



## THE SIXTH FRAMEWORK PROGRAMME

The sixth framework programme covers Community activities in the field of research, technological development and demonstration (RTD) for the period 2002 to 2006



# S M A R T

Statistical Multilingual Analysis  
for Retrieval and Translation

D2.2: APPLICATION OF MARKOV APPROACHES TO SMT

Version 1.0  
29/September/2008



# Executive Summary

| VERSION | DATE       | AUTHOR                                                                                                                      |
|---------|------------|-----------------------------------------------------------------------------------------------------------------------------|
| 0.1     | 20/06/2008 | Craig Saunders, Skeleton                                                                                                    |
| 0.2     | 08/09/2008 | Craig Saunders, George Foster, Matti Kääriäinen, Sandor Szedmak, Zhouan Wang, Initial Draft                                 |
| 0.3     | 12/09/2008 | Craig Saunders, George Foster, Matti Kääriäinen, Sandor Szedmak, Zhouan Wang, Marc Dymetman, Draft for Review               |
| 1.0     | 29/09/2008 | Craig Saunders, George Foster, Matti Kääriäinen, Sandor Szedmak, Zhouan Wang, Marc Dymetman, Incorporated Reviewer Comments |

|                      |                                               |
|----------------------|-----------------------------------------------|
| TITLE                | D2.2: Application of Markov approaches to SMT |
| STATE                | Draft                                         |
| CONFIDENTIALITY      | PU                                            |
| AUTHOR(S)            | Craig Saunders                                |
| CONFIDENTIALITY      | SMART consortium                              |
| PARTICIPANT PARTNERS | Helsinki, Southampton, UCL, XRCE, NRC         |
| WORKPACKAGE          | WP2                                           |
| ABSTRACT             |                                               |
| KEYWORDS             |                                               |
| REFERENCES           |                                               |
| COMMENTS             |                                               |
| REVIEWER             | Juho Rousu                                    |

# Contents

|          |                                               |          |
|----------|-----------------------------------------------|----------|
| <b>1</b> | <b>Overview of this document</b>              | <b>4</b> |
| <b>2</b> | <b>Machine Translation Systems</b>            | <b>5</b> |
| 2.1      | Portage . . . . .                             | 5        |
| 2.1.1    | Experimental Settings . . . . .               | 6        |
| 2.2      | LSR Approach . . . . .                        | 6        |
| 2.3      | Sinuhe . . . . .                              | 7        |
| 2.3.1    | Building a phrase table . . . . .             | 8        |
| 2.3.2    | The translation model . . . . .               | 9        |
| 2.3.3    | Experiments . . . . .                         | 14       |
| 2.4      | MMR based approaches . . . . .                | 17       |
| 2.5      | Model behind the learning approach . . . . .  | 18       |
| 2.6      | Notations . . . . .                           | 21       |
| 2.7      | Learning elements . . . . .                   | 21       |
| 2.7.1    | Similarity measure of words . . . . .         | 21       |
| 2.7.2    | Word features . . . . .                       | 23       |
| 2.7.3    | Output labels of the words . . . . .          | 24       |
| 2.8      | Learning problem . . . . .                    | 25       |
| 2.9      | Example . . . . .                             | 26       |
| 2.10     | Multiview learning . . . . .                  | 32       |
| 2.11     | Phrase extraction . . . . .                   | 33       |
| 2.12     | Sentence translation . . . . .                | 34       |
| 2.13     | Matrax . . . . .                              | 35       |
| 2.13.1   | Non-contiguous phrases . . . . .              | 35       |
| 2.13.2   | Definition and library construction . . . . . | 36       |
| 2.13.3   | The Model . . . . .                           | 37       |
| 2.13.4   | Parameter Estimation . . . . .                | 39       |
| 2.13.5   | Decoder . . . . .                             | 42       |



|          |                               |           |
|----------|-------------------------------|-----------|
| <b>3</b> | <b>Experimental Results</b>   | <b>43</b> |
| 3.1      | Experimental Set-up . . . . . | 43        |
| 3.2      | Resources . . . . .           | 44        |
| 3.2.1    | Portage . . . . .             | 44        |
| 3.2.2    | Matrax . . . . .              | 44        |
| 3.2.3    | Results . . . . .             | 46        |
| 3.3      | Conclusions . . . . .         | 46        |

# Chapter 1

## Overview of this document

This document forms deliverable 2.2 “Application of Markov-based approaches to SMT” of the SMART project. It represents a comparison of the current state of the full translation systems developed within the project, resulting mainly from Workpackages 2 and 3 thus far. The evaluation suite as developed in Workpackage 6 is used to provide a platform which ensure that all systems operate on the same data, and as such their translations can be automatically evaluated using a range of metrics and statistical significance between methods can be estimated. In order to reduce the length of this document, where a technique has been included in a previous deliverable we simply reference it, only including newly developed material in this document (however previous systems are also used in the tests for comparison). Accordingly, we first describe each of the translation systems used in Chapter 2 and then detail the results of the experimental study and give some interpretation in Chapter 3. The final chapter summarises the findings and describes anticipated directions for the future in relation to specific workpackage tasks.



## Chapter 2

# Machine Translation Systems

In this chapter we describe each of the systems that will be used in the tests. The baseline for the SMART project is the Portage system as it existed at the start of the project. We also give brief descriptions here of how techniques from previous deliverables have been applied to the benchmark data sets, and furthermore give an outline of the Matrax translation system from XRCE. The novel aspects specific to SMART in this section is the development of an MMR-based approach to word alignment and the introduction of a phrase-based exponential model for translation called *Sinuhe*.

### 2.1 Portage

Portage (2006) represents the baseline system and forms one of the benchmarks upon which the SMART project will be evaluated. Portage is a standard two-pass phrase-based system, similar to [1]. Features in the first-pass loglinear model include phrase scores derived from symmetrized IBM2 word alignments (using both relative-frequency and IBM1-based “lexical” estimates of conditional phrase probabilities, with a phrase-length limit of seven), four-gram language model probability, word count, and phrase-displacement distortion. The second-pass (rescoring) model additionally uses IBM2 scores and untranslated-word heuristics (in both directions), word, phrase, and length posterior probabilities, and quote and parenthesis mismatch indicators. Parameter tuning is performed using Och’s minimum-error-rate algorithm [2]. The system is trained on lowercase text, and a separate post-processing step is used to restore case



information to the output using an HMM-based approach.

### 2.1.1 Experimental Settings

For the “full” training condition, three 1000-sentence development sets were extracted from the training material by sampling at fixed intervals. The first two sets were used for training the first- and second-pass loglinear models, while the third was held in reserve (not used at all yet). The remaining training material was used for phrase-pair extraction and language model training.

A similar scheme was used for the 10k and 50k conditions, except that only two 500-sentence development sets were used.

## 2.2 LSR Approach

The Least Squares Regression (LSR) algorithm follows the kernel regression approach introduced in Deliverable D2.1. Concretely, we embed both the input and output sentences into respective feature spaces, where the features are the counts of word  $n$ -grams occurring in the sentence. Then we seek a matrix-represented parameter to map the input feature vector to the output one. To obtain the solution, kernel ridge regression is employed which avoids dealing with the explicit features by applying the so-called ‘kernel trick’. Finally, we decode the translation sentence from its feature vector predicted by the regression model via searching in all the possible translations for the one whose feature vector has the smallest Euclidean distance to our prediction. During this process, a beam search algorithm is utilized, with the support of a phrase table to generate the potential translation candidates. In addition, we also add a language model (LM) to the decoding procedure to gain better quality outputs.

As a kernel method, due to its computational complexities, our system will only participate the tasks with 10k training sentences and 1k test sentences. In all these experiments, our model is parameterized as follows. We use the blended tri-gram kernel that counts the word uni-grams, bi-grams, and tri-grams simultaneously to induce both the input and output feature spaces. The language models used in the experiments are Kneser-Ney smoothed tri-gram models trained on the target language training sentences. To integrate the LM into our regression framework, we choose the tighter integration approach introduced in Deliverable D3.2 Chapter 2, in which we add a weighted LM score to the objective function of our ridge regression as an additional loss, and seek solution minimizing the LM-regularized least-squares loss on the training set. In the decoding process, we set the beam size to 100. The phrase-tables selected to generate candidate translations are extracted from the corresponding training



sets. In addition, there are two parameters to be tuned, the ridge regression regularization coefficient and the weight for the LM. In our case, it will be too expensive to do cross-validation; therefore only a restricted range of values for the parameters are evaluated. Preliminary experiments show that the model is insensitive to the ridge regression regularization coefficient, and a value around 0.01 will be proper for it. The LM weight should make the LM scores comparable to the regression scores, which is set to  $0.5 \times 10^6$  in this task.

## 2.3 Sinuhe

In this section we give a high-level description of **Sinuhe**, a phrase-based SMT system based on a conditional exponential family translation model. The translation model can also be viewed as a conditional Markov model with variable length history, so **Sinuhe** can be seen as a “Markov approach to SMT”. We first describe how **Sinuhe**’s phrase-level features are extracted, then move on to describe our proposed translation model and how its parameters are estimated from data, and finally describe how the model can be used to predict translations (by itself or combined with a language model).

**Related work** The translation model we propose is built around the same phrase-level features that are used in standard phrase-based statistical machine translation systems [1], and is thus closely related to them. However, we use the phrase-level features quite differently from previous approaches. In a nutshell, we use the phrases to define a conditional exponential family probability distribution for feature vectors of translations, train the model parameters discriminatively by optimizing a global training criterion, and solve optimization problems related to training the model parameters and predicting the most likely translations using efficient exact algorithms. Our models and algorithms have their foundations in well-known machine learning methods, so the novel part in our work is the exiting new mix of familiar ingredients.

The work most closely resembling ours is the approach developed by Blunson et al [3]. Unlike us, their work is based on a model that uses synchronous grammars, but also they propose a global discriminative log-linear model that takes into account multiple derivations. This is quite analogous to our use of overlapping phrases. They show that their model can be trained on more than a hundred thousand sentences, and that the resulting translation model produces better translations than the translation model of a baseline system **Hiero**. Though very interesting, their model has several shortcomings not shared by the model proposed by us. For example, the model has problems with reachability of reference translations, has not been scaled up to larger datasets, and lacks an



efficient exact decoding procedure. Another closely related approach is that by Tillmann et al [4], who propose a discriminative block bigram prediction model. However, as the name suggests, the model is restricted to phrases of length at most two. Yet another discriminative approach that is somewhat related to ours is that of Liang et al [5], who augment a standard phrase-based model with additional features whose weights are trained discriminatively by a variant of the perceptron algorithm [5]. Also their approach is non-probabilistic, has problems with reachability of reference translations, and has not been scaled up to large enough numbers of features so that biphrase features could be used directly.

### 2.3.1 Building a phrase table

We use the tools provided with `Moses` [6] to build the phrase table. We begin with a sentence aligned parallel corpus  $S$  consisting of source and target language sentence pairs  $(x, y)$ . The words within the sentences are aligned using Giza++ in both directions, and the alignments are symmetrized using the tools provided with `Moses`. The word alignment process maps the original sentence pairs  $(x, y)$  into word-aligned sentence pairs  $(x, a, y)$ , where  $a$  is a bipartite graph that represents the word alignment between the source and target words. In general,  $a$  may link each source and target word to any number of words in the opposite language. Other word alignment strategies could be used as well, but we haven't experimented with any yet.

Following [7], we extract from each aligned sentence pair  $(x, a, y)$  all the phrase pairs (or biphases)  $(x', a', y')$  that satisfy the following criteria: (1)  $x'$  and  $y'$  are consecutive word subsequences of  $x$  and  $y$ , respectively, and both have length at most  $k$ , (2)  $a'$  is the alignment between words of  $x'$  and  $y'$  induced by  $a$ , (3)  $a'$  contains at least one link, and (4) there are no links in  $a$  that have just one end in  $x'$  or  $y'$ . Each aligned training sentence  $(x, a, y)$  thus generates a number of potentially overlapping aligned biphrase features  $(x', a', y')$ . In our experiments, we chose  $k = 8$ . Unlike in `Moses`, we do not map the aligned biphases  $(x', a', y')$  back to non-aligned biphases  $(x', y')$ , but retain the alignments.

To reduce the number of phrase table entries, the phrase table is simplified by dropping all but the top  $K = 20$  most frequently extracted aligned biphases  $(x', a', y')$  for each source phrase  $x'$ . Also `Moses` employs an analogous pruning strategy. We use two additional criteria to prune the set of extracted biphases  $(x', a', y')$  that are used in `Sinuhe`. First, to simplify and speed up the dynamic programming algorithms used in training the model parameters and predicting translations, we require that the first and the last words of the source phrase of each aligned biphrase is aligned to one or more words of the target phrase. Second, we filter out all aligned biphases that occur only once in the training



corpus. This is commonly done also in the current phrase-based SMT systems, since doing so has been observed to reduce the phrase table size dramatically without much impact on translation quality. Our primary motivation, however, is to use this filtering as a leave-one-out overfitting avoidance procedure for feature extraction. Without this leave-one-out procedure, all the aligned biphrases that occur in the aligned training sentences would actually be present in the feature set of the model, whereas the feature vectors for future test data would be considerably more sparse. We believe the leave-one-out procedure helps in removing this bias, but our experiments on this are still ongoing. The leave-one-out procedure reduces the number of features significantly, but in our experiments, millions of features per language pair still remain.

### 2.3.2 The translation model

#### Features

The features of our conditional exponential translation model are binary indicator vectors that indicate which of the aligned biphrases that occur in the phrase table in a candidate translation, and where in the translation they occur. More specifically, a source sentence  $x$  aligned to a candidate translation  $y$  by an alignment  $a$  is represented by a binary feature vector  $\phi(x, a, y)$ , where  $\phi(x, a, y)_{(x', a', y'), i}$  is 1 iff the aligned biphrase  $(x', a', y')$  occurs at source sentence position  $i$ , and 0 otherwise. Here,  $(x', a', y')$  occurs in  $(x, a, y)$  iff  $(x', a', y')$  would have been extracted from  $(x, a, y)$  by the phrase extraction process. The weights for the aligned biphrases are tied together by mapping the binary feature vector  $\phi$  (indexed by aligned biphrase and position pairs) to an integral feature vector  $\tilde{\phi}$  (indexed by aligned biphrases) by the formula  $\tilde{\phi}_{(x', a', y')} = \sum_i \phi_{(x', a', y'), i}$ . The “real” features that drive the translation process are thus the lowest level binary features  $\phi$ , whereas the higher level representation  $\tilde{\phi}$  is convenient as it lives in the same space as the parameter vector for the model.

#### The model

Instead of modelling the conditional distribution  $P(y|x)$  directly, we model the conditional distribution  $P(\phi(x, a, y)|x)$ . We use the following conditional exponential model:

$$P(\phi(x, a, y)|x) = \frac{\exp(w \cdot \tilde{\phi}(x, a, y))}{\sum_{\phi \in \Phi_x} \exp(w \cdot \tilde{\phi})}$$

Here,  $w$  is a parameter vector with one component for each aligned biphrase feature, i.e., for each extracted aligned biphrase that survived the pruning. The



## D2.2: Application of Markov approaches to SMT

---

set  $\Phi_x$  of feasible feature vectors for  $x$  includes all feature vectors  $\phi$  satisfying the following criteria:

1. There exists a translation  $y'$  and an alignment  $a'$  such that all active features in  $\phi$  occur in  $(x, a', y')$
2. Features corresponding to aligned biphrases that occur inside aligned biphrases whose features are active in  $\phi$  are active in  $\phi$ .

Thus, in addition to all feature vectors  $\phi(x, a', y')$  that can be realized by real translations, the set  $\Phi_x$  may contain feature vectors  $\phi$  that lack some features that in any real translation would have been forced active by other active features. This is because we cannot force sub-phrases implying super-phrases in our dynamic programming algorithms, but only super-phrases forcing sub-phrases. Ideally, it would be better if we could restrict  $\Phi_x$  to consist of realizable feature vectors only as some of the probability mass is now wasted on non-realizable feature vectors. However, we hope and believe that the non-realizable feature vectors that now creep into  $\Phi_x$  have little effect in practice.

The choice of modelling  $P(\phi(x, a, y)|x)$  instead of  $P(y|x)$  is crucial, both from a modelling and from a computational perspective. From a modelling perspective, the crucial point is that any aligned sentence pair  $(x, a, y)$  has an associated feature vector  $\phi(x, a, y)$  that receives non-zero probability by our model. In conventional phrase-based SMT systems such as **Moses** that (implicitly or explicitly) model  $P(y|x)$  directly, it often happens that reference translations  $y$  in the training corpus are not reachable from their associated source sentences  $x$ , and thus receive zero probability independent of the model's parameters. This problem is not marginal: In our experiments on the Spanish to English Europarl task, around 80% of the reference translations on a test set were unreachable by using a standard phrase-based decoder similar to that used by **Moses**. The unreachability problem rules out the direct use of maximum likelihood type methods. Thus, in training the log-linear weights of conventional phrase-based SMT systems, one has to resort to reachable pseudo-references that may bias the training process in complicated ways, or to non-probabilistic methods like minimum error rate training.

From a computational perspective, working with feature vectors  $\phi(x, a, y)$  instead of the underlying targets and alignments means that the translation model models the ordering and choice of words in  $y$  only partially. This way, when computing the normalizing constants and feature expectations, we do not have to deal with the unbounded set of all possible translations  $y$  and alignments  $a$  for a source  $x$ . Instead, we can operate on a smaller number of equivalence classes given by  $\phi(x, a, y)$ . The price for this computational convenience is that the model cannot directly predict unique translations. Fortunately, our



experiments show that even very simple strategies are sufficient for finding a reasonable representative  $y$  for the most probable equivalence class.

Though the number of feature vectors for a single source sentence  $x$  may be large (more than  $10^{200}$ ), evaluating the partition function and the related feature expectations can be done efficiently by dynamic programming. The key insight that makes this possible is that feature dependencies are introduced only through biphrase overlaps. This means that all partial feature vectors that overlap in the same way with the feature under consideration can be dealt with simultaneously. The dynamic programming is still non-trivial due to the various ways in which the aligned biphases of different (source and target) lengths can overlap each other. However, our implementation shows that efficient implementation is possible.

### Learning the model parameters

**The objective** To estimate the model parameters  $w$  we use maximum a posteriori (MAP) estimation. To control overfitting, we regulate the parameters by a suitably scaled Gaussian prior. This prior models our prior belief that the observed feature occurrence counts randomly deviate from their “true” values, and that the absolute deviations can be expected to be the larger the larger the counts are. Following [8], we use an estimate for the variance of the total occurrence count of an aligned biphrase to scale the respective dimension of the Gaussian prior. For simplicity, we use the observed occurrence count as the variance estimate. This can be justified by assuming that the occurrence counts follow a Poisson distribution, though we have not checked how close to truth this assumption is.

Combining the model with the prior gives the negative log-posterior

$$\mathcal{L}(w) = \sum_{(x',a',y')} \frac{w_i^2}{2\sigma_i^2} - \sum_{(x,a,y) \in S} \log P(\phi(x, a, y)|x) + C,$$

where the sum over  $(x', a', y')$  is understood to go over all aligned biphrase features in the model. The prior variance  $\sigma_i^2$  for feature  $i$  is set to  $\alpha$  times the reciprocal of the estimated variance of the respective feature, where the regularization parameter  $\alpha > 0$  is the only free parameter of the system.

**Optimization** For simplicity, we use only first order gradient methods. As usual, the gradient of  $\mathcal{L}(w)$  w.r.t.  $w$  can be written as

$$\nabla \mathcal{L}(w) = \sum_i \frac{w_i}{\sigma_i^2} - \sum_{(x,a,y) \in S} [\tilde{\phi}(x, a, y) - \mathbb{E}_w[\tilde{\phi}(x, a', y')|x]],$$



where  $\mathbb{E}_w[\tilde{\phi}(x, a', y')|x]$  denotes the expectation of the aligned biphrase occurrence count features given the model with parameters  $w$ . Computing the feature expectations can be done by combining the results of a left-to-right and right-to-left dynamic programming sweep over the source sentence. Due to the many possible kinds of feature overlaps and the many-to-many nature of the phrase-level features, the dynamic programming is technically a bit involved. The technical details are omitted here for brevity, but can be extracted from the source code of our implementation if desired.

Inspired by the work in [9], we decided to use stochastic gradient ascent instead of ordinary gradient ascent. To do this, at each step  $t$ , we sample with replacement  $b$  examples from  $S$  to obtain the batch  $S_t$ . We start from  $w_0 = 0$ , and use the update rule

$$w_{t+1} = w_t - \eta \nabla \mathcal{L}_t(w_t),$$

where  $\eta > 0$  is a fixed learning rate, and  $\nabla \mathcal{L}_t(w)$  is the gradient of the negative log-posterior

$$\mathcal{L}_t(w) = \frac{b}{|S|} \sum_i \frac{w_i^2}{2\sigma_i^2} - \sum_{(x,a,y) \in S_t} \log P(\phi(x, a, y)|x)$$

with respect to batch  $S_t$ . By processing the data in small batches, we only need to store the part of  $w$  in memory that corresponds to features that are active in the batch – the effects of regularization to other parts of  $w$  can be dealt with lazily when the features are next time loaded into memory. Thus, we can handle weight vectors that would otherwise be too large to store in memory.

Another advantage of the stochastic gradient method is that it can be parallelized to speed up the training. We use a Berkeley DB database for storing the features and feature weights and for concurrency control, which means that all processes that access the database must be able to `mmap` to the filesystem where the database is stored. Otherwise, the amount of parallelism in training is unlimited.

### Prediction

**Prediction with the translation model alone** Prediction is done in two phases. First, we find by dynamic programming the highest probability feature vector  $\hat{\phi}(x)$  defined by

$$\hat{\phi}(x) = \arg \max_{\phi(x,a,y) : x \text{ covered}} P(\phi(x, a, y)|x).$$

Note that we restrict the search to feature vectors that cover the whole of  $x$ , i.e., each word in  $x$  is covered by at least one aligned biphrase  $(x', a', y')$  whose



indicator in  $\phi(x, a, y)$  is on. The motivation for this is to force the system to translate all words in the source sentence even if its model correctly predicts that the “correct” translation for the words in their context has most likely not been seen yet. To make it possible to cover also previously unseen source words, for each source word that otherwise could not be covered, we implicitly add an extra aligned biphrase with zero weight to the phrase table that copies the source word to the target as is. This is again a standard trick in phrase-based SMT.

The second step in the prediction is to construct a translation  $y$  from the highest probability feature vector  $\hat{\phi}(x)$ . We do this by simply concatenating the target sides of the phrases with non-zero indicators in  $\hat{\phi}(x)$  in the order induced by their positions in the source sentence. Thus, at this point, this prediction strategy has no phrase-level reordering, and no model for explicitly encouraging fluency of target language output. One option would be to use  $n$ -best rescoring, but we haven’t done that yet though our implementation already supports generating  $n$ -best lists.

**Predicting with an integrated LM** The prediction strategy outlined in the previous section is simple and conceptually clean. The proposed translation model is expressive enough to produce quite fluent output, but suffers from data sparsity that limits the amount of fluency enhancing phrase overlaps, and also the model’s limitations in modelling target word ordering. To address, our second less elegant prediction strategy integrates the proposed translation model with a language model, and predicts translations using a beam search based decoder with support for phrase overlaps. The beam decoder also supports limited phrase-level distortion, that is, swapping the order of the translations of two consecutive non-overlapping phrases.

The beam decoder is driven by a weighted combination of the following features.

1. Translation model:  $\log P(\phi(x, a, y)|x)$
2. Language model: log-probability given by a smoothed  $n$ -gram language model to  $y$
3. Translation length: number of words in translation
4. Distortion: number of source words in phrases whose translations have been swapped

We have not implemented any automated methods for tuning the weights for these features yet, but have resorted to manual tuning.



The implementation of the beam decoder differs from that in systems like **Pharaoh** and **Moses** in that we structure the beam search around state transitions, not around states. This means that we first choose a phrase (i.e., state transition), and apply it to all hypotheses (states) to which it can be applied simultaneously. It seems that when phrase overlaps are allowed, this strategy is much more efficient than a search procedure that expands one hypothesis at a time by applying all applicable phrases to it. Furthermore, our strategy eliminates the need for future scores, and is implementationally very similar to the dynamic programming procedures used in training the model and in the first prediction strategy.

### 2.3.3 Experiments

In this section we report initial experimental results on **Sinuhe** (with and without a language model) on a set of translation experiments that use the Europarl corpus as training and testing data. The experiments reported here are on translating all the 10 non-English Europarl languages into English. In these experiments, we used **Moses** and **Pharaoh** as comparison points. For further experimental results on the official SMART evaluation tasks, we refer the reader to the experimental results section 3.

#### Translation into English

In these experiments, we used version 3 of the Europarl parallel corpus [10]. The corpus contains parallel text automatically extracted from the proceedings of the European Parliament in 11 European languages, and contains tens of millions of words per language. We lower-cased and tokenized all the data, and used it to train **Sinuhe** and **Moses** for translating the 10 non-English languages in the corpus into English. We reserved 10000 randomly chosen sentence pairs per language pair for development and testing purposes. The rest of the data was used to train the translation and language models of **Moses** as explained in [11]. The log-linear parameters of **Moses** were tuned on 2000 sentences of development data. The feature weights for the **Sinuhe** beam decoder were chosen manually to produce reasonable looking translation quality on the Spanish to English task. We haven't had time for systematic experiments on tuning the weights, so the weights we chose may or may not be close to optimal. The weight for the translation model is 1.0, the weight for the language model is 0.2, the weight for the translation length 0.7, and the weight for the distortion feature is 0.3. We used these same weights for all the translation tasks without further tuning.

We used the phrases extracted by **Moses** as the aligned biphrase features for **Sinuhe**. The parameters for **Sinuhe**'s model were trained on the same data from



which the phrases were extracted by the stochastic gradient method explained in section 2.3.2. In training the parameters, we chose  $b = 10$ ,  $\eta = 0.1$ , and  $\alpha = |S|/10^8$ . We did not run any experiments with other values of  $b$  or  $\eta$ , but tested that the value chosen for  $\alpha$  is reasonable though not optimal. We run the stochastic gradient for about  $3 \times 10^6$  iterations, at which point the value of the optimized function seemed to have stabilized. Depending on the language pair, the training took from one to three weeks with 5 parallel training processes on a standard quad core machine. Unfortunately, we do not have more exact timing results as the machines were used by other people also and we had to interrupt the training from time to time (to make backups etc). As a language model for **Sinuhe**<sup>LM</sup> – the version of **Sinuhe** that uses a language model – we used the SRILM toolkit [12] to train a 5-gram language model with Kneser-Ney smoothing (`-kndiscount`), interpolation (`-interpolate`), and open vocabulary (`-unk`).

We used **Pharaoh** and **Moses** as comparison points in our experiments. The **Pharaoh** SMT system is a commonly used baseline system (see, e.g., [10]) from which the state-of-the-art system **Moses** has evolved. The translation quality of **Moses** is very competitive, and at par with the winners of the last WMT translation challenges [11]. The  $P(y|x)_{\text{Moses}}$  system is a version of **Moses** in which all features but the forward translation model has zero weight. Thus, by comparing the translations produced by **Sinuhe** and  $P(y|x)_{\text{Moses}}$  we can implicitly compare the underlying translation models. A more direct comparison of the probabilities predicted by the translation models was not possible as there is no feasible way to normalize the “probabilities” produced by **Moses**.

We measure the performance of the SMT systems with the BLEU score [13] that measures the similarity of  $n$ -grams consisting of consecutive words in the systems’ outputs to those in reference translations produced by human translators. The BLEU scores for **Pharaoh** are the ones reported in [10], and the BLEU scores for **Sinuhe**, **Moses**, and  $P(y|x)_{\text{Moses}}$  are computed by us with the same training and test data splits. It should be noted that using different data splits for **Pharaoh** and the other systems may affect the comparison.

The results in Table 2.1 show that **Sinuhe** and **Sinuhe**<sup>LM</sup> produce translations with quality similar to those produced by **Pharaoh**, but worse than those produced by **Moses**. The translation quality of **Sinuhe** is superior to that of  $P(y|x)_{\text{Moses}}$ , and **Sinuhe**<sup>LM</sup> is clearly better than the plain version of **Sinuhe**. These results serve at least as a sanity check – **Sinuhe** and **Sinuhe**<sup>LM</sup> produce reasonable translations as their translation quality (wrt BLEU score) is not worse than that of the baseline **Pharaoh**, and only somewhat worse than the state-of-the-art **Moses**. Also, the fact that **Sinuhe** produces far better translations than  $P(y|x)_{\text{Moses}}$  is evidence that the proposed translation model is indeed superior to the model used in **Moses**. As far as translation speed is concerned,



|                            | da    | de    | el    | es    | fi    | fr    | it    | nl    | pt    | sv    |
|----------------------------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| <b>Pharaoh</b>             | 28.5  | 25.3  | 27.2  | 30.5  | 21.8  | 30.0  | 27.8  | 23.0  | 30.1  | 30.2  |
| $P(y x)_{\text{Moses}}$    | 23.99 | 20.24 | 21.88 | 27.40 | 16.82 | 25.70 | 23.05 | 19.98 | 25.57 | 26.45 |
| <b>Simuhe</b>              | 27.63 | 22.37 | 25.07 | 31.38 | 18.26 | 30.31 | 26.85 | 22.54 | 29.78 | 29.65 |
| <b>Simuhe<sup>LM</sup></b> | 29.11 | 23.91 | 26.41 | 33.15 | 20.56 | 31.39 | 28.50 | 23.78 | 31.52 | 31.18 |
| <b>Moses</b>               | 32.10 | 27.84 | 30.07 | 36.26 | 24.79 | 34.42 | 31.89 | ?.?   | 34.82 | 34.46 |

Table 2.1: The translation quality of the SMT systems w.r.t. BLEU score on the task of translating 10 European languages to English. **Moses** repeatedly crashed for the nl-en language pair, hence the missing entry in the table.



**Sinuhe** is clearly the best of the systems, but also **Sinuhe**<sup>LM</sup> is significantly faster than **Moses**. For example, translating the 2000 test sentences from Swedish to English with **Sinuhe** takes 31 seconds, whereas the same task takes 1 minute and 20 seconds with **Sinuhe**<sup>LM</sup> (including the time for loading the language model). The translation time taken by **Moses** depends on whether the **Moses** parameters are optimized for translation speed or translation quality, but even if they are optimized for speed, **Sinuhe** and **Sinuhe**<sup>LM</sup> are clearly faster than **Moses**. However, we did not do exact timing experiments on **Moses** yet, so we cannot make any final quantitative claims on the relative translation speeds of the different systems.

We do not have solid explanations why **Moses** produces better translations (wrt BLEU) than **Sinuhe**<sup>LM</sup>. At least part of the reason is likely to be overfitting, in two forms. First, **Moses** and hence also the phrase extraction process used by it, has been carefully designed to work well on the Europarl corpus. It is not at all clear that the phrase extraction heuristics that serve **Moses** best are optimal for extracting features **Sinuhe**. Second, **Moses** gains advantage from its ability to exploit long and rare phrase translations that are specific to the Europarl corpus. The leave-one-out pruning and regularization employed in **Sinuhe** and **Sinuhe**<sup>LM</sup> may (and hopefully do) make it less likely to rely much on such long and rare phrases. Thus, one could hope that **Sinuhe** and **Sinuhe**<sup>LM</sup> would fare better in the comparisons on test data from other domains, but this remains to be tested. Other possible explanations include that the parameters for **Sinuhe**<sup>LM</sup> were not optimized whereas those of **Moses** were by the automatic MERT training procedure, and that **Sinuhe**<sup>LM</sup> has no reverse translation model. And of course the fact that **Sinuhe**<sup>LM</sup> was written 3 days before running the experiments reported here, whereas **Moses** has been tuned for years to produce good BLEU results on Europarl.

## 2.4 MMR based approaches

This section summarises the amendments that have been made on the prototype of the machine learning based translator since it was introduced in deliverable D2.1 and further extended in D4.2. There were two main issues with the previous system, these being:

1. The earlier version generated too large a phrase dictionary, which contained too many false positive phrase translations. The size of this phrase dictionary was one of main obstacles in scaling up the training system, and the large number of false translations caused a bottleneck in the decoder.
2. The previous system was restricted to derive the relationship between



the target and source phrases by considering the relations between the individual target and source words only.

## 2.5 Model behind the learning approach

In this section and in the sequel when we use the term “word” it is assumed that for each of languages there is a set of labels identifying the word types, the meaning of these words. The corresponding phrases and the sentences of a language are ordered subsets of the label set belonging to this language.

Our aim is to translate phrases of a source language into phrases belonging to a target language. To this end we are given a set of pairs of sentences supposed to be translations of each other. A way of stating the translation task of the phrases is the following:

- Choose a pair of sentences.
- Assign labels to the words of the source sentence as indicators of a phrase to be translated, say the label is equal to 1 if the word is part of the phrase and 0 otherwise.
- Find a similar labelling of the words of the target sentence, namely, words in the translation of the source phrase receive label 1 and the remaining others are set to 0.

In this way we can formulate a binary classification problem on the words of the target sentence. To realise this learning task we need to characterise the words by embedding them into a feature space which expresses the relationship between them, e.g. how frequently they co-occur and how close their relative positions are (either within a sentence for same-language statistics, or between aligned sentences for translation statistics). Thus, one task that we need to solve is to build a relevant feature space for both the source and the target languages.

Unfortunately to solve a binary classification problem for all reasonable phrases in all sentence pairs of the training set can be computationally infeasible. To overcome this, the classification of all the phrases considered in the source is solved as one structural learning problem exploiting the capability of Maximum Margin Regression (MMR). In this setting we receive a score for each target word showing how strongly or weakly they relate to the source phrases and source words. From these scores we can obtain a sentence-wide alignment and also an alignment between source and target phrases.

To recover the target phrases we exploit a model to approximate the semantic relations. In this model we assume that the words occurring in a sentence are of the range of a set of functions defined on the semantic contents. For each



language, there is a set of functions of this kind, and the semantic contents are assumed to be a set of subsets of all possible semantic identities. This type of set system is known as a ring of sets and is closed under intersection, union and set difference. A related approach is sometimes used in mathematical linguistics by applying Heyting algebra with similar but extended properties, see in [14].

So, if there are two sentences which are translation of each other then we can think of these as values of some corresponding functions which project the common (semantic) contents into different language environments. Formally we can describe these relations by defining the semantic contents by  $(\mathcal{U}, \mathbb{U}_{\mathcal{U}})$ , where  $\mathcal{U}$  is the semantic universe and  $\mathbb{U}_{\mathcal{U}} \subset 2^{\mathcal{U}}$  denotes the set system of the semantic objects. Furthermore we have

$$\begin{aligned} \forall \mathcal{S}_u, \mathcal{S}_v \in \mathbb{U}_{\mathcal{U}} \Rightarrow & \mathcal{S}_u \cup \mathcal{S}_v \in \mathbb{U}_{\mathcal{U}}, \\ & \mathcal{S}_u \cap \mathcal{S}_v \in \mathbb{U}_{\mathcal{U}}, \\ & \mathcal{S}_u \setminus \mathcal{S}_v \in \mathbb{U}_{\mathcal{U}}, \\ & \mathcal{S}_v \setminus \mathcal{S}_u \in \mathbb{U}_{\mathcal{U}}. \end{aligned} \tag{2.1}$$

A language  $\mathcal{L}$  is given by a set of functions  $\mathcal{F}_{\mathcal{L}}$  mapping the semantic objects into an ordered subset of words of this language. We might think of this system which follows the theory of the manifolds, namely a space of complex irregular objects described by simpler spaces. Here the semantic space plays the role of the manifold with a tricky structure and the observable word sequences are the simple spaces to somehow express this structure. There could possibly be a distinct function in  $\mathcal{F}_{\mathcal{L}}$  for each of the semantic objects which maps to a concrete target language. Our assumption is that these functions transfer the structure of the semantic space into the the set words of a given language. So we have a set system with similar properties in case of a language  $(\mathcal{W}_{\mathcal{L}}, \mathbb{W}_{\mathcal{L}})$ , where  $\mathcal{W}_{\mathcal{L}}$  is the set of all words of language  $\mathcal{L}$  and  $\mathbb{W}_{\mathcal{L}}$  denotes a set of finite subsets of  $\mathcal{W}_{\mathcal{L}}$  including the empty subset as well. Thus, we have

$$f_{\mathcal{L}} : \mathbb{U}_{\mathcal{U}} \Rightarrow \mathbb{W}_{\mathcal{L}}, f_{\mathcal{L}} \in \mathcal{F}_{\mathcal{L}} \tag{2.2}$$

A sentence in this setting is simply a long phrase which happens to have a specific content. In this way two sentences are translation of each other if they are derived from the same (semantic) content, and obviously we must assume that the translation is perfectly or at least approximately correct. Thus, the translation of a phrase  $\mathcal{P}$  of the source language is a phrase  $\mathcal{R}$  in the target language which is expressible by

$$\mathcal{R} = f_{\mathcal{L}_{target}} f_{\mathcal{L}_{source}}^{-1}(\mathcal{P}), \tag{2.3}$$

where  $f^{-1}$  denotes the inverse function of  $f$ . Hence the translation projects back the source into the semantic space and then this inverse image is mapped into the target language.

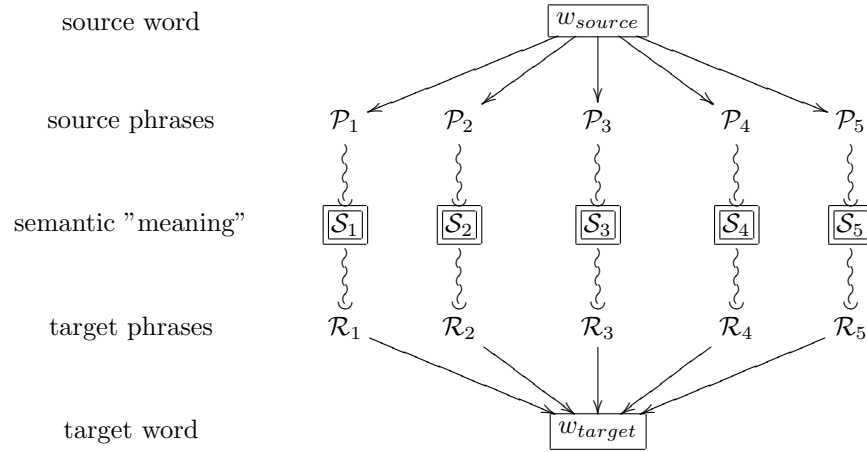


Figure 2.1: The translation flow; the relationship between the source and the target words are expressed via the relationship of the corresponding phrases.

A consequence of this framework is that if two source phrases  $\mathcal{P}_1$  and  $\mathcal{P}_2$  are translated into phrases  $\mathcal{R}_1$  and  $\mathcal{R}_2$  of the target language, then we have the relations

$$\begin{aligned}
 \mathcal{P}_1 &\Leftrightarrow \mathcal{R}_1, \\
 \mathcal{P}_2 &\Leftrightarrow \mathcal{R}_2, \\
 \mathcal{P}_1 \cap \mathcal{P}_2 &\Leftrightarrow \mathcal{R}_1 \cap \mathcal{R}_2, \\
 \mathcal{P}_1 \cup \mathcal{P}_2 &\Leftrightarrow \mathcal{R}_1 \cup \mathcal{R}_2, \\
 \mathcal{P}_1 \setminus \mathcal{P}_2 &\Leftrightarrow \mathcal{R}_1 \setminus \mathcal{R}_2, \\
 \mathcal{P}_2 \setminus \mathcal{P}_1 &\Leftrightarrow \mathcal{R}_2 \setminus \mathcal{R}_1.
 \end{aligned} \tag{2.4}$$

Where one, or even both, of intersections and differences can be empty. The aim of a correct translation is to fulfil approximately the rules enumerated in (2.4). Thus the translation procedure projects the source words into the semantic space via the source phrases and the corresponding target phrases show the target words those which have the highest similarity to these set of phrases.

The flow of the translation is represented on Figure 2.1, and this is the general setting and picture that is in mind when we consider the similarity measures and features in the following sections.



## 2.6 Notations

The following symbols are employed in the description of the translation algorithm.

| Symbol                                        | Meaning                                                                                                |
|-----------------------------------------------|--------------------------------------------------------------------------------------------------------|
| $\mathcal{S}_S$                               | set of source sentences                                                                                |
| $\mathcal{S}_T$                               | set of target sentences                                                                                |
| $\mathcal{S}_s$                               | a source sentence with index $s$ , it is an ordered set of source words                                |
| $\mathcal{S}_t$                               | a target sentence with index $s$ , it is an ordered set of target words                                |
| $w_{s t,i}$                                   | a word of source sentence $s$ or a target sentence $t$ in position $i$                                 |
| $\mathcal{W}_S$                               | entire set of all source words                                                                         |
| $\mathcal{W}_t$                               | entire set of all target words                                                                         |
| $\mathcal{S}[w]$                              | set of sentences containing word $w$ , whose subscript shows it is a set of source or target sentences |
| $\mathcal{I}[w, \mathcal{S}_{s t}]$           | set of positions of word $w$ in sentence $\mathcal{S}$ in source $s$ or in target $t$                  |
| $\mathcal{I}[w]$                              | the set of $\mathcal{I}[w, \mathcal{S}_{s t}]$ for all sentences containing word $w$                   |
| $\mathcal{S}_1 \Leftrightarrow \mathcal{S}_2$ | the sentences are translations of each other                                                           |
| $\langle \mathbf{u}, \mathbf{v} \rangle$      | the inner product of the vectors $\mathbf{u}$ and $\mathbf{v}$                                         |
| $\mathbf{A}'$                                 | the transpose of the matrix $\mathbf{A}$                                                               |
| $\ \mathbf{W}\ _{Frobenius}$                  | the Frobenius norm of the matrix $\mathbf{W}$ and it is equal to $\text{trace}(\mathbf{W}'\mathbf{W})$ |
| $\mathbf{A} \bullet \mathbf{B}$               | is the pointwise product of the matrices $\mathbf{A}$ and $\mathbf{B}$                                 |

## 2.7 Learning elements

### 2.7.1 Similarity measure of words

#### Word descriptor

The descriptor of a word is a set of the indices of the sentences containing this word and in each of the sentences a set of the positions of this word. The positions are normalised by sentence length to allow them to be comparable across sentences.

$$\mathcal{D}[w] = \mathcal{S}[w] \text{ and } \mathcal{I}[w], \text{ where } \mathcal{I}[w] = \{\mathcal{I}[w, S] | S \in \mathcal{S}[w]\} \quad (2.5)$$



### Similarity measure

When the next similarity measure is defined we assume that words in a sentence expressing semantically related contents generally fall close to each other in their positions within the sentence independently of their order.

First we define the similarity between two words of the same language, i.e. when they are chosen only from the source sentences or from the target sentences. The similarity measure of two words  $w_1$  and  $w_2$ , within one sentence  $\mathcal{S}$  with length  $n_{\mathcal{S}}$  can be computed in several different ways. Two possible methods corresponding to a Linear and a Gaussian case are shown below:

**Linear:**

$$s_{\mathcal{S}}(w_1, w_2) = \begin{cases} 1 - \min_{i_1 \in \mathcal{I}[w_1, \mathcal{S}], i_2 \in \mathcal{I}[w_2, \mathcal{S}]} \frac{|i_1 - i_2|}{n_{\mathcal{S}}} \\ \text{if } \mathcal{I}[w_1, \mathcal{S}] \neq \emptyset \text{ and } \mathcal{I}[w_2, \mathcal{S}] \neq \emptyset, \\ 0 \quad \text{otherwise.} \end{cases} \quad (2.6)$$

**Gaussian:**

$$s_{\mathcal{S}}(w_1, w_2) = \begin{cases} \exp \left( - \frac{\left( \min_{i_1 \in \mathcal{I}[w_1, \mathcal{S}], i_2 \in \mathcal{I}[w_2, \mathcal{S}]} \frac{|i_1 - i_2|}{n_{\mathcal{S}}} \right)^2}{\sigma} \right) \\ \text{if } \mathcal{I}[w_1, \mathcal{S}] \neq \emptyset \text{ and } \mathcal{I}[w_2, \mathcal{S}] \neq \emptyset, \\ 0 \quad \text{otherwise,} \end{cases} \quad (2.7)$$

where  $\sigma$  is the variance of the Gaussian.

Now consider two words  $w_1$  (source) and  $w_2$  (target) from different languages, where the sentences  $\mathcal{S}_1$  and  $\mathcal{S}_2$  are indicated as translations of each other (i.e.  $\mathcal{S}_1 \leftrightarrow \mathcal{S}_2$ ). Then the similarity between them can be obtained as:

**Linear:**

$$s_{\mathcal{S}_1, \mathcal{S}_2}(w_1, w_2) = \begin{cases} 1 - \min_{i_1 \in \mathcal{I}[w_1, \mathcal{S}_1], i_2 \in \mathcal{I}[w_2, \mathcal{S}_2]} \left| \frac{i_1}{n_{\mathcal{S}_1}} - \frac{i_2}{n_{\mathcal{S}_2}} \right| \\ \text{if } \mathcal{I}[w_1, \mathcal{S}_1] \neq \emptyset \text{ and } \mathcal{I}[w_2, \mathcal{S}_2] \neq \emptyset, \\ 0 \quad \text{otherwise} \end{cases} \quad (2.8)$$

**Gaussian:**

$$s_{\mathcal{S}_1, \mathcal{S}_2}(w_1, w_2) = \begin{cases} \exp \left( - \frac{\left( \min_{i_1 \in \mathcal{I}[w_1, \mathcal{S}_1], i_2 \in \mathcal{I}[w_2, \mathcal{S}_2]} \left| \frac{i_1}{n_{\mathcal{S}_1}} - \frac{i_2}{n_{\mathcal{S}_2}} \right| \right)^2}{\sigma} \right) \\ \text{if } \mathcal{I}[w_1, \mathcal{S}_1] \neq \emptyset \text{ and } \mathcal{I}[w_2, \mathcal{S}_2] \neq \emptyset, \\ 0 \quad \text{otherwise,} \end{cases} \quad (2.9)$$

where  $\sigma$  is again the variance of the Gaussian.



The similarity measures can then be computed with respect to the entire databases, in the case when both words from the same languages

$$s(w_1, w_2) = \frac{\sum_{\mathcal{S} \in \mathcal{S}[w_1] \cap \mathcal{S}[w_2]} s_{\mathcal{S}}(w_1, w_2)}{|\mathcal{S}[w_1] \cup \mathcal{S}[w_2]|}, \quad (2.10)$$

and in the case when the languages are different

$$s(w_1, w_2) = \frac{1}{|\mathcal{S}_s[w_1] \cup \mathcal{S}_s[w_2]|} \sum_{\substack{\mathcal{S}_1 \in \mathcal{S}_s[w_1] \cap \mathcal{S}_s[w_2] \\ \mathcal{S}_2 \in \mathcal{S}_t[w_1] \cap \mathcal{S}_t[w_2] \\ \mathcal{S}_1 \leftrightarrow \mathcal{S}_2}} s_{\mathcal{S}_1, \mathcal{S}_2}(w_1, w_2), \quad (2.11)$$

where we must also bear in mind that the sentences correspond to each other, thus

$$|\mathcal{S}_s[w_1] \cup \mathcal{S}_s[w_2]| = |\mathcal{S}_t[w_1] \cup \mathcal{S}_t[w_2]|. \quad (2.12)$$

## 2.7.2 Word features

For each word occurring in a sentence, features are generated using the similarity measures above using a set of words from a pair of sentences (e.g. by defining a 'window of interest' in each sentence).

The feature vector of a source word  $w_s$  relative to a sentence pair  $\mathcal{S}_s, \mathcal{S}_t$ ,  $\mathcal{S}_s \leftrightarrow \mathcal{S}_t$  is obtained by

$$\begin{aligned} \phi_{\mathcal{S}_s, \mathcal{S}_t}(w_s) &= \underbrace{(s(w_s, w_{s_1}), \dots, s(w_s, w_{s_{n_{\mathcal{S}_s}}}))}_{(w_{s_1}, \dots, w_{s_{n_{\mathcal{S}_s}}}) = \mathcal{S}_s}, \underbrace{(s(w_s, w_{t_1}), \dots, s(w_s, w_{t_{n_{\mathcal{S}_t}}}))}_{(w_{t_1}, \dots, w_{t_{n_{\mathcal{S}_t}}}) = \mathcal{S}_t}, \\ w_s &\in \mathcal{S}_s, \end{aligned} \quad (2.13)$$

which is the concatenation of the similarities relative to the other words taken out of the source and the target sentences. In case of a word  $w_t$  of the target we have

$$\begin{aligned} \phi_{\mathcal{S}_s, \mathcal{S}_t}(w_t) &= \underbrace{(s(w_t, w_{s_1}), \dots, s(w_t, w_{s_{n_{\mathcal{S}_s}}}))}_{(w_{s_1}, \dots, w_{s_{n_{\mathcal{S}_s}}}) = \mathcal{S}_s}, \underbrace{(s(w_t, w_{t_1}), \dots, s(w_t, w_{t_{n_{\mathcal{S}_t}}}))}_{(w_{t_1}, \dots, w_{t_{n_{\mathcal{S}_t}}}) = \mathcal{S}_t}, \\ w_t &\in \mathcal{S}_t. \end{aligned} \quad (2.14)$$

We can summarise the structure of the feature matrix containing the feature vectors in its rows in the following way

$$\phi_{\mathcal{S}_s, \mathcal{S}_t}(\mathcal{S}_s, \mathcal{S}_t) = \begin{bmatrix} SS & ST \\ TS & TT \end{bmatrix}, \quad (2.15)$$



where the sub-matrix  $SS$  consists of the similarities between the source words,  $TT$  contains the similarities between the target words and  $ST$  and its transpose  $ST' = TS$  comprise the similarities between the source and the target words.

One can interpret the sub-matrices  $SS$  and  $TT$  as language models to the source and the target language respectively. Following this logic the roles of  $ST$  and  $TS$  reflect the translation model.

### 2.7.3 Output labels of the words

The labels of the source words indicate the existence of words and phrases. If we assume a source sentence  $\mathcal{S}_s$  is covered by a set of phrases  $\mathcal{P}_{\mathcal{S}_s} = \mathcal{P}_1, \dots, \mathcal{P}_{N_{\mathcal{S}_s}}$ , which are ordered subsets of the sentence  $\mathcal{S}_s$ , then the label vector of a source word  $w_s$  is equal to

$$\begin{aligned} \psi_{\mathcal{P}_{\mathcal{S}_s}}(w_s) &= ([w_s \in \mathcal{P}_1], \dots, [w_s \in \mathcal{P}_{N_{\mathcal{S}_s}}]) \\ [w_s \in \mathcal{P}] &= \begin{cases} 1 & \text{if } w_s \in \mathcal{P} \\ 0 & \text{if } w_s \notin \mathcal{P}. \end{cases} \end{aligned} \quad (2.16)$$

We can also derive label vectors to the target words assuming similar phrase covering of the target sentence  $\mathcal{R}_{\mathcal{S}_t} = \mathcal{R}_1, \dots, \mathcal{R}_{N_{\mathcal{S}_t}}$ :

$$\begin{aligned} \psi_{\mathcal{R}_{\mathcal{S}_t}}(w_t) &= ([w_t \in \mathcal{R}_1], \dots, [w_t \in \mathcal{R}_{N_{\mathcal{S}_t}}]) \\ [w_t \in \mathcal{R}] &= \begin{cases} 1 & \text{if } w_t \in \mathcal{R} \\ 0 & \text{if } w_t \notin \mathcal{R}. \end{cases} \end{aligned} \quad (2.17)$$

We may apply a weighted schema down-scaling the effect of the longer phrases, e.g. in case of exponentially decreasing weight system with base value 2, then the label vector takes the form

$$\begin{aligned} \psi_{\mathcal{P}_{\mathcal{S}_s}}(w_s) &= ([w_s \in \mathcal{P}_1], \dots, [w_s \in \mathcal{P}_{N_{\mathcal{S}_s}}]) \\ [w_s \in \mathcal{P}] &= \begin{cases} 2^{-|\mathcal{P}|} & \text{if } w_s \in \mathcal{P} \\ 0 & \text{if } w_s \notin \mathcal{P} \end{cases}, \end{aligned} \quad (2.18)$$

where  $|\mathcal{P}|$  denotes the length of the phrase measured in the number of words occurring in it.

**Example 1.** *If a source sentence  $\mathcal{S}_s$  is covered by ngrams, subsets of consecutive words  $w_1, \dots, w_{n_{\mathcal{S}_s}}$  of the sentence, then the matrix of the output labels when the sentence length is equal to  $n_{\mathcal{S}_s} = 4$  takes the following form:*



| Words | Phrases |   |   |   |       |   |   |       |   |       |
|-------|---------|---|---|---|-------|---|---|-------|---|-------|
|       | 1gram   |   |   |   | 2gram |   |   | 3gram |   | 4gram |
|       | 1       | 2 | 3 | 4 | 5     | 6 | 7 | 8     | 9 | 10    |
| 1     | 1       | 0 | 0 | 0 | 1     | 0 | 0 | 1     | 0 | 1     |
| 2     | 0       | 1 | 0 | 0 | 1     | 1 | 0 | 1     | 1 | 1     |
| 3     | 0       | 0 | 1 | 0 | 0     | 1 | 1 | 1     | 1 | 1     |
| 4     | 0       | 0 | 0 | 1 | 0     | 0 | 1 | 0     | 1 | 1     |

(2.19)

## 2.8 Learning problem

In the learning problem we are going to predict the labels of a word with a known feature vector. The labels can describe how the words relate to the others in semantical sense, i.e. if a word is in the intersection of two phrases then it probably conveys information of the intersection of the meaning of the phrases as well.

With the input features and output labels we can setup the learning problem in the following form

$$\begin{aligned}
 \min \quad & \frac{1}{2} \|\mathbf{W}\|_{Frobenius}^2 + C \sum_{s=1}^{n_{\mathcal{S}_s}} \xi_s \\
 \text{w.r.t.} \quad & \mathbf{W} \text{ linear operator, } \boldsymbol{\xi} \text{ loss,} \\
 \text{s.t.} \quad & \langle \boldsymbol{\psi}_{\mathcal{P}_{\mathcal{S}_s}}(w_s), \mathbf{W} \boldsymbol{\phi}_{\mathcal{S}_s, \mathcal{S}_t}(w_s) \rangle \geq 1 - \xi_s, \quad w_s \in \mathcal{S}_s, \\
 & \boldsymbol{\xi} \geq \mathbf{0},
 \end{aligned}
 \tag{2.20}$$

where  $C > 0$  is the penalty constant to the loss term in the objective function. Here we consider the source words only when the  $\mathbf{W}$  is computed assuming that the source words constitute the training set. Within these relations the target words form the test set and the labels of the target words, the target phrases, relating to the source phrases are predicted.

The optimum solution to the learning problem can be expressed in a closed form:

$$\mathbf{W} = \sum_{w_s \in \mathcal{S}_s} \alpha_{w_s} \boldsymbol{\psi}_{\mathcal{P}_{\mathcal{S}_s}}(w_s) \boldsymbol{\phi}_{\mathcal{S}_s, \mathcal{S}_t}(w_s)', \tag{2.21}$$

where the  $\alpha_{w_s}, w_s \in \mathcal{S}_s$  are the Lagrangians introduced to each of the constraint in (2.20).

Predicting the labels for a target word  $w_t$  we have

$$\mathbf{W} \boldsymbol{\phi}_{\mathcal{S}_s, \mathcal{S}_t}(w_t). \tag{2.22}$$

In this way we receive a real vector and not binary indicators of the phrases (which is what we are after). The real values partially express the uncertainty of the labels. To deal with this uncertainty we are predicting not directly the



labels of a target word but the relations between the source words and the target words. We can measure the strength of this relations between  $w_s$  and  $w_t$  by

$$\begin{aligned}\mathcal{R}(w_s, w_t) &= \langle \psi_{\mathcal{P}_{\mathcal{S}_s}}(w_s), \mathbf{W} \phi_{\mathcal{S}_s, \mathcal{S}_t}(w_t) \rangle \\ &= \sum_{w_r \in \mathcal{S}_s} \alpha_{w_r} \kappa_{\psi}(w_s, w_r) \kappa_{\phi}(w_r, w_t) \\ \text{where} & \\ \kappa_{\psi}(w, w_s) &= \langle \psi_{\mathcal{P}_{\mathcal{S}_s}}(w), \psi_{\mathcal{P}_{\mathcal{S}_s}}(w_s) \rangle, \\ \kappa_{\phi}(w_s, w_t) &= \langle \phi_{\mathcal{S}_s, \mathcal{S}_t}(w_s), \phi_{\mathcal{S}_s, \mathcal{S}_t}(w_t) \rangle.\end{aligned}\tag{2.23}$$

This measure says that if the margin is large then the similarity in semantical sense between these words is high.

The strongest relation gives us the alignment most likely to a target word  $w_t$ , and to a source word  $w_s$

$$\begin{aligned}\hat{w}_s(w_t) &= \arg \max_{w \in \mathcal{S}_s} \mathcal{R}(w, w_t), \\ \hat{w}_s(w_s) &= \arg \max_{w \in \mathcal{S}_t} \mathcal{R}(w, w_s).\end{aligned}\tag{2.24}$$

This problem has a simple solution since the values of the functions  $\kappa_{\psi}$  and  $\kappa_{\phi}$  can be computed in advance for all pairs of the source and the target words. Since the optimisation problem (2.24) is entirely symmetric with respect to handling of the source and the target words, the direction of 'translation' can be reversed simply by taking the transpose of  $\mathbf{W}$ . In this way we can built up a directed bipartite graph between the source and the target words where the arcs connecting the nodes, the source or target words, are drawn into the graph if a source word highly relates to a target and vice versa. To illustrate this process we now demonstrate the procedure on a concrete sentence.

## 2.9 Example

The words of the source sentence in French and the target sentence in English are shown in Table 2.2:

In the next tables the indices of the words are used to identify the corresponding words. The values of the components in the feature vectors demonstrated on Figure 2.2.

Applying (2.23) the margin based similarity between the source and the target words can be computed, and the relations are shown in figure 2.3.

The optimum alignments are computed by the MMR predictor by the following procedure:

- The MMR learning tool is trained twice, first the feature vectors of the source words are considered as training set and the feature vectors of the target words play the role of the test to be predicted, in the second round the roles are swapped between the source and the target.



## D2.2: Application of Markov approaches to SMT

---

| source words | word index | target words | word index |
|--------------|------------|--------------|------------|
| centreware   | 0          | centreware   | 0          |
| regroupe     | 1          | is           | 1          |
| plusieurs    | 2          | a            | 2          |
| applications | 3          | suite        | 3          |
| et           | 4          | of           | 4          |
| pilotes      | 5          | applications | 5          |
| d            | 6          | and          | 6          |
| imprimante   | 7          | printer      | 7          |
| et           | 8          | fax          | 8          |
| de           | 9          | drivers      | 9          |
| télécopie    | 10         | that         | 10         |
| compatibles  | 11         | support      | 11         |
| avec         | 12         | the          | 12         |
| la           | 13         | xerox        | 13         |
| ligne        | 14         | document     | 14         |
| de           | 15         | centre       | 15         |
| produits     | 16         | line         | 16         |
| xerox        | 17         | of           | 17         |
| document     | 18         | products     | 18         |
| centre       | 19         |              |            |

Table 2.2: The words and their indices of the French source sentence, and the words of their indices of the English target sentence

- After the first training for each target word the most similar source word or words are derived and for each source word the most similar target word or words are selected as well. Note, the optimal pairs of the source and target words can be different if the views are changed. After the second training the same procedure is applied providing further pairs of similar words.

These alignments are displayed in Table 2.3. After superposing these alignments the derived correspondence between the source and target words can be seen in Table 2.4.

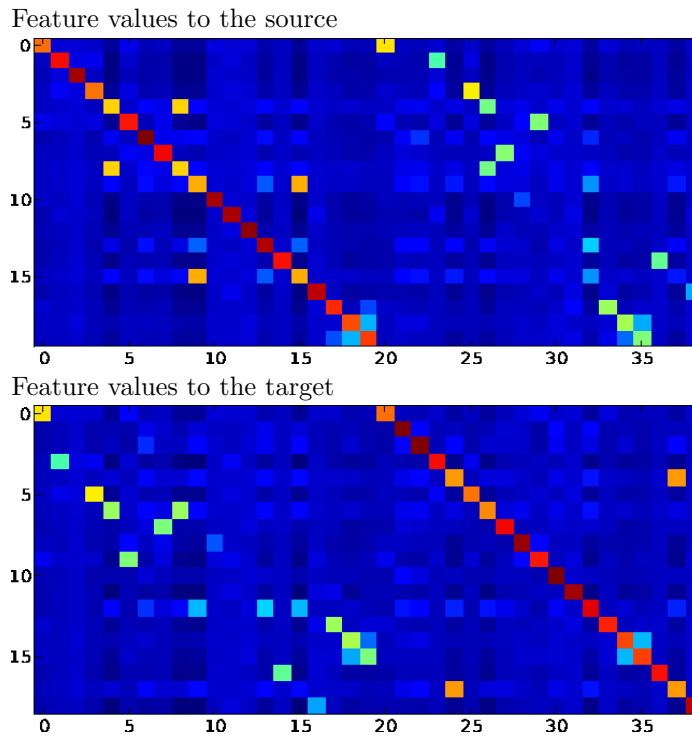


Figure 2.2: Demonstration of the feature vectors, bright colours show higher values.

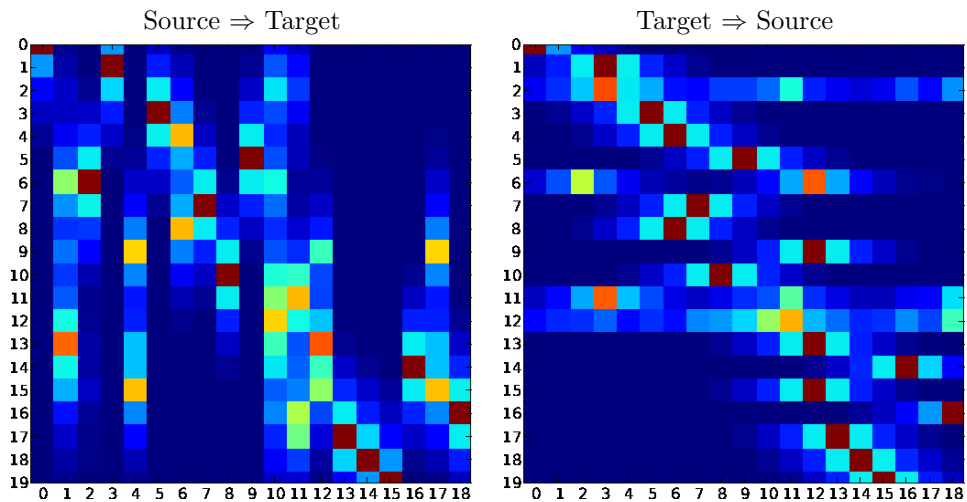


Figure 2.3: The margin based similarity map between source and target words predicted from the source, first image, and from the target, second image. The row indices identify the French words and the column indices the English ones. The bright red shows the strong the dark blue the weak relations



| Training set: source words, test set: target words |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |
|----------------------------------------------------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|
| Most corresponding target to each source word      |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |
| source word indices                                |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |
| 0                                                  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| 0                                                  | 3 | 5 | 5 | 6 | 9 | 2 | 7 | 6 | 4 | 8  | 11 | 10 | 12 | 16 | 4  | 18 | 13 | 14 | 15 |
| target word indices                                |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |

| Training set: target words, test set: source words |   |   |   |   |   |    |   |   |    |    |    |    |    |    |    |    |    |    |    |
|----------------------------------------------------|---|---|---|---|---|----|---|---|----|----|----|----|----|----|----|----|----|----|----|
| Most corresponding target to each source word      |   |   |   |   |   |    |   |   |    |    |    |    |    |    |    |    |    |    |    |
| source word indices                                |   |   |   |   |   |    |   |   |    |    |    |    |    |    |    |    |    |    |    |
| 0                                                  | 1 | 2 | 3 | 4 | 5 | 6  | 7 | 8 | 9  | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| 0                                                  | 3 | 3 | 5 | 6 | 9 | 12 | 7 | 6 | 12 | 8  | 3  | 11 | 12 | 16 | 12 | 18 | 13 | 14 | 15 |
| target word indices                                |   |   |   |   |   |    |   |   |    |    |    |    |    |    |    |    |    |    |    |

| Training set: source words, test set: target words |    |   |   |    |   |   |   |    |   |    |    |    |    |    |    |    |    |    |  |
|----------------------------------------------------|----|---|---|----|---|---|---|----|---|----|----|----|----|----|----|----|----|----|--|
| Most corresponding source to each target word      |    |   |   |    |   |   |   |    |   |    |    |    |    |    |    |    |    |    |  |
| target word indices                                |    |   |   |    |   |   |   |    |   |    |    |    |    |    |    |    |    |    |  |
| 0                                                  | 1  | 2 | 3 | 4  | 5 | 6 | 7 | 8  | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |  |
| 0                                                  | 13 | 6 | 1 | 15 | 3 | 8 | 7 | 10 | 5 | 12 | 11 | 13 | 17 | 18 | 19 | 14 | 15 | 16 |  |
| source word indices                                |    |   |   |    |   |   |   |    |   |    |    |    |    |    |    |    |    |    |  |

| Training set: target words, test set: source words |   |   |   |   |   |   |   |    |   |    |    |    |    |    |    |    |    |    |  |
|----------------------------------------------------|---|---|---|---|---|---|---|----|---|----|----|----|----|----|----|----|----|----|--|
| Most corresponding source to each target word      |   |   |   |   |   |   |   |    |   |    |    |    |    |    |    |    |    |    |  |
| target word indices                                |   |   |   |   |   |   |   |    |   |    |    |    |    |    |    |    |    |    |  |
| 0                                                  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8  | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |  |
| 0                                                  | 0 | 6 | 1 | 1 | 3 | 4 | 7 | 10 | 5 | 12 | 12 | 13 | 17 | 18 | 19 | 14 | 14 | 16 |  |
| source word indices                                |   |   |   |   |   |   |   |    |   |    |    |    |    |    |    |    |    |    |  |

Table 2.3: Predicted alignments



| source words | aligned target words(occurrences)        |
|--------------|------------------------------------------|
| centroware   | centroware(4)                            |
| regroupe     | suite(4) of(1)                           |
| plusieurs    | applications(1) suite(1) $\emptyset$ (2) |
| applications | applications(4)                          |
| et           | and(3) $\emptyset$ (1)                   |
| pilotes      | drivers(4)                               |
| d'           | a(3) the(1)                              |
| imprimante   | printer(4)                               |
| et           | and(3) $\emptyset$ (1)                   |
| de           | of(1) the(1) $\emptyset$ (2)             |
| télécopie    | fax(4)                                   |
| compatibles  | support(2) suite(1) the(1)               |
| avec         | that(3) support(1)                       |
| la           | the(4)                                   |
| ligne        | line(3) of(1)                            |
| de           | of(2) the(1) $\emptyset$ (1)             |
| produits     | products(4)                              |
| xerox        | xerox(4)                                 |
| document     | document(4)                              |
| centre       | centre(4)                                |

Table 2.4: From source to target alignment derived of four different views.  $\emptyset$  denotes the empty word.



## 2.10 Multiview learning

The approach described in this section gives rise to an extension of the optimisation framework used in the phrase prediction. It allows us to exploit more sophisticated nonlinear relationships among the words of a sentence.

We may consider the features defined by (2.13) and (2.14) as concatenation of two different sources of input properties, namely the word relations to the source words and to the target words. After splitting the feature vectors we have in case of the source words

$$\begin{aligned}
 \text{view 1:} \quad \phi_{\mathcal{S}_s}(w_s) &= \underbrace{(s(w_s, w_{s_1}), \dots, s(w_s, w_{s_{n_{\mathcal{S}_s}}}))}_{(w_{s_1}, \dots, w_{s_{n_{\mathcal{S}_s}}}) = \mathcal{S}_s}, \\
 \text{view 2:} \quad \phi_{\mathcal{S}_t}(w_s) &= \underbrace{(s(w_s, w_{t_1}), \dots, s(w_s, w_{t_{n_{\mathcal{S}_t}}}))}_{(w_{t_1}, \dots, w_{t_{n_{\mathcal{S}_t}}}) = \mathcal{S}_t}, \tag{2.25} \\
 w_s &\in \mathcal{S}_s,
 \end{aligned}$$

and for the target words

$$\begin{aligned}
 \text{view 1:} \quad \phi_{\mathcal{S}_s}(w_t) &= \underbrace{(s(w_t, w_{s_1}), \dots, s(w_t, w_{s_{n_{\mathcal{S}_s}}}))}_{(w_{s_1}, \dots, w_{s_{n_{\mathcal{S}_s}}}) = \mathcal{S}_s}, \\
 \text{view 2:} \quad \phi_{\mathcal{S}_t}(w_t) &= \underbrace{(s(w_t, w_{t_1}), \dots, s(w_t, w_{t_{n_{\mathcal{S}_t}}}))}_{(w_{t_1}, \dots, w_{t_{n_{\mathcal{S}_t}}}) = \mathcal{S}_t}, \tag{2.26} \\
 w_t &\in \mathcal{S}_t.
 \end{aligned}$$

Joining the features in an additive way in the optimisation problem

$$\begin{aligned}
 \min \quad & \frac{1}{2} (\|\mathbf{W}_{\mathcal{S}_s}\|_{Frobenius}^2 + \|\mathbf{W}_{\mathcal{S}_t}\|_{Frobenius}^2) + C \sum_{s=1}^{n_{\mathcal{S}_s}} \xi_s \\
 \text{w.r.t.} \quad & \mathbf{W}_{\mathcal{S}_s}, \mathbf{W}_{\mathcal{S}_t} \text{ linear operators, } \boldsymbol{\xi} \text{ loss,} \\
 \text{s.t.} \quad & \langle \psi_{\mathcal{P}_{\mathcal{S}_s}}(w_s), \mathbf{W}_{\mathcal{S}_s} \phi_{\mathcal{S}_s}(w_s) + \mathbf{W}_{\mathcal{S}_t} \phi_{\mathcal{S}_t}(w_s) \rangle \geq 1 - \xi_s, \quad w_s \in \mathcal{S}_s, \\
 & \boldsymbol{\xi} \geq \mathbf{0}, \tag{2.27}
 \end{aligned}$$

will provide the same prediction that we received solving (2.20), since it just cut the terms in the summation of the inner product into two parts and they will be reunited in the dual problem. However choosing a different approach by applying outer product between the views we can obtain a significantly distinct solution

$$\begin{aligned}
 \min \quad & \frac{1}{2} \|\mathbf{W}_{\mathcal{S}_s \otimes \mathcal{S}_t}\|_{Frobenius}^2 + C \sum_{s=1}^{n_{\mathcal{S}_s}} \xi_s \\
 \text{w.r.t.} \quad & \mathbf{W}_{\mathcal{S}_s \otimes \mathcal{S}_t} \text{ linear operator, } \boldsymbol{\xi} \text{ loss,} \\
 \text{s.t.} \quad & \langle \psi_{\mathcal{P}_{\mathcal{S}_s}}(w_s), \mathbf{W}_{\mathcal{S}_s \otimes \mathcal{S}_t} [\phi_{\mathcal{S}_s}(w_s) \otimes \phi_{\mathcal{S}_t}(w_s)] \rangle \geq 1 - \xi_s, \quad w_s \in \mathcal{S}_s, \\
 & \boldsymbol{\xi} \geq \mathbf{0}, \tag{2.28}
 \end{aligned}$$



The additive case (2.27) implies the following dual problem

$$\begin{aligned}
 \min \quad & \frac{1}{2} \boldsymbol{\alpha} \mathbf{K}_\psi \bullet [\mathbf{K}_{S_s} + \mathbf{K}_{S_t}] \boldsymbol{\alpha} - \mathbf{1}' \boldsymbol{\alpha} \\
 \text{w.r.t.} \quad & \boldsymbol{\alpha} \in \mathbb{R}^{n_{S_s}} \\
 \text{s.t.} \quad & \mathbf{0} \leq \boldsymbol{\alpha} \leq \mathbf{1},
 \end{aligned} \tag{2.29}$$

where  $\mathbf{K}_\psi$  is the kernel of the label vectors,  $\mathbf{K}_{S_s}$  and  $\mathbf{K}_{S_t}$  are the kernels of the two input views.

The productive case (2.28) however gives the dual

$$\begin{aligned}
 \min \quad & \frac{1}{2} \boldsymbol{\alpha} \mathbf{K}_\psi \bullet [\mathbf{K}_{S_s} \bullet \mathbf{K}_{S_t}] \boldsymbol{\alpha} - \mathbf{1}' \boldsymbol{\alpha} \\
 \text{w.r.t.} \quad & \boldsymbol{\alpha} \in \mathbb{R}^{n_{S_s}} \\
 \text{s.t.} \quad & \mathbf{0} \leq \boldsymbol{\alpha} \leq \mathbf{1},
 \end{aligned} \tag{2.30}$$

where the pointwise summation between the views is replaced with a pointwise multiplication, thus the relations between the views become nonlinear, where all the components of a view meet with all the components of the other view.

## 2.11 Phrase extraction

The phrase extraction is executed by the following algorithm:

- From a given sentence compute the four possible alignments between the source and the target words, see the example in Table 2.3.
- From the four predicted alignment a score is derived to all pairs of source and target words. The score in the current version is only the sum of maximum margin based alignments computed by (2.23). Obviously, if the four possible alignment agree on the same pair the corresponding score is higher.
- Collect to a source phrase all the words of the target sentence which aligned to any word of this phrase.
- Drop the words with score under a certain threshold.
- Sort the candidate target words into the order of the target sentence.
- A target phrase is accepted in the current version:
  - if contains no more than one gap with length 1.
  - if its length differs from the source phrase with at most 1 word.



## 2.12 Sentence translation

The sentence translation follows the schema presented in Deliverable 4.2. Here we simply summarise the main steps rather than repeat the technical detail:

- Find to each of the source phrases(ngrams) the translated target phrase with the highest score multiplied with the length of the phrase from the phrase dictionary.
- The candidate phrases are put into bags corresponding to each word of the source sentence. A candidate phrase corresponding to a source phrase is stored into every bag of the words of the source phrase.
- The phrases are preprocessed based on the following consistency rules:
  - Candidate phrase contained by another phrase in the same bag is dropped.
  - Long phrases have no short phrases as subset are deleted as well.
  - If a bag contains more than one candidate phrase then the phrases whose intersections with all other phrases in that, in the previous, and in the next bags are empty are deleted too.
- After filtering the phrases are reordered by a dynamic programming based algorithm. The distance between two consecutive phrases is computed as a sum of the following components
  - The size of the symmetric difference of the phrases normalised to the length of the union of the two phrases.
  - The distortion factor derived as a distance between the bags, corresponding to the source words, containing those phrases.
  - the score of the phrase would be included into the set of phrases covering the target phrases.
- After reordering the phrases are concatenated into a list of words.
- The list of target word is postprocessed.
  - The order of two words is reversed if their current order contradicts with the order of any candidate phrases. If the order provided by two phrases differs then the order of the phrase with the higher score is forced.
  - If two consecutive words are the same one of them is deleted.
  - If two consecutive word pairs are the same then one pair is ignored as well.



## 2.13 Matrax

This section provides a brief description of Matrax [15], a phrase-based translation system which uses *non-contiguous* bi-phrases as its basic building blocks.

Most phrase-based models proposed so far only deal with multi-word units that are sequences of contiguous words on both the source and the target side. Matrax, however, is designed to deal with multi-word expressions that need not be contiguous in either or both the source and the target side.

### 2.13.1 Non-contiguous phrases

Why should it be a good thing to use phrases composed of possibly non-contiguous sequences of words? In doing so we expect to improve translation quality by better accounting for additional linguistic phenomena as well as by extending the effect of contextual semantic disambiguation and example-based translation inherent in phrase-based MT. An example of a phenomenon best described using non-contiguous units is provided by English phrasal verbs. Consider the sentence “Mary *switches* her table lamp *off*”. Word-based statistical models would be at odds when selecting the appropriate translation of the verb. If French were the target language, for instance, corpus evidence would come from both examples in which “switch” is translated as “allumer” (to switch on) and as “éteindre” (to switch off). If many-to-one word alignments are not allowed from English to French, as it is usually the case, then the best thing a word-based model could do in this case would be to align “off” to the empty word and hope to select the correct translation from “switch” only, basically a 50-50 bet. While handling inseparable phrasal verbs such as “to run out” correctly, previously proposed phrase-based models would be helpless in this case. A comparable behavior is displayed by German separable verbs. Moreover, non-contiguous linguistic units are not limited to verbs. Negation is formed, in French, by inserting the words “ne” and “pas” before and after a verb respectively. So, the sentence “Pierre ne mange pas” and its English translation display a complex word-level alignment (Figure 2.4) most current models cannot account for.<sup>1</sup>

Flexible idioms, allowing for the insertion of linguistic material, are other phenomena best modeled with non-contiguous units.

---

<sup>1</sup>An exception is Chiang’s system [16], where *hierarchical bi-phrases* can be used to account for gappy constructions, using a form of synchronous context-free grammar for decoding. By contrast Matrax uses non-hierarchical bi-phrases in conjunction with a “flat” decoder.

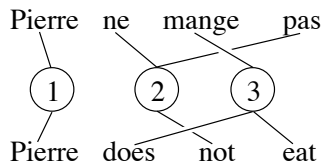


Figure 2.4: An example of a complex alignment associated with different syntax for negation in English and French.

### 2.13.2 Definition and library construction

We define a *bi-phrase* as a pair comprising a *source phrase* and a *target phrase*:  $b = \langle \tilde{s}, \tilde{t} \rangle$ . Each of the source and target phrases is a sequence of words and gaps (indicated by the symbol  $\diamond$ ); each gap acts as a placeholder for exactly one unspecified word. For example,  $\tilde{w} = w_1 w_2 \diamond w_3 \diamond \diamond w_4$  is a phrase of length 7, made up of two contiguous words  $w_1$  and  $w_2$ , a first gap, a third word  $w_3$ , two consecutive gaps and a final word  $w_4$ . To avoid redundancy, phrases may not begin or end with a gap. If a phrase does not contain any gaps, we say it is *contiguous*; otherwise it is *non-contiguous*. Likewise, a bi-phrase is said to be *contiguous* if both its phrases are contiguous.

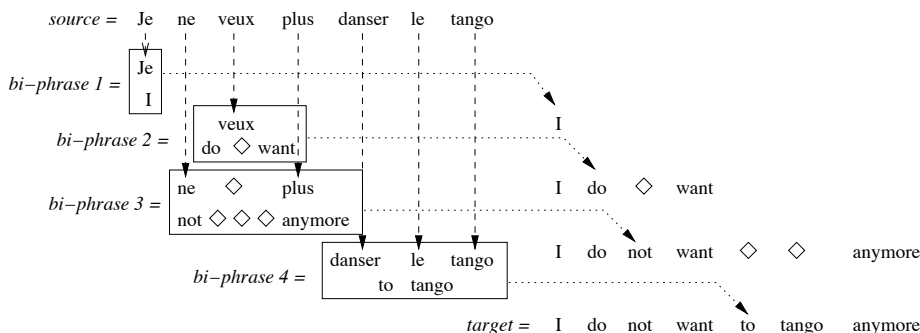


Figure 2.5: Combining bi-phrases to produce a translation.

The translation of a source sentence  $s$  is produced by combining together bi-phrases so as to cover the source sentence, and produce a well-formed target-language sentence (i.e. without gaps). A complete translation for  $s$  can be described as an ordered sequence of bi-phrases  $b_1 \dots b_K$ . When piecing together the final translation, the target-language portion  $\tilde{t}_1$  of the first bi-phrase  $b_1$  is first laid down, then each subsequent  $\tilde{t}_k$  is positioned on the first “free” position in the target language sentence, i.e. either the leftmost gap, or the right end of



the sequence. Figure 2.5 illustrates this process with an example.

To produce translations, our approach therefore relies on a collection of bi-phrases, what we call a *bi-phrase library*. Such a library is constructed from a corpus of existing translations, aligned at the word level.

Two strategies come to mind to produce non-contiguous bi-phrases for these libraries. The first is to align the words using a “standard” word alignment technique, such as the *Refined Method* described in [17] (the intersection of two IBM Viterbi alignments, forward and reverse, enriched with alignments from the union) and then generate bi-phrases by combining together individual alignments that co-occur in the same pair of sentences. This is the strategy that is usually adopted in other phrase-based MT approaches [18, 19]. Here, the difference is that we are not restricted to combinations that produce strictly contiguous bi-phrases.

The second strategy is to rely on a word-alignment method that naturally produces many-to-many alignments between non-contiguous words, such as the method described in [20]. By means of a matrix factorization, this method produces a parallel partition of the two texts, seen as sets of word tokens. Each token therefore belongs to one, and only one, subset within this partition, and corresponding subsets in the source and target make up what are called *cepts*. For example, in Figure 2.4, these cepts are represented by the circles numbered 1, 2 and 3; each cept thus connects word tokens in the source and the target, regardless of position or contiguity. These cepts naturally constitute bi-phrases, and can be used directly to produce a bi-phrase library.

Obviously, the two strategies can be combined, and it is always possible to produce increasingly large and complex bi-phrases by combining together co-occurring bi-phrases, contiguous or not. One problem with this approach, however, is that the resulting libraries can become very large. With contiguous phrases, the number of bi-phrases that can be extracted from a single pair of sentences typically grows quadratically with the size of the sentences; with non-contiguous phrases, however, this growth is exponential. As it turns out, the number of available bi-phrases for the translation of a sentence has a direct impact on the time required to compute the translation; we will therefore typically rely on various filtering techniques, aimed at keeping only those bi-phrases that are more likely to be useful. For example, we may retain only the most frequently observed bi-phrases, or impose limits on the number of cepts, the size of gaps, etc.

### 2.13.3 The Model

In statistical machine translation, we are given a source language input  $s_1^J = s_1 \dots s_J$ , and seek the target-language sentence  $t_1^I = t_1 \dots t_I$  that is its most likely



translation:

$$\hat{t}_1^I = \operatorname{argmax}_{t_1^I} Pr(t_1^I | s_1^J) \quad (2.31)$$

Our approach is based on a direct approximation of the posterior probability  $Pr(t_1^I | s_1^J)$ , using a log-linear model:

$$Pr(t_1^I | s_1^J) = \frac{1}{Z_{s_1^J}} \exp \left( \sum_{m=1}^M \lambda_m h_m(t_1^I, s_1^J) \right)$$

In such a model, the contribution of each *feature function*  $h_m$  is determined by the corresponding model parameter  $\lambda_m$ ;  $Z_{s_1^J}$  denotes a normalization constant. This type of model is now quite widely used for machine translation [21, 18], and is also the approach underlying Portage.

Additional variables can be introduced in such a model, so as to account for hidden characteristics, and the feature functions can be extended accordingly. For example, our model must take into account the actual set of bi-phrases that was used to produce this translation:

$$Pr(t_1^I, b_1^K | s_1^J) = \frac{1}{Z_{s_1^J}} \exp \left( \sum_{m=1}^M \lambda_m h_m(t_1^I, s_1^J, b_1^K) \right)$$

Our model relies on nine feature functions, which we describe here.

- The *bi-phrase* feature function  $h_{bp}$ : it represents the probability of producing  $t_1^I$  using some set of bi-phrases, under the assumption that each source phrase produces a target phrase independently of the others:

$$h_{bp}(t_1^I, s_1^J, b_1^K) = \sum_{k=1}^K \log Pr(\tilde{t}_k | \tilde{s}_k) \quad (2.32)$$

Individual bi-phrase probabilities  $Pr(\tilde{t}_k | \tilde{s}_k)$  are estimated based on occurrence counts in the word-aligned training corpus.

- The *reverse bi-phrase* feature function  $h_{rbp}$ : similar to  $h_{bp}$ , but conditioning the source phrase on the target phrase:

$$h_{rbp}(t_1^I, s_1^J, b_1^K) = \sum_{k=1}^K \log Pr(\tilde{s}_k | \tilde{t}_k) \quad (2.33)$$



- The *compositional bi-phrase* feature function  $h_{comp}$ : this is introduced to compensate for  $h_{bp}$ 's strong tendency to overestimate the probability of rare bi-phrases; it is computed as in equation (2.33), except that bi-phrase probabilities are computed based on individual word translation probabilities, somewhat as in IBM model 1 [22]:

$$Pr(\tilde{t}|\tilde{s}) = \frac{1}{|\tilde{s}||\tilde{t}|} \prod_{t \in \tilde{t}} \sum_{s \in \tilde{s}} Pr(t|s)$$

- The *reverse compositional bi-phrase* feature function  $h_{rcomp}$ , similar to  $h_{comp}$ , but where the bi-phrase probabilities are computed in the opposite direction:

$$Pr(\tilde{s}|\tilde{t}) = \frac{1}{|\tilde{t}||\tilde{s}|} \prod_{s \in \tilde{s}} \sum_{t \in \tilde{t}} Pr(s|t)$$

- The *target language* feature function  $h_{tl}$ : this is based on a  $N$ -gram language model of the target language. As such, it ignores the source language sentence and the decomposition of the target into bi-phrases, to focus on the actual sequence of target-language words produced by the combination of bi-phrases:

$$h_{tl}(t_1^I, s_1^J, b_1^K) = \sum_{i=1}^I \log Pr(t_i | t_{i-N+1}^{i-1})$$

- The *word-count* and *bi-phrase count* feature functions  $h_{wc}$  and  $h_{bc}$ : these control the length of the translation and the number of bi-phrases used to produce it:

$$h_{wc}(t_1^I, s_1^J, b_1^K) = I \quad h_{bc}(t_1^I, s_1^J, b_1^K) = K$$

- The *reordering* feature function  $h_{reord}(t_1^I, s_1^J, b_1^K)$ : it measures the amount of reordering between bi-phrases of the source and target sentences.
- the *gap count* feature function  $h_{gc}$ : It takes as value the total number of gaps (source and target) within the bi-phrases of  $b_1^K$ , thus allowing the model some control over the nature of the bi-phrases it uses, in terms of the discontinuities they contain.

#### 2.13.4 Parameter Estimation

The values of the  $\lambda$  parameters of the log-linear model can be set so as to optimize a given criterion. For instance, one can maximize the likelihood of



some set of training sentences. Instead, and as suggested by Och [23], we chose to maximize directly the quality of the translations produced by the system, as measured with a machine translation evaluation metric.

Say we have a set of source-language sentences  $S$ . For a given value of  $\lambda$ , we can compute the set of corresponding target-language translations  $T$ . Given a set of *reference* (“gold-standard”) translations  $R$  for  $S$  and a function  $E(T, R)$  which measures the “error” in  $T$  relative to  $R$ , then we can formulate the parameter estimation problem as<sup>2</sup>:

$$\hat{\lambda} = \operatorname{argmin}_{\lambda} E(T, R)$$

As pointed out by Och, one notable difficulty with this approach is that, because the computation of  $T$  is based on an  $\operatorname{argmax}$  operation (see eq. 2.31), it is not continuous with regard to  $\lambda$ , and standard gradient-descent methods cannot be used to solve the optimization. Och proposes two workarounds to this problem: the first one relies on a direct optimization method derived from Powell’s algorithm; the second introduces a smoothed (continuous) version of the error function  $E(T, R)$  and then relies on a gradient-based optimization method.

We have opted for this last approach. Och shows how to implement it when the error function can be computed as the sum of errors on individual sentences. Unfortunately, this is not the case for such widely used MT evaluation metrics as BLEU [24] and NIST [25]. We show here how it can be done for NIST; a similar derivation is possible for BLEU.

The NIST evaluation metric computes a weighted  $n$ -gram precision between  $T$  and  $R$ , multiplied by a factor  $B(S, T, R)$  that penalizes short translations. It can be formulated as:

$$B(S, T, R) \times \sum_{n=1}^N \frac{\sum_{s \in S} I_n(t_s, r_s)}{\sum_{s \in S} C_n(t_s)} \quad (2.34)$$

where  $N$  is the largest  $n$ -gram considered (usually  $N = 4$ ),  $I_n(t_s, r_s)$  is a weighted count of common  $n$ -grams between the target ( $t_s$ ) and reference ( $r_s$ ) translations of sentence  $s$ , and  $C_n(t_s)$  is the total number of  $n$ -grams in  $t_s$ .

To derive a version of this formula that is a continuous function of  $\lambda$ , we will need multiple translations  $t_{s,1}, \dots, t_{s,K}$  for each source sentence  $s$ . The general idea is to weight each of these translations by a factor  $w(\lambda, s, k)$ , proportional to the score  $m_{\lambda}(t_{s,k}|s)$  that  $t_{s,k}$  is assigned by the log-linear model for a given

---

<sup>2</sup>For the sake of simplicity, we consider a single reference translation per source sentence, but the argument can easily be extended to multiple references.



$\lambda$ :

$$w(\lambda, s, k) = \left[ \frac{m_\lambda(t_{s,k}|s)}{\sum_{k'} m_\lambda(t_{s,k'}|s)} \right]^\alpha$$

where  $\alpha$  is the *smoothing factor*. Thus, in the smoothed version of the NIST function, the term  $I_n(t_s, r_s)$  in equation (2.34) is replaced by  $\sum_k w(\lambda, s, k) I_n(t_{s,k}, r_s)$ , and the term  $C_n(t_s)$  is replaced by  $\sum_k w(\lambda, s, k) C_n(t_{s,k})$ . As for the brevity penalty factor  $B(S, T, R)$ , it depends on the total length of translation  $T$ , i.e.  $\sum_s |t_s|$ . In the smoothed version, this term is replaced by  $\sum_s \sum_k w(\lambda, s, k) |t_{s,k}|$ . Note that, when  $\alpha \rightarrow \infty$ , then  $w(\lambda, s, k) \rightarrow 0$  for all translations of  $s$ , except the one for which the model gives the highest score, and so the smooth and normal NIST functions produce the same value. In practice, we determine some “good” value for  $\alpha$  by trial and error (5 works fine).

We thus obtain a scoring function for which we can compute a derivative relative to  $\lambda$ , and which can be optimized using gradient-based methods. In practice, we use the *OPT++* implementation of a quasi-Newton optimization [26]. As observed by Och, the smoothed error function is not convex, and therefore this sort of minimum-error rate training is quite sensitive to the initialization values for the  $\lambda$  parameters. Our approach is to use a random set of initializations for the parameters, perform the optimization for each initialization, and select the model which gives the overall best performance.

Globally, parameter estimation proceeds along these steps:

1. Initialize the training set: using random parameter values  $\lambda_0$ , for each source sentence of some given set of sentences  $S$ , we compute multiple translations. (In practice, we use the  $M$ -best translations produced by our decoder; see Section 2.13.5).
2. Optimize the parameters: using the method described above, we find  $\lambda$  that produces the best smoothed NIST score on the training set.
3. Iterate: we then re-translate the sentences of  $S$  with this new  $\lambda$ , combine the resulting multiple translations with those already in the training set, and go back to step 2.

Steps 2 and 3 can be repeated until the smoothed NIST score does not increase anymore<sup>3</sup>.

---

<sup>3</sup>It can be seen that, as the set of possible translations for  $S$  stabilizes, we eventually reach a point where the procedure converges to a maximum. In practice, however, we can usually stop much earlier.



### 2.13.5 Decoder

The decoder that we implemented is a version of the beam-search stack decoder described in [27], extended to cope with non-contiguous phrases. Each translation is the result of a sequence of *decisions*, each of which involves the selection of a bi-phrase and of a target position. The final result is obtained by combining decisions, as in Figure 2.5. *Hypotheses*, corresponding to partial translations, are organised in a sequence of priority stacks, one for each number of source words covered. Hypotheses are extended by filling the first available uncovered position in the target sentence; each extended hypotheses is then inserted in the stack corresponding to the updated number of covered source words. Each hypothesis is assigned a score which is obtained as a combination of the actual feature function values and of admissible heuristics, adapted to deal with gaps in phrases, estimating the future cost for completing a translation. Each stack undergoes both threshold and histogram pruning. Whenever two hypotheses are indistinguishable as far as the potential for further extension is concerned, they are merged and only the highest-scoring is further extended. Complete translations are eventually recovered in the “last” priority stack, i.e. the one corresponding to the total number of source words: the best translation is the one with the highest score, and that does not have any remaining gaps in the target.

## Chapter 3

# Experimental Results

In this section we detail the overall set-up for the experimental evaluation and also where possible indicate the resources used for training each of the algorithms. The process was aided by the SMART evaluation suite<sup>1</sup> which allowed participants to upload translations and directly compare the performance.

### 3.1 Experimental Set-up

The data used in the experiments was taken from Version 3 of the Europarl data. Five splits of the data were constructed, with two language pairs: Spanish - English and French - English. The training sizes for the languages were just under 1.3M sentences for the Spanish corpus, and just over 1.3M sentences for the French. Test sizes were 10,000 sentences in each case. For the data, some preprocessing was done, mainly to remove outliers. Any sentence with more than 80 words in it was removed, and any sentences that were clearly 'junk' (for example those that simply contained one character) were also removed. Sentences were then tokenized by tokenizer.perl program<sup>2</sup>. We note that not all current algorithms are scalable to such large training and test set sizes, therefore when submitting test translations, participants can choose to indicate that only the first 10,000 or 50,000 sentences from each split were used (or the entire set). All translations for the test sets are submitted using the evaluation suite developed within SMART, therefore allowing common evaluation metrics to be applied to all methods, and automatic significance testing across algorithms to be carried out. Submission and results are available via a web interface so all

---

<sup>1</sup><http://isambard.ijs.si/smart>

<sup>2</sup>Provided at <http://www.statmt.org/europarl/> in the tool package.



participants can get feedback on their results and compare to other algorithms. For those that can be scaled to the full data set, submitting translations for the restricted training sizes allows comparison to less efficient methods, and also shows how performance of a technique increases with more training data.

In addition to training on a subset of sentences, users can also submit translations for only the first 1,000 test sentences (in the case that decoding for a method is particularly slow). Note that for any technique which submits the full test set, the first 1,000 is automatically extracted for a direct comparison to methods that can only scale to a test size of 1,000. The evaluation metrics used were BLEU and NIST scores, and were performed in both a lower-case only setting (predictions and references are converted to lower-case) and a true-case setting; the former being chosen for those algorithms not yet adapted to cases and also to ensure translation accuracy rather than re-casing accuracy is measured.

## 3.2 Resources

In the following subsections we give some approximate notes on the resources used for the different algorithms. Note that in many cases due to the training set size and short duration of the evaluation period, many of the algorithms were run on multiple processors or clusters so any timings given are approximate. Given however that a large part of the remainder of the workpackage focusses on improving the efficiency of the various techniques, it was deemed beneficial to give attention to this (often neglected or completely unreported) aspect; even if they are only a guideline.

### 3.2.1 Portage

Tables 3.1 and 3.2 show the number of parameters and the training times, respectively, for each condition, averaged over all five folds. Both tables are broken down by model component as follows: language model (LM), phrase table (TM), first-pass loglinear model, second-pass loglinear model, and truecasing model. The phrase table and loglinear models were trained 10-ways parallel, with each parallel job incurring a non-negligible model-load cost. (In other words, training these components using a single processor would not take 10 times as long as shown here.)

### 3.2.2 Matrax

Due to time constraints, Matrax participated only in the English-Spanish track: 5-fold cross validation on 1.3M / 50K / 10K sentences for training, and 1K /



| condition |        | number of parameters |        |             |             |           |         |
|-----------|--------|----------------------|--------|-------------|-------------|-----------|---------|
|           |        | LM                   | TM     | loglinear 1 | loglinear 2 | truecaser | total   |
| fren      | 10k    | 128k                 | 134k   | 7           | 29          | 132k      | 394k    |
| enfr      | 10k    | 142k                 | 134k   | 7           | 29          | 147k      | 423k    |
| esen      | 10k    | 128k                 | 137k   | 7           | 29          | 131k      | 396k    |
| enes      | 10k    | 139k                 | 137k   | 7           | 29          | 149k      | 425k    |
| fren      | 50k    | 508k                 | 732k   | 7           | 29          | 469k      | 1,708k  |
| enfr      | 50k    | 562k                 | 732k   | 7           | 29          | 517k      | 1,811k  |
| esen      | 50k    | 508k                 | 727k   | 7           | 29          | 469k      | 1,704k  |
| enes      | 50k    | 556k                 | 727k   | 7           | 29          | 533k      | 1,816k  |
| fren      | 1,257k | 7,623k               | 20,707 | 7           | 29          | 526k      | 33,590k |
| enfr      | 1,257k | 8,165k               | 20,707 | 7           | 29          | 548k      | 34,350k |
| esen      | 1,306k | 7,459k               | 22,508 | 7           | 29          | 516k      | 35,131k |
| enes      | 1,306k | 7,844k               | 22,508 | 7           | 29          | 564k      | 35,992k |

Table 3.1: Portage Model Sizes

10K sentences for test. In these experiments, the main options and parameters set in Matrax were the following:

- Building the bi-phrase library: to construct our bi-phrase library according to the procedure described in Section 2.13.2, we considered only one possible word-alignment per training sentence pair, and extracted a maximum of 2 cepts per bi-phrase (or 3 cepts for the smaller training sets), allowing non-contiguous bi-phrases with a maximum of 4 gaps, and filtering bi-phrases occurring less than 2 times in the corpus.
- Estimating the model parameters: to tune the model parameters we set aside 2K sentences from the training set and followed the procedure described in Section 2.13.4. We generated n-best lists of 200 translations, produced by a decoder using a beam of width 12 and stack of size 48,000, with a maximum of reordering between bi-phrases of the source and target sentences of 5. We iterated through this procedure 3 times, with 100 optimization rounds, and a perturbation of 0.25 in the initialization values.
- Decoding: to produce the final translations, Matrax decoder (see Section 2.13.5) was set to use a beam of width 12 and stack of size 48,000, with a maximum reordering of 5.

The training and decoding was done with lower cased data. For the submission, the translations were recased using the WMT recasing tools (<http://www.statmt.org/wmt08/scripts.tgz>) trained on the same parallel data used to train Matrax.



| condition |        | training time (minutes) |     |             |             |           |       |
|-----------|--------|-------------------------|-----|-------------|-------------|-----------|-------|
|           |        | LM                      | TM  | loglinear 1 | loglinear 2 | truecaser | total |
| fren      | 10k    | 0                       | 70  | 48          | 66          | 1         | 185   |
| enfr      | 10k    | 0                       | 72  | 40          | 59          | 1         | 172   |
| esen      | 10k    | 0                       | 71  | 35          | 53          | 1         | 160   |
| enes      | 10k    | 0                       | 71  | 32          | 46          | 1         | 150   |
| fren      | 50k    | 1                       | 74  | 35          | 62          | 2         | 174   |
| enfr      | 50k    | 1                       | 74  | 33          | 55          | 2         | 165   |
| esen      | 50k    | 1                       | 73  | 24          | 42          | 2         | 142   |
| enes      | 50k    | 1                       | 73  | 37          | 38          | 1         | 150   |
| fren      | 1,257k | 7                       | 337 | 39          | 47          | 5         | 435   |
| enfr      | 1,257k | 10                      | 341 | 59          | 51          | 5         | 466   |
| esen      | 1,306k | 7                       | 245 | 43          | 46          | 3         | 344   |
| enes      | 1,306k | 10                      | 289 | 45          | 44          | 4         | 392   |

Table 3.2: Portage Training Times

### 3.2.3 Results

This section details the results obtained so far. Note that the *Sinuhe* and *Sinuhe*<sup>LM</sup> models were only trained on one-fold. However given the very low standard deviation of the other models, it should not be too difficult to assess any significant differences between the techniques. Note that in the following tables, only the lower-case evaluations are reported as we are currently focussing on translation systems, rather than the quality of the recaser. Full results however are available from <http://isambard.ijs.si/smart/AllRes.aspx>. Note also that for entries without standard deviations, the result is for one-fold only. We also include results for the state-of-the-art system Pharaoh, for which 100,000 sentences was the training size limit using hardware comparable to the MMR approach. A 4-gram language model trained on the CMU toolkit was used, with a 1000 sentence development set.

## 3.3 Conclusions

In this deliverable we have described the recent progress made with respect to the development of the translation systems, and also provided an initial set of empirical results to assist in evaluating the project at this stage. The full baseline set of results from the Portage system have been obtained, and this is the baseline to which methods should be compared in the final evaluation. As expected, the methods developed so far still have some additional gains to



## D2.2: Application of Markov approaches to SMT

| Train | Method    | BLEU 10K            | NIST 10K            | BLEU 1K             | NIST 1K             |
|-------|-----------|---------------------|---------------------|---------------------|---------------------|
| Full  | Portage   | $0.3138 \pm 0.0022$ | $8.0414 \pm 0.0475$ | $0.2897 \pm 0.0026$ | $6.8986 \pm 0.0262$ |
| Full  | Sinuhe    | 0.2597              | 7.1647              | 0.2391              | 6.1942              |
| Full  | Sinuhe-LM | 0.2797              | 7.6502              | 0.2519              | 6.5514              |
| 100K  | Pharaoh   |                     |                     | $0.2491 \pm 0.0032$ | $6.2971 \pm 0.0488$ |
| 50K   | Portage   | $0.2382 \pm 0.0027$ | $6.8293 \pm 0.0344$ | $0.2434 \pm 0.0011$ | $6.2651 \pm 0.0310$ |
| 50K   | MMR       |                     |                     | $0.1592 \pm 0.0031$ | $5.3312 \pm 0.0301$ |
| 10K   | Portage   | $0.1892 \pm 0.0013$ | $5.9151 \pm 0.0168$ | $0.1901 \pm 0.0034$ | $5.4609 \pm 0.0394$ |
| 10K   | LSR       |                     |                     | $0.1490 \pm 0.0035$ | $4.8063 \pm 0.0378$ |

Table 3.3: Results for the English to French Task, ordered by size of training set used.

| Train | Method    | BLEU 10K            | NIST 10K            | BLEU 1K             | NIST 1K             |
|-------|-----------|---------------------|---------------------|---------------------|---------------------|
| Full  | Portage   | $0.3288 \pm 0.0020$ | $8.4330 \pm 0.0190$ | $0.3060 \pm 0.0008$ | $7.2165 \pm 0.0361$ |
| Full  | Sinuhe    | 0.2889              | 7.8689              | 0.2662              | 6.7746              |
| Full  | Sinuhe-LM | 0.3021              | 8.1032              | 0.2785              | 6.9527              |
| 100K  | Pharaoh   |                     |                     | $0.2661 \pm 0.0034$ | $6.7453 \pm 0.0361$ |
| 50K   | Portage   | $0.2525 \pm 0.0024$ | $6.5669 \pm 0.0277$ | $0.2525 \pm 0.0024$ | $6.5669 \pm 0.0277$ |
| 50K   | MMR       |                     |                     | $0.1828 \pm 0.0031$ | $5.7289 \pm 0.0232$ |
| 10K   | Portage   | $0.1992 \pm 0.0071$ | $6.4014 \pm 0.0329$ | $0.1997 \pm 0.0030$ | $5.8563 \pm 0.0417$ |
| 10K   | LSR       |                     |                     | $0.1654 \pm 0.0039$ | $5.2329 \pm 0.0501$ |

Table 3.4: Results for the French to English Task, ordered by size of training set used.

| Train | Method    | BLEU 10K            | NIST 10K            | BLEU 1K             | NIST 1K             |
|-------|-----------|---------------------|---------------------|---------------------|---------------------|
| Full  | Portage   | $0.3473 \pm 0.0022$ | $8.5573 \pm 0.0604$ | $0.3146 \pm 0.0028$ | $7.1362 \pm 0.0330$ |
| Full  | Matrax    | $0.2805 \pm 0.0036$ | $7.5888 \pm 0.0909$ | $0.2574 \pm 0.0036$ | $6.4513 \pm 0.0456$ |
| Full  | Sinuhe    | 0.2973              | 7.7946              | 0.26911             | 6.5219              |
| Full  | Sinuhe-LM | 0.3240              | 8.1987              | 0.2920              | 6.7268              |
| 100K  | Pharaoh   |                     |                     | $0.2700 \pm 0.0026$ | $6.5577 \pm 0.0587$ |
| 50K   | Portage   | $0.2657 \pm 0.0011$ | $7.3043 \pm 0.0260$ | $0.2611 \pm 0.0020$ | $6.4879 \pm 0.0463$ |
| 50K   | Matrax    | $0.2623 \pm 0.0052$ | $7.1857 \pm 0.0538$ | $0.2523 \pm 0.0076$ | $6.3157 \pm 0.0852$ |
| 50K   | Pharaoh   |                     |                     | $0.2516 \pm 0.0040$ | $6.3068 \pm 0.0643$ |
| 50K   | MMR       |                     |                     | $0.1650 \pm 0.0020$ | $5.4501 \pm 0.0589$ |
| 10K   | Portage   | $0.2118 \pm 0.0006$ | $6.3090 \pm 0.0348$ | $0.2086 \pm 0.0018$ | $5.7008 \pm 0.0408$ |
| 10K   | Matrax    | $0.2203 \pm 0.0024$ | $6.3835 \pm 0.0253$ | $0.2110 \pm 0.0027$ | $5.6785 \pm 0.0294$ |
| 10K   | LSR       |                     |                     | $0.1510 \pm 0.0027$ | $4.9121 \pm 0.0294$ |

Table 3.5: Results for the English to Spanish Task, ordered by size of training set used.



## D2.2: Application of Markov approaches to SMT

| Train | Method    | BLEU 10K            | NIST 10K            | BLEU 1K             | NIST 1K             |
|-------|-----------|---------------------|---------------------|---------------------|---------------------|
| Full  | Portage   | $0.3516 \pm 0.0013$ | $8.7959 \pm 0.0037$ | $0.3206 \pm 0.0027$ | $7.3608 \pm 0.0075$ |
| Full  | Sinuhe    | 0.3165              | 8.3355              | 0.2844              | 6.9595              |
| Full  | Sinuhe-LM | 0.3321              | 8.5891              | 0.3026              | 7.1725              |
| 100K  | Pharaoh   |                     |                     | $0.2754 \pm 0.0018$ | $6.7876 \pm 0.0122$ |
| 50K   | Portage   | $0.2705 \pm 0.0025$ | $7.5642 \pm 0.0301$ | $0.2611 \pm 0.0059$ | $6.6407 \pm 0.0398$ |
| 50K   | Pharaoh   |                     |                     | $0.2556 \pm 0.0013$ | $6.5230 \pm 0.0075$ |
| 50K   | MMR       |                     |                     | $0.1870 \pm 0.0023$ | $5.9437 \pm 0.0160$ |
| 10K   | Portage   | $0.2167 \pm 0.0033$ | $6.6109 \pm 0.0280$ | $0.2091 \pm 0.0051$ | $5.8965 \pm 0.0459$ |
| 10K   | LSR       |                     |                     | $0.1649 \pm 0.0054$ | $5.1834 \pm 0.0339$ |

Table 3.6: Results for the Spanish to English Task, ordered by size of training set used.

be made in terms of both performance and scalability. The results however are encouraging. The Sinuhe system was able to run in the short term using the full training set, and the NIST score is just one point below Portage. The MMR method at this stage does not scale to the full data set, and the performance is slightly less than Sinuhe; however there is no true language model in the system as yet, and also the decoder currently being used is a very simplistic search. Therefore, with the addition of these two components one would expect the performance to rise significantly, as evidenced by the increase in Sinuhe's performance with a language model.

In summary, the results are encouraging, and the scalability of the systems is perhaps the main remaining issue, which was anticipated in the proposal and is the main focus of the last year in workpackage 2. Integration of language models from workpackage 3 into the MMR-based approaches, and a test of using MMR-style word alignments as a replacement for GIZA in both Pharaoh and Sinuhe systems are ongoing, and should both provide improvements on the above results in future.

At this point in the project however, it is very beneficial to have a full set of results for the baseline system, and to see that the methods currently under development are not so far away in terms of performance, with many of the deficits of the newer systems already starting to be addressed.

# Bibliography

- [1] Philipp Koehn, Franz Josef Och, and Daniel Marcu. 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, Morristown, NJ, USA, 2003. Association for Computational Linguistics.
- [2] Franz Josef Och. Minimum error rate training in statistical machine translation. In *ACL '03: Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, pages 160–167, Morristown, NJ, USA, 2003. Association for Computational Linguistics.
- [3] Phil Blunson, Trevor Cohn, and Miles Osborne. A discriminative latent variable model for statistical machine translation. In *ACL*, 2008.
- [4] Cristoph Tillmann and Tong Zhang. A block bigram prediction model for statistical machine translation. *ACM Transactions on Speech and Language Processing*, 4(3), 2007.
- [5] Percy Liang, Alexandre Bouchard-Cote, Dan Klein, and Ben Taskar. An end-to-end discriminative approach to machine translation. In *ACL*, 2006.
- [6] Philipp Koehn, Hieu Hoang, Alexandra Birch, and Chris Callison-Burch et al. Moses: Open source toolkit for statistical machine translation. In *ACL*, 2007.
- [7] Franz Josef Och, Christoph Tillmann, and Hermann Ney. Improved alignment models for statistical machine translation. In *EMNLP*, pages 20–28, 1999.
- [8] Stanley Chen and Ronald Rosenfeld. A survey of smoothing techniques for ME models. *IEEE Transactions Speech and Audio Processing*, 8(1):37–50, 2000.



- [9] S. V. N. Vishwanathan, Nicol N. Schraudolph, Mark W. Schmidt, and Kevin P. Murphy. Accelerated training of conditional random fields with stochastic gradient methods. In *ICML*, 2006.
- [10] Philipp Koehn. Europarl: A parallel corpus for statistical machine translation. In *MT Summit*, 2005.
- [11] ACL second workshop on statistical machine translation. <http://www.statmt.org/wmt07>, 2007.
- [12] Andread Stolcke. Srilm - an extensible language modeling toolkit. In *Proceedings of International Conference in Spoken Language Processing*, 2002.
- [13] K. Papineni, S. Roukos, T. Ward, and W. Zhu. BLEU: a method for automatic evaluation of machine translation. In *ACL*, 2002.
- [14] B.H. Partee, A. ter Meulen, and R.E. Wall. *Mathematical Methods in Linguistics*. Studies in Linguistics and Philosophy, Volume 30. Kluwer, 1990.
- [15] Michel Simard, Nicola Cancedda, Bruno Cavestro, Marc Dymetman, Éric Gaussier, Cyril Goutte, Kenji Yamada, Philippe Langlais, and Arne Mauser. Translating with non-contiguous phrases. In *HLT/EMNLP*, 2005.
- [16] David Chiang. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the ACL*, pages 263–270, Ann Arbor, Michigan, 2005.
- [17] Franz Josef Och and Hermann Ney. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, 29(1):19–51, March 2003.
- [18] Richard Zens and Hermann Ney. Improvements in Phrase-Based Statistical Machine Translation. In *Proc. of the HLT-NAACL 2003 Conference*, Edmonton, Canada, 2003.
- [19] Franz Josef Och and Hermann Ney. The Alignment Template Approach to Statistical Machine Translation. *Computational Linguistics*, 30(4):417–449, 2004.
- [20] Cyril Goutte, Kenji Yamada, and Eric Gaussier. Aligning words using matrix factorisation. In *Proc. ACL'04*, pages 503–510, 2004.
- [21] Christoph Tillmann and Fei Xia. A phrase-based unigram model for statistical machine translation. In *Proc. of the HLT-NAACL 2003 Conference*, Edmonton, Canada, 2003.



## D2.2: Application of Markov approaches to SMT

---

- [22] Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311, 1993.
- [23] Franz Och. Minimum error rate training in statistical machine translation. In *ACL'03: 41st Ann. Meet. of the Assoc. for Computational Linguistics*, 2003.
- [24] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the ACL*, pages 311–318, Philadelphia, USA, 2002.
- [25] George Doddington. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proc. ARPA Workshop on Human Language Technology*, 2002.
- [26] J. C. Meza. OPT++: An Object-Oriented Class Library for Nonlinear Optimization. Technical Report SAND94-8225, Sandia National Laboratories, Albuquerque, USA, March 1994.
- [27] Philipp Koehn. *Noun Phrase Translation*. PhD thesis, University of Southern California, 2003.