GREAT: a finite-state machine translation toolkit implementing a Grammatical Inference Approach for Transducer Inference (GIATI)

Jorge González and **Francisco Casacuberta**
Departamento de Sistemas Informáticos y Computación
Instituto Tecnológico de Informática
Universidad Politécnica de Valencia
{jgonzalez,fcn}@dsic.upv.es

Abstract

GREAT is a finite-state toolkit which is devoted to Machine Translation and that learns structured models from bilingual data. The training procedure is based on grammatical inference techniques to obtain stochastic transducers that model both the structure of the languages and the relationship between them. The inference of grammars from natural language causes the models to become larger when a less restrictive task is involved; even more if a bilingual modelling is being considered. GREAT has been successful to implement the GIATI learning methodology, using different scalability issues to be able to deal with corpora of high volume of data. This is reported with experiments on the EuroParl corpus, which is a state-of-the-art task in Statistical Machine Translation.

1 Introduction

Over the last years, grammatical inference techniques have not been widely employed in the machine translation area. Nevertheless, it is not unknown that researchers are trying to include some structured information into their models in order to capture the grammatical regularities that there are in languages together with their own relationship.

GIATI (Casacuberta, 2000; Casacuberta et al., 2005) is a grammatical inference methodology to infer stochastic transducers in a bilingual modelling approach for statistical machine translation.

From a statistical point of view, the translation problem can be stated as follows: given a source sentence $\mathbf{s} = \mathbf{s}_1 \dots \mathbf{s}_J$, the goal is to find a target sentence $\hat{\mathbf{t}} = \mathbf{t}_1 \dots \mathbf{t}_{\hat{I}}$, among all possible target strings $\mathbf{t}$, that maximises the posterior probability:

$$\hat{\mathbf{t}} = \underset{\mathbf{t}}{\operatorname{argmax}} \Pr(\mathbf{t}|\mathbf{s}) \qquad (1)$$

The conditional probability $\Pr(\mathbf{t}|\mathbf{s})$ can be replaced by a joint probability distribution $\Pr(\mathbf{s},\mathbf{t})$ which is modelled by a stochastic transducer being inferred through the GIATI methodology (Casacuberta et al., 2004; Casacuberta and Vidal, 2004):

$$\hat{\mathbf{t}} = \underset{\mathbf{t}}{\operatorname{argmax}} \Pr(\mathbf{s},\mathbf{t}) \qquad (2)$$

This paper describes GREAT, a software package for bilingual modelling from parallel corpus.

GREAT is a finite-state toolkit which was born to overcome the computational problems that previous implementations of GIATI (Picó, 2005) had in practice when huge amounts of data were used. Even more, GREAT is the result of a very meticulous study of GIATI models, which improves the treatment of smoothing transitions in decoding time, and that also reduces the required time to translate an input sentence by means of an analysis that will depend on the granularity of the symbols.

Experiments for a state-of-the-art, voluminous translation task, such as the EuroParl, are reported. In (González and Casacuberta, 2007), the so called phrase-based finite-state transducers were concluded to be a better modelling option for this task than the ones that derive from a word-based approach. That is why the experiments here are exclusively related to this particular kind of GIATI-based transducers.

The structure of this work is as follows: first, section 2 is devoted to describe the training procedure, which is in turn divided into several lines, for instance, the finite-state GIATI-based models are defined and their corresponding grammatical inference methods are described, including the techniques to deal with tasks of high volume of data; then, section 3 is related to the decodification process, which includes an improved smoothing behaviour and an analysis algorithm that performs according to the granularity of the bilingual symbols in the models; to continue, section 4 deals

Proceedings of the EACL 2009 Workshop on Computational Linguistic Aspects of Grammatical Inference, pages 24–32,
Athens, Greece, 30 March 2009.

with an exhaustive report on experiments; and finally, the conclusions are stated in the last section.

2 Finite state models

A stochastic finite-state automaton $\mathcal{A}$ is a tuple (Γ, Q, i, f, P), where Γ is an alphabet of symbols, Q is a finite set of states, functions $i : Q \to [0, 1]$ and $f : Q \to [0, 1]$ refer to the probability of each state to be, respectively, initial and final, and parcial function $P : Q \times \{\Gamma \cup \varepsilon\} \times Q \to [0, 1]$ defines a set of transitions between pairs of states in such a way that each transition is labelled with a symbol from Γ (or the empty string ε), and is assigned a probability. Moreover, functions i, f, and P have to respect the *consistency* property in order to define a distribution of probabilities on the free monoid. Consistent probability distributions can be obtained by requiring a series of local constraints which are similar to the ones for stochastic regular grammars (Vidal et al., 2005):

- $\sum_{q \in Q} i(q) = 1$
- $\forall q \in Q : \sum_{\gamma \in \{\Gamma \cup \varepsilon\}, q' \in Q} P(q, \gamma, q') + f(q) = 1$

A stochastic finite-state transducer is defined similarly to a stochastic finite-state automaton, with the difference that transitions between states are labelled with pairs of symbols that belong to two different (input and output) alphabets, that is, $(\Sigma \cup \varepsilon) \times (\Delta \cup \varepsilon)$. Then, given some input and output strings, **s** and **t**, a stochastic finite-state transducer is able to associate them a joint probability $\Pr(\mathbf{s}, \mathbf{t})$. An example of a stochastic finite-state transducer can be observed in Figure 1.

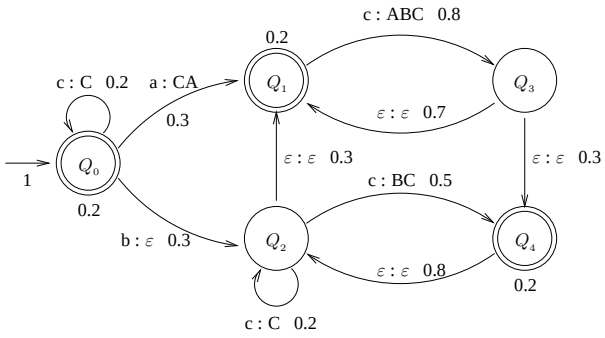

Figure 1: A stochastic finite-state transducer

2.1 Inference of stochastic transducers

The GIATI methodology (Casacuberta et al., 2005) has been revealed as an interesting approach to infer stochastic finite-state transducers through the modelling of languages. Rather than learning translations, GIATI first converts every pair of parallel sentences in the training corpus into a corresponding extended-symbol string in order to, straight afterwards, infer a language model from.

More concretely, given a parallel corpus consisting of a finite sample C of string pairs: first, each training pair $(\bar{x}, \bar{y}) \in \Sigma^\star \times \Delta^\star$ is transformed into a string $\bar{z} \in \Gamma^\star$ from an extended alphabet, yielding a string corpus S; then, a stochastic finite-state automaton $\mathcal{A}$ is inferred from S; finally, transition labels in $\mathcal{A}$ are turned back into pairs of strings of source/target symbols in $\Sigma^\star \times \Delta^\star$, thus converting the automaton $\mathcal{A}$ into a transducer $\mathcal{T}$.

The first transformation is modelled by some labelling function $\mathcal{L} : \Sigma^\star \times \Delta^\star \to \Gamma^\star$, while the last transformation is defined by an inverse labelling function $\Lambda(\cdot)$, such that $\Lambda(\mathcal{L}(C)) = C$. Building a corpus of extended symbols from the original bilingual corpus allows for the use of many useful algorithms for learning stochastic finite-state automata (or equivalent models) that have been proposed in the literature on grammatical inference.

2.2 Phrase-based n-gram transducers

Phrase-based n-gram transducers represent an interesting application of the GIATI methodology, where the extended symbols are actually bilingual phrase pairs, and n-gram models are employed as language models (González et al., 2008). Figure 2 shows a general scheme for the representation of n-grams through stochastic finite state automata.

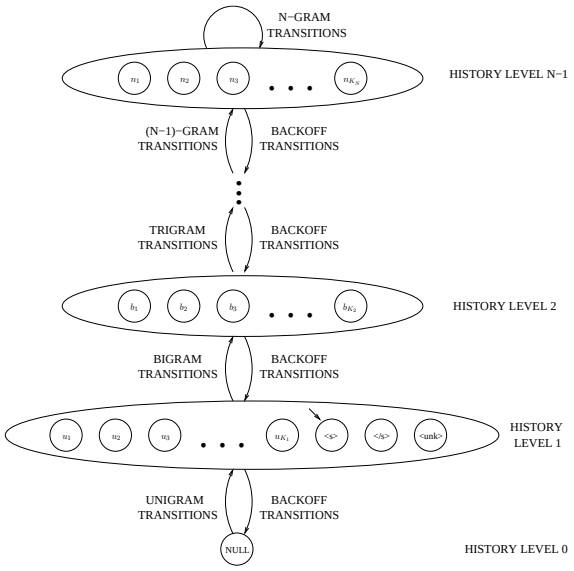

Figure 2: A finite-state n-gram model

The states in the model refer to all the n-gram histories that have been seen in the string corpus S in training time. Consuming transitions jump from states in a determined layer to the one immediately above, increasing the history level. Once the top level has been reached, n-gram transitions allow for movements inside this layer, from state to state, updating the history to the last $n-1$ seen events.

Given that an n-gram event $\Gamma_{n-1}\Gamma_{n-2}\ldots\Gamma_2\Gamma_1\Gamma_0$ is statistically stated as $\Pr(\Gamma_0|\Gamma_{n-1}\Gamma_{n-2}\ldots\Gamma_2\Gamma_1)$, then it is appropriately represented as a finite state transition between their corresponding up-to-date histories, which are associated to some states (see Figure 3).

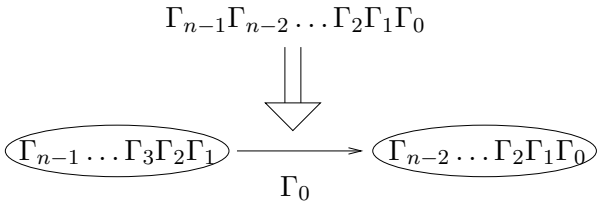

Figure 3: Finite-state representation of n-grams

Therefore, transitions are labelled with a symbol from Γ and every extended symbol in Γ is a translation pair coming from a phrase-based dictionary which is inferred from the parallel corpus.

Nevertheless, backoff transitions to lower history levels are taken for smoothing purposes. If the lowest level is reached and no transition has been found for next word $\mathbf{s}_j$, then a transition to the <unk> state is fired, thus considering $\mathbf{s}_j$ as a non-starting word for any bilingual phrase in the model. There is only 1 initial state, which is denoted as <s>, and it is placed at the 1st history level.

The inverse labelling function is applied over the automaton transitions as in Figure 4, obtaining a single transducer (Casacuberta and Vidal, 2004).

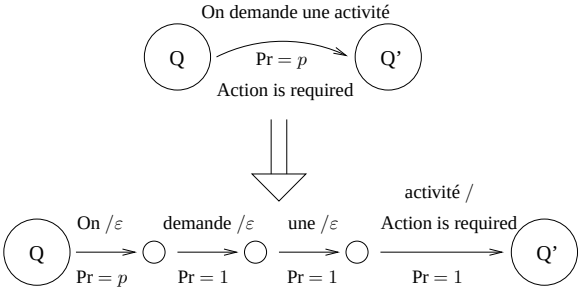

Figure 4: Phrase-based inverse labelling function

Intermediate states are artificially created since they do not belong to the original automaton model. As a result, they are non-final states, with only one incoming and one outcoming edge each.

2.3 Transducer pruning via n-gram events

GREAT implements this pruning technique, which is inspired by some other statistical machine translation decoders that usually filter their phrase-based translation dictionaries by means of the words in the test set sentences (Koehn et al., 2007).

As already seen in Figure 3, any n-gram event is represented as a transition between their corresponding historical states. In order to be able to navigate through this transition, the analysis must have reached the $\Gamma_{n-1}\ldots\Gamma_3\Gamma_2\Gamma_1$ state and the remaining input must fit the source elements of Γ_0. In other words, the full source sequence from the n-gram event $\Gamma_{n-1}\ldots\Gamma_3\Gamma_2\Gamma_1\Gamma_0$ has to be present in the test set. Otherwise, its corresponding transition will not be able to be employed during the analysis of the test set sentences. As a result, n-gram events that are not in the test set can skip their transition generation, since they will not be affected during decoding time, thus reducing the size of the model. If there is also a backoff probability that is associated to the same n-gram event, its corresponding transition generation can be skipped too, since its source state will never be reached, as it is the state which represents the n-gram event. Nevertheless, since trained extended-symbol n-gram events would typically include more than n source words, the verification of their presence or their absence inside the test set would imply hashing all the test-set word sequences, which is rather impractical. Instead, a window size is used to hash the words in the test set, then the trained n-gram events are scanned on their source sequence using this window size to check if they might be skipped or not. It should be clear that the bigger the window size is, the more n-gram rejections there will be, therefore the transducer will be smaller. However, the translation results will not be affected as these disappearing transitions are unreachable using that test set. As the window size increases, the resulting filtered transducer is closer to the minimum transducer that reflects the test set sentences.

3 Finite state decoding

Equation 2 expresses the MT problem in terms of a finite state model that is able to compute the ex-

pression $\Pr(\mathbf{s}, \mathbf{t})$. Given that only the input sentence is known, the model has to be parsed, taking into account all possible **t** that are compatible with **s**. The best output hypothesis $\hat{\mathbf{t}}$ would be that one which corresponds to a path through the transduction model that, with the highest probability, accepts the input sequence as part of the input language of the transducer.

Although the navigation through the model is constrained by the input sentence, the search space can be extremely large. As a consequence, only the most scored partial hypotheses are being considered as possible candidates to become the solution. This search process is very efficiently carried out by a beam-search approach of the well known Viterbi algorithm (Jelinek, 1998), whose temporal asymptotic cost is $\Theta(J \cdot |Q| \cdot M)$, where M is the average number of incoming transitions per state.

3.1 Parsing strategies: from words to phrases

The trellis structure that is commonly employed for the analysis of an input sentence through a stochastic finite state transducer has a variable size that depends on the beam factor in a dynamic beam-search strategy. That way, only those nodes scoring at a predefined threshold from the best one in every stage will be considered for the next stage.

A word-based parsing strategy would start with the initial state <s>, looking for the best transitions that are compatible with the first word $\mathbf{s}_1$. The corresponding target states are then placed into the output structure, which will be used for the analysis of the second word $\mathbf{s}_2$. Iteratively, every state in the structure is scanned in order to get the input labels that match the current analysis word $\mathbf{s}_i$, and then to build an output structure with the best scored partial paths. Finally, the states that result from the last analysis step are then rescored by their corresponding final probabilities.

This is the standard algorithm for parsing a source sentence through an non-deterministic stochastic finite state transducer. Nevertheless, it may not be the most appropriate one when dealing with this type of phrase-based n-gram transducers.

As it must be observed in Figure 4, a set of consecutive transitions represent only one phrase translation probability after a given history. In fact, the path from Q to Q' should only be followed if the remaining input sentence, which has not been analysed yet, begins with the full input sequence *On demande une activité*. Otherwise, it should not be taken into account. However, as far as the words in the test sentence are compatible with the corresponding transitions, and according to the phrase score, this (word) synchronous parsing algorithm may store these intermediate states into the trellis structure, even if the full path will not be accomplished in the end. As a consequence, these entries will be using a valuable position inside the trellis structure to an idle result. This will be not only a waste of time, but also a distortion on the best score per stage, reducing the effective power of the beam parameter during the decoding. Some other better analysis options may be rejected because of their a-priori lower score. Therefore, this decoding algorithm can lead the system to a worse translation result. Alternatively, the beam factor can be increased in order to be large enough to store the successful paths, thus more time will be required for the decoding of any input sentence.

On the other hand, a phrase-based analysis strategy would never include intermediate states inside a trellis structure. Instead, these artificial states are tried to be parsed through until an original state is being reached, i.e. Q' in Figure 4. Word-based and phrase-based analysis are conceptually compared in Figure 5, by means of their respective edge generation on the trellis structure.

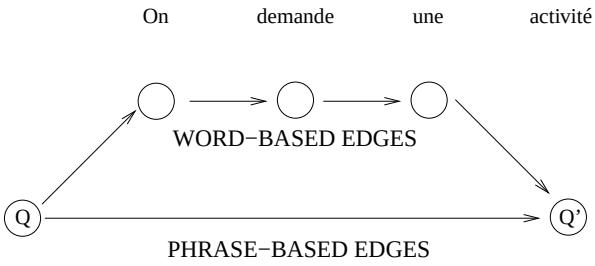

Figure 5: Word-based and phrase-based analysis

However, in order to be able to use a scrolled two-stage implementation of a Viterbi phrase-based analysis, the target states, which may be positioned at several stages of distance from the current one, are directly advanced to the next one. Therefore, the nodes in the trellis must be stored together with their corresponding last input position that was parsed. In the same manner, states in the structure are only scanned if their position indicator is lower than the current analysis word. Otherwise, they have already taken it into account so they are directly transfered to the next stage. The algorithm remains synchronous with the words in the input sentence, however, on this

particular occasion, states in the i-th step of analysis are guaranteed to have parsed **at least** until the i-th word, but maybe they have gone further. Figure 6 is a graphical diagram about this concept.

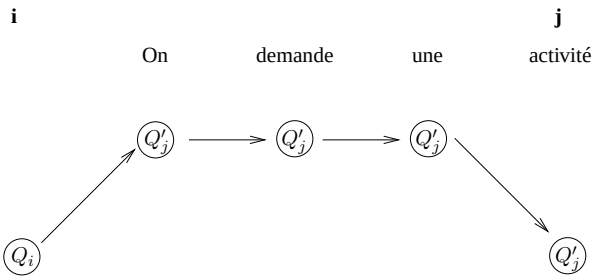

Figure 6: A phrase-based analysis implementation

Moreover, all the states that are being stored in the successive stages, that is, the ones from the original topology of the finite-state representation of the n-gram model, are also guaranteed to lead to a final state in the model, because if they are not final states themselves, then there will always be a successful path towards a final state.

GREAT incorporates an analysis strategy that depends on the granularity of the bilingual symbols themselves so that a phrase-based decoding is applied when a phrase-based transducer is used.

3.2 Backoff smoothing

Two smoothing criteria have been explored in order to parse the input through the GIATI model. First, a standard backoff behaviour, where backoff transitions are taken as failure transitions, was implemented. There, backoff transitions are only followed if there is not any other successful path that has been compatible with the remaining input.

However, GREAT also includes another more refined smoothing behaviour, to be applied over the same bilingual n-gram transducers, where smoothing edges are interpreted in a different way.

GREAT suggests to apply the backoff criterion according to its definition in the grammatical inference method which incorporated it into the language model being learnt and that will be represented as a stochastic finite-state automaton. In other words, from the n-gram point of view, backoff weights (or finite-state transitions) should only be employed if no transitions are found in the n-gram automaton for a current **bilingual** symbol. Nevertheless, input words in translation applications do not belong to those bilingual languages. Instead, input sequences have to be analysed in such a way as if they could be internally representing any possible bilingual symbol from the extended vocabulary that matches their source sides. That way, bilingual symbols are considered to be a sort of input, so the backoff smoothing criterion is then applied to each compatible, bilingual symbol.

For phrase-based transducers, it means that for a successful transition $(\bar{x}, \bar{y})$, there is no need to go backoff and find other paths consuming that bilingual symbol, but we must try backoff transitions to look for any other successful transition $(\bar{x}', \bar{y}')$, which is also compatible with the current position.

Conceptually, this procedure would be as if the input sentence, rather than a source string, was actually composed of a left-to-right bilingual graph, being built from the expansion of every input word into their compatible, bilingual symbols as in a category-based approach. Phrase-based bilingual symbols would be graphically represented as a sort of skip transitions inside this bilingual input graph.

This new interpretation about the backoff smoothing weights on bilingual n-gram models, which is not a priori a trivial feature to be included, is easily implemented for stochastic transducers by considering backoff transitions as ε/ε transitions and keeping track of a dynamic list of forbidden states every time a backoff transition is taken.

An outline about the management of state activeness, which is integrated into the parsing algorithm, is shown below:

ALGORITHM

```
for Q in {states to explore}
  for Q-Q' in {transitions}  (a)
    if Q' is active
      [...]
      set Q' to inactive
  if Q is not NULL
    if Q not in the top level
      for Q' in {inactive states}
        set Q'  to active
        Q'' := backoff(Q')
        set Q'' to inactive
    Q := backoff(Q)
    GoTo (a)
  else
    [...]
    for Q' in {inactive states}
      set Q' to active
[...]
```

END ALGORITHM

The algorithm will try to translate several consecutive input words as a whole phrase, always allowing a backoff transition in order to cover all the compatible phrases in the model, not only the ones which have been seen after a given history, but after all its suffixes as well. A dynamic list of forbidden states will take care to accomplish an exploration constraint that has to be included into the parsing algorithm: a path between two states Q and Q' has necessarily to be traced through the minimum number of backoff transitions; any other Q-Q' or Q-Q" paths, where Q" is the destination of a Q'-Q" backoff path, should be ignored. This constraint will cause that only one transition per bilingual symbol will be followed, and that it will be the highest in the hierarchy of history levels. Figure 7 shows a parsing example over a finite-state representation of a smoothed bigram model.

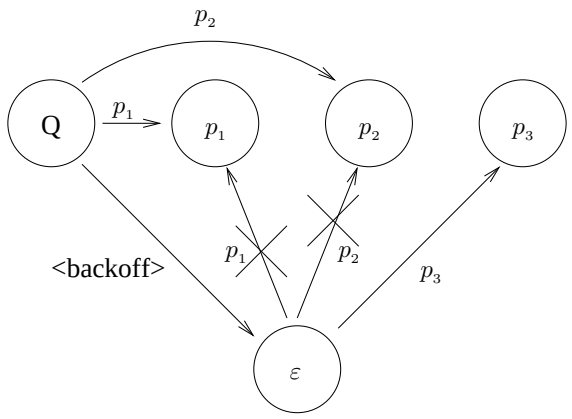

Figure 7: *Compatible edges for a bigram model*.

Given a reaching state Q, let us assume that the transitions that correspond to certain bilingual phrase pairs p_1, p_2 and p_3 are all compatible with the remaining input. However, the bigram (Q, p_3) did not occur throughout the training corpus, therefore there is no a direct transition from Q to p_3. A backoff transition enables the access to p_3 because the bigram (Q, p_3) turns into a unigram event that is actually inside the model. Unigram transitions to p_1 and p_2 must be ignored because their corresponding bigram events were successfully found one level above.

4 Experiments

GREAT has been successfully employed to work with the French-English EuroParl corpus, that is, the benchmark corpus of the NAACL 2006 shared task of the Workshop on Machine Translation of the Association for Computational Linguistics. The corpus characteristics can be seen in Table 1.

Table 1: *Characteristics of the Fr-En EuroParl.*

		French	**English**
Training	Sentences	688031	
	Run. words	15.6 M	13.8 M
	Vocabulary	80348	61626
Dev-Test	Sentences	2000	
	Run. words	66200	57951

The EuroParl corpus is built on the proceedings of the European Parliament, which are published on its web and are freely available. Because of its nature, this corpus has a large variability and complexity, since the translations into the different official languages are performed by groups of human translators. The fact that not all translators agree in their translation criteria implies that a given source sentence can be translated in various different ways throughout the corpus.

Since the proceedings are not available in every language as a whole, a different subset of the corpus is extracted for every different language pair, thus evolving into somewhat a different corpus for each pair of languages.

4.1 System evaluation

We evaluated the performance of our methods by using the following evaluation measures:

BLEU *(Bilingual Evaluation Understudy) score*: This indicator computes the precision of unigrams, bigrams, trigrams, and tetragrams with respect to a set of reference translations, with a penalty for too short sentences (Papineni et al., 2001). BLEU measures accuracy, not error rate.

WER *(Word Error Rate)*: The WER criterion calculates the minimum number of editions (substitutions, insertions or deletions) that are needed to convert the system hypothesis into the sentence considered ground truth. Because of its nature, this measure is very pessimistic.

Time. It refers to the average time (in milliseconds) to translate one word from the test corpus, without considering loading times.

4.2 Results

A set of experimental results were obtained in order to assess the impact of the proposed techniques in the work with phrase-based n-gram transducers.

By assuming an unconstrained parsing, that is, the successive trellis structure is large enough to store all the states that are compatible within the analysis of a source sentence, the results are not very sensitive to the n-gram degree, just showing that bigrams are powerful enough for this corpus. However, apart from this, Table 2 is also showing a significant better performance for the second, more refined behaviour for the backoff transitions.

Table 2: *Results for the two smoothing criteria.*

	n				
Backoff	1	2	3	4	5
baseline					
BLEU	26.8	26.3	25.8	25.7	25.7
WER	62.3	63.9	64.5	64.5	64.5
GREAT					
BLEU	26.8	**28.0**	27.9	27.9	27.9
WER	62.3	**61.9**	62.0	62.0	62.0

From now on, the algorithms will be tested on the phrase-based *bigram* transducer, being built according to the GIATI method, where backoff is employed as ε/ε transitions with forbidden states.

In these conditions, the results, following a word-based and a phrase-based decoding strategy, which are in function of the dynamic beam factor, can be analysed in Tables 3 and 4.

Table 3: *Results for a word-based analysis.*

beam	Time (ms)	BLEU	WER
1.00	0.1	0.4	94.6
1.02	0.3	12.8	81.9
1.05	5.2	20.0	74.0
1.10	30.0	24.9	68.2
1.25	99.0	27.1	64.6
1.50	147.0	27.5	62.9
2.00	173.6	27.8	62.1
3.50	252.3	28.0	61.9

From the comparison of Tables 3 and 4, it can be deduced that a word-based analysis is iteratively taking into account a quite high percentage of useless states, thus needing to increase the beam parameter to include the successful paths into the analysis. The price for considering such a long list

Table 4: *Results for a phrase-based analysis.*

beam	Time (ms)	BLEU	WER
1.00	0.2	19.8	71.8
1.02	0.4	22.1	68.6
1.05	0.7	24.3	66.0
1.10	2.4	26.1	64.2
1.25	7.0	27.1	62.8
1.50	9.7	27.5	62.3
2.00	11.4	27.8	62.0
3.50	**12.3**	28.0	61.9

of states in every iteration of the algorithm is in terms of temporal requirements.

However, a phrase-based approach only stores those states that have been successfully reached by a full phrase compatibility with the input sentence. Therefore, it takes more time to process an individual state, but since the list of states is shorter, the search method performs at a better speed rate. Another important element to point out between Tables 3 and 4, is about the differences on quality results for a same beam parameter in both tables. Word-based decoding strategies suffer the effective reduction on the beam factor that was mentioned on section 3.1 because their best scores on every analysis stage, which determine the exploration boundaries, may refer to a no way out path. Logically, these differences are progressively reduced as the beam parameter increases, since the search space is explored in a more exhaustive way.

Table 5: *Number of trained and survived n-grams.*

	n-grams	
Window size	unigrams	bigrams
No filter	1,593,677	4,477,382
2	299,002	512,943
3	153,153	141,883
4	130,666	90,265
5	126,056	78,824
6	125,516	77,341

On the other hand, a phrase-based extended-symbol bigram model, being learnt by means of the full training data, computes an overall set of approximately 6 million events. The application of the n-gram pruning technique, using a growing window parameter, can effectively reduce that number to only 200,000. These n-grams, when represented as transducer transitions, suppose a reduction from 20 million transitions to only those

500,000 that are affected by the test set sentences. As a result, the size of the model to be parsed decreases, therefore, the decoding time also decreases. Tables 5 and 6 show the effect of this pruning method on the size of the transducers, then on the decoding time with a phrase-based analysis, which is the best strategy for phrase-based models.

Table 6: *Decoding time for several windows sizes.*

Window size	Edges	Time (ms)
No filter	19,333,520	362.4
2	2,752,882	41.3
3	911,054	17.3
4	612,006	12.8
5	541,059	11.9
6	531,333	11.8

Needless to say that BLEU and WER keep on their best numbers for all the transducer sizes as the test set is not present in the pruned transitions.

5 Conclusions

GIATI is a grammatical inference technique to learn stochastic transducers from bilingual data for their usage in statistical machine translation. Finite-state transducers are able to model both the structure of both languages and their relationship. GREAT is a finite-state toolkit which was born to overcome the computational problems that previous implementations of GIATI present in practice when huge amounts of parallel data are employed.

Moreover, GREAT is the result of a very meticulous study of GIATI models, which improves the treatment of smoothing transitions in decoding time, and that also reduces the required time to translate an input sentence by means of an analysis that will depend on the granularity of the symbols.

A pruning technique has been designed for n-gram approaches, which reduces the transducer size to integrate only those transitions that are really required for the translation of the test set. That has allowed us to perform some experiments concerning a state-of-the-art, voluminous translation task, such as the EuroParl, whose results have been reported in depth. A better performance has been found when a phrase-based decoding strategy is selected in order to deal with those GIATI phrase-based transducers. Besides, this permits us to apply a more refined interpretation of backoff transitions for a better smoothing translation behaviour.

Acknowledgments

This work was supported by the "Vicerrectorado de Innovación y Desarrollo de la Universidad Politécnica de Valencia", under grant 20080033.

References

Francisco Casacuberta and Enrique Vidal. 2004. Machine translation with inferred stochastic finite-state transducers. *Comput. Linguistics*, 30(2):205–225.

Francisco Casacuberta, Hermann Ney, Franz Josef Och, Enrique Vidal, Juan Miguel Vilar, Sergio Barrachina, Ismael García-Varea, David Llorens, César Martínez, and Sirko Molau. 2004. Some approaches to statistical and finite-state speech-to-speech translation. *Computer Speech & Language*, 18(1):25–47.

Francisco Casacuberta, Enrique Vidal, and David Picó. 2005. Inference of finite-state transducers from regular languages. *Pattern Recognition*, 38(9):1431–1443.

F. Casacuberta. 2000. Inference of finite-state transducers by using regular grammars and morphisms. In A.L. Oliveira, editor, *Grammatical Inference: Algorithms and Applications*, volume 1891 of *Lecture Notes in Computer Science*, pages 1–14. Springer-Verlag. 5th International Colloquium Grammatical Inference -ICGI2000-. Lisboa. Portugal.

J. González and F. Casacuberta. 2007. Phrase-based finite state models. In *Proceedings of the 6th International Workshop on Finite State Methods and Natural Language Processing (FSMNLP)*, Potsdam (Germany), September 14-16.

J. González, G. Sanchis, and F. Casacuberta. 2008. Learning finite state transducers using bilingual phrases. In *9th International Conference on Intelligent Text Processing and Computational Linguistics. Lecture Notes in Computer Science*, Haifa, Israel, February 17 to 23.

Frederick Jelinek. 1998. *Statistical Methods for Speech Recognition*. The MIT Press, January.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *ACL*. The Association for Computer Linguistics.

K. Papineni, S. Roukos, T. Ward, and W. Zhu. 2001. Bleu: a method for automatic evaluation of machine translation.

David Picó. 2005. *Combining Statistical and Finite-State Methods for Machine Translation*. Ph.D. thesis, Departamento de Sistemas Informáticos y Computación. Universidad Politécnica de Valencia, Valencia (Spain), September. Advisor: Dr. F. Casacuberta.

E. Vidal, F. Thollard, F. Casacuberta C. de la Higuera, and R. Carrasco. 2005. Probabilistic finite-state machines - part I. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(7):1013–1025.