Mach Translat (2011) 25:145–160
DOI 10.1007/s10590-011-9097-6

GREAT: open source software for statistical machine translation

Jorge González · Francisco Casacuberta

Received: 15 October 2010 / Accepted: 25 July 2011 / Published online: 28 August 2011

Abstract In this article, the first public release of GREAT as an open-source, statistical machine translation (SMT) software toolkit is described. GREAT is based on a bilingual language modelling approach for SMT, which is so far implemented for n-gram models based on the framework of stochastic finite-state transducers. The use of finite-state models is motivated by their simplicity, their versatility, and the fact that they present a lower computational cost, if compared with other more expressive models. Moreover, if translation is assumed to be a subsequential process, finite-state models are enough for modelling the existing relations between a source and a target language. GREAT includes some characteristics usually present in state-of-the-art SMT, such as phrase-based translation models or a log-linear framework for local features. Experimental results on a well-known corpus such as Europarl are reported in order to validate this software. A competitive translation quality is achieved, yet using both a lower number of model parameters and a lower response time than the widely-used, state-of-the-art SMT system Moses.

Keywords Statistical machine translation · Monotonic bilingual segmentation · Grammatical inference · Language modelling · Stochastic finite-state transducers

1 Introduction

Statistical machine translation (SMT) is a pattern recognition approach to machine translation where models are automatically inferred from the analysis of bilingual text corpora (Koehn 2010). Under this framework, several free/open-source

J. González (✉) · F. Casacuberta
Departamento de Sistemas Informáticos y Computación, Instituto Tecnológico de Informática, Universitat Politècnica de València, València, Spain
e-mail: jgonzalez@dsic.upv.es

software packages are nowadays publicly available (Ortiz et al. 2005; Koehn et al. 2007; Li et al. 2009). The toolkit GREAT is presented here as an efficient SMT approach, developed in the context of González's (2009) PhD thesis.

In this article, the first release of GREAT as an open-source SMT software toolkit is described, which may be freely downloaded from http://prhlt.iti.upv.es/page/software. GREAT is based on a bilingual language modelling approach for SMT, which is so far implemented for n-gram models based on the framework of *stochastic finite-state transducers* (SFSTs). The software offers room for other language models that future developers may want to incorporate, whether they are finite-state representable models (Mariño et al. 2006; Pérez et al. 2008) or they come from a more complex, non-regular grammatical inference framework.

SFSTs constitute a class of statistical models that is being used in SMT (Amengual et al. 2000; Casacuberta and Vidal 2004) and in other applications of natural language processing (Karttunen 2001) with success. Advantages of these models are their great simplicity and easy integration with the conventional automatic speech recognition models (hidden Markov models), thereby allowing the use of the simple and traditional Viterbi-beam-search techniques for efficient speech translation (Vidal 1997). Besides their versatility, they present a lower computational cost than other more expressive models, thus being more interesting in practice. Moreover, most translation relations are subsequential (Berstel 1979), therefore SFSTs seem a priori powerful enough for modelling most existing relations between a source and a target language. SFSTs are also able to implement and to approach other SMT models (Bangalore and Riccardi 2002; Kumar et al. 2006).

Similar to what happens with other statistical models, SFSTs can also be learned automatically from bilingual corpora (Casacuberta and Vidal 2007). One of these SFST estimation techniques is the *grammatical inference and alignments for transducer inference* (GIATI) method (Casacuberta and Vidal 2004; Andrés-Ferrer et al. 2008). The basic idea of this technique, which is not only limited to SFSTs, lies in the use of a language model of bilingual segments.

State-of-the-art SMT is strongly based on phrases (variable-length n-grams) and on a log-linear statistical approach (Och and Ney 2002). GREAT is able to consider both of them by means of the implementation of a log-linear framework for phrase-based SFSTs. Experimental results on a widely used, well-known translation task such as Europarl are reported in order to validate this software package. They show a competitive translation quality, yet using a clearly lower number of parameters than that of a state-of-the-art SMT competitor system, Moses (Koehn et al. 2007), and far smaller translation time requirements.

This article is organized as follows: Section 2 presents the statistical framework of GREAT together with a formal definition for SFSTs; Sect. 3 is devoted to SFST training; Sect. 4 documents the GREAT decoder; Sect. 5 documents GREAT's use of a log-linear modelling framework. Some empirical results are reported in Sect. 6. The conclusions that can be drawn from this study, along with some future working plans, are presented in the last section.

2 Statistical framework

Given a source sentence $\mathbf{s}$, the goal of SMT is to find a target sentence $\hat{\mathbf{t}}$, among all possible target strings $\mathbf{t}$, that maximises the probability:

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

where $\Pr(\mathbf{t} \mid \mathbf{s})$ can be replaced by $\Pr(\mathbf{s}, \mathbf{t})$ since the relation between both probability distributions allows us to dispense with the third-party term, $\Pr(\mathbf{s})$, which does not depend on the maximization variable:

$$\begin{aligned} \hat{\mathbf{t}} &= \underset{\mathbf{t}}{\operatorname{argmax}} \frac{\Pr(\mathbf{s}, \mathbf{t})}{\Pr(\mathbf{s})} \\ &= \underset{\mathbf{t}}{\operatorname{argmax}} \Pr(\mathbf{s}, \mathbf{t}) \end{aligned} \tag{2}$$

Although there exist different approaches to estimating $\Pr(\mathbf{s}, \mathbf{t})$ (Marcu and Wong 2002), the joint probability distribution is also adequately modelled by using a language modelling framework for monotonic bilingual string segmentations (Simard and Plamondon 1998).

The concept of monotonic bilingual segmentation is defined for SMT by means of two functions, $\mu_0^K : \Sigma^\star \to \mathbb{N}$ for the source sentence $\mathbf{s}$, and $\rho_0^K : \Delta^\star \to \mathbb{N}$ for the target sentence $\mathbf{t}$:

$$\begin{gathered} \mu_0^K(\mathbf{s}) \to \{0, \ldots, |\mathbf{s}|\} \qquad \rho_0^K(\mathbf{t}) \to \{0, \ldots, |\mathbf{t}|\} \\ \mu_0(\mathbf{s}) = 0 \quad \mu_K(\mathbf{s}) = |\mathbf{s}| \qquad \rho_0(\mathbf{t}) = 0 \quad \rho_K(\mathbf{t}) = |\mathbf{t}| \\ \forall k = 1 \ldots K : \mu_k(\mathbf{s}) \geq \mu_{k-1}(\mathbf{s}) \wedge \rho_k(\mathbf{t}) \geq \rho_{k-1}(\mathbf{t}) \end{gathered}$$

that monotonically divide them into the same number K of substrings:

$$\begin{gathered} \mathbf{s} = \bar{s}_1 \ldots \bar{s}_k \ldots \bar{s}_K \\ \mathbf{t} = \bar{t}_1 \ldots \bar{t}_k \ldots \bar{t}_K \\ \forall k = 1 \ldots K : \bar{s}_k = \mathbf{s}_{\mu_{k-1}+1}^{\mu_k} \wedge \bar{t}_k = \mathbf{t}_{\rho_{k-1}+1}^{\rho_k} \end{gathered}$$

therefore, substrings $\bar{s}_k$ and $\bar{t}_k$ are aligned with each other, $\forall k = 1 \ldots K$. Hence, a string of K symbols γ can be derived from $\mu_0^K(\mathbf{s})$ and $\rho_0^K(\mathbf{t})$:

$$\begin{gathered} \gamma = \gamma_1 \cdots \gamma_k \cdots \gamma_K \\ \forall k = 1 \ldots K : \gamma_k = \bar{s}_k \bar{t}_k \end{gathered}$$

This process is applied to all sentence pairs in the training set, after which a string corpus is generated on the basis of a bilingual vocabulary, and from which a language model technique can be employed to learn.

This is the formalism of GREAT, which is so far implemented for finite-state language models based on a *consistent* modelling of Pr(**s**, **t**) by means of SFSTs. Other authors (Kanthak et al. 2005) are also working in this joint approach, but they use *weighted finite-state transducers* (WFSTs) (Mohri et al. 2002) instead, which do not require the modelling of consistent probability distributions.

2.1 Stochastic finite-state transducers

Stochastic finite-state transducers are defined by means of a tuple $(\Sigma, \Delta, Q, q_0, f, P)$, where Σ is the alphabet of input symbols, Δ is the alphabet of output symbols, Q is a finite set of states, $q_0 \in Q$ is the initial state, $f : Q \to [0, 1]$ is the probability that states may be final states, and the partial function $P : Q \times \Sigma^\star \times \Delta^\star \times Q \to [0, 1]$ defines a set of edges between pairs of states in such a way that every edge is labelled with an input string in $\Sigma^\star$, also with an output string in $\Delta^\star$, and is weighted by a transition probability.

Moreover, aiming to model Pr(**s**, **t**), those functions are constrained in order to be able to define a consistent probability distribution on the Cartesian product $\Sigma^\star \times \Delta^\star$ (Vidal et al. 2005).

$$\forall q \in Q : \sum_{\sigma \in \Sigma^\star, \delta \in \Delta^\star, q' \in Q} P(q, \sigma, \delta, q') + f(q) = 1$$

This description is complemented by the graphical example in Fig. 1, extracted from the parallel corpus used in the experiments in Sect. 6.

Under other constraints however, SFSTs may define conditional models instead. For example, SFSTs for Pr(**s** | **t**) or Pr(**t** | **s**) are discussed in Sect. 5.1.

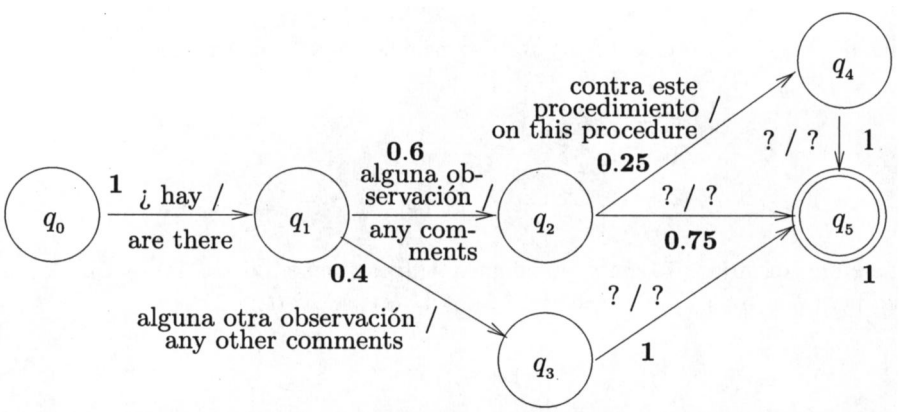

Fig. 1 Graphical description of SFSTs. States are denoted as q_i, $\forall i : 0 \leq i \leq 5$, and represented as *circles* in the diagram, where a *double circle* means a final state. Edges are represented by means of *direction arrows* between pairs of states, in such a way that transition labels are shown in the format "input string"/"output string". In this particular example, the model is used to address Spanish-to-English SMT. Transition and final probabilities are in *bold*

3 Model training with GREAT

Only a few techniques to learn SFSTs can be found in the literature (Bangalore and Riccardi 2002; Knight and Al-Onaizan 1998; Casacuberta and Vidal 2007). However, a relation between regular translations generated by SFSTs and regular languages over some alphabet of string pairs was established through morphisms (Berstel 1979). This property was used to propose a method of inference of SFSTs based on the inference of *stochastic finite-state automata* (SFSAs) (Casacuberta and Vidal 2004). This method, which has been widely used in SMT applications (Casacuberta and Vidal 2007; Pérez et al. 2008; González and Casacuberta 2009), is known as GIATI and is the training framework of GREAT.

The GIATI algorithm establishes that a parallel corpus is to be processed by using monotonic bilingual segmentations to get a string corpus whose symbols belong to a new bilingual alphabet $\Gamma \subset \Sigma^\star \times \Delta^\star$. Then, a language model can be trained from this new sample. Although GIATI was initially designed as an SFST inference algorithm (and so it is implemented in GREAT), the core idea is that any language modelling technique can be applied to the new string corpus. When a language modelling framework based on finite-state models is used, an SFSA is inferred from such a synthetic corpus. That model can also be seen as an SFST for $\Pr(\mathbf{s}, \mathbf{t})$ as described in Sect. 2.1, once the transition labels in Γ are traced back to their separate input and output strings, as a direct derivation from the morphism theorem (Medvedev 1964). The current release of GREAT is based on n-gram-based SFSAs and it works with `SRILM`[1], which is then required as additional software. Future GREAT versions will also interface with other n-gram toolkits.

The tool to train a translation model with GREAT is named `giati` and a configuration file must be provided: `giati -c <config_file>`. This file points to several working directories and establishes the value of some training parameters using the format `PARAMETER = VALUE`. Details about the most practical settings are given in the `README` file. Only the ones related to the translation functions are discussed here.

Different setups are provided through several training parameters, either instantiated in the configuration file or via the command-line. Let us analyse them by means of the three sequential steps of `giati`:

1. A string set is extracted from the word-aligned training corpus based on the concept of monotonic bilingual sentence segmentation. New symbols are built by concatenating source and target words using two control characters, so that original words can be restored afterwards. The software looks for possible control characters in a wide range in such a way that they are not included in the training corpus.

For example, for the sentences "El coche rojo" and "The red car", a bilingual string like "El/The coche–rojo/red–car" can be constructed, where characters '/' and '–' are the aforementioned control characters. Two individual symbols are used, 'El/The' and 'coche–rojo/red–car', whose source and target words can later be decoded in step 3 of `giati` thanks to these control characters.

[1] Available from http://www.speech.sri.com/projects/srilm/.

If the --end command-line flag is included, training is not actually performed, but the control characters that would have been employed next are output. This is useful if step 1 of giati is skipped because an external bilingual string corpus is provided later, instead of using the embedded segmentation algorithms of GREAT. This alternative file has to follow the same format, so the control characters computed with --end are needed to build it. The --segment command-line flag skips this step, thus starting giati from step 2.

If neither --end nor --segment are included into the command-line, a string set is extracted from the training parallel corpus according to the following parameters in the configuration file:

- ALIGN refers to a word-alignments file for the training corpus following the GIZA++ format (Och and Ney 2003). Alignments are processed to extract monotonic bilingual segmentations from them according to which ALGORITHM (see next option below) is selected. Although they may be provided by any means, alignments can also be statistically learnt by means of GIZA++[2] (Och and Ney 2003). In this case, basic GIZA++ training is usually enough for GREAT.
- ALGORITHM determines the extraction algorithm to be used. So far, there are only two bilingual segmentation algorithms implemented, one for a *word-based* and one for a *phrase-based* SMT approach (Koehn 2010), described in González (2009). They are selected by setting this parameter to 1 or 3, respectively. Odd numbers are used because even numbers are reserved for future developments on *factored* SMT (Koehn and Hoang 2007).
- SIMBEXTMAX limits the number of words in the aligned segments. For the example above, the symbol 'El/The' involves two words, whereas the symbol 'coche−rojo/red−car' involves four words.

2. An n-gram model is trained from the string set obtained in step 1. Probabilities for bilingual symbols to be arranged in sequence are learnt. The NGRAM parameter refers here to the n-gram order requested to train.

3. Finally, an SFST is built from the n-gram model trained in step 2. The command-line option -F <integer> deals with filtering the SFST building by means of the test corpus (González and Casacuberta 2009). This technique implements a sliding window of the size indicated by -F to determine if any trained n-gram represents an SFST path reachable by some of the source sentences that are going to be translated later. If not, then they are not represented in the SFST structure. The larger the window size, the more constrained the reachability, and therefore the smaller the SFSTs are (until convergence is achieved). By default, -F is set to 0, which means no filtering at all.

[2] Available from http://code.google.com/p/giza-pp.

4 Decoding with GREAT

Finite-state search is efficiently accomplished with the well-known Viterbi algorithm (Viterbi 1967). Under the application of this decoding framework to SFSTs (Picó and Casacuberta 2001), several beam-search techniques (Steinbiss et al. 1994) can be introduced in order to reduce translation time requirements.

A graph structure known as *trellis* is employed by the Viterbi algorithm in order to organize the search space represented by SFSTs. A search strategy where the trellis includes phrase-based transitions was adopted with the aim of adapting the decoding procedure to a phrase-based SMT approach, thus allowing us to develop a more effective and efficient beam-search decoding algorithm for phrase-based SFSTs (González and Casacuberta 2009). Furthermore, this technique is algorithmically equivalent to a search strategy based on words in the case of using word-based SFSTs.

The tool to translate a test corpus with GREAT is called `refx`, which requires that a model is provided: `refx -m <model> -t <test_set>`.

Additional running options can be introduced via the command-line as well. The most important ones are `-b <integer>` and the flag `--smooth`. The `-b` option triggers a beam-search technique based on histogram pruning (Steinbiss et al. 1994). In general terms, it means that only the number indicated by `-b` of partial hypotheses are taken into account in the trellis from stage to stage, i.e. the best scored ones. The smaller this number is, the more constrained the search process is, and therefore the faster it is. However, translation results can be rather poor if the number considered is too small. As a consequence, a trade-off between quality and time requirements is usually empirically determined in order to reduce the response time without a significant quality loss. On the other hand, the flag `--smooth` is related to SFST smoothing. Its explanation is discussed in the next subsection.

4.1 Smoothing

Language models typically suffer from a sparseness problem (Rosenfeld 1996). GREAT, which is based on a *bilingual* language modelling approach for SMT, is even more prone to it. Smoothing in SMT is typically handled by a recovery method which allows us to assign a probability to any given sentence pair. Therefore, the probability distributions of the SFSTs being inferred must be modelled in order to deal with unseen events out of training data, thus aiming for a larger coverage in SMT applications. This process is internally carried out by the `giati` training tool, which always takes smoothing into account to estimate the models.

Smoothing in GREAT is so far applied at the language level, thus currently relying on well-known approaches for n-gram smoothing. Ours is based on the *backoff* method, which introduces some penalties for level downgrading within hierarchical language models. In Kneser and Ney (1995), backoff was also applied to n-gram models, and in Torres and Varona (2001) and in Llorens et al. (2002) it was applied to SFSAs. As Fig. 2 suggests, backoff is implemented for SFSAs based on n-gram models by means of different top-down failure transitions, that is, used only if there

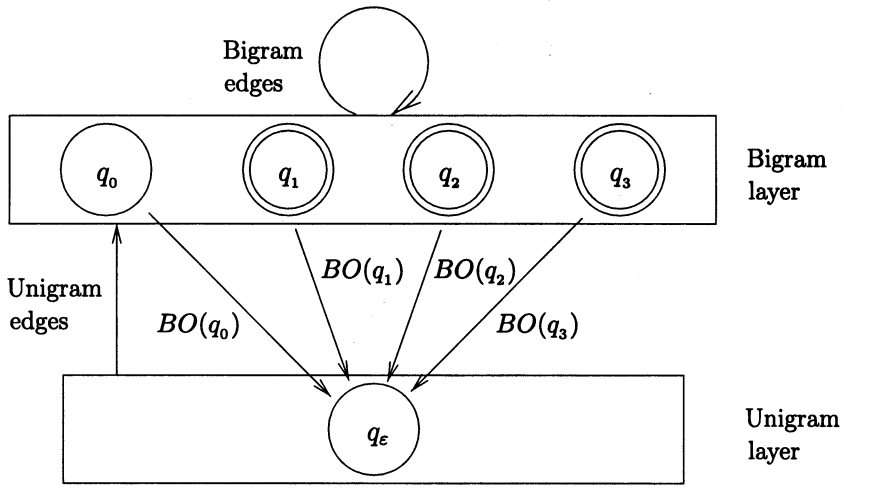

Fig. 2 Efficient representation of backoff for bigram-based SFSAs. Backoff (BO) is expressed as failure transitions from the bigram layer to the unigram layer. Unigrams go in the other direction and bigrams link states within the bigram layer

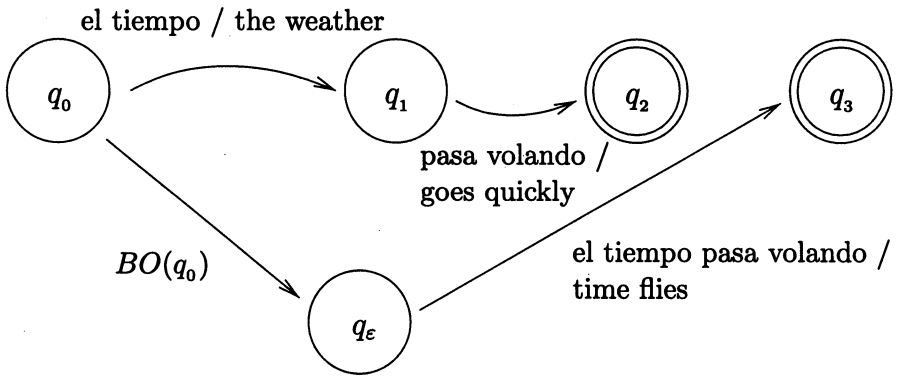

Fig. 3 Structural pruning of an SFST that is based on a bilingual bigram model. Backoff edges do not consume any symbol but if they are considered failure transitions, then given the input sentence "el tiempo pasa volando", the output sentence "time flies" can not be found in the SFST

is not any other successful transition available, between the layers representing consecutive order n-grams.

Nevertheless, this equivalence is only in terms of language modelling. In our context however, a language model of bilingual symbols is used as a translation device, where only the source sentence is known in running time. If backoff edges are interpreted with a failure meaning, then the search is partially bound without even considering the scores. This effect causes some translations of the source sentence to be ignored, even though their relation is explicitly modelled in the SFST, which implies a significant loss in translation quality (González 2009).

For example, Fig. 3 shows that, as an SFSA where transition labels are treated as a whole, both strings "el tiempo / the weather" and "pasa volando / goes quickly" (from

q_0 to q_1, and then from q_1 to q_2) and "el tiempo pasa volando / time flies" (from q_0, down to q_ε, then up again to q_3) are accepted by the model taking into account that backoff edges are failure edges, thus modelling two paths with different translations for the same input sentence. Nevertheless, as an SFST looks for the translation of the input string "el tiempo pasa volando", if backoff transitions are considered failure transitions, then only the output string "the weather goes quickly" will be found. This is because a successful transition is found from q_0 (from q_0 to q_1) and therefore the backoff edge $q_0 \rightarrow q_\varepsilon$ can not be taken into account. As a result, the translation "time flies" remains hidden within the SFST because it can not be discovered by means of such a search algorithm. This is the built-in behaviour for `refx` regarding backoff and thus it is related to an incomplete exploration of the search space that sometimes may be convenient for reducing response time.

Nevertheless, if a full exploration of the search space is preferred, a more accurate backoff interpretation algorithm is also implemented. For n-gram-based SFSTs, the objective is that the definition of n-grams must be respected in such a way that the exploration of the network has to be equivalent in smoothing terms. Therefore, backoff edges must not be considered failure transitions with regard to the source words but as for every bilingual label that the input is compatible with.

This behaviour is implemented by means of the interpretation of backoff edges as ε/ε edges instead (ε means the empty string), and avoiding the access to some eventually inactive states (González and Casacuberta 2009). After exhausting all possible transition options labelled with any pair of strings, the corresponding backoff edge is taken and other possible transitions using different string pairs are explored. This second backoff algorithm is applied if `refx` is launched together with the `--smooth` flag.

Smoothing is still an open problem in SMT (Foster et al. 2006) and SFSTs are not an exception: they are not completely smoothed just by smoothing the corresponding SFSAs. Future research on SFST smoothing should be based on specific criteria for SFSTs as well.

5 Log-linear modelling with GREAT

Sections 2, 3, and 4 are related to a statistical framework where one SFST modelling $\Pr(\mathbf{s}, \mathbf{t})$ is used to find the most likely translation $\hat{\mathbf{t}}$ of a given source sentence $\mathbf{s}$. This section is devoted to extending this framework by means of a log-linear approach (Och and Ney 2002), where several SFSTs[3] featuring different models are properly combined in order to decide in conjunction which is the overall best translation.

In general terms, a log-linear statistical framework is enunciated as:

$$\hat{\mathbf{t}} = \underset{\mathbf{t}}{\operatorname{argmax}} \sum_{m=1}^{M} \lambda_m \, h_m(\mathbf{s}, \mathbf{t}) \tag{3}$$

[3] Abuse of notation: SFSTs refers to WFSTs, whether they are stochastic or not.

where $h_m(\mathbf{s}, \mathbf{t})$ is the evaluation of model m over $(\mathbf{s}, \mathbf{t})$, also known as *feature*, M is the number of models, and each λ_m is a trainable weight.

GREAT makes this framework available for phrase-based SFSTs, where an arbitrary number of phrase-based local features can be used. The main model of such a scenario will always be an SFST for $\Pr(\mathbf{s}, \mathbf{t})$, which can be learnt by means of the `giati` tool, as shown in Sect. 3. Several phrase-based local features are proposed, e.g. translation models for $\Pr(\mathbf{s} \mid \mathbf{t})$ and $\Pr(\mathbf{t} \mid \mathbf{s})$ or length models, but users may also define others of their own interest. The application of phrase-based local features in the log-linear scenario simplifies the modelling of a sentence-based feature $h_m(\mathbf{s}, \mathbf{t})$ by means of the use of some other lower-context (based on phrases) models $\eta_m(\bar{s}, \bar{t})$.

The `local` tool computes both direct and inverse translation models from the corresponding word-based statistical dictionaries also estimated by GIZA++. For an inverse phrase-based model, i.e. the probability of a source phrase $\bar{s}$ given a target phrase $\bar{t}$, the following equation is performed: $\eta_m(\bar{s}, \bar{t}) = \Pr(\bar{s}|\bar{t}) \approx \text{Poisson}(|\bar{s}|, |\bar{t}|) \prod_{\mathbf{s}_j \in \bar{s}} \sum_{\mathbf{t}_i \in \bar{t}} \Pr(\mathbf{s}_j|\mathbf{t}_i)$, where a Poisson probability distribution is used for normalization on the basis of the work of Singh and Husain (2007). A similar equation is applied for direct translation models. A target-sentence length model is also computed by the `local` tool regarding a phrase-based approach. This feature is formulated as $\eta_m(\bar{s}, \bar{t}) = |\bar{t}|$.

The usage of `local` is similar to `refx` as for their running options. Nevertheless, two options are particularly needed here: first, `-f 2` selects the n-gram (and not the SFST) model, where a `.vocabext` file contains the set of bilingual phrase pairs; second, `-C 'xy'` specifies the two `x` and `y` control characters used to build the language model vocabulary during training. For more information on control characters and why they are needed, please refer to step 1 of the `giati` training tool, described in Sect. 3. After execution, a `.probs` file will be created from the `.vocabext` file where the former extends the latter with three scores per phrase pair according to the corresponding local features presented above.

5.1 The SFST log-linear framework

Phrase-based local features are expressed as a particular case of SFSTs in order to establish a homogeneous statistical framework to work with. Figure 4 shows the relation between these features and SFSTs.

Every feature $h_m(\mathbf{s}, \mathbf{t})$ is computed as the Viterbi score for $\mathbf{s}$ and $\mathbf{t}$ according to the corresponding m-th SFST in the log-linear approach, what defines the optimal sequence of phrase pairs that best applies. However, this framework is constrained in such a way so that all the SFSTs have to define the same segmentation for a given sentence pair:

$$K > 0$$
$$\mathbf{s} = \bar{s}_1 \ldots \bar{s}_K$$
$$\mathbf{t} = \bar{t}_1 \ldots \bar{t}_K$$

source phrase	target phrase	score
se requiere una acción	action is required	p
...		
...		

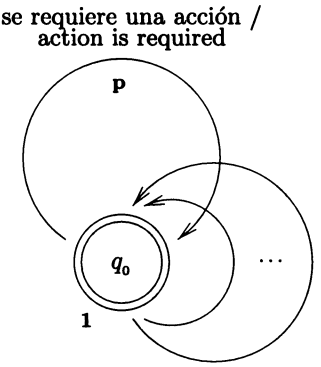

Fig. 4 Equivalent representation of phrase-based local features through SFSTs. Every table entry is embedded within a looping transition of a phrase-based SFST with no topology at all. Scores are correspondingly stored as transition probabilities

$$h_m(\mathbf{s},\mathbf{t}) = \begin{cases} \sum_{k=1}^{K} \log P_m\left(q_{k-1}, \bar{s}_k, \bar{t}_k, q_k\right) + \log f_m(q_K) & m = 1 \\ \sum_{k=1}^{K} \log P_m\left(q_0, \bar{s}_k, \bar{t}_k, q_0\right) = \sum_{k=1}^{K} \log \eta_m\left(\bar{s}_k, \bar{t}_k\right) & m > 1 \end{cases}$$

assuming that $m = 1$ refers to the main SFST, i.e. the one for $\Pr(\mathbf{s},\mathbf{t})$, where $q_0 \ldots q_K$ is the path for the aforesaid bilingual segmentation. Phrase-based local features are represented by a model number $m > 1$.

Then, Eq. 3 can be expressed as follows:

$$\begin{aligned} \hat{\mathbf{t}} &= \operatorname*{argmax}_{\mathbf{t}} \sum_{m=1}^{M} \lambda_m\, h_m(\mathbf{s},\mathbf{t}) \\ &= \operatorname*{argmax}_{\mathbf{t}} \left(\lambda_1 \left[\sum_{k=1}^{K} \log P_1\left(q_{k-1}, \bar{s}_k, \bar{t}_k, q_k\right) + \log f_1(q_K) \right] \right. \\ &\quad \left. + \sum_{m=2}^{M} \lambda_m \sum_{k=1}^{K} \log \eta_m\left(\bar{s}_k, \bar{t}_k\right) \right) \\ &= \operatorname*{argmax}_{\mathbf{t}} \left(\sum_{k=1}^{K} \left[\lambda_1 \log P_1\left(q_{k-1}, \bar{s}_k, \bar{t}_k, q_k\right) + \sum_{m=2}^{M} \lambda_m \log \eta_m\left(\bar{s}_k, \bar{t}_k\right) \right] \right. \\ &\quad \left. + \lambda_1 \log f_1(q_K) \right) \end{aligned} \tag{4}$$

which can be tackled by means of a Viterbi search in the main SFST through the contribution of all the models (along with their relative weights) over the phrase pair $(\bar{s}_k, \bar{t}_k)$ that extends a partial hypothesis.

To that end, local scores are embedded into the main SFST by means of defining an extended transition function $P : Q \times \Sigma^\star \times \Delta^\star \times Q \rightarrow \mathbb{R}^M$. The `refx` tool builds such an extended SFST by using the `-n` *M* option (together with the aforementioned options `-m`, `-t`, `-f 2`, and `-C 'xy'`), as `-n` stands for the number of scores per transition which SFSTs are built with.

The tool to train the set of λ-weights with GREAT is named `simplex` and it uses the same syntax as the `refx` decoding tool (see Sect. 4). It is based on the BLEU evaluation measure (Papineni et al. 2002) and it is applied to extended SFSTs using development data.

Once the set of optimal weights is established by means of `simplex`, a standard SFST with only one score per transition, but that accounts for all the SFSTs in the log-linear framework, can be built. This process is also carried out by the `refx` tool by means of several running options (`-g`, `-i`, `-d`, and `-l`) that allow us to instantiate the set of λ-weights: `-g` for weighting the main model; `-i` and `-d` for inverse and direct translation models, respectively; and `-l` refers to the length model. The number of scores per transition must therefore be set to 1 (`-n 1`) so that all the scores in a transition are recombined into a global score by means of computing the square-bracketed expression of Eq. 4. Therefore, the integrated SFST keeps the structure of the main SFST, while only the scores originally estimated by `giati` are modified. As a consequence, the integrated model does not increase the decoding cost, and so users may add as many phrase-based local features as they like.

6 Experimental results

Experiments on the Europarl corpus (Koehn 2005) were conducted using the partitions that were established in the 1st workshop on Statistical Machine Translation of the North American Chapter of the Association for Computational Linguistics.

The Europarl corpus comprises 11 different languages from the proceedings of the European Parliament, which are electronically published on the web. However, we only focus here on the German–English, Spanish–English and French–English tasks since these were the language pairs that were selected for the above-mentioned workshop. The characteristics of the Europarl corpus are presented in Table 1.

Our SFST-based log-linear framework was tuned by means of another disjoint partition of parallel data, of similar dimensions to the one used for testing. The main SFST was based on a bigram model, where `SIMBEXTMAX` was set to 14 (7 words per segment and language), and `-F 8` was applied as regards the SFST filtering degree. While at decoding time, `-b 50` was used and the `--smooth` flag was activated. The Moses system (Koehn et al. 2007) is used for comparison purposes. In this case, a standard experimental setup was configured for Moses. Models were not binarised, as they were not binarised in GREAT either.

Table 1 Characteristics of the Europarl corpus for the training and test partitions

Subset features	German	English	Spanish	English	French	English
Train						
Sentences	751k		731k		688k	
Run. words	15.3M	16.1M	15.7M	15.2M	15.6M	13.8M
Vocabulary	195k	66k	103k	64k	80k	62k
Test						
Sentences	2000		2000		2000	
Run. words	54k	58k	60k	58k	66k	58k
OOV words	377	127	207	125	139	133

Thousands and millions of the referred items are respectively denoted as *k* and *M*

Table 2 Translation results for the Europarl corpus by using GREAT and Moses

Translation task	System	BLEU	WER	Phrase pairs	Response time
German → English	GREAT	23.5	67.7	1.8M	25 s
	Moses	24.6	66.8	12M	9 min 03 s
English → German	GREAT	15.9	74.2	1.9M	45 s
	Moses	17.5	72.5	13M	21 min 00 s
Spanish → English	GREAT	28.6	59.9	1.9M	32 s
	Moses	30.5	58.3	19M	10 min 49 s
English → Spanish	GREAT	28.1	61.0	1.7M	20 s
	Moses	29.7	59.5	19M	12 min 43 s
French → English	GREAT	29.3	58.9	1.5M	18 s
	Moses	30.3	57.7	15M	17 min 34 s
English → French	GREAT	29.3	62.9	1.6M	50 s
	Moses	31.3	60.5	16M	12 min 48 s

The experimental setup for Moses is based on eight statistical models, including phrase-based reordering, direct and inverse phrase-based translation probabilities, direct and inverse word-based lexical weighting, phrase-based and word-based penalties, and a word-based target language model

The results, which are presented in Table 2 along with the ones obtained with Moses, show the viability of GREAT to perform reasonably well on benchmark data frequently used in SMT studies, using however a lower parameter set and less translation time.

Confidence intervals at a level of 95% were computed for BLEU, following the bootstrap resampling technique described by Koehn (2004). These turned to be around ± 0.9 points for all the translation experiments, and are omitted in Table 2 for the sake of clarity.

These results allow us to compare GREAT with the state-of-the-art in SMT. For all the translation directions, GREAT displays an ability to deal with the Europarl task, yielding a performance close to that of Moses. Confidence intervals are overlapped in 4 of the 6 cases, and where not, they are close to overlapping. It appears that bilingual phrases are better exploited by GREAT, as it attains almost the same translation quality

by using only 10% of the number of phrase pairs considered in Moses. Translation time requirements are also strongly favourable with GREAT in all cases. As Moses also uses a lot more models than GREAT, we conclude that GREAT presents an interesting trade-off between its performance and the amount of resources needed to get it.

7 Conclusions and future work

In this article, the first public release of GREAT as an open-source SMT software toolkit is presented. GREAT is based on a bilingual language modelling approach for SMT, which is so far implemented for n-gram models based on the framework of SFSTs.

The validity of GREAT to build SMT systems has been confirmed by means of experiments on a well-known task based on Europarl. Results close to those yielded by a state-of-the-art system, Moses, have been achieved, while exhibiting, on the one hand, far less translation time, and on the other hand, also a lower number of model parameters.

Future work on factored SMT and on additional log-linear features is planned to be developed on the basis of SFST composition operations.

Acknowledgements Study was supported by the EC (FEDER, FSE), the Spanish government (MICINN, MITyC, "Plan E", under Grants MIPRCV "Consolider Ingenio 2010", iTrans2 TIN2009-14511, and erudito.com TSI-020110-2009-439), and the Generalitat Valenciana (Grant Prometeo/2009/014).

References

Amengual JC, Benedí JM, Casacuberta F, Castaño MA, Castellanos A, Jiménez VM, Llorens D, Marzal A, Pastor M, Prat F, Vidal E, Vilar JM (2000) The EUTRANS-I speech translation system. Mach Transl 15(1-2):75–103

Andrés-Ferrer J, Juan-Císcar A, Casacuberta F (2008) Statistical estimation of rational transducers applied to machine translation. Appl Artif Intell 22(1–2):4–22

Bangalore S, Riccardi G (2002) Stochastic finite-state models for spoken language machine translation. Mach Transl 17(3):165–184

Berstel J (1979) Transductions and context-free languages. B.G. Teubner, Stuttgart, Germany

Casacuberta F, Vidal E (2004) Machine translation with inferred stochastic finite-state transducers. Comput Linguist 30(2):205–225

Casacuberta F, Vidal E (2007) Learning finite-state models for machine translation. Mach Learn 66(1): 69–91

Foster G, Kuhn R, Johnson H (2006) Phrasetable smoothing for statistical machine translation. In: Proceedings of the 11th Conference on Empirical Methods in Natural Language Processing, Stroudsburg, PA, pp 53–61

González J (2009) Aprendizaje de transductores estocásticos de estados finitos y su aplicación en traducción automática. PhD thesis, Universitat Politècnica de València. Advisor: Casacuberta F

González J, Casacuberta F (2009) GREAT: a finite-state machine translation toolkit implementing a grammatical inference approach for transducer inference (GIATI). In: Proceedings of the EACL Workshop on Computational Linguistic Aspects of Grammatical Inference, Athens, Greece, pp 24–32

Kanthak S, Vilar D, Matusov E, Zens R, Ney H (2005) Novel reordering approaches in phrase-based statistical machine translation. In: Proceedings of the ACL Workshop on Building and Using Parallel Texts: Data-Driven Machine Translation and Beyond, Ann Arbor, MI, pp 167–174

Karttunen L (2001) Applications of finite-state transducers in natural language processing. In: Proceedings of the 5th Conference on Implementation and Application of Automata, London, UK, pp 34–46

Kneser R, Ney H (1995) Improved backing-off for n-gram language modeling. In: Proceedings of the 20th IEEE International Conference on Acoustic, Speech and Signal Processing, Detroit, MI, pp 181–184

Knight K, Al-Onaizan Y (1998) Translation with finite-state devices. In: Proceedings of the 3rd Conference of the Association for Machine Translation in the Americas, Langhorne, PA, pp 421–437

Koehn P (2004) Statistical significance tests for machine translation evaluation. In: Proceedings of the 9th Conference on Empirical Methods in Natural Language Processing, Barcelona, Spain, pp 388–395

Koehn P (2005) Europarl: a parallel corpus for statistical machine translation. In: Proceedings of the 10th Machine Translation Summit, Phuket, Thailand, pp 79–86

Koehn P (2010) Statistical machine translation. Cambridge University Press, Cambridge, UK

Koehn P, Hoang H (2007) Factored translation models. In: Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, Prague, Czech Republic, pp 868–876

Koehn P, Hoang H, Birch A, Callison-Burch C, Federico M, Bertoldi N, Cowan B, Shen W, Moran C, Zens R, Dyer C, Bojar O, Constantin A, Herbst E (2007) Moses: open source toolkit for statistical machine translation. In: Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics, Prague, Czech Republic, pp 177–180

Kumar S, Deng Y, Byrne W (2006) A weighted finite state transducer translation template model for statistical machine translation. Nat Lang Eng 12(1):35–75

Li Z, Callison-Burch C, Dyer C, Ganitkevitch J, Khudanpur S, Schwartz L, Thornton WNG, Weese J, Zaidan OF (2009) Joshua: an open source toolkit for parsing-based machine translation. In: Proceedings of the ACL Workshop on Statistical Machine Translation, Morristown, NJ, pp 135–139

Llorens D, Vilar JM, Casacuberta F (2002) Finite state language models smoothed using n-grams. Int J Pattern Recognit Artif Intell 16(3):275–289

Marcu D, Wong W (2002) A phrase-based, joint probability model for statistical machine translation. In: Proceedings of the 7th Conference on Empirical Methods in Natural Language Processing, Morristown, NJ, pp 133–139

Mariño JB, Banchs RE, Crego JM, de Gispert A, Lambert P, Fonollosa JAR, Costa-jussà MR (2006) N-gram-based machine translation. Comput Linguist 32(4):527–549

Medvedev YT (1964) On the class of events representable in a finite automaton. In: Moore EF (ed) Sequential machines selected papers. Addison Wesley, Reading, MA

Mohri M, Pereira F, Riley M (2002) Weighted finite-state transducers in speech recognition. Comput Speech Lang 16(1):69–88

Och FJ, Ney H (2002) Discriminative training and maximum entropy models for statistical machine translation. In: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, Philadelphia, PA, pp 295–302

Och FJ, Ney H (2003) A systematic comparison of various statistical alignment models. Comput Linguist 29(1):19–51

Ortiz D, García-Varea I, Casacuberta F (2005) Thot: a toolkit to train phrase-based statistical translation models. In: Proceedings of the 10th Machine Translation Summit, Phuket, Thailand, pp 141–148

Papineni K, Roukos S, Ward T, Zhu WJ (2002) Bleu: a method for automatic evaluation of machine translation. In: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, Philadelphia, PA, pp 311–318

Pérez A, Torres MI, Casacuberta F (2008) Joining linguistic and statistical methods for Spanish-to-Basque speech translation. Speech Commun 50:1021–1033

Picó D, Casacuberta F (2001) Some statistical-estimation methods for stochastic finite-state transducers. Mach Learn 44:121–142

Rosenfeld R (1996) A maximum entropy approach to adaptive statistical language modeling. Comput Speech Lang 10:187–228

Simard M, Plamondon P (1998) Bilingual sentence alignment: balancing robustness and accuracy. Mach Transl 13(1):59–80

Singh AK, Husain S (2007) Exploring translation similarities for building a better sentence aligner. In: Proceedings of the 3rd Indian International Conference on Artificial Intelligence, Pune, India, pp 1852–1863

Steinbiss V, Tran BH, Ney H (1994) Improvements in beam search. In: Proceedings of the 3rd International Conference on Spoken Language Processing, Yokohama, Japan, pp 2143–2146

Torres MI, Varona A (2001) k-TSS language models in speech recognition systems. Comput Speech Lang 15(2):127–149

Vidal E (1997) Finite-state speech-to-speech translation. In: Proceedings of the 22nd IEEE International Conference on Acoustic, Speech and Signal Processing, Munich, Germany, pp 111–114

Vidal E, Thollard F, de la Higuera C, Casacuberta F, Carrasco RC (2005) Probabilistic finite-state machines–Part II. IEEE Trans Pattern Anal Mach Intell 27(7):1025–1039

Viterbi A (1967) Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. IEEE Trans Inf Theory 13(2):260–269