Prediction of Words in Statistical Machine Translation using a Multilayer Perceptron

Alexandre Patry
Université de Montréal
2920 chemin de la Tour
Montréal (Québec), Canada H3T 1J4
patryale@iro.umontreal.ca

Philippe Langlais
Université de Montréal
2920 chemin de la Tour
Montréal (Québec), Canada H3T 1J4
felipe@iro.umontreal.ca

Abstract

We propose to estimate the probability that a target word appears in the translation of a given source sentence using a multilayer perceptron. At the expense of ignoring word order and repetition, our model does not assume word alignments and consider all source words jointly when evaluating the probability of a target word.

We compared our model against IBM1 which does not consider word order either. Our model was comparable with IBM1 when predicting the target words that should appear in the translation of a source sentence. When our model was extended to include alignment information, it surpassed IBM1 on all the metrics we used.

1 Introduction

When translating a sentence, a phrase-based model (Koehn et al., 2003) divides it in phrases that are translated individually and then reunited using a language model (usually a trigram) and a reordering model. A side effect of this strategy is that the dependencies between a target phrase and the words outside of its aligned source phrase are only captured by the language model.

For example, such a system could be asked to translate *plant* in French knowing only that the last two words produced are *de cette* (*of this*). It is easy to imagine a sentence where the translation should be *plante* the botanical term ("The leaves of this plant are red.") and another where it should be *usine* the building ("The walls of this plant are red.").

To overcome this limitation, Vickrey et al. (2005) proposed to condition the alignment score on features observed outside of the alignment. This technique has improved word alignments (Zhao and Xing, 2006), phrase-based translation (Carpuat and Wu, 2007; Stroppa et al., 2007) and hierarchical translation (Chan et al., 2007).

Another alternative explored is to select (Hildebrand et al., 2005) or weight (Lü et al., 2007) the training material according to the source sentence. In this setting, the context of an alignment is captured by giving a greater importance to translations occurring in similar contexts.

Finally, Bangalore et al. (2007) presented a global lexical selection model that gets rid of alignments altogether and conditions the probability of each target word on the whole source sentence. They used logistic regression models representing a source sentence with a set of ngrams and classifying each target word as present or absent in the translation. The selected words are then passed to a second module, which searches for their best ordering.

Our work is in line with their idea and introduces a global lexical selection model which uses a multilayer perceptron taking as input a bag-of-words. The differences between the two approaches are discussed in sections 3.2 and 5.

Because our model does not consider word order, we compared it against IBM1 which does not consider it either. When they were asked to predict the target words that should translate a source sentence, both IBM1 and multilayer perceptron were comparable. We also evaluated an extension to our model, which includes information on word alignments. This extension fared better than IBM1 for all metrics we used.

We describe the mathematical foundation of our work in section 2 and multilayer perceptrons in sec-

tion 3. We present and discuss empirical results in section 4 and 5 and then conclude in section 6.

2 Mathematical Model

Statistical machine translation models estimate the probability of a target sentence given a source sentence from a bitext. As bitexts are very small compared to the number of possible translations, independence assumptions are needed to obtain a model that will adapt to new source sentences. Most translation models assume sentences can be broken in independent alignments (words, phrases or trees) and then compute statistics on these alignments. Because we want to capture information outside of the alignments, we put forward different hypotheses:

1. Word ordering and repetitions are insignificant.

2. Target words are independent of each other.

These assumptions, which are obviously dubious, allow us to model the sentences as bags-of-words and to split the target sentence prediction into independent word predictions:

$$\begin{aligned} \Pr(\mathbf{t}|\mathbf{s}) &= \prod_{t \in \mathbf{t}} \Pr(t|\mathbf{s}) \prod_{t \in \mathcal{V}_T \setminus \mathbf{t}} \Pr(\bar{t}|\mathbf{s}) \\ \Pr(\bar{t}|\mathbf{s}) &= 1 - \Pr(t|\mathbf{s}) \end{aligned} \tag{1}$$

where $\mathbf{s}$ and $\mathbf{t}$ are source and target bag-of-words and $\mathcal{V}_T$ is the set of all target words. The two products compute respectively the probability that selected and unselected words translate $\mathbf{s}$.

We are now left with the evaluation of $\Pr(t|\mathbf{s})$, the probability of t being in the translation of $\mathbf{s}$. This probability can be approximated by any binary classifier, in our case we chose a multilayer perceptron.

3 Multilayer Perceptron

A perceptron is a linear classifier whose output is a function applied on the weighted average of its inputs:

$$y = f_o(b + \sum_s w_s x_s) \tag{2}$$

where x_s is an input, w_s its weight, b a constant bias and $f_o : \mathcal{R} \to \mathcal{R}$ the output function.

A perceptron whose input is the output of other perceptrons is called a multilayer perceptron (MLP):

$$\begin{aligned} y &= f_o(b + \sum_j w_j h_j) \\ h_j &= f_a(b_j + \sum_s w_{js} x_s) \end{aligned} \tag{3}$$

where h_j is an hidden unit and f_a is called the activation function.

According to Equation 1, one classifier is needed for each target word. Instead of having many independent MLP with a single output unit, we opted for a single MLP with many output units:

$$\begin{aligned} y_{t|\mathbf{s}} &= \mathrm{sigm}(b_t + \sum_j w_{tj} h_j) \\ h_j &= \tanh(b_j + \sum_s w_{js} x_s) \\ \mathrm{sigm}(z) &= (1 + e^{-z})^{-1} \\ x_s &= \begin{cases} 1 & \text{if } s \in \mathbf{s}, \\ 0 & \text{otherwise.} \end{cases} \end{aligned} \tag{4}$$

When the weights are optimized correctly, $y_{t|\mathbf{s}}$ approximates $\Pr(t|\mathbf{s})$.

3.1 Training

Because we want the outputs of our MLP to model posterior class probability, we optimize it to minimize negative log-likelihood of the training bitext:

$$\begin{aligned} E &= -\log \prod_{i=1}^{n} \mathrm{p}(\mathbf{t}^i|\mathbf{s}^i) \\ &= -\sum_{i=1}^{n} \sum_{t \in \mathbf{t}^i} \log \mathrm{p}(t|\mathbf{s}^i) + \sum_{t \notin \mathbf{t}^i} \log \mathrm{p}(\bar{t}|\mathbf{s}^i) \end{aligned} \tag{5}$$

where $\mathbf{s}^i$ and $\mathbf{t}^i$ are the sentence pairs in the training bitext.

The partial derivative for the weights of this MLP for the sentence pair $(\mathbf{s}^i, \mathbf{t}^i)$ encoded in binary vec-

tors $(\mathbf{x}^i, \tau^i)$ are (Bishop, 1995):

$$\begin{aligned}
\frac{\partial E}{\partial w_{tj}} &= h_j(y_t^i - \tau_t^i) \\
\frac{\partial E}{\partial h_j} &= \sum_t (y_{t|\mathbf{s}}^i - \tau_t^i) w_{tj} \\
\frac{\partial h_j}{\partial w_{js}} &= x_s^i (1 - \tan^2(\sum_s w_{js} x_s^i)) \\
\frac{\partial E}{\partial w_{js}} &= \frac{\partial E}{\partial h_j} \frac{\partial h_j}{\partial w_{js}}
\end{aligned} \tag{6}$$

Once these derivatives are known, the weights can be optimized using any gradient-based algorithm. Section 4.4 describes the training algorithm we used in our experiments.

3.2 Motivations for MLP

Bangalore et al. (2007) modeled Equation 1 using logistic regression, which is equivalent to a perceptron (Equation 2) with a sigmoid output function. Compared to perceptrons, MLPs are more complex to train and are not convex but they scale better to the size of the vocabularies, they model problem that are not linearly separable and they can assign non-zero probabilities to words that never co-occurred in the training corpus.

The number of weights of a logistic regression model is in $O(|\mathcal{V}_S| \cdot |\mathcal{V}_T|)$ where $\mathcal{V}_S$ and $\mathcal{V}_T$ are respectively the set of source and target words known to the system. Because of its hidden layer, the number of weights in our MLP is in $O(|\mathcal{V}_S|h + h|\mathcal{V}_T|)$ where h is the number of hidden units, which was always smaller than the vocabulary sizes in our experiments.

The hidden layer of our MLPs acts as a continuous space where the source words are projected. Even though this is not obvious to control, we hope the hidden units will capture concepts or topics. Translating a source bag-of-words is thus a two-step operation. It is first projected in a concept space that is then projected in the space of target words. This way, the probability of a target word can be increased by source words that never co-occurred with it as long as they express common concepts.

4 Experiments

4.1 Data

We evaluated our system on a subset of the English and French parts of EUROPARL (Koehn, 2005), a corpus extracted from the proceedings of the European Parliament.

We used $100,000$ sentence pairs to train our models (TRAIN), $10,000$ to compare them at different training stages (DEV) and $2,000$ other sentence pairs (TEST) to evaluate the models that performed best on DEV.

4.2 Baselines

The first baseline against which we compared our model is an IBM1 model computed by GIZA++ (Och and Ney, 2003) with its default configuration[1], the only alignment model presented by Brown et al. (1993) which does not account for the order of source words. We evaluated IBM1 as:

$$\Pr(t|\mathbf{s}) = \frac{1}{1+|\mathbf{s}|}\left(\Pr(t|\epsilon) + \sum_{s \in \mathbf{s}} \Pr(t|s)\right) \tag{7}$$

where ϵ is a special *null word*.[2] When Och et al. (2004) evaluated different rescoring features to enhance statistical machine translation, they got one of their best improvement with IBM1.

Our second baseline is a perceptron (Equation 2) where each source word is linked to its 10 best translations according to IBM1.[3]

4.3 Multilayer Perceptrons

The results reported in this section were obtained from a MLP with 500 hidden units (henceforth MLP). MLP has no links between units representing source and target words, thus no notion of word alignment. Since word alignments are a reality in high-quality parallel texts, we evaluated a second model (MLP+) which contains 500 hidden units to which we added the direct connections that are found in our baseline perceptron.

[1] Five EM iterations and only lexicon entries with a probability higher than 1^{-7} were considered.

[2] We abuse the notation and allow word repetition in $\mathbf{s}$ and $\mathbf{t}$ for IBM1.

[3] Our implementation was too slow to contain all the IBM1 connections.

Each training iteration lasted approximately 12 hours on computers equipped with two *Intel Xeon Quadcore* processors; each model fitted in one gigabyte of RAM.

4.4 Training

We optimized the weights of our MLP using a conjugate gradient method (CG). As in gradient descent, at each iteration, CG selects a step direction, a step size and then update the weights. While gradient descents step in the gradient direction, CG steps in a modified gradient direction such that each step is orthogonal to that of preceding iterations. We selected the step size using BRENT line search algorithm (Press et al., 1992, Chapter 10.2).

We sped up training by ignoring all words occurring less than five times in TRAIN. The French vocabulary was reduced from $40,383$ to $15,314$ words and English vocabulary from $30,155$ to $11,970$ words. The sentential ratio of target vocabulary words that could not be predicted by the different models is presented in Table 1. Even though MLP and MLP+ ignored all words appearing less than five times in TRAIN, they still obtained a better coverage than PERCEPTRON and IBM1 thanks to the layer of hidden units which allows the models to assign probabilities to target words that never co-occurred with any of the words in the source sentence.

We also mini-batched the sentence pairs in groups of 100 and updated the weights for each of them. Gradients computed for such mini-batches are noisy, we thus reset the memory of previous directions before processing each mini-batch with a probability of 5%.

We regularized our models with a L1 penalty that was used only when computing the step size. Had we applied the penalty on updates, weights associated with rare source words would have always been pushed toward zero. We set the penalty weight to 0.1.

Finally, weights were initialized with random values following a uniform distribution between -0.1 and 0.1. Because we expected an empty target bag-of-words given an empty source bag-of-words, we fixed biases to 0 and did not update them.[4]

[4]Since $\text{sigm}^{-1}(0) \rightarrow -\infty$, we fixed output biases to $\text{sigm}^{-1}(0.01) \approx -4.595$.

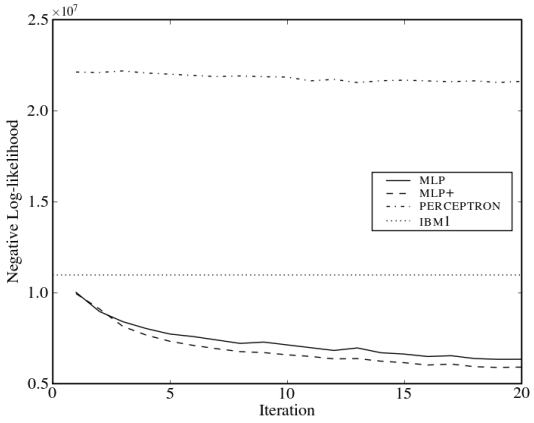

Figure 1: Negative log-likelihood of the TRAIN corpus decreased with successive training iterations.

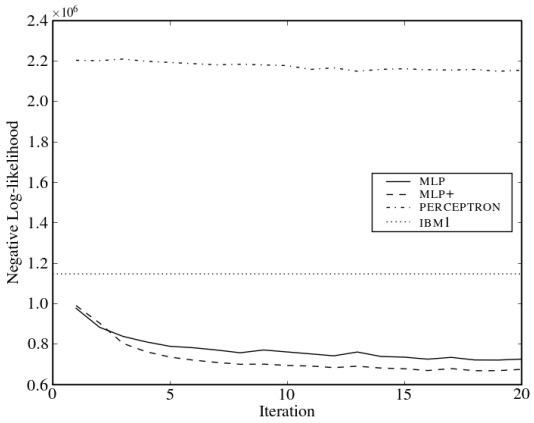

Figure 2: Negative log-likelihood of the DEV corpus decreased with successive training iterations; no overfitting was observed.

4.5 Negative Log-Likelihood

To make sure that our models were learning as we expected them to, we evaluated their negative log-likelihood at each training iteration (Figures 1 and 2). This was the case for our three perceptron-based models, which saw their negative log-likelihood decrease over iterations on TRAIN without overfitting on DEV.

Regarding the different model architectures, hidden units brought a large decrease in negative log-likelihood, but the best results were obtained with MLP+ which have both hidden units and direct connections.

Negative log-likelihood helps to see how each

Model	Target OOV (%)		
	TRAIN	DEV	TEST
MLP and MLP+	1.9	2.5	2.4
PERCEPTRON	12.8	22.4	21.2
IBM1	0.002	4.0	3.9

Table 1: Percentage of target words to which no probability was assigned by the different models averaged over sentences.

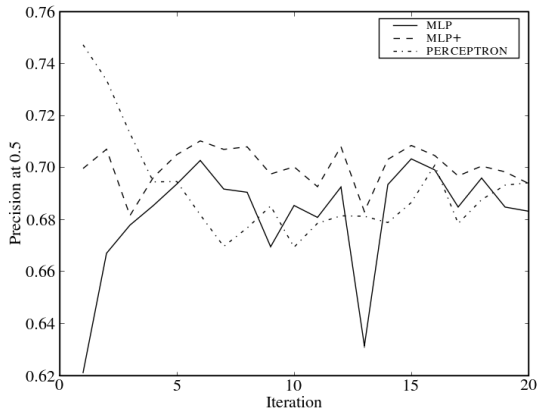

Figure 3: Average precision of words predicted by our models.

model improves during training but is difficult to interpret, especially when comparing models with different architectures. The figures of IBM1 should thus be seen as a sanity check. The following sections looks at how the models compare in different situations.

4.6 Bag-of-words Predictions

To assess how good the different models were at predicting target words from a bag of source words, we computed their average precision and recall. For each sentence in DEV, we forwarded the source words to the models and all target words predicted with a probability higher than 0.5 were retained. We then computed precision and recall using the target sentences of DEV as references. Figures 3 and 4 present precision and recall averaged over all the sentences of DEV.

The increase in recall indicates that as the iterations pass, more and more words are predicted. Precision tends to stabilize so the ratio of valid words

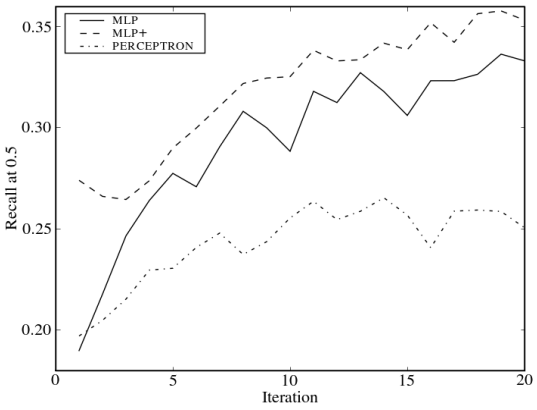

Figure 4: Average recall of words predicted by our models.

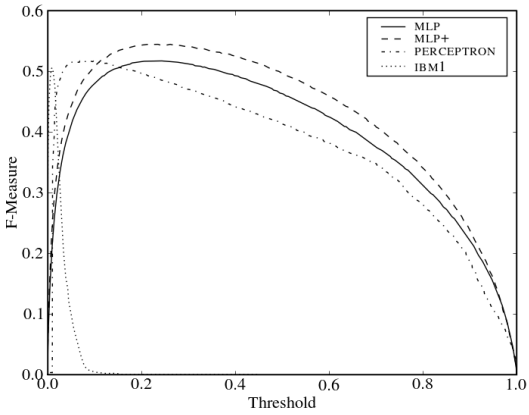

Figure 5: F-measure obtained by varying the threshold.

stabilizes as well. The precision of MLP dropped at iteration 9, which is the iteration where we observed the largest increase in negative log-likelihood on DEV.

All predictions of IBM1 yielded a probability below 0.5 so they were not discussed in this section.

4.7 Truncated Lists of Words

In the previous section, we showed that precision and recall increase as the negative log-likelihood improve when the threshold is 0.5. For this section, we selected the models with the best negative log-likelihood on DEV and evaluated their f-measure as the threshold varied (Figure 5).

The best f-measure of IBM1 (0.506) is close to that of PERCEPTRON (0.517) and MLP (0.517), but

Model	It.	Thres.	Prec.	Rec.
MLP	19	0.221	0.53	0.50
MLP+	18	0.214	0.55	0.54
PERCEPTRON	19	0.086	0.54	0.50
IBM1	–	0.009	0.47	0.55

Table 2: Precision (Prec.) and recall (Rec.) at threshold (Thres.) yielding the best f-measure on TEST for models computed at iteration (It.) giving the lowest negative log-likelihood to DEV.

Model	Threshold			
	$(0, 10]$	$(10, 20]$	$(20, 30]$	$(30, \infty]$
MLP	0.215	0.236	0.216	0.225
MLP+	0.207	0.216	0.233	0.255
PERC.	0.078	0.076	0.054	0.089
IBM1	0.029	0.015	0.009	0.006

Table 3: Thresholds yielding the highest f-measure on TEST for source sentences of different sizes.

we observed larger values for MLP+ (0.544). Detailed results are presented in Table 2 and an example of the output of the best configurations is presented in Figure 6.

Figure 5 shows that good thresholds must be selected in a small range for IBM1 and in a wider range for the other models. We ran a second set of experiments to see how source sentences lengths affected this threshold. The results of these experiments are presented in Table 3 and 4.

The thresholds of all models had variations, but they are much more important for IBM1. We selected the thresholds obtained from sentences containing one to ten words and used them to extract words from sentences containing 30 words and more. While IBM1 f-measure dropped from 0.56 to 0.15 on long sentences, it dropped by less than 0.01 on the other three models. The probabilities computed by IBM1 could probably be set using a heuristic considering the number of source words, but this is out of the scope of this paper.

Model	F-Measure			
	$(0, 10]$	$(10, 20]$	$(20, 30]$	$(30, \infty]$
MLP	0.52	0.51	0.52	0.52
MLP+	0.54	0.53	0.54	0.56
PERC.	0.44	0.48	0.52	0.55
IBM1	0.49	0.50	0.54	0.56

Table 4: Best f-measures obtained on TEST for source sentences of different sizes.

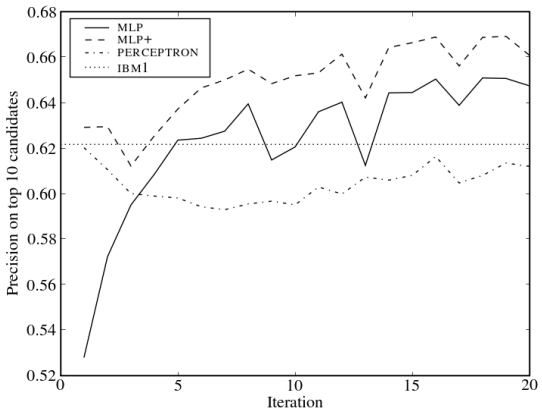

Figure 7: Precision on ten first target words averaged over the sentences of DEV.

On shorter sentences, MLP+ and MLP had better f-measures than IBM1, but on longer sentences MLP is worse than the three other models which connect source and target words directly.

4.8 Sorted Lists of Words

Bag-of-words prediction can be interpreted as an information retrieval problem where source sentences are queries and words in target sentences are relevant documents. Using this formulation, we evaluated the output of the different models as sorted lists of words using precision at ten (Figure 7) and mean average precision (Figure 8).

For each sentence in DEV, we computed the precision on the ten best target words (Figure 7). While IBM1 had a precision of 0.62, our best MLP had 0.65 and our best MLP+ 0.67, which means that we expect MLP+ to get one more word right than IBM1 when two lists of ten words are evaluated.[5]

We evaluated the complete lists of target words using mean average precision (MAP), which aver-

[5]When we take into account the number of sentences containing less than 10 words, the upper bound for this metric changes from 1 to 0.95

Source	particular attention have been paid to the issue of employment
Reference	la question de l' emploi fait l' objet d' une attention particulière
MLP	attention(0.92), question(0.75), l'(0.71), de(0.65), *été*(0.47), *à*(0.46), la(0.46)
MLP+	attention(0.78), l'(0.75), question(0.69), emploi(0.60), de(0.54), la(0.52), particulière(0.50)
PERCEPTRON	emploi(0.70) *a*(0.55) attention(0.52), de(0.38), *à*(0.36), l'(0.35), question(0.30)
IBM1	de(0.08) emploi(0.06) la(0.05), attention(0.05), *a*(0.05), question(0.04), *été*(0.04)

Figure 6: Example outputs where probabilities are specified between parentheses and words that are not in the reference are italicized.

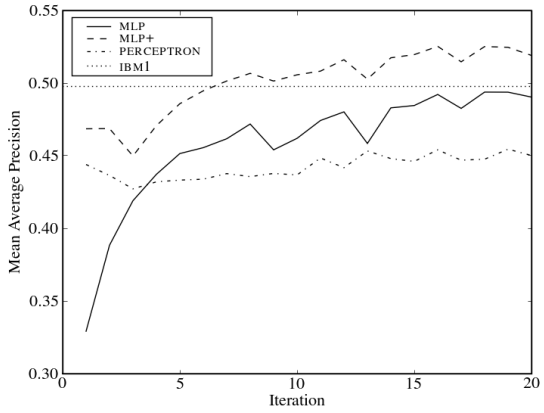

Figure 8: Mean average precision averaged over the sentences of DEV.

ages the precision computed at all the points in the list where a valid word is found (Figure 8).[6] Even if MLP and MLP+ predicted $15,340$ words for every sentence and IBM1 predicted 3239 words on average, MLP was comparable to IBM1 and MLP+ was superior by more than 2%.

4.9 Number of Hidden Units

We trained models using 10, 100 and 1000 hidden units. When no direct connections are included, 10 and 100 units are clearly not enough. For a given source sentence, the probability of most word tends to approach zero, so using a multilayer perceptron containing too few hidden units drives almost all probabilities to zero. When using 1000 units, the results were comparable to those obtained when using 500 units and no overfitting was observed either.

Models with 10 or 100 hidden units to which we added direct connections outperformed PERCEPTRON but were inferior to models with 500 or 1000 hidden units. Results obtained with 500 and 1000 hidden units were still very similar.

[6]For instance, a list containing three valid words at positions 2, 5 and 12 would get a mean average precision of $\frac{1}{3}(\frac{1}{2}+\frac{2}{5}+\frac{3}{12})=0.38$.

5 Discussion

To our knowledge, Bangalore et al. (2007) were the first to introduce a translation model that was not explicitly assuming word alignments. To do so, they used a logistic regression model taking as input ngrams of source words.

Whereas they modeled source words dependencies using ngrams, we captured them automatically by using a multilayer perceptron. Our input space is smaller, but it does not contain any information about relative word positions.

We were very encouraged by MLP, which does not encode word alignment information in its architecture but had results comparable to IBM1 on a parallel text of good quality. The main advantage of MLP over the other models is that its coverage and space complexity do not depend directly on word co-occurrences. Such a model could thus be trained on complete document pairs whose sentence alignments are unknown or undefined, much like comparable corpora.

We intend to use our model as a feature in a system designed to rescore the top ranked candidates of a statistical machine translation system. We expect the improvement to be at least as good as that possible with IBM1 models if not better.

6 Conclusion

We proposed to translate bags-of-words using a multilayer perceptron. While being more difficult to optimize, multilayer perceptrons offer many advantages over the famous IBM1 model. No assumption

of independence between source words is made, hidden units offer a way to customize the model and prior information can easily be added as direct links between the input and output units. Our experiments suggest that models without direct connections are comparable to IBM1 but models including direct connections surpass it.

The major showstopper for our models is their exorbitant training time. While we selected the number of hidden units empirically, many algorithms were proposed to grow a multilayer perceptron as needed (Bishop, 1995, Chapter 9.5). The training time could also be reduced using parallelization. Bengio et al. (2003) propose hints on how to parallelize the training of large multilayer perceptrons.

This work is still preliminary and experiments are needed to verify if our models can help improving a statistical machine translation system. As no word alignment is assumed to optimize our model or ensure that its space complexity is reasonable, we plan to investigate how they could capture information from comparable corpora where word and sentence alignments are undefined.

References

Srinivas Bangalore, Patrick Haffner, and Stephan Kanthak. 2007. Statistical machine translation through global lexical selection and sentence reconstruction. In *ACL*.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, Christian Jauvin, Jaz K, Thomas Hofmann, Tomaso Poggio, and John Shawe-taylor. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

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

Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: parameter estimation. *Computational Linguistics*, 19(2):263–311.

Marine Carpuat and Dekai Wu. 2007. Improving statistical machine translation using word sense disambiguation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 61–72.

Yee Seng Chan, Hwee Tou Ng, and David Chiang. 2007. Word sense disambiguation improves statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 33–40, Prague, Czech Republic.

Almut Silja Hildebrand, Matthias Eck, Stephan Vogel, and Alex Waibel. 2005. Adaptation of the translation model for statistical machine translation based on information retrieval. In *Proceedings of the European Association for Machine Translation 10th Annual Conference*, Budapest, Hungary.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *NAACL '03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 48–54, Edmonton, Canada.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Proc. of MT Summit*.

Yajuan Lü, Jin Huang, and Qun Liu. 2007. Improving statistical machine translation performance by training data selection and optimization. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 343–350.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Franz Josef Och, Daniel Gildea, Sanjeev Khudanpur, Anoop Sarkar, Kenji Yamada, Alex Fraser, Shankar Kumar, Libin Shen, David Smith, Katherine Eng, Viren Jain, Zhen Jin, and Dragomir Radev. 2004. A smorgasbord of features for statistical machine translation. In Daniel Marcu Susan Dumais and Salim Roukos, editors, *HLT-NAACL 2004: Main Proceedings*, pages 161–168, Boston, Massachusetts, USA.

William Press, Saul Teukolsky, William Vetterling, and Brian Flannery. 1992. *Numerical Recipes in C*. Cambridge University Press, Cambridge, UK, 2nd edition.

Nicolas Stroppa, Antal van den Bosch, and Andy Way. 2007. Exploiting source similarity for smt using context-informed features. In *Proceedings of the 11th International Conference on Theoretical and Methodological Issues in Machine Translation*, pages 231–240, Skövde, Sweden.

David Vickrey, Luke Biewald, Marc Teyssier, and Daphne Koller. 2005. Word-sense disambiguation for machine translation. In *HLT '05: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 771–778, Vancouver, Canada.

Bing Zhao and Eric P. Xing. 2006. BiTAM: Bilingual topic admixture models for word alignment. In *Proceedings of the COLING/ACL on Main conference poster sessions*, pages 969–976, Sydney, Australia.