them with respect to a "full" model which incorporated all available word pairs:

$$G_{st} = \frac{1}{|\mathcal{T}|} \log \frac{p(\mathcal{T}|\mathcal{S})}{p_{\bar{s}t}(\mathcal{T}|\mathcal{S})},$$

where $p_{\bar{s}t}$ denotes the full IBM1 model p with the parameter $p(t|s)$ set to zero and the resulting distribution $p(w|s)$ renormalized. The advantage of this method is that it gives a measure of each parameter's worth in the presence of other parameters. As is the previous section, this is an approximation because determining the true gain would require retraining $p_{\bar{s}t}$ and not merely renormalizing.

A problem with IBM1 gains is that they are not very robust. If the corpus contains a sentence pair $(\mathbf{s},\mathbf{t})$ which consists only of a single word pair (s,t), then G_{st} will contain the term $\frac{1}{|\mathcal{T}|} \log \frac{p(t|s)+p(t|s_0)}{p(t|s_0)}$, so if $p(t|s_0)$ is close to zero (as is frequently the case), G_{st} will be close to infinity, even though (s,t) may occur only once in the training corpus. To remedy this, I computed gains with respect to a linear combination of IBM1 and a smoothing model u, of the form $\lambda p(w|\mathbf{s}) + (1-\lambda)u(w|\mathbf{h},\mathbf{s})$. In the experiments reported below, I used a uniform distribution for u, with $\lambda = .99$.[5]

Smoothed IBM1 gains can be computed in parallel in a single pass over the training corpus using the algorithm in figure 1. The line marked with an asterisk takes into account the increase in $p(t|s)$ due to renormalizing the distribution $p(w|s)$ after setting $p(t'|s)$ to

[5] Another interesting choice for u would be the interpolated trigram, which would make the method described here more similar to the MEMD gain ranking described in the previous section.

segment	file pairs	sentence pairs	English tokens	French tokens
train	922	1,639,250	29,547,936	31,826,112
held-out	30	54,758	978,394	1,082,350
test	30	53,676	984,809	1,103,320

Table 1: Corpus segmentation. The *held-out* segment was used to train combining weights for the trigram and the overall linear model; the *train* segment was used for all other training.

```
for all word pairs (s,t):   G_st ← 0
for each sentence pair (s,t) ∈ (S,T):
  for each token t in t:
    K ← Σ_{j=0}^{|s|} p(t|s_j) + ((1−λ)(|s|+1))/λ u(t|h,s)
    for each word s in s:
      G_st ← G_st + log K/(K − f_s(s)p(t|s))
      for all t' ≠ t:
        G_st' ← G_st' + log K/(K + f_s(s) p(t|s)p(t'|s)/(1−p(t'|s)))   *
for all (s,t):   G_st ← G_st/|T|
```

Figure 1: Algorithm for IBM1 gains. $\mathbf{f_s}(s)$ gives the number of times s occurs in $\mathbf{s}$.

zero, for each word $t' \neq t$ in the vocabulary. To speed up the algorithm, I performed this step only for those t' such that $p(t'|s) \geq .01$. This causes the gains for pairs (s, t') such that $p(t'|s) < .01$ to be slightly overestimated, but since the gains of such pairs are low in any case, the ranking of the most valuable pairs is unlikely to be radically affected.

4 Experiments

I ran experiments on the Canadian Hansard corpus, with English as the source language and French as the target language. After sentence alignment using the method described in (Simard et al., 1992), the corpus was split into disjoint segments as shown in table 1. To evaluate performance, I used perplexity: $p(\mathcal{T}|\mathcal{S})^{-1/|\mathcal{T}|}$, where p is the model being evaluated, and $(\mathcal{S}, \mathcal{T})$ is the test corpus. Perplexity is a good indicator of performance for the TransType application described in the introduction, and it has also been used in the evaluation of full-fledged SMT systems (Al-Onaizan et al., 1999). To ensure a fair comparison, all models used the same target vocabulary.

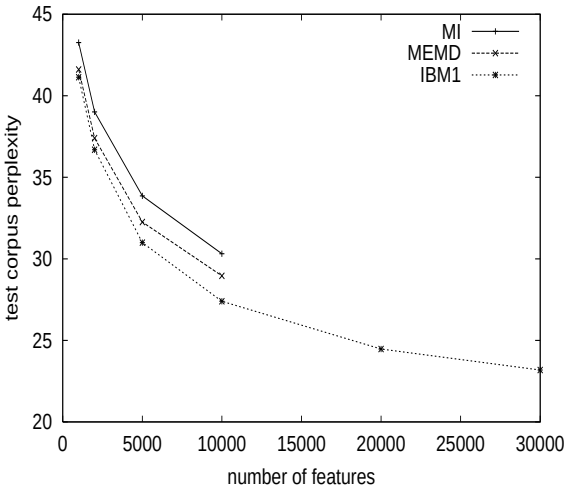

Figure 2: MEMD performance versus number of features for various feature-selection methods.

To compare MEMD feature-selection methods, I first ranked all 35 million bilingual word pairs cooccurring within aligned sentence pairs in the training corpus using the MI and IBM1 gains methods. Because the MEMD gains method was much more expensive, it was used to rank only a short list of approximately 160,000 pairs derived by merging the top 100,000 candidates from each of the other methods. As shown in table 2, the three methods give substantially different rankings, even among the top-ranked pairs. For each method, I trained MEMD models on a sequence of successively larger feature sets consisting of the top-ranked word pairs for that method. The results are shown in figure 2. Due to time constraints,[6] 20,000- and 30,000-feature models were trained only for the IBM1 feature sets, which outperformed the other

[6] A 30,000 feature MEMD model takes approximately 6 days to train on a 750MHz Pentium.

MI		MEMD gains		IBM1 gains	
:	:	mr.	m.	and	et
mr.	m.	i	je	government	gouvernement
we	nous	we	nous	we	nous
i	je	?	?	,	,
?	?	government	gouvernement	:	:
offenders	loi	grant	accorder	gucci	gucci
deleted	règlement	closer	plus	depreciation	amortissement
interlake	felix	imperial	imperial	endorse	appuyer
woodbine	beaches	same	la	indeed	vraiment
question	ai	stabilization	grain	appalled	consterné

Table 2: Pairs ranked 1–5 (top box) and 20000-20005 for each feature-selection method.

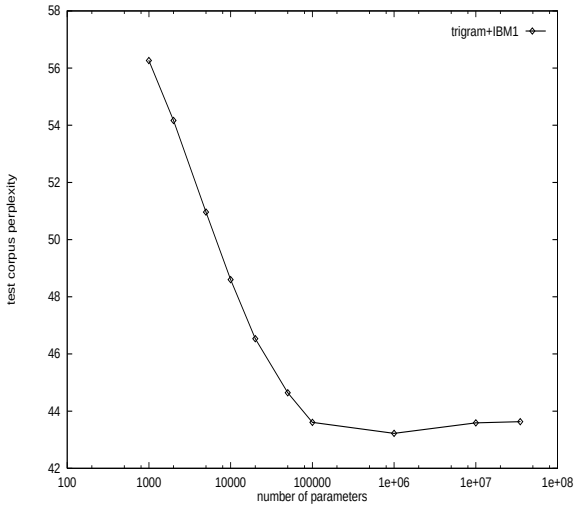

Figure 3: Performance of the linear model versus number of IBM1 parameters.

model	word pairs	ppx	Δ
3G	—	61.0	—
3G+IBM1	34,969,331	43.6	0%
3G+IBM1	1,000,000	43.2	0.9%
MEMD	1,000	41.1	5.7%
MEMD	30,000	23.2	46.9%

Table 3: Comparison of model performances. The *word pairs* column gives the number of word pairs selected by the IBM1 gain ranking method, the *ppx* column gives test corpus perplexity, and the Δ column gives the perplexity drop as a percentage of the baseline. 3G is the trigram model and '+' denotes linear interpolation.

methods by a small margin.

Since the number of features in the MEMD models was much smaller than the number of parameters in the full IBM1, before comparing the MEMD and linear models I wanted to be sure that any performance difference was not due to IBM1 overfitting the training corpus. To eliminate this possibility, I optimized the number of IBM1 parameters by training linear models with various sizes of translation parameter sets obtained from the IBM1 gain ranking. As shown in figure 3, the larger linear models do exhibit a very slight overtraining effect, with the optimum parameter set size around 1M, compared to 35M parameters in the full model.

Table 3 presents final results for various linear and MEMD models. The MEMD models give a striking improvement over the linear models, with a 1000-feature MEMD model performing better than the best linear model (despite containing 1000 times fewer word-pair parameters), and the best MEMD model yielding a perplexity reduction of more than 45% over the baseline linear model.

5 Discussion

The main result of this paper is that the MEMD framework appears to be a much more effective way to combine information from different sources than linear interpolation, at least for the problem studied here. It is fairly easy to see intuitively why this should

be the case: MEMD essentially multiplies predictive scores arising from different sources rather than averaging them. This gives information sources which assign either very high or very low scores much more influence over the final result. When such scores are based upon reliable evidence, this will lead to better models.

One somewhat surprising result of these experiments was that the IBM1 gains feature selection method resulted in better models than the MEMD gains method, despite the fact that the latter is based on a much more direct measure of each feature's worth within the MEMD model. A possible explanation for this is that the gain over the reference trigram is not a good predictor of the gain in the presence of many other features; this is borne out by the fact that, for very small feature sets (on the order of 100 words and less), the MEMD method did outperform the IBM1 method. Another explanation is inaccuracies in the gain approximations computed by Printz' method, which involves many numerical parameters that require tuning. Further investigation is required into this and other techniques for finding valid word pairs, since all methods tested yielded significant quantities of noise beyond 30,000 pairs. Because the source vocabulary contains about 50,000 words this is obviously an unrealistically small number of translations.

Although the main use for the model I have described in this paper is in applications like TransType which need to make rapid predictions of upcoming target text, it is interesting to speculate about whether a MEMD model for $p(w|\mathbf{h},\mathbf{s})$ could also be useful for SMT. Compared to the standard noisy channel approach, this has the advantage of permitting much less complex search procedures; of allowing any information which is directly observable in the training corpus to be very easily incorporated into the model via boolean features; and of an estimation procedure where translation model parameters can be optimized for use with an existing language model.[7] Disadvantages include the high cost of training MEMD models, the fact that $p(w|\mathbf{h},\mathbf{s})$ is somewhat less general than $p(\mathbf{s}|\mathbf{t})$ for building realistic translation models; and the lack of a mechanism equivalent to the EM algorithm for incorporating "hidden" variables into MEMD models (see (Foster, 2000) for a discussion of this problem).

6 Conclusion

The problem of searching for the best target text in statistical translation applications can be greatly simplified if the fundamental distribution $p(\mathbf{t}|\mathbf{s})$ is expanded directly in terms of the distribution $p(w|\mathbf{h},\mathbf{s})$, rather than using the standard noisy-channel approach. I compared a simple linear model for $p(w|\mathbf{h},\mathbf{s})$ based on IBM's model 1 with an equivalent MEMD model, and found that the MEMD model has over 45% lower test corpus perplexity, despite using two orders of magnitude fewer parameters. I also compared several methods for selecting MEMD word-pair features, and found that a simple method which ranks pairs according to their gain within model 1 offers slightly better performance and significantly lower computational cost than a more general MEMD feature-selection algorithm due to Printz. Finally, I suggest that it may be fruitful to explore the idea of using a MEMD model for $p(w|\mathbf{h},\mathbf{s})$ as an alternative to the noisy-channel approach to SMT.

Acknowledgements

This work was carried out as part of the TransType project at RALI, funded by the Natural Sciences and Engineering Research Council of Canada. I wish to thank Guy Lapalme and Andreas Eisele for comments on the paper, and Philippe Langlais for inspiring discussions.

References

Yaser Al-Onaizan, Jan Curin, Michael Jahr, Kevin Knight, John Lafferty, Dan Melamed, Franz-Josef Och, David Purdy, Noah A. Smith, and David Yarowsky. 1999. Statistical machine translation: Final report,

[7] In principle, both language and translation components could be trained simultaneously.

JHU workshop 1999. Technical report, The Center for Language and Speech Processing, The Johns Hopkins University, www.clsp.jhu.edu/ws99/projects/mt/final_report.

Adam L. Berger, Stephen A. Della Pietra, and Vincent J. Della Pietra. 1996. A Maximum Entropy approach to Natural Language Processing. *Computational Linguistics*, 22(1):39–71.

Christopher M. Bishop. 1995. *Neural Networks for Pattern Recognition*. Oxford.

Peter F. Brown, Stephen A. Della Pietra, Vincent Della J. Pietra, and Robert L. Mercer. 1993. The mathematics of Machine Translation: Parameter estimation. *Computational Linguistics*, 19(2):263–312, June.

S. Della Pietra, V. Della Pietra, and J. Lafferty. 1995. Inducing features of random fields. Technical Report CMU-CS-95-144, CMU.

George Foster, Pierre Isabelle, and Pierre Plamondon. 1997. Target-text Mediated Interactive Machine Translation. *Machine Translation*, 12:175–194.

George Foster. 2000. Incorporating position information into a Maximum Entropy / Minimum Divergence translation model. In *Proceedings of the 4th Computational Natural Language Learning Workshop (CoNLL)*, Lisbon, Portugal, September. ACL SigNLL.

Ismael García-Varea, Francisco Casacuberta, and Hermann Ney. 1998. An iterative, DP-based search algorithm for statistical machine translation. In ICSLP-98 (ICS, 1998), pages 1135–1138.

1998. *Proceedings of the 5th International Conference on Spoken Language Processing (ICSLP) 1998*, Sydney, Australia, December.

F. Jelinek and R. L. Mercer. 1980. Interpolated estimation of Markov source parameters from sparse data. In E. S. Gelsema and L. N. Kanal, editors, *Pattern Recognition in Practice*. North-Holland, Amsterdam.

Ph. Langlais and G. Foster. 2000. Using context-dependent interpolation to combine statistical language and translation models for interactive MT. In *Content-Based Multimedia Information Access (RIAO)*, Paris, France, April.

Philippe Langlais, Sébastien Sauvé, George Foster, Elliott Macklovitch, and Guy Lapalme. 2000. A comparison of theoretical and user-oriented evaluation procedures of a new type of interactive MT. In *Second International Conference On Language Resources and Evaluation (LREC)*, pages 641–648, Athens, Greece, June.

S. Niessen, S. Vogel, H. Ney, and C. Tillmann. 1998. A DP based search algorithm for statistical machine translation. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics (ACL) and 17th International Conference on Computational Linguistics (COLING) 1998*, pages 960–967, Montréal, Canada, August.

Franz Josef Och, Christoph Tillmann, and Hermann Ney. 1999. Improved alignment models for statistical machine translation. In *Proceedings of the 4nd Conference on Empirical Methods in Natural Language Processing (EMNLP)*, College Park, Maryland.

Harry Printz. 1998. Fast computation of Maximum Entropy/Minimum Divergence feature gain. In ICSLP-98 (ICS, 1998), pages 2083–2086.

Ronald Rosenfeld. 1996. A maximum entropy approach to adaptive statistical language modelling. *Computer Speech and Language*, 10:187–228.

Michel Simard, George F. Foster, and Pierre Isabelle. 1992. Using cognates to align sentences in bilingual corpora. In *Proceedings of the 4th Conference on Theoretical and Methodological Issues in Machine Translation (TMI)*, Montréal, Québec.

Ye-yi Wang and Alex Waibel. 1998. Fast decoding for statistical machine translation. In ICSLP-98 (ICS, 1998), pages 2775–2778.