Statistical Machine Translation

Sophie Arnoult, Gideon Maillette de Buyj Wenniger and Andrea Schuch

September 24, 2007

1 Introduction

Our goal in this project was to implement models 1 to 3 from the 1993 IBM paper by Brown et. al., *"The Mathematics of Statistical Machine Translation: Parameter Estimation"*. We briefly present these models in this introduction. We discuss our implementation of the models in section 2 and the experimental setup in section 3. Test results and their analysis are presented in section 4, and discussed in section 5.

All the IBM models, and Statistical Machine Translation (SMT) in general, model the problem of finding the best English translation for a French sentence[1] as a noisy channel: The French sentence $\mathbf{f}$[2] to translate is considered to be a 'corruption' of the English sentence $\mathbf{e}$ as depicted in Figure 1. Which English sentence is at the source of its French counterpart is unknown, but we can look for sentences that maximise $P(\mathbf{e}|\mathbf{f})$.

e
message
Noisy Channel
f
observation

Figure 1: The noisy channel model of machine translation

As by Bayes' theorem,

$$P(\mathbf{e}|\mathbf{f}) = \frac{P(\mathbf{e}) \cdot P(\mathbf{f}|\mathbf{e})}{P(\mathbf{f})} \tag{1}$$

[1]We follow the convention that the source language of the translation task is French, and the target language English.

[2]See the Appendix for a list of notations

the best English translation for a French sentence is:

$$\hat{\mathbf{e}} = \arg\max_{e} P(\mathbf{e})P(\mathbf{f}|\mathbf{e}) \tag{2}$$

The translation problem is thus reduced to a language task, to obtain $P(\mathbf{e})$, and a translation task, to obtain $P(\mathbf{f}|\mathbf{e})$[3].

The IBM models focus on the translation task only. To emphasize that the English sentence represents a 'collection' of concepts, the words of the sentence are called *cepts*. Each cept generates one or more words in the French translation. Conversely, each word in the French sentence is said to be *aligned* to a cept in the English sentence. To account for the possibility that French words have no counterpart in the English translation, one says that these words are aligned to the *empty cept*, e_0. Figure 2 shows a possible alignment between two sentences. In this case, the last three words of the French sentence are aligned to the same word in the English sentence. Note that the IBM model of alignment does not allow for the reverse, that is to say, a French word cannot be aligned with more than one English word. Generally, a group of words cannot be aligned to another group of words with the IBM models. This would in fact require a Phrase-Based approach.

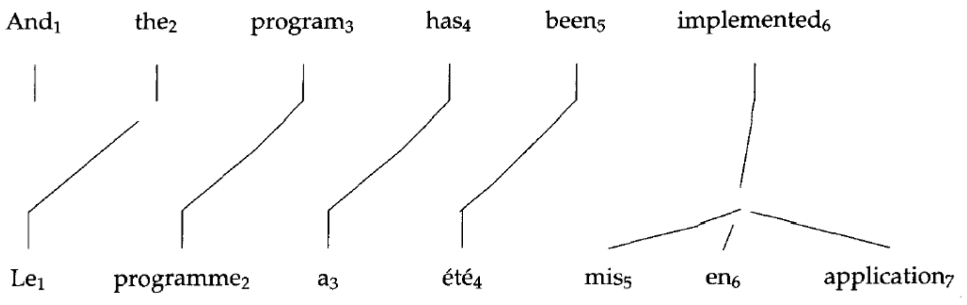

Figure 2: Alignment between a French and an English sentence, after [1]. In the IBM notation, this alignment is represented as [2, 3, 4, 5, 6, 6, 6].

Each IBM model builds on the previous one, each increasing the complexity of the alignment model. In all cases, the translation probability $P(\mathbf{f}|\mathbf{e})$ is seen as the sum on all alignments of the conditional probabilities $P(\mathbf{f}, \mathbf{a}|\mathbf{e})$, where $\mathbf{a}$ is an alignment between the French and the English sentences:

$$P(\mathbf{f}|\mathbf{e}) = \sum_{\mathbf{a}} P(\mathbf{f}, \mathbf{a}|\mathbf{e}) \tag{3}$$

[3]The translation model informs us on what sentences are good translations, while the language model ensures that these sentences are well-formed. By combining these models, we thus get better results than if we were to look directly for the sentence that maximises $P(e|f)$.

The conditional probability $P(\mathbf{f},\mathbf{a}|\mathbf{e})$ can itself be expressed as:

$$P(\mathbf{f},\mathbf{a}|\mathbf{e}) = P(m|\mathbf{e})\prod_{j=1}^{m} P(a_j|a_1^{j-1}, f_1^{j-1}, m, \mathbf{e})P(f_j|a_1^j, f_1^{j-1}, m, \mathbf{e}) \tag{4}$$

The following sections present the main assumptions taken by models 1 to 3, along with the principal equations and the algorithm proposed to compute $P(\mathbf{f}|\mathbf{e})$.

1.1 Model 1

Model 1 takes $P(m|\mathbf{e})$ to be a parameter ϵ independent of $\mathbf{e}$ and m. The term $P(a_j|a_1^{j-1}, f_1^{j-1}, m, \mathbf{e})$ is assumed to depend only on l and to be equal to $(l+1)^{-1}$, and $P(f_j|a_1^j, f_1^{j-1}, m, \mathbf{e})$ is assumed to depend only on f_j and e_{a_j}, and can thus be expressed as the conditional probability $t(f_j|e_{a_j})$. Equation 4 can therefore be rewritten as:

$$P(\mathbf{f},\mathbf{a}|\mathbf{e}) = \frac{\epsilon}{(l+1)^m}\prod_{j=1}^{m} t(f_j|e_{a_j}) \tag{5}$$

Importantly, these assumptions make it possible to compute $P(\mathbf{f}|\mathbf{e})$ efficiently, as Equation 3 can be rewritten as:

$$P(\mathbf{f}|\mathbf{e}) = \frac{\epsilon}{(l+1)^m}\prod_{j=1}^{m}\sum_{i=0}^{l} t(f_j|e_i) \tag{6}$$

The translation probabilities $t(f_j|e_i)$ can be computed using:

$$t(f|e) = \lambda_e^{-1}\sum_{s=1}^{S} c(f|e;\mathbf{f}^{(s)},\mathbf{e}^{(s)}) \tag{7}$$

where:

$$c(f|e;\mathbf{f},\mathbf{e}) = \frac{t(f|e)}{t(f|e_0)+...+t(f|e_l)}\sum_{j=1}^{m}\delta(f,f_j)\sum_{i=0}^{l}\delta(e,e_i) \tag{8}$$

and λ_e is a normalisation parameter ($\sum_f t(f|e) = 1$):

$$\lambda_e = \sum_f\sum_{s=1}^{S} c(f|e;\mathbf{f}^{(s)},\mathbf{e}^{(s)}) \tag{9}$$

The algorithm used to compute the translation probabilities $t(f_j|e_i)$ is the following:

1. Choose initial values for $t(f|e)$. As $P(\mathbf{f}|\mathbf{e})$ has a unique local maximum in this model, it is sufficient to take these initial values to be equal and different from zero.

2. Compute the counts $c(f|e;\mathbf{f}^{(S)},\mathbf{e}^{(S)})$ for each pair of sentences using Equation 8.
3. For each e appearing in the English sentences, compute λ_e using Equation 9, and $t(f|e)$ using Equation 7.
4. Repeat steps 2 and 3 until the values of $t(f|e)$ have converged to the desired degree.

1.2 Model 2

Model 2 makes the same assumptions as Model 1, except for the term $P(a_j|a_1^{j-1},f_1^{j-1},m,\mathbf{e})$, which is assumed to depend on j, a_j, m and l, and can thus be expressed as $a(a_j|j,m,l)$. Equation 4 can then be rewritten as:

$$P(\mathbf{f},\mathbf{a}|\mathbf{e}) = \epsilon \prod_{j=1}^{m} t(f_j|e_{a_j})a(a_j|j,m,l) \tag{10}$$

As in model 1, $P(\mathbf{f}|\mathbf{e})$ can be expressed simply as:

$$P(\mathbf{f}|\mathbf{e}) = \epsilon \prod_{j=1}^{m} \sum_{i=0}^{l} t(f_j|e_i)a(i|j,m,l) \tag{11}$$

The translation probabilities $t(f_j|e_i)$ can be computed as in Model 1, with the counts $c(f|e;\mathbf{f},\mathbf{e})$ being now expressed as:

$$c(f|e;\mathbf{f},\mathbf{e}) = \sum_{j=1}^{m}\sum_{i=0}^{l} \frac{t(f|e)a(i|j,m,l)\delta(f,f_j)\delta(e,e_i)}{t(f|e_0)a(0|j,m,l)+...+t(f|e_l)a(l|j,m,l)} \tag{12}$$

The alignment probabilities can be computed by:

$$a(i|j,m,l) = \mu_{jml}^{-1} \sum_{s=1}^{S} c(i|j,m,l;\mathbf{f},\mathbf{e}) \tag{13}$$

where:

$$c(i|j,m,l;\mathbf{f},\mathbf{e}) = \frac{t(f_j|e_i)a(i|j,m,l)}{t(f_j|e_0)a(0|j,m,l)+...+t(f_j|e_l)a(l|j,m,l)} \tag{14}$$

and μ_{jml} is a normalisation parameter ($\sum_i a(i|j,m,l)=1$):

$$\mu_{jml} = \sum_{i=0}^{l}\sum_{s=1}^{S} c(i|j,m,l;\mathbf{f},\mathbf{e}) \tag{15}$$

These parameters can be computed using the following algorithm:

1. The initial values for $t(f|e)$ and $a(a_j|j,m,l)$ are taken to be those of Model 1. The initial values for the latter are thus $(l+1)^{-1}$.
2. For each pair of sentences, compute $c(f|e;\mathbf{f},\mathbf{e})$ and $c(i|j,m,l;\mathbf{f},\mathbf{e})$ using Equation 12 and Equation 14, respectively.
3. For each e appearing in the English sentences, compute λ_e using Equation 9, and $t(f|e)$ using Equation 7. Besides, for each value taken by $\langle j,m,l\rangle$, compute μ_{jml} using Equation 15 and $a(i|j,m,l)$ using Equation 13.
4. Repeat steps 2 and 3 until the values of $t(f|e)$ and of $a(i|j,m,l)$ have converged to the desired degree.

1.3 Model 3

Model 3 proposes a more elaborate model of alignments, by introducing two new parameters, namely *distortions* and *fertilities*, beside the previously used translation probabilities. The fertility ϕ of a cept e in $\mathbf{e}$ is the number of words generated by this cept in $\mathbf{f}$. The distortion $d(j|i,m,l)$ represents the probability that a cept at position i in a sentence $\mathbf{e}$ of length l generates a word at position j in a sentence $\mathbf{f}$ of length m. This parameter thus models the 'corruption' of word order caused by translation. Note that different cepts are allowed to map to the same index in the target sentence. This means that the model is deficient, as it can assign a non-zero probability to strings that are actually impossible as they would have more than one word per position. This is the price paid for keeping the dependencies of probabilities limited and in that way the mathematics simple.

In Model 3, Equation 4 is not used anymore to compute $P(\mathbf{f},\mathbf{a}|\mathbf{e})$, instead this probability is written as:

$$P(\mathbf{f},\mathbf{a}|\mathbf{e}) = \begin{pmatrix} m-\phi_0 \\ \phi_0 \end{pmatrix} p_0^{m-2\phi_0} p_1^{\phi_0} \prod_{i=1}^{l} \phi_i! n(\phi_i|e_i) \times \prod_{j=1}^{m} t(f_j|e_{a_j}) d(j|a_j,m,l) \quad (16)$$

with p_0 and p_1 some pair of auxiliary parameters ($p_0+p_1=1$), ϕ_i the fertility of a word at index i, $n(\phi_i|e_i)$ the fertility probability ($\sum_\phi n(\phi|e)=1$) and $d(j|a_j,m,l)$ the distortion probability ($\sum_j d(j|i,m,l)=1$).

Equation 3 cannot be simplified anymore as it was in the previous models. Instead, a limited set of best alignments $\mathcal{S}$ is estimated by taking the Viterbi alignments from Model 2 and then doing a greedy search to improve it performing small (local) changes. The summation of formula 3 is then limited to the alignments of $\mathcal{S}$.

$$\mathcal{S} = \mathcal{N}(b^{\infty}(V(\mathbf{f}|\mathbf{e};2))) \bigcup \cup_{ij} \mathcal{N}(b^{\infty}_{i \leftarrow j}(V_{i \leftarrow j}(\mathbf{f}|\mathbf{e};2))) \tag{17}$$

where $V(\mathbf{f}|\mathbf{e};2)$ is the unconstrained Viterbi alignment found by Model 2, and $V_{i \leftarrow j}(\mathbf{f}|\mathbf{e};2)$ is the pegged Viterbi alignment for ij (i.e. the Viterbi alignment found for a fixed i,j pair) found by Model 2[4]. Followingly, $\mathcal{N}(b^{\infty}(V(\mathbf{f}|\mathbf{e};2)))$ and $\mathcal{N}(b^{\infty}_{i \leftarrow j}(V_{i \leftarrow j}(\mathbf{f}|\mathbf{e};2)))$ are the sets of best alignments that can be found starting from $V(\mathbf{f}|\mathbf{e};2)$ or $V_{i \leftarrow j}(\mathbf{f}|\mathbf{e};2)$ and iteratively taking the neighbouring alignments that maximise the alignment probability, until this probability cannot be improved anymore by local changes.
Alignments are called neigbouring if they can be reached by performing one of the following move or swap operations:

1. Move from non-empty cept to non-empty cept (j maps to $i' \neq 0$ instead of $i \neq 0$)
2. Move from non-empty to empty cept[5]
3. Move from empty cept to non-empty cept[6]
4. Swap between two non-empty cepts

Finding the best alignments is done by performing a hill climbing or greedy local search over alignment improvements, where at every iteration one chooses the best swap or move. Since these operations only manipulate a small amount of the factors that determine the alignment probability (Equation 16), it is possible and advantageous to take the old alignment probability and then multiply it by the factor of probability change that corresponds to performing the operation, as in Equation 18. Note that Equation 18 corresponds to a move from a non-empty to a non-empty cept. Other operations require an adaptation of this formula.

$$P(\mathbf{a}'|\mathbf{e},\mathbf{f}) = P(\mathbf{a}|\mathbf{e},\mathbf{f}) \frac{\phi_{i'}+1}{\phi_i} \frac{n(\phi_{i'}+1|e_{i'})}{n(\phi_{i'}|e_{i'})} \frac{n(\phi_i-1|e_i)}{n(\phi_i|e_i)} \frac{t(f_j|e_{i'})}{t(f_j|e_i)} \frac{d(j|i',m,l)}{d(j|i,m,l)} \tag{18}$$

The translation probabilities and distortion probabilities required for the first estimation of $\mathcal{S}$ are obtained from Model 2. The initial estimations for the fertility probabilities as given by [1] being quite complex, we decided instead to use the following probability distribution as an initial estimate, taking $\phi_{max} = 10$:

[4]These alignments are computed using Model 2, as Model 3 does not allow for computing them efficiently.

[5]This comes down to swapping an non-empty cept with the empty cept.

[6]id.

Table 1: Initial probability distribution for the fertilities

ϕ	0	1	2	3	4 to ϕ_{max}
$n(\phi\|e)$	0.2	0.65	0.1	0.04	$0.01/(\phi_{max}-3)$

Once a set of alignments is obtained, the computation of counts and probabilities for the different parameters can be obtained in the same manner as with the previous models. Equation 19 to Equation 23 give the equations for the counts, and Equation 24 to Equation 27 the equations for the probabilities.

$$\begin{aligned}
c(f|e;\mathbf{f},\mathbf{e}) &= \sum_{\mathbf{a}} P(\mathbf{a}|\mathbf{e},\mathbf{f}) \sum_{j=1}^{m} \delta(f,f_j)\delta(e,e_{a_j}) && (19)\\
c(j|i,m,l;\mathbf{f},\mathbf{e}) &= \sum_{\mathbf{a}} P(\mathbf{a}|\mathbf{e},\mathbf{f})\delta(i,a_j) && (20)\\
c(\phi|e;\mathbf{f},\mathbf{e}) &= \sum_{\mathbf{a}} P(\mathbf{a}|\mathbf{e},\mathbf{f}) \sum_{i=1}^{l} \delta(\phi,\phi_i)\delta(e,e_i) && (21)\\
c(0;\mathbf{f},\mathbf{e}) &= \sum_{\mathbf{a}} P(\mathbf{a}|\mathbf{e},\mathbf{f})(m-2\phi_0) && (22)\\
c(1;\mathbf{f},\mathbf{e}) &= \sum_{\mathbf{a}} P(\mathbf{a}|\mathbf{e},\mathbf{f})\phi_0 && (23)
\end{aligned}$$

$$\begin{aligned}
t(f|e) &= \lambda_e^{-1} \sum_{s=1}^{S} c(f|e;\mathbf{f}^{(s)},\mathbf{e}^{(s)}) && (24)\\
d(j|i,m,l) &= \mu_{iml}^{-1} \sum_{s=1}^{S} c(j|i,m,l;\mathbf{f}^{(s)},\mathbf{e}^{(s)}) && (25)\\
n(\phi|e) &= \nu_e^{-1} \sum_{s=1}^{S} c(\phi|e;\mathbf{f}^{(s)},\mathbf{e}^{(s)}) && (26)\\
p_k &= \xi^{-1} \sum_{s=1}^{S} c(k;\mathbf{f}^{(s)},\mathbf{e}^{(s)}) && (27)
\end{aligned}$$

The algorithm for Model 3 is thus the following:

1. Initialise the values of $t(f|e)$ to those of Model 2, adapt the alignment probabilities of Model 2 to initialise the distortion probabilities $d(j|i,m,l)$, use Table 1 to initialise the fertility probabilities, and initialise p_0 and p_1.

2. Compute a set of alignments using Equation 17 and Equation 18 (or a variant of the latter).

3. Update the counts for the parameters, for each pair of sentences, using Equation 19 to Equation 23.
4. Update the probabilities for the parameters using Equation 24 to Equation 27 using the counts computed in step 3.
5. Repeat step 2 to 4 until the parameters have converged to the desired degree.

2 Implementation

We implemented models 1 to 3 in java. The different classes, that we grouped into four packages, are briefly presented in section 2.1[7]. Section 2.2 contains general notes concerning our implementation.

2.1 Packages and classes

- **preprocessing**
  - *ConfigFile*: reads a file "configFile.txt" containing a number of parameters that can be specified by the user. These parameters are stored as fields in the class, and can hence be directly used by other classes.
  - *DataReader*: extracts sentences from a Europarl bilingual corpus and stores the sentences of each language into a list.
  - *SentenceAlignment*: reads two lists of sentences, one for each language, pairs these sentences and stores them in a list of *SentencePair* objects, and filters out pairs for which (one of) the sentences does not cover the 'n'[8] most frequent words, or pairs for which (one of) the sentences are longer than a number of words specifiable by the user.
- **general**
  - *Main*: the class containing the *main* method.
  - *Model1*: the class used to implement Model 1.
  - *Model2*: the class used to implement Model 2.
  - *Evaluation*: the class used to perform evaluation given manually aligned sentences.
- **model3**
  - *Model3*: the class used to implement Model 3.

[7]See the javadoc for the complete documentation
[8]Specifiable by the user

  - *Partition*: the class used to represent partitions.
  - *SAAP*: allows for the coupling of a *SentencePair* object with an alignment. This class is primarily used to implement alignment moves and swaps, and the corresponding probability estimations.

- **utilities**
  - *SentencePair*: allows for the representation and manipulation of a sentence pair.
  - *Model*: specifies an interface for the models.
  - *JML*: allows to represent a $\langle j, m, l\rangle$ triplet as used by Model 2.
  - *IML*: allows to represent an $\langle i, m, l\rangle$ triplet as used by Model 3.
  - *ProbabilityCell*: allows for the representation of a cell in the various look-up tables used for this project. The cell contains a 'probability' field, a 'counts' field and a 'delta' field used to evaluate probability convergence.
  - *LookUpTable*: this generic class allows a uniform representation of the various tables used in this project to store probabilities and counts (translation, alignments, distortion and fertilities). A LookUpTable object is represented as a two-layer HashMap $\langle E_1, \langle E_2, \text{ProbabilityCell}\rangle\rangle$. See also Table 2 and Figure 3. Note that the entries of these tables are not specified at initialisation, instead they are added to the tables when they are encountered in the data.

Table 2: LookUpTable instantiation types

	row parameter	row type E_1	column parameter	column type E_2
Translations	e	String	f	String
Alignments	$\langle j, m, l\rangle$	JML	i	Integer
Distortions	$\langle i, m, l\rangle$	IML	j	Integer
Fertilities	e	String	ϕ	Integer

2.2 Implementation notes

2.2.1 Corpus-specific implementation

We used the Europarl sentence-aligned corpus Dutch/English for testing. The corpus, obtainable from http://www.statmt.org/europarl/, is stored in two folders, 'en' and 'nl'. Each folder contains a number of text files, named

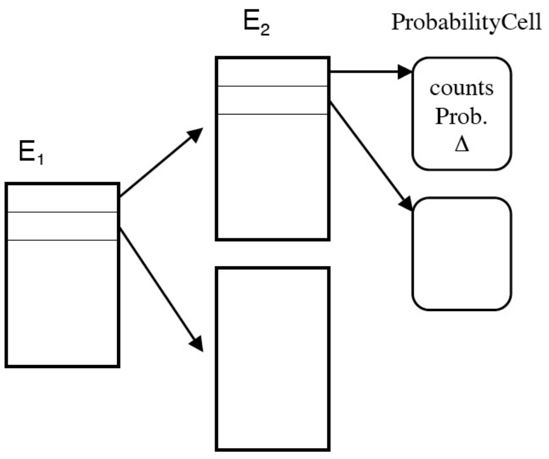

Figure 3: The LookUpTable structure

after the date their content was recorded. The language of this content is specified only by the folder in which it is kept. The *DataReader* class is taylored to read a corpus in this format, its methods would require adaptation should another corpus be used.

The *SentenceAlignment* class' principal method, *mergeSentences*, inputs two lists of already aligned sentences. We did not deal with alignment in this project.

2.2.2 Iterating over sentences

Early tests revealed that a naive implementation of the algorithms presented in section 1.1 to section 1.3 led to extremely long run times. We found it was much more efficient to iterate first over sentences, and then over parameters, rather than the other way around. For Model 1 for instance, instead of considering every possible $\langle e, f \rangle$ pair, and updating the corresponding entry in the ef-table for every sentence pair, we iterate over every sentence pair, and update the ef-table for the relevant $\langle e, f \rangle$ pairs.

2.2.3 Computational cost of Model 3

The complexity of Model 3 lies mainly in the computation of the different alignments. At each iteration, we need to consider $m \cdot (n+1)$ pegged Viterbi alignments (each of the m French words can be pegged to each of the $n+1$ English cepts). For each alignment, one can perform $m \cdot n$ moves and $m \cdot (m-1)/2$ swaps. Therefore, each iteration of the Model 3 algorithm requires $m^2 \cdot (n+1) \cdot (n+(m-1)/2)$ operations to find the best alignments[9]. This corresponds to a complexity in $O(K^4)$, where K is the length of the sentences (m and n) assuming those have the same order of length. Having made this observation, it is easy to see that the length of the sentences is a great factor in the computation time of model 3, and very long sentences cannot be aligned in realistic time. To limit the time required for generating the results, we therefore limited the maximum length of both the target and the source sentences to ten words.

2.2.4 Efficiency in generating initial alignments

An important efficiency gain in the implementation is obtained from the insight that pegged Viterbi alignments of Model 1 and Model 2 differ only from their unconstrained counterpart on the alignment of the word that is pegged. Therefore, such pegged alignments are efficiently generated from the original unconstrained Viterbi alignment by just changing the alignment of this one pegged word. Additionally by the same principle the alignment probabilities of such pegged alignments can be efficiently computed from the probability of the unconstrained Viterbi Alignment. This is done by taking this unconstrained probability and then multiplying by the factor of change of the probability, which can be determined using Equation 16. Note that this probability recomputation is performed by the *move* method in the *SAAP* class.

2.2.5 Machine Precision

The move and swap methods, which are at the core of the alignment probability computation of model 3, require a lot of multiplications and divisions of probability values. At some point in our testing we discovered a very subtle bug that allowed probabilities to grow infinitely, resulting from the fact that $(A/B) \cdot (B/A)$ can differ from 1 as a result of limited machine precision. We solved this problem by rewriting this kind of operations as a quotient of products $(A \cdot B)/(B \cdot A)$ instead.

2.2.6 Convergence criterion

We used a simple but robust method to test for convergence: We consider that a model has converged when all of its parameters have, and in turn, we consider that a parameter has converged when the probability of all its entries in the corresponding look-up table have changed by less than a certain value since the latest iteration. The difference between the former and the present probability value is stored for each entry as Δ. The convergence

[9]More operations can be required as finding a best alignment is itself an iterative process. I turns out however that the alignment probability converges in a small number of steps, so we can consider this contribution to complexity a constant.

test consists therefore in checking whether all Δ values have fallen below a threshold specified by the user. We used a threshold of 0.001.

3 Experimental Setup

We first tested our implementation on the toy corpus presented in section B of the Appendix. We then went on testing on different amounts of data from the Europarl corpus, varying the constraints we put on the input sentence pairs: in all cases we performed filtering on words by keeping only the sentence pairs were both sentences covered the 1000 most frequent words in the corresponding language, and in all but the first set-up, we added a constraint on sentence length, namely that both sentences in each sentence pair should be shorter than 10 words. Table 3 sums up the different set-ups used for testing on the Europarl corpus.

Table 3: Set-up summary for the tests on the Europarl corpus

	Set-Up 1	Set-Up 2	Set-Up 3
training set	Yes	Yes	Yes
begin date	96-04-15	96-04-15	96-04-15
end date	00-01-01	96-10-01	99-01-01
Nb sentences	7166	586	4302
test set	Yes	Yes	No
begin date	00-01-01	96-10-01	n.a.
end date	03-09-26	97-01-01	n.a.
Tested models	1, 2	1, 2, 3	1, 2, 3
Word filtering	Yes	Yes	Yes
Sentence filtering	No	Yes	Yes

The labels ‘training’ and ‘test’ sets are rather misleading as we are not performing supervised learning on the data. The distinction between the two sets was intended for evaluation: By extracting 20 sentences from the ‘training’ set, and 20 sentences from the ‘test’ set that do not appear in the first set, we were able to compare the performance of the models on seen and unseen sentences.

Set-Up 1 is the first set-up we used. We ran Models 1 and 2 on 7166 sentences, resulting from the filtering of sentences that covered the 1000 most frequent words in the Europarl data spanning from 96-04-15 to 00-01-01 (excluded). No limit on sentence length was specified (*no sentence filtering*).

Set-Up 2 uses a much smaller amount of data (586 sentences). Also, the length of the sentences was limited to 10. As a result, the sentences we used

for evaluation were much simpler than for Set-Up 1.
As we found that limiting sentence length considerably decreased run-time, we also ran the models on an increased amount of data, leading to Set-Up 3 (4302 sentences). We did not create a new set of evaluation sentences for this set-up, and we therefore did not use a test set.

4 Results and Analysis

We present in section 4.1 the results obtained from testing our implementation on the toy corpus. Results concerning the tests performed on the Europarl corpus are presented in the remainder of this section: The evaluation method is presented in section 4.2, the evaluation results in section 4.3, and an analysis of these results in section 4.4.

4.1 Toy corpus

Our toy corpus, which is given in section B, consists of a small number of very short, one-to-one aligned sentences and phrases, built from a minimal vocabulary. As such, it is not representative of natural language, but it allowed us to get some insights in the working of the different models. The small size of this corpus also allowed us to obtain readable results in a short time.

With an initial version of our toy corpus, that contained all but the four last pairs of sentences (separated from the rest by an empty line in the Appendix), all models found the alignment [0, 2, 3, 0, 5, 6] for the pair of sentences “a dog and a man sleep / een hond en een man slapen”. The alignment of ‘een’ with the empty cept can be explained in all cases by the preferred alignment of e_0 with the most frequent word in our toy corpus, to the effect that the translation probability is higher for ‘een’/e_0 than for ‘een’/‘a’.

After adding the four last sentences to our corpus, we found that Model 2 and Model 3 both correctly found the alignment [1, 2, 3, 4, 5, 6], while Model 1 showed no improvement.

Experimenting with these four sentences, by adding them one or two at the time, revealed the following points:

- The extra sentences give information that removes the ambiguity in the alignment of ‘a’/‘een’ as they contain this word only once. Both models tend to profit equally from this specific information.
- When adding only one sentence, the results were somewhat surprising, as Model 2 performed better than Model 3 in some cases, and worse

in others. The fact that our corpus is not completely balanced in the occurrences of words makes it possible that similar sentences can yet have different information values to the different models. Besides, as Model 3 only searches from the alignments found by Model 2, Model 3 is not guaranteed to return the best alignment its model prescribes (search error).

- Model 3 prefers to align multiple words to the same cept, which can happen when the distortion table indicates no preference. This can be explained by the higher probability assigned by Equation 16 to alignments where cepts have a higher fertility, as this probability is proportional to $\Pi\phi_i!$, where ϕ_i is the fertility of each cept. This reflects the generative process from which Equation 16 is derived: After fertilities are determined, a set of ϕ_i alignment positions for that cept can be chosen in $\phi_i!$ different ways.

4.2 Evaluation

We evaluated our implementation by randomly selecting 20 sentences from the training set and 20 other sentences from the test set for Set-up 1 and Set-up 2, using only the 20 sentences from the Set-up 2 training set to evaluate Set-up 3[10].
In all cases, we each manually aligned theses sentences, which resulted in three manual alignments per sentence. We then compared the alignment found by the Model under evaluation with our manual alignments, counting 1 point for each "correctly aligned" word, and then averaging over the sentence length and the number of manual alignments. Consider the following example:

```
hoe staat het daarmee
what is the situation here
manual alignment 1: [1, 4, 0, 5]
manual alignment 2: [1, 2, 4, 5]
manual alignment 3: [1, 2, 4, 5]
sentences13.txt[4, 4, 3, 4] score: 8.33
```

The alignment found in this case by Model 3 scores only 1 point ($a_2 = 4$) with manual alignment 1. This score is then divided by 12, as there are 3 manual alignments and the sentence length is 4. The final score is represented as a percentage.
Followingly, the score found for each sentence pair is averaged over the number of sentences, resulting in an overall score for the model under evaluation.

[10]We thus have four sets of 20 sentences. The sentences extracted from the training set of Set-Up 1 are presented in section C of the Appendix, and the sentences extracted from the training set of Set-Up 2, which are used to evaluate Set-Up 2 and Set-Up 3, are presented in section D

4.3 Results

The results of the evaluation of the models for our different set-ups are presented in Table 4, with a reminder of the main set-up characteristics. As explained above, Set-up 1 uses a different set of evaluation sentences than Set-up 2 and 3, as it does not filter sentences for length whereas the latter set-ups do. Note that for Model 2 in Set-up 1, only 200 iterations

Table 4: Evaluation results

	Set-Up 1	Set-Up 2	Set-Up 3
Nb sentences	7166	586	4302
Max. sentence length	n.a.	10	10
Model 1:			
training set	74.85	68.78	77.85
test set	73.43	67.07	n.a.
Model 2:			
training set	75.56	65.31	78.93
test set	70.0	65.52	n.a.
Model 3:			
training set	n.a.	66.11	80.79
test set	n.a.	58.21	n.a.

were performed, as running with an unlimited sentence length made each iteration quite slow. We also did not wait for Model 3 to converge, letting it run only 100 iterations in each case.

4.4 Analysis

One can see from Table 4 that performance improves with the model complexity, provided there is enough data to get a good approximation of the involved parameters. With a limited amount of data as in Set-up 2, the performance is lower for all models, but the more complex models 2 and 3 'suffer' more than Model 1, which is more robust as it only considers translation parameters.

As was expected, the different models tend to perform less well on unseen sentences, showing that the parameter estimations improve with more data. One should remark here that as our data consists only of sentences covering the 1000 most frequent words, the risk of encountering an unknown word in an unseen sentence was close to zero. Similarly, by limiting the sentence length to 10 in Set-ups 2 and 3, we decreased the risk of encountering a sentence of unknown length, which would have prevented Model 2 or Model

3 of finding certain alignments, or distortions respectively. Retrospectively, it is thus a matter of chance that Model 2 was able to find alignments for all the evaluation sentences of Set-Up 1.

Finally, one can note that the models performed better in Set-Up 3 as in Set-Up 1, although they dealt with less data. This has to do with the limitation on sentence length, which has a twofold effect. First, if all the sentences in the data have a limited length, the parameter estimation for these sentences is better than if part of the sentences are below this length, and part of them above, as relatively less data is then available for the corresponding lengths. Secondly, as sentence length decreases, the a priori chance that an alignment is correct increases, so better results can be expected for shorter sentences. As mentioned in section 2.2.3, limiting sentence length also limits the complexity of Model 3, resulting in shorter run times. Nevertheless, one cannot limit sentence length too strongly, as this is not very adequate for natural language. The evaluation sentences extracted from the training sets of Set-up 1 and Set-ups 2/3 are presented in the Appendix, giving an idea of the influence of sentence length limitation on a corpus.

5 Discussion and Conclusion

We implemented Model 1-3 of the IBM models in Java, and were able to get good results, as the alignment accuracy found by all models is in the range of 70-80%. Furthermore we could see that the more complex models indeed have the potential to outperform the more simple ones, provided there is enough data. We used rather strict restrictions on our data (limited amount of words covered and limited sentence length) to get results in a reasonable time, but our code would be able to handle looser, and more natural, assumptions.

Our implementation is now quite fast for Model 1, which takes about 300ms per iteration for Set-Up 3, and for Model 2 (about 1.3 seconds per iteration), but it might still be improved for Model 3 (around 150s per iteration). The inherent complexity of Model 3 and in particular of the selection of alignments, limits the room for improvement, but a number of ideas might still prove useful.
One idea would be to use integers to represent words: Letting the algorithm compare integers instead of strings could significantly increase the performance, at the cost of a loss of readability of the code.
Subsequently, another improvement might consist in using bi-dimensional arrays instead of two-layer HashMaps to store the different parameters: With limiting assumptions on the length of sentences and the number of

words, the resulting increase in space use would still be manageable, while the storage and lookup time would decrease from logarithmic to constant. Another improvement would be to reorder the generation of swaps and moves to enable the storage of sub-results that can be shared between different local changes of an alignment, rather than recomputing them.

Interesting future work would be to implement the other components required for translation, namely the language model and the search method. While the language model is readily available, the search method is a project in itself, because of the size of the search space. Hereby, a nice insight is that local change methods similar to the swap and move methods of model 3 are also used in recomputing the alignment probabilities in greedy, local search methods for the best translation ([2], [3]).

A Notations

e	English word
f	French word
$\mathbf{e}$	English sentence
$\mathbf{f}$	French sentence
l	length of English sentence
m	length of French sentence
i	word index in English sentence
j	word index in French sentence
e_0	the empty cept
$\mathbf{a}$	alignment between a French and an English sentence
a_j	the index of the English word that is aligned to the French word at index j

B Toy corpus

The man sleeps .	De man slaapt .
The woman sleeps .	De vrouw slaapt .
The dog sleeps .	De hond slaapt .
A man sleeps .	Een man slaapt .
A dog sleeps .	Een hond slaapt .
A woman sleeps .	Een vrouw slaapt .
A man .	Een man .
A dog .	Een hond .
A woman .	Een vrouw .
A .	Een .
Woman .	Vrouw .
Man .	Man .
Dog .	Hond .
A man and a dog .	Een man en een hond .
A man and a woman .	Een man en een vrouw .
A dog and a woman .	Een hond en een vrouw .
A dog and a man .	Een hond en een man .
A woman and a man .	Een vrouw en een man .
A man and a woman sleep .	Een man en een vrouw slapen .
A dog and a man sleep .	Een hond en een man slapen .
A dog and a woman sleep .	Een hond en een vrouw slapen .
A dog and the man sleep .	Een hond en de man slapen .
The dog and a woman sleep .	De hond en een vrouw slapen .
A dog and the woman sleep .	Een hond en de vrouw slapen .
The dog and a man sleep .	De hond en een man slapen .

C Evaluation sentences for Set-up 1

```
dat kan geen goede zaak zijn
this cannot be right

maar zij hebben niets in die richting gedaan
but they did nothing about it

het debat is gesloten
the debate is closed

de lidstaten en de commissie zijn het er echter over eens dat de
samenwerking al per 1 januari 1999 moet beginnen
however the member states and the commission have both now agreed
that the cooperation shall start on 1 january 1999

ik ben blij met het antwoord van de commissaris
i welcome the commissioner s reply

het parlement neemt de resolutie aan
parliament adopted the resolution

wat kunnen we doen om te helpen
what can we do to help

het parlement neemt de resolutie aan
parliament adopted the motion for a resolution

wij moeten hen daarbij steunen
we must support them in this respect

dit feit is niet nieuw
this is not a new situation

wat wil onze collega
what does the honourable member want

daarom heb ik tegen het verslag gestemd
on the basis of the above i voted against this report

het probleem is heel ernstig
the problem is a very serious one

mijn vraag gaat over dit politieke probleem en daarom heb ik haar
```

ook gesteld
my question referred to this political issue and that is why i asked it

ook zal zij dan een besluit moeten nemen over de maatregelen waarmee
zij deze doelstellingen wil bereiken
at that time the european union will also have to decide on future measures
to be taken in order to achieve these objectives

mevrouw de voorzitter dit is een goed verslag
madam president this is a good report

ik wel
i certainly have done

ik denk dat wij hier allemaal achter kunnen staan
that is something i believe we can all support

daar kan dus nu ook niet over gestemd worden
so this cannot be put to the vote

dat kan ik in antwoord op uw vraag wel stellen
that would be my response to your question

D Evaluation sentences for Set-up 2/3

dat is niet het geval
this is not the case

het debat is gesloten
the debate is closed

u doet niets
you do nothing

dit is echter een bijzonder belangrijk verslag
however this is an extremely important report

het is altijd al zo geweest
it always has been

ik heb twee vragen
i have two questions

dit is dus niet genoeg
so this is not enough

ik vind het goed wat de commissie heeft voorgesteld
i think the commission has made good proposals

dat is onze wens
that is our demand

het probleem is heel ernstig
the problem is a very serious one

wat zou u dan gedaan hebben
what would you have done

het parlement neemt de resolutie aan
parliament adopted the resolution

dit is het begin van het ierse voorzitterschap
this is the start of the irish presidency

dat is de waarde ervan
that is its value

dat moet duidelijk zijn
this must be quite clear

dit moet gezegd worden
this is something which has to be said

hoe staat het daarmee
what is the situation here

ze hebben hun werk goed gedaan
they have done well

dat is een zeer belangrijk punt
that is a very important point

ik zou daarover het volgende willen zeggen
let me make two things completely clear

References

[1] Brown et. Al. (1993): The mathematics of statistical machine translation: parameter estimation. Computational Linguistics Vol. 19, No. 2

[2] Germann, U. (2003) : Greedy decoding for statistical machine translation in almost linear time. In Proceedings of the HLT-NAACL-03. Edmonton, Alberta, Canada.

[3] Germann, U. et al (2001) : Fast decoding and optimal decoding for machine translation. In Proc. of the 39th Annual Meeting of the Association for Computational Linguistics (ACL). Toulouse, France.

[4] The Europarl parallel corpus:
http://www.statmt.org/europarl/