A Syntactic Pattern Recognition Approach to Computer Assisted Translation

Jorge Civera[1], Juan M. Vilar[2], Elsa Cubel[1], Antonio L. Lagarda[1], Sergio Barrachina[2], Francisco Casacuberta[1], Enrique Vidal[1], David Picó[1], and Jorge González[1]

[1] Departamento de Sistemas Informáticos y Computación, Universitat Politècnica de València
Instituto Tecnológico de Informática, E-46071 València, Spain
tt2iti@iti.upv.es
http://www.iti.upv.es/~prhlt
[2] Departamento de Lenguajes y Sistemas Informáticos
Universitat Jaume I, E-12071 Castellón de la Plana, Spain

Abstract. It is a fact that current methodologies for automatic translation cannot be expected to produce high quality translations. An alternative approach is to use them as an aid to manual translation. We focus on a possible way to help human translators: to interactively provide completions for the parts of the sentences already translated. We explain how finite state transducers can be used for this task and show experiments in which the keystrokes needed to translate printer manuals were reduced to nearly 25% of the original.

1 Introduction

It is becoming increasingly clear that current automatic translation methodologies cannot be expected to produce high quality translation in the near future. An alternative way to take advantage of the technologies developed is to use them in order to help human translators. One such approach, proposed by [1], can be explained as follows: the translator begins to type the translation and the system guesses the best completion for the text typed so far. The user can then accept the suggestion of the computer or part of it. This should reduce the amount of work of the translator.

This approach has two important aspects: the models need to provide adequate completions and they have to do so efficiently. To fulfill these two requirements, we have decided to use Stochastic Finite State Transducers (SFST) since they have proved in the past to be able to provide adequate translations [2–4] and, as we show in this paper, efficient parsing algorithms can be easily adapted in order to provide completions.

The rest of the paper is structured as follows. The following section presents the general setting for machine translation and finite state models. In section 3, the search procedure for an interactive translation is presented. Experimental results are presented in section 4. Finally, some conclusions and future work are explained in section 5.

2 Machine Translation with Finite-State Transducers

Given a source sentence s, the goal of MT is to find a target sentence $\hat{\mathbf{t}}$ that maximizes:

$$\hat{\mathbf{t}} = \underset{\mathbf{t}}{\operatorname{argmax}} \Pr(\mathbf{t} \mid \mathbf{s}) = \underset{\mathbf{t}}{\operatorname{argmax}} \Pr(\mathbf{t}, \mathbf{s}) \approx \underset{\mathbf{t}}{\operatorname{argmax}} \Pr_{\mathcal{T}}(\mathbf{t}, \mathbf{s}) \tag{1}$$

A. Fred et al. (Eds.): SSPR&SPR 2004, LNCS 3138, pp. 207–215, 2004.

The joint distribution $\Pr(\mathbf{t}, \mathbf{s})$ can be modeled by Stochastic Finite State Transducers (SFST) $\mathcal{T}$ [5]. They have been successfully applied into many translation tasks [2–4]. Furthermore, there exist efficient parsing algorithms like Viterbi[6] for the best parse and REA [7] for the n-best parses.

One possible way of inferring SFSTs from training data is the Grammatical Inference and Alignments for Transducer Inference[1] (GIATI) technique [8]. Given a finite sample of string pairs, it works in three steps:

1. Building training strings. Each training pair is transformed into a single string from an extended alphabet to obtain a new sample of strings. The "extended alphabet" contains words or substrings from source and target sentences coming from training pairs.
2. Inferring a (stochastic) regular grammar. Typically, a smoothed n-gram is inferred from the sample of strings obtained in the previous step.
3. Transforming the inferred regular grammar into a transducer. The symbols associated to the grammar rules are transformed into source/target symbols by applying an adequate transformation, thereby transforming the grammar inferred in the previous step into a transducer.

The transformation of a parallel corpus into a corpus of single sentences is performed with the help of statistical alignments: each word is joined with the word in the target sentence it is aligned to, creating an "extended word". This joining is done taking care not to invert the order of the output words. The third step is trivial with this arrangement. In our experiments, the alignments are obtained using the GIZA software [9, 10], which implements IBM statistical models [11, 12].

3 Interactive Search

In the previous section the training process undergone to generate a SFST $\mathcal{T}$ from a parallel corpus was described. The aim of interactive search is to find a suffix of target sentence $\hat{\mathbf{t}}_s$ that maximizes the *a posteriori* probability given a SFST $\mathcal{T}$, a source sentence $\mathbf{s}$ and a prefix of the target sentence $\mathbf{t}_p$ produced by a human translator:

$$\hat{\mathbf{t}}_s = \underset{\mathbf{t}_s}{\operatorname{argmax}} \Pr(\mathbf{t}_s \mid \mathbf{s}, \mathbf{t}_p) \approx \underset{\mathbf{t}_s}{\operatorname{argmax}} \Pr_{\mathcal{T}}(\mathbf{t}_p \mathbf{t}_s, \mathbf{s}) \tag{2}$$

This equation is similar to the one for general translation but in this case, the optimization is performed over the set of target suffixes rather than the set of complete target sentences.

The solution to this problem has been devised in two phases. The first phase copes with the extraction of a word graph $\mathcal{W}$ from a SFST $\mathcal{T}$ given a source sentence $\mathbf{s}$. In a second phase, the search of the best translation (or translations) is performed over the word graph $\mathcal{W}$.

[1] The previous name of this technique was MGTI - Morphic-Generator Transducer Inference.

3.1 Word Graph Derivation

A word graph is a compact representation of all the possible translations that a SFST $\mathcal{T}$ can produce from a given source sentence **s** together with the probabilities of those translations. In fact, the word graph could be seen as a kind of weighted finite state automaton in which the probabilities are not normalized.

The construction of the word graph is reminiscent of the intersection of two automata: a SFST and a linear DFA (deterministic finite state automaton) representing the input sentence. We will explain it through a simple example. Assume that we have to translate the sentence "haga clic en siguiente ." (click next) using the SFST of Figure 1. The first step is to build the DFA of Figure 2. It is easy to see that the DFA has as many states as words in the source sentence plus one and the ith word of the sentence connects states $i-1$ and i.

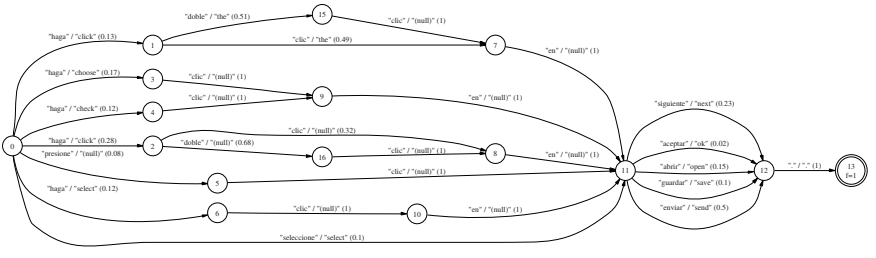

Fig. 1. A transducer inferred from a parallel corpus

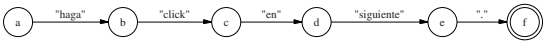

Fig. 2. A DFA representing the sentence: "haga clic en siguiente."

From these two automata, we can easily build the word graph. For the moment, assume that the output of the arcs of the SFST have at most one word. The states of the word graph will be pairs composed of a state of the DFA and a state of the SFST. The initial state of the word graph will be the pair composed of the initial states of those automata. The probability of a pair being final will be the final probability of the corresponding state in the SFST. Now, assume that p and r are states of the DFA, p' and r' are states of the SFST and that there is an arc from p to r with input w in the DFA and another arc in the SFST from p' to r' with input w and output y. Then, the word graph will have an arc from $q \equiv (p, p')$ to $q' \equiv (r, r')$ with input y (remember that the word graph represents sentences of the output language, i.e. possible translations). This arc will have the same probability as the arc (p', w, y, r') in the SFST that we will denote as $P(q, y, q')$. The final-state probability of each state q will be denoted as $P_F(q)$. The result of this process in our example can be seen in Figure 3.

There are a couple of minor issues to deal with in this construction. On one hand, the output symbol for a given arc could be empty string (which are represented by "(null)" in the Figures) or could contain more than one word. Since the word graph generated

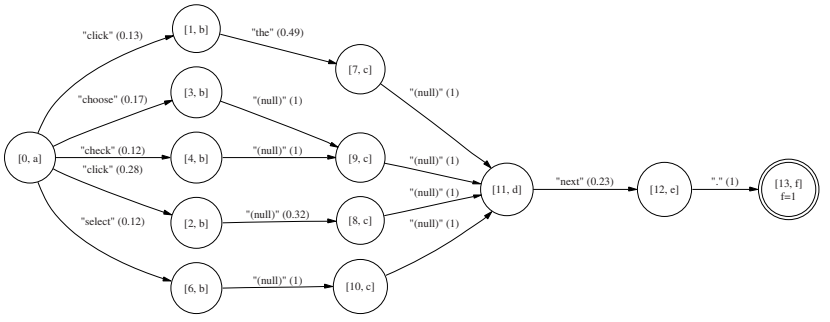

Fig. 3. Word graph resulting from the SFST in Figure 1 and the DFA in Figure 2. Isolated states are not shown

is not deterministic, the inclusion of empty outputs coming from the SFST is integrated easily. In the case of arcs with more than one word, auxiliary states were created in order to assign only one word for each arc. On the other hand, it is possible to have words in the input sentence that do not belong to the input vocabulary in the SFST. This problem is solved with the introduction of a special "unknown word" in the input vocabulary of the SFST.

3.2 Search of n-Best Translations Given a Prefix of the Target Sentence

Once the word graph is constructed, it can be used for finding the best completions for the part of the translation typed by the human translator. Not that the word graph depends only on the input sentence, so it is used repeatedly for finding the completions of all the different prefixes provided by the translator.

Ideally, the task would be to find the target suffix $\mathbf{t}_s$ that maximizes the probability *a posteriori* given a prefix $\mathbf{t}_p$ of the target sentence and the input sentence. In practice, however, it may happen that $\mathbf{t}_p$ is not present in the word graph $\mathcal{W}$. The solution is to use not $\mathbf{t}_p$ but a prefix $\mathbf{t}'_p$ that minimizes the edition distance with $\mathbf{t}_p$ and is compatible with $\mathcal{W}$. Therefore, the score of a target translation $\mathbf{t} \equiv \mathbf{t}_p \cdot \mathbf{t}_s$ is characterized by two functions, the edition cost between the target prefix $\mathbf{t}_p$ and the optimal prefix $\mathbf{t}'_p$ found in the word graph $\mathcal{W}$ and the *a posteriori* probability of $\mathbf{t}_s$ ($Pr(\mathbf{t}_s \mid \mathbf{t}'_p)$). However, the list of n-best translations has been prioritized first by minimum edition cost and then by *a posteriori* probability to value more significantly those translations that were closer to the user preferences.

Let q_p be the state(s) in $\mathcal{W}$ that is (are) reached from the initial state using $\mathbf{t}'_p$ and let $\mathcal{P}(\mathcal{W}, \mathbf{t}_p, q_p)$ be the set of possible paths $(q_0, t_1, q_1), \ldots, (q_{m-1}, t_m, q_m)$ in $\mathcal{W}$ from $q_0 = q_p$ that produce the translation suffix $\mathbf{t}_s = t_1, \ldots, t_m$ of length m . $Pr(\mathbf{t}_s \mid \mathbf{t}'_p)$ is calculated as:

$$Pr(\mathbf{t}_s \mid \mathbf{t}'_p) = \sum_{\mathcal{P}_m \in \mathcal{P}(\mathcal{W}, \mathbf{t}_s, q_p)} \prod_{1 \le i \le m} P(q_{i-1}, t_i, q_i) P_F(q_m) \tag{3}$$

The search for the $\mathbf{t}_s$ that maximize $Pr(\mathbf{t}_s \mid \mathbf{t}'_p)$ has been demonstrated to be an NP-hard problem, so the Viterbi approach [6] will be adopted to make feasible the calculation of $Pr(\mathbf{t}_s \mid \mathbf{t}'_p)$, that is:

$$\widehat{Pr}(\mathbf{t}_s \mid \mathbf{t}'_p) = \max_{\mathcal{P}_m \in \mathcal{P}(\mathcal{W}, \mathbf{t}_s, q_p)} \prod_{1 \leq i \leq m} P(q_{i-1}, t_i, q_i) P_F(q_m) \tag{4}$$

This simplification is imperative because of the real-time constraints under which our prototype is required to run. However, a better approximation to $Pr(\mathbf{t}_s \mid \mathbf{t}'_p)$ will be presented in next section.

The algorithm proposed to solve this search problem is an adapted version of the Recursive Enumeration Algorithm (REA) described in [7] that integrates the minimum edition cost algorithm in the search procedure. This algorithm consist on two parts:

– Forward search that calculates the 1-best path from the initial state q_0 to every state in the word graph $\mathcal{W}$. Paths in the word graph are weighted not only based on their *a posteriori* probability, but also on their edition cost respect to the target sentence prefix.

  To this purpose, ficticious edges have been inserted into the word graph to represent edition operations like insertion, substitution and deletion. These edition operations have been included in the word graph in the following way:

  - **Insertion:** An insertion edge has been inserted as a loop for each state in the word graph.
  - **Deletion:** A deletion edge is added for each arc in the word graph having the same source and target state than its sibling arc.
  - **Substitution:** Each arc in the word graph is treated as a substitution edge whose edition cost is proportional to the levenshtein distance between the symbol associated with this arc and the word prefix employed to traverse this arc during the search.

  For example, in Figure 3 if the user would type "press" as an initial prefix, we would have two different classes of translation sentences. Those ones that applying an insertion operation would start from the initial stage at state $[0, a]$, and those ones that applying a deletion or substitution operation would depart from states in the second stage.

– Backward search that enumerates candidates for the k-best path along the $(k-1)$-best path. This recursive algorithm defines the next best path that arrives at a given state q as the next best path that reaches q' plus the arc leaving from q' to q, being $(q', b, q) \in E$. If this next best path arriving at state q' has not been calculated yet, then the next best path procedure is called recursively until a 1-best path is found or no best paths are found.

To reduce the computational cost of the search, the beam-search technique has been implemented. During the word graph construction, two beam coefficients were employed to penalize those edges leading to backoff states over those ones arriving at

normal states. Finally, a third beam coefficient controls how far in terms of number of edition operations a hypothesis could be from the best hypothesis in a given stage during the parsing procedure.

3.3 An Improved Approximation to the Translation Probability

A better approximation to the true translation probability of equation (3) can be obtained on the base of the Viterbi n-best path approach. This approximation is achieved by summing up the probability of those paths with the same translation $\mathbf{t}_s$ in the set of n-best paths $\mathcal{P}_N(\mathcal{W}, \mathbf{t}_s, q_p)$:

$$Pr_N(\mathbf{t}_s \mid \mathbf{t}'_p) = \sum_{\mathcal{P}_m \in \mathcal{P}_N(\mathcal{W}, \mathbf{t}_s, q_p)} \prod_{1 \leq i \leq m} P(q_{i-1}, t_i, q_i) P_F(q_m) \tag{5}$$

Indeed, the Viterbi approximation could be considered to be a particular case of the n-best approximation, that is, when the number of paths is 1. As the reader may expect, this approximation improves as the size of $\mathcal{P}_N(\mathcal{W}, \mathbf{t}_s, q_p)$ increases. However it should be noted that $Pr_N(\mathbf{t}_s \mid \mathbf{t}'_p)$ follows a log-wise growth, since most of the probability associated with the translation $\mathbf{t}_s$ is accumulated in the first n-best paths.

4 Experimental Results

4.1 Corpus Features

We performed experiments using the so-called Xerox corpus [13]. This corpus consists in a collection of technical Xerox manuals written in English, Spanish, French and German. The English versions are the original and the rest are translation of them. The sizes (in thousands of words) of the subsets used can be seen in Table 1.

Table 1. Features of the Xerox Corpus: training, vocabulary and test sizes are measured in thousands of words

	EN / ES	EN / DE	EN / FR
TRAINING	600/700	600/500	600/700
VOCABULARY	26 / 30	25 / 27	25 / 37
TEST	8 / 9	9 / 10	11 / 10
PERPLEXITY (3-gram)	107/60	93/169	193/135

4.2 Translation Quality Evaluation

We have used three different measures in order to assess the techniques presented:

1. *Translation Word Error Rate* (TWER). It is defined as the minimum number of word substitution, deletion and insertion operations to convert the target sentence provided by the transducer into the reference translation. Also known as edit distance.

2. *Character Error Rate* (CER). Edit distance in terms of characters between the target sentence provided by the transducer and the reference translation.
3. *Key-Stroke Ratio* (KSR). Number of key-strokes that are necessary to achieve the reference translation plus the acceptance key-stroke divided by the number of running characters.

These experiments were performed with GIATI transducers based on trigrams. The results are shown in Table 2. On the leftmost column appears the language pair employed for each experiment, English (En), Spanish (Es), French (Fr) and German (De). The main two central columns compare the results obtained with 1-best translation to 5-best translations. In the latter case, the target sentence out of the five suggested translations that minimizes most the correspondent error measure was selected.

Table 2. Results for the Xerox Corpus comparing 1-best to 5-best translations

	GIATI 3-gram (1-best)			GIATI 3-gram (5-best)		
XRCE 2	KSR	CER	TWER	KSR	CER	TWER
En-Es	29.1	30.3	43.1	26.2	25.0	37.8
Es-En	33.5	35.5	51.4	29.7	28.1	45.2
En-Fr	58.5	54.3	73.8	53.7	48.5	69.6
Fr-En	58.4	55.3	71.9	54.0	49.5	67.7
En-De	66.2	62.8	81.3	60.1	56.7	77.2
De-En	59.0	61.5	78.5	53.9	55.1	73.3

The best results were obtained between English and Spanish language pairs, in which the human translator would only need to type 25% of the total reference sentences. In theory, this could result in a factor of 4 increase in the productivity of human translators.

Furthermore, in all cases there is a clear and significant improvement in error measures when we move from 1 to 5-best translations. This gain in translation quality diminishes in a log-wise fashion as we increase the number of best translations. Pair of languages as English and French present somewhat higher error rates, as is also the case between English and German, reflecting the complexity of the task faced in these experiments.

4.3 A Comparative Evaluation: Viterbi vs. n-Best Approximation

An approximation to the true translation probability based on the n-best path was introduced in section 3.3. Some experiments were performed to assess the evolution of the translation quality as the calculation of the translation *a posteriori* probability improves. To this purpose a simplified version of the Xerox corpus was employed to reduce the impact of noise due to preprocess and postprocess phases.

The most important conclusion that could be extracted from these results is the adequacy of the simpler and direct Viterbi approach as an approximation to the actual *a posteriori* probability of a target sentence. As it can be observed from Table 3, the evolution of TWER rates across an increasing number of n-best translations does not

Table 3. TWER comparative table across different number of n-best paths based on a simplified version of XRCE2

TWER	Viterbi	5-best	10-best	20-best	50-best	100-best	200-best	500-best	1000-best
En-Es	31.7	31.9	32.1	32.0	32.2	32.3	32.3	32.4	32.4
Es-En	35.9	35.3	35.4	35.5	35.6	35.7	35.7	35.7	35.7
En-Fr	60.7	60.7	61.0	61.0	60.8	60.7	60.7	60.8	60.8
Fr-En	56.1	57.0	57.0	57.1	57.0	57.2	57.1	57.2	57.3
En-De	69.7	69.8	69.7	69.8	69.8	69.7	69.7	69.8	69.9
De-En	63.1	63.2	63.3	63.5	63.5	63.4	63.6	63.6	63.6

show a consistent positive growth of the translation quality. It is even negative for large n in most cases. A possible reason for these results is that for large n translations with lower quality are more frequent among the set of n-best translations, so summing up the probability of equal translations favors those ones that even being less probable have more repetitions.

5 Conclusions and Future Work

Finite-state transducers can be used for computer assisted translation. These models can be learned from parallel corpus, but the number of states/transitions can be too high. The concept of interactive search has been introduced in this paper along with some efficient techniques (word graph derivation and n-best) that solve the parsing problem given a prefix of the target sentence undeolve the parsing problem given a prefix of the target sentence under real-time constraints.olve the parsing problem given a prefix of the target sentence under real-time constraints.

The promising results achieved in the first experiments provide a new field in machine translation still to be explored, in which the human expertise is combined with automatic translation techniques to increase productivity without sacrificing high-quality translation.

Moreover, an alternative approach to Viterbi approximation based on the n-best idea was explained and the results obtained with it confirm the appropriateness of the Viterbi approach in real applications.

Finally, the introduction of morpho-syntactic information and/or bilingual categories in finite-state transducers are topics that leave an open door to future research.

Acknowledgements

This work has been supported by the European Union under the IST Programme (IST-2001-32091) and the Spanish project TIC2003-08681-C02-02.

References

1. Langlais, P., Foster, G., Lapalme, G.: Unit completion for a computer-aided translation typing system. Machine Translation **15** (2000) 267–294
2. Amengual, J.C., Benedí, J.M., Castano, A., Castellanos, A., Jiménez, V.M., Llorens, D., Marzal, A., Pastor, M., Prat, F., Vidal, E., Vilar, J.M.: The EuTrans-I speech translation system. Machine Translation **15** (2000) 75–103

3. Casacuberta, F., Llorens, D., Martínez, C., Molau, S., Nevado, F., Ney, H., Pastor, M., Picó, D., Sanchis, A., Vidal, E., Vilar, J.M.: Speech-to-speech translation based on finite-state transducers. In: International Conference on Acoustic, Speech and Signal Processing. Volume 1., IEEE Press (2001)
4. Vidal, E.: Finite-state speech-to-speech translation. In: Int. Conf. on Acoustics Speech and Signal Processing (ICASSP-97), proc., Vol.1, Munich (1997) 111–114
5. Picó, D., Casacuberta, F.: Some statistical-estimation methods for stochastic finite-state transducers. Machine Learning **44** (2001) 121–142
6. Viterbi, A.: Error bounds for convolutional codes and a asymtotically optimal decoding algorithm. IEEE Transactions on Information Theory **13** (1967) 260–269
7. Jiménez, V.M., Marzal, A.: Computing the k shortest paths: a new algorithm and an experimental comparison. In Vitter, J.S., Zaroliagis, C.D., eds.: Algorithm Engineering. Volume 1668 of Lecture Notes in Computer Science., London, Springer-Verlag (1999) 15–29
8. Casacuberta, F., Ney, H., Och, F.J., E. Vidal, J.M.V., Barrachina, S., Garcia-Varea, I., D. Llorens, C.M., Molau, S., Nevado, F., Pastor, M., Pico, D., Sanchis., A.: Some approaches to statistical and finite-state speech-to-speech translation. Computer Speech and Language **18** (2004) 25–47
9. Al-Onaizan, Y., Curin, J., Jahr, M., Knight, K., Lafferty, J., Melamed, D., Och, F.J., Purdy, D., Smith, N., Yarowsky, D.: Statistical machine translation (1999)
10. Och, F.J., Ney, H.: Improved statistical alignment models. In: ACL00, Hongkong, China (2000) 440–447
11. Brown, P.F., Cocke, J., Pietra, S.D., Pietra, V.J.D., Jelinek, F., Lafferty, J.D., Mercer, R.L., Rossin, P.S.: A statistical approach to machine translation. Computational Linguistics **16** (1990) 79–85
12. Brown, P.F., Pietra, S.D., Pietra, V.J.D., Mercer, R.L.: The mathematics of statistical machine translation: Parameter estimation. Computational Linguistics **19** (1993) 263–312
13. SchlumbergerSema S.A., de Informática, I.T., für Informatik VI, R.W.T.H.A.L., en Linguistique Informatique Laboratory University of Montreal, R.A., Soluciones, C., Gamma, S., Europe, X.R.C.: TT2. TransType2 - computer assisted translation. Project technical annex. (2001)