The Tenjinno Machine Translation Competition

Bradford Starkie[1], Menno van Zaanen[2], and Dominique Estival[3]

[1] Starkie Enterprises Pty. Ltd.,
52 Boisdale Street, Surrey Hills (Melbourne), Victoria 3127 Australia,
bstarkie@starkieenterprises.com,
http://www.bradstarkie.id.au
[2] Macquarie University,
North Ryde (Sydney), NSW 2109 Australia,
menno@ics.mq.edu.au,
http://www.ics.mq.edu.au/~menno
[3] Appen Pty. Ltd.,
Level 6, North Tower 1, Railway Street, Chatswood (Sydney), NSW 2067 Australia
destival@appen.com.au,
http://www.ics.mq.edu.au/~destival

Abstract. This paper describes the Tenjinno Machine Translation Competition held as part of the International Colloquium on Grammatical Inference 2006. The competition aimed to promote the development of new and better practical grammatical inference algorithms used in machine translation. Tenjinno focuses on formal models used in machine translation. We discuss design issues and decisions made when creating the competition. For the purpose of setting the competition tasks, a measure of the complexity of learning a transducer was developed. This measure has enabled us to compare the competition tasks to other published results, and it can be seen that the problems solved in the competition were of a greater complexity and were solved with lower word error rates than other published results. In addition the complexity measures and benchmark problems can be used to track the progress of the state-of-the-art into the future.

1 Introduction

This paper describes the Tenjinno machine translation competition, held in conjunction with the eighth International Colloquium on Grammatical Inference (ICGI 2006). The competition aimed to measure and improve upon the current state-of-the-art in grammatical inference. Tenjinno was the successor to the earlier Abbadingo [1], Gowachin and Omphalos [2, 3] competitions.

At the ICGI 2004 it was decided that it would be desirable to have a competition task that more closely resembles a real world application compared to earlier competitions; hence the task of machine translation was chosen. However, although described as a machine translation competition, it was in fact a transducer inference competition.

Machine translation has obvious applications in the translation of human languages, but the problem of translation is more general and has applications beyond natural language. For instance, it is a cornerstone of genomics that the translation of DNA into living things is a predominately deterministic process. For this reason, we invited submission from practitioners from all areas of computer science including machine learning, natural language processing, formal languages, machine translation and bioinformatics.

The competition differed from other machine translation competitions such as the DARPA machine translation competitions [4] in the following important ways:

- The languages were artificial and were derived from formal models.
- We expected that competitors would infer the model used to derive the data exactly and the metric used to measure the winner reflected this expectation.

In Section 2, we present the task we chose for the competition and describe the two formalisms that were used. In Section 3, we describe how the competition was set up and explain the measure used to estimate the current state-of-the art, and we also explain the name Tenjinno. In Section 4, we give properties of the problems that were proposed and their current status.

2 Task Description

Essentially, we generated several automatic machine translation systems that were based on finite state transducers and syntax directed translation schemas. Using these formal models, we generated sentences in the source language and corresponding translations in the target language. This generated data, containing both source and target sentences aligned on a sentence by sentence basis, was provided to the competitors who used these sentences to train their translation systems.

Test data, containing only source sentences, was also provided. The task for the competitors was to generate the corresponding target sentences. The translated sentences could then be submitted to the competition Oracle, which determined whether or not the problem had been solved.

There were four problems proposed on the web site and these were ranked according to the difficulty of the translation process. At the close of the competition, the competitor who had solved the highest-ranking problem would be declared the winner of the competition overall. The two lowest ranking problems were to translate sentences produced from Finite State Transducers (FSTs), while the higher ranking problems were to translate sentences produced from Syntax Directed Translation Schema (SDTS).

Finite State Transducers (FST) were chosen as one of the formalisms for the Tenjinno competition mainly because there has been a history of research into the inference of FSTs published at previous ICGIs. The SDTS formalism was chosen because, unlike FSTs, SDTSs are expressive enough to define the reordering of constituents when translating from one language to another.

Reordering of constituents is a common task when translating between natural languages and its formalisation is an important issue for Machine Translation.

For instance, it is often necessary to translate from Subject Verb Object (SVO) languages into Subject Object Verb (SOV) languages, with arbitrarily large constituents as either subjects or objects. However, the inference of SDTSs has not been well researched yet and it was hoped that the Tenjinno competition would spur interest into this field.

In the Tenjinno competition, some of the transducers were deterministic and some were non-deterministic, and labelled as such. The problems labelled non-deterministic were ranked higher (i.e. more difficult) than those labelled deterministic. This distinction was made because it has been shown in previous competitions [2, 3] that the inference of non-deterministic grammars is a more difficult task.

Where the transducers were non-deterministic, they can be shown to be unambiguous. This was a deliberate design choice of the competition, due the fact that the number of translations for a sentence can quickly become significantly large when the transducer is ambiguous.

2.1 Finite State Transducers

A Finite State Transducer (FST) is a formalism that can translate strings to strings directly using a finite state machine.

An FST (which can also be seen as a regular syntax-directed translation scheme) [5] T is a tuple $\langle N, \Sigma, \Delta, R, I \rangle$ where N is a finite set of non-terminal symbols or states, I is the initial state, Σ is a finite set of input terminal symbols and Δ is a finite set of output terminals.

R is the set of rules which can take one of the two following forms:

- $A \rightarrow \mathrm{a}B, \mathrm{w}B$ for $A, B \in N, \mathrm{a} \in \Sigma, \mathrm{w} \in \Delta^*$ or
- $A \rightarrow \mathrm{a}, \mathrm{w}$ for $\mathrm{a} \in \Sigma, \mathrm{w} \in \Delta^*$.

The right-hand side of rules are separated into two parts by a comma. Each of the parts represents information about a language; the source language first, followed by the target language. Table 1 gives an example of a finite state transducer.

Table 1. A non-deterministic Finite State Transducer.

$S \rightarrow$una A, a A
$S \rightarrow$la A, the A
$A \rightarrow$camera B, room B
$A \rightarrow$camera, room
$A \rightarrow$camera C, C
$B \rightarrow$doppia, with two beds
$C \rightarrow$doppia, double room
$C \rightarrow$singola, single room

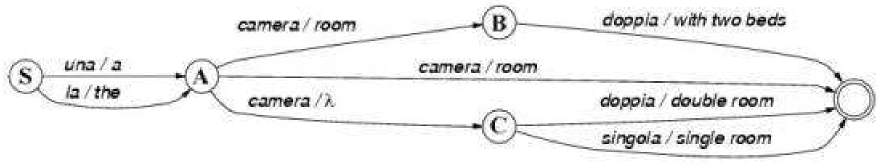

Fig. 1. The Finite State Transducer of Table 1 shown diagrammatically.

An input string “x” can be mapped to the output string “y” denoted (x, y) if there is a translation from $(S, S) \Rightarrow (\mathrm{x}_1 A_1, \mathrm{y}_1 A_1) \Rightarrow (\mathrm{x}_1\mathrm{x}_2 A_2, \mathrm{y}_1\mathrm{y}_2 A_2) \Rightarrow \cdots \Rightarrow (\mathrm{x}, \mathrm{y})$ where $\Rightarrow$ denotes the application of a rule.

If there is a translation from (S, S) to (x, y), (x, y) is known as a translation pair. For instance, given the FST shown in Table 1, (una camera doppia, a double room) can be shown to be a translation pair as follows: $(S, S) \Rightarrow (\mathrm{una}A, \mathrm{a}A) \Rightarrow (\mathrm{una\ camera}C, \mathrm{a}C) \Rightarrow$ (una camera doppia, a double room).

We say that a finite transducer is deterministic if, for each rule of the form $A \rightarrow \mathrm{a}B, \mathrm{w}B$ where $\mathrm{a} \in \Sigma \cup \{\epsilon\}$, $\mathrm{w} \in \Delta^*$ and $B \in N \cup \{\epsilon\}$, there does not exist any other rule of the form $A \rightarrow \mathrm{a}C, \mathrm{h}C$ where $C \in N \cup \{\epsilon\}$ and $\mathrm{h} \in \Delta^*$. Although every language that is accepted by a non-deterministic finite-state automaton is also accepted by a deterministic finite-state automaton, there exist non-deterministic finite-state transducers for which there is no equivalent deterministic finite-state transducer [6].

2.2 Syntax Directed Translation Schema

A Syntax-Directed Translation Schema (SDTS) is a transducer in which both the input and output languages are described using a context-free grammar, and there is a one to one mapping between parse trees in the input language and parse trees in the output language, and similarly a one to one mapping between parse trees in the output language and parse trees in the input language.

An SDTS [7] T is a tuple $\langle N, \Sigma, \Delta, R, I\rangle$ where N is a finite set of non-terminal symbols or states, I is the initial state, Σ is a finite set of input terminal symbols and Δ is a finite set of output terminals.

R is the set of rules which can take on the form $A \rightarrow \mathrm{a}, \mathrm{b}$ where $\mathrm{a} \in (N \cup \Sigma)^*$ and $\mathrm{b} \in (N \cup \Delta)^*$ and the non-terminals in “a” are a permutation of the non-terminals in “b”. If a non-terminal B appears more than once in “a” or “b” then we use superscripts to indicate the association. This association is an intimate part of the rule. For example, in the rule $A \rightarrow B^{(1)}CB^{(2)}, B^{(2)}B^{(1)}C$. Table 2 gives an example of a syntax directed translation schema.

From Table 2, it can be seen that $(0\ 0\ 1\ 1\ 1, \mathrm{b\ b\ b\ a\ a})$ is a translation pair as follows:

Table 2. A simple Syntax-Directed Translation Schema.

$S \rightarrow 0AS, SA\mathrm{a}$
$A \rightarrow 0SA, AS\mathrm{a}$
$S \rightarrow 1, \mathrm{b}$
$A \rightarrow 1, \mathrm{b}$

$(S, S) \Rightarrow (0AS, SA\mathrm{a}) \Rightarrow (0\ 0SAS, SAS\mathrm{a\ a}) \Rightarrow (0\ 0\ 1AS, SA\mathrm{b\ a\ a}) \Rightarrow$
$\Rightarrow (0\ 0\ 1\ 1S, S\mathrm{b\ b\ a\ a}) \Rightarrow (0\ 0\ 1\ 1\ 1, \mathrm{b\ b\ b\ a\ a})$. Note that in this example, the input sequence is always leftmost expanded, but the output sequence is not.

It can be seen that finite state transducers are a sub-class of simple syntax-directed translation schemas [8]. It can also be seen that input and output languages defined by a FST must be regular, but in the case of SDTSs the input and output languages can be context-free and not regular. This enables SDTSs to translate individual words in an input text according to long range dependencies between words in the input text, whereas such dependencies cannot be represented using a FST. For example, consider the SDTS shown in Table 3. In this example, whether or not the word "b" is translated into an "f" or a "g" depends upon whether or not either an "a" or a "c" was presented earlier in the input text. Here, $(S, S) \Rightarrow^* (\mathrm{a\ c\ d\ b}, \mathrm{j\ f})$ and $(S, S) \Rightarrow^* (\mathrm{c\ c\ d\ b}, \mathrm{j\ g})$.

Table 3. An SDTS that is not a FST.

$S \rightarrow \mathrm{a}S\mathrm{b}, S\mathrm{f}$
$S \rightarrow \mathrm{a\ b}, \mathrm{f}$
$S \rightarrow \mathrm{c}S\mathrm{b}, S\mathrm{g}$
$S \rightarrow \mathrm{c\ b}, \mathrm{g}$
$S \rightarrow \mathrm{a}S\mathrm{d}, S\mathrm{h}$
$S \rightarrow \mathrm{a\ d}, \mathrm{h}$
$S \rightarrow \mathrm{c}S\mathrm{d}, S\mathrm{j}$
$S \rightarrow \mathrm{c\ d}, \mathrm{j}$

3 Competition Philosophy

In this section, we describe the underlying philosophy we followed while designing the Tenjinno competition and define the requirements of the competition. We also discuss how we ranked the competition problems and explain the measure we used to estimate the current state-of-the-art. Finally, we explain how Tenjinno, the name of the competition, was derived.

3.1 Requirements

The target transducers and the training and testing examples were created with the following objectives in mind:

1. The learning task should be sufficiently difficult. Specifically, the problems should be just outside of the current state-of-the-art, but not so difficult that it is unlikely that a winner would be found.
2. It should be provable either that the training sentences are sufficient to identify the target transducer or that it is at least possible to give an estimation of the probability of the training sentences being sufficient.

To determine whether or not requirement 1 was met, a measure of the complexity of the learning task was derived, similar to that used for the Omphalos Competition [2, 3]. The measure was derived by creating a model of the learning task based upon a brute force search. In this model, the learner is presented with an infinite stream of translation pair sets, where each set consists of a string in the input language and all translations of that sentence. After each presentation the learner constructs a set of hypothetical SDTS each of the form $\langle N, \Sigma, \Delta, R, I\rangle$ where Σ is the set of input terminal symbols observed in the translation pair sets and Δ is the set of output terminals observed in the translation pair sets, and no rule in R is longer than the length of any sequence observed in the translation pair sets. The brute force learner initially assumes that $|N| = 1$ and constructs all possible SDTSs. The learner then discards all SDTSs that are inconsistent with the observed translation pair sets. If no SDTS remains in the hypothesis set, then the number of non-terminals is increased and the process is repeated until at least one SDTS consistent with the observed translation pair sets is found.

This process can be shown to terminate because, for any given number of non-terminals, there are a finite number of possible SDTSs and it is known that there exists at least one SDTS that is consistent with the observed translation pair sets (i.e. the SDTS used to generate the translation pair sets). When there are more than one candidate SDTS, the learner selects one according to any linear ordering where one SDTS is considered more favourable than another if it contains fewer non-terminals and shorter rules.

It can be seen that for any SDTS, when the set of observation pairs used to train the model includes at least one translation pair generated for each of the rules, it causes the learning algorithm to construct a set of hypothesis SDTSs that includes the target SDTS. If a selected hypothesis is inconsistent with the target SDTS, then at some finite point in time a translation pair set will be presented to the learner that causes the learner to discard the incorrect hypothesis.

Therefore, provided enough translation pairs are presented to this algorithm (and an infinite amount of memory and computing time is available), this algorithm will always identify the target SDTS. The size of the hypothesis space that needs to be created by such a learning algorithm forms the basis of the complexity measure. This model is also used to determine whether the training data is sufficient to enable competitors to identify the target transducer. An upper bound on the number of samples required is derived using PAC learning theory [9], but it was decided to construct significantly smaller test sets than is suggested using the PAC learning equations, based upon the knowledge that the upper bounds indicated from the PAC learning equations assume that all hypotheses in the hypothesis space are mutually exclusive, which is not the case for the brute force algorithm.

3.2 The Ranking of the Competition Problems

The Tenjinno competition adopted a linear ordering for the competition problems based upon the formalism used, the complexity of the problems, and whether or not the formal model from which the data was derived is deterministic.

It is well understood that the class of transducers that can be described using FSTs is a proper subset of the class of transducers that can be described using SDTSs [7]. Similarly, it is well known that the class of transducers that can be described using deterministic FSTs is a subset of the class of transducers that can be described using non-deterministic FSTs. The class of transducers that can be described using deterministic SDTSs is also a subset of the class of transducers that can be described using non-deterministic SDTSs [6].

Solving the problem generated from a non-deterministic model is considered to be more difficult, therefore, if both problems were solved, the competitor who had solved the non-deterministic problem would be declared the winner.

In addition, a complexity measure for each learning task was constructed to enable a comparison between the competition problems and problems with well known benchmark corpora. In addition, during the life of the competition the organisers reserved the right to add more difficult problems or easier problems, as indicated by the complexity measure.

Table 4 describes the equations used to measure the complexity of the competition problems. Different measures are used for the different classes of problems.

Because the FST equation is the simpler of the two, we will describe how it was derived but leave it to the reader to determine why the SDTS equation describes the number of possible SDTS, given the input parameters.

The complexity measure for the FST tasks describes an upper bound on the number of possible FSTs that could be constructed given a finite set of input symbols, output symbols and non-terminals of size σ, δ and n respectively, where there are at most L non-terminal and output symbols on the right hand side of the longest rule.

Firstly it can be seen that if, given the input parameters, there are R possible rules, then the number of possible FSTs is 2^R. This is because in any given FST each possible rule can either be present or not. Therefore the exponent of the complexity measure describes the number of possible rules. Each rule has one non-terminal on the left hand side which can take on n different values, and one input symbol from the input language on the right hand side which can take on one of σ different values, hence the exponent begins with $n \times \sigma$. The right hand side can have either 0 or 1 non-terminals. We consider both cases by adding both possibilities using the summation $\sum_{j=0}^{1}$ where j describes the number of non-terminals. There are n choices of non-terminals hence there are n^j possible ways of selecting j non-terminals. Note that when there are 0 non-terminals $n^j = 1$. If the right hand side has j non-terminals there can be up to $k = L - j$ terminal symbols on the right hand side. There are δ choices of output terminals, and these terminals can be repeated. Therefore there are δ^k ways of selecting them. Note that L defines the maximum length of a right-hand side; there can be less than L symbols. We can consider all possibilities using the summation $\sum_{k=0}^{L-j} \delta^k$. Hence the number of possible FSTs given the input parameters is $2^{n \times \sigma \times \sum_{j=0}^{1} (n^j \times \sum_{k=0}^{L-j} \delta^k)}$.

Similarly, the complexity measure for the SDTS tasks describes an upper bound on the number of possible SDTSs that could be constructed given a finite set of input symbols, output symbols and non-terminals of size s, d and n respectively, where there are at most L symbols on the right hand side of the longest rule. Specifically the number of non-terminals symbols and output symbols on any given rule is less than L, and the number of non-terminals symbols and input symbols on any given rule is less than L.

Table 4. Formalisms with their complexity measures.

Formalism	Complexity Measure
SDTS	$2^{n \times \sum_{i=0}^{L} (\sum_{j=0}^{i} (\binom{n+j-1}{j} \times \binom{\sigma+i-j-1}{i-j} \times i! \times \sum_{k=j}^{L} (\binom{\delta+k-j-1}{k-j} \times k!)))}$
FST	$2^{n \times \sigma \times \sum_{j=0}^{1} (n^j \times \sum_{k=0}^{L-j} \delta^k)}$

where

n=the number of non-terminals
σ=the size of the input lexicon
δ =the size of the output lexicon
L=the length of the longest rule

Construction of Non-deterministic Transducers The finite state transducer of problem 2 can be shown to be non-deterministic but unambiguous. Firstly, the transducer is non-deterministic because it has non-deterministic state transitions such as $S \rightarrow \mathrm{a\ b\ c}X, \mathrm{c}X$ and $S \rightarrow \mathrm{a\ b\ c}Z, \mathrm{b}Z$. However the correct state transition can always be determined by looking at most three characters ahead; that is, the underlying context-free backbone is LL(3), which implies that the underlying grammar is unambiguous.

The syntax directed translation schema of problem 4 was constructed as follows. First a randomly generated LR(1) grammar was constructed. Because this grammar was LR(1) it was known to be unambiguous. This grammar included center recursion to ensure that it was not regular. A construct of the form $\mathrm{a}^n\mathrm{b}^m$ where $n, m > 1$ and $m \neq n$ was then added to one of the non-terminals. this was known to be unambiguous because it was the union of two disjoint LR(1) grammars. The SDTS was constructed such the mapping from input symbols to output symbols differed depending upon whether $m > n$ or $n < m$. Any parser that can parse these sentence needs to determine whether $m > n$ or $n > m$ before deciding which rule to apply. This would require an arbitrary amount of lookahead, and thus this transducer is non-deterministic and cannot be represented by a deterministic SDTS.

3.3 Estimating the Current State-of-the-Art

To ensure that the competition problems where just beyond the state-of-the-art, the proceedings of recent conferences were examined to identify the performance of different inference techniques on some corpora commonly used for the inference of machine translation systems. The complexities of the problems in the Tenjinno competition were then set to be an order of magnitude larger than these problems. In addition, the complexities of the problems were set to be large enough that they could not be solved using a brute force search in the time in which the competition was open (i.e. there are approximately 1.57×10^{13} microseconds in a six month period).

Table 5 contains a list of some corpora commonly used for the inference of machine translation systems. This data is derived from Casacuberta [5], Amengual et al. [10], Matusov et al. [11]. Some of the earlier experiments listed here are described in Vidal and Casacuberta [12] as being indicative of the state-of-the-art. Conveniently the documents from which this table has been compiled describe experiments in which n-gram finite state transducers were inferred from the given corpora. In addition, the corpora were described in sufficient detail in those papers to allow for easy comparison between the complexity of the learning tasks and the word error rates obtained.

Although there is yet to be a published technique that can achieve a 0% word error rate on any of these corpora, it should be noted that these corpora are natural language corpora, which are unlikely to be defined using finite state transducers. Therefore, we would expect that the word error rates obtained using the techniques described in the papers on data defined using finite state transducers would be better than that listed in Table 5.

It seemed appropriate to begin with competition tasks that were of a slightly greater complexity to those listed in this table. When calculating the complexity of the underlying models we used the somewhat arbitrary assumption that it is possible to learn the structure of the underlying models from the training examples without n-gram smoothing, that is whether or not the probability of each possible transition is either zero or non-zero. Therefore we have estimated the number of non-terminals to be either one quarter of the number of words in the training examples, or the number of possible states in the given n-gram $\{(d+1)^{n-1}\}$ whichever is smallest.

We have also included a column which gives an estimate of the complexity of an SDTS that might be used to describe the target language. The SDTS measure is estimated using the given vocabularies and assuming that the target SDTS could be described using a similar number of non-terminals to the number of Penn Treebank tags. Specifically, we selected the number of non-terminals to be 36 and have arbitrarily selected the length of the longest rule to be 7.

Most of the corpora in Table 5 [5, 10, 11] have large vocabularies. Some other commonly used corpora, such as the Canadian Hansard, are used because of the close alignment between word orders in the source and target languages (e.g. French and English). However, as was mentioned earlier, word reordering is of crucial importance in translation between natural languages and it is one of the difficult problems for machine translation. For this reason, although the problems in the Tenjinno competition have a complexity similar to those listed in Table 5, the Tenjinno problems have smaller vocabularies and derive more of their complexity from non-monotonic alignment.

Using the results of Table 5 the complexities of the Tenjinno competition problems were selected to be just outside of the complexities listed in Table 6.

Table 5. Properties of commonly used corpora for the inference of machine translation systems.

Corpus	Translation pair	Source lexicon	Target lexicon	Word error rate
EuTrans-0	Spanish $\rightarrow$English	683	514	0.74%
EuTrans-I	Spanish $\rightarrow$English	686	513	9.7%
EuTrans-II	Italian $\rightarrow$English	2,459	1,712	28.3%
Verbmobil	German$\rightarrow$English	7,939	4,672	36.2%
Basic Travel Corpus	Chinese$\rightarrow$English	7,643	6,982	48%

Table 6. Estimated complexity of commonly used corpora for the inference of machine translation systems.

Corpus	$\log_2$ of FSM complexity	$\log_2$ of SDTS complexity
EuTrans-0	1.89×10^{19}(7 class n-gram)	2.30×10^{37}
EuTrans-I	2.14×10^{22}(4-gram)	3.45×10^{39}
EuTrans-II	3.16×10^{24}(4-gram)	6.87×10^{46}
Verbmobil	3.89×10^{28}(4-gram)	6.91×10^{48}
Basic Travel Corpus	3.99×10^{28}(4-gram)	8.56×10^{50}

3.4 The name Tenjinno

Since the conference was held in Japan, the ICGI competition committee decided to have a Japanese inspired name for the ICGI 2006 competition and that ideally the name should be related to language and learning. Using these requirements, we decided to use the Japanese name "Tenjin" as the basis for the name of the competition. Tenjin is the Shinto kami of scholarship, the deified Sugawara no Michizane.

Tenjin is a word that appeared frequently on the Internet at the time of the competition (645,000 hits on Google, measured before the start of the competition) and therefore we have added the Japanese particle "no" to the end of the word, so that it became the "Tenjin no" competition, or loosely Tenjin's competition. Although "no" should be a separate word, by conjugating it with Tenjin (i.e. Tenjinno), we created something more unique. For instance at the time of the competition, "Tenjin no" returned 97,000 hits, "Tenjin-no" 1,060 hits, but "Tenjinno" returned only 12 hits, and the website of the Tenjinno competition was the highest ranked result.

4 Participation and Results

The Tenjinno Competition started on 3rd January 2006. The website (`http://www.ics.mq.edu.au/~tenjinno/`) had been made available a few days before, but the oracle came online on 3rd January. In total, while the competition was open, the website received 2,550 visits. The datasets were downloaded by users from 63 different sites. Overall, users from 200 sites have accessed the website.

Table 7 lists the competition problems, along with their complexities and the status of each problem i.e. whether or not the problem was solved. The problems are listed in order of increasing difficulty. The winner of the competition overall was the competitor that solved the highest ranking problem according to this table.

Table 7. Properties of the problems of Tenjinno.

Problem	Formalism	$\log_2$ Complexity	Status
1	Deterministic FST	7.27×10^{31}	Solved
2	Non Deterministic Unambiguous FST	7.27×10^{31}	Unsolved
3	Deterministic SDTS	7.24×10^{34}	Unsolved
4	Non Deterministic Unambiguous SDTS	8.28×10^{34}	Unsolved

Problem 1 was solved by Alexander Clark (Royal Holloway University of London) on 6th April 2006, who was also the winner of the Omphalos competition [2]. At the closing of the competition on 1st July 2006 the remaining problems (2, 3, and 4) were unsolved and thus Alexander Clark was declared the winner overall.

5 Conclusions

In contrast to previous ICGI competitions the problems in the Tenjinno competition mirrored a real world task. Set in the context of machine translation,

Table 8. Important dates for the Tenjinno competition.

Date	Event
31 December 2005	Competition details available on the website
1 January 2006	Competition begins
1 July 2006	Competition closes
2 July 2006	Competition winner announced
September 2006	Tenjinno session at ICGI-2006

the task was to learn a translation between sentences in two artificial paired languages. This task is interesting from a formal perspective, where one may try different approaches, for example learning grammars for both languages and finding a mapping between them or learning one grammar and tree operations on the resulting derivation. It is also interesting from an application perspective, because machine translation has always been an important research topic in natural language processing and machine translation applications have become more successful recently and have gained more visibility.

For the purpose of setting the competition tasks, extensions to the complexity measure used in the Omphalos competition were used and are described in this paper. These measures have enabled us to compare the competition problems with other published results in the field of machine translation. Using this measure, we can see that the problems solved in the competition were of a greater complexity and were solved with lower word error rates than other published results. One reason for this is undoubtedly because, unlike real world tasks, it is known that the training and test data are derived from formal models similar to those learnt by the grammatical inference software. Despite this we believe that the competition has played a role in focusing research onto an important area of grammatical inference. In addition the complexity measures and benchmark problems can be used to track the progress of the state-of-the-art into the future.

Bibliography

[1] Kevin J. Lang, Barak A. Pearlmutter, and Rodney A. Price. Results of the Abbadingo One DFA learning competition and a new evidence-driven state merging algorithm. In V. Honavar and G. Slutzki, editors, *Proceedings of the Fourth International Conference on Grammar Inference*, volume 1433 of *Lecture Notes in AI*, pages 1–12, Berlin Heidelberg, Germany, 1998. Springer-Verlag.

[2] Bradford Starkie, François Coste, and Menno van Zaanen. The Omphalos context-free grammar learning competition. In Paliouras and Sakakibara [13], pages 16–27.

[3] Bradford Starkie, François Coste, and Menno van Zaanen. Progressing the state-of-the-art in grammatical inference by competition. *AI Communications*, 18(2):93–115, 2005.

[4] National Institute of Standards and Technology. The 2006 NIST machine translation evaluation plan (MT06), 2006. URL `http://www.nist.gov/speech/tests/mt/doc/mt06_evalplan.v4.pdf`.

[5] F. Casacuberta. Inference of finite-state transducers by using regular grammars and morphisms. In Arlindo L. Oliveira, editor, *Grammatical Inference: Algorithms and Applications (ICGI); Lisbon, Portugal*, volume 1891 of *Lecture Notes in AI*, pages 1–14, Berlin Heidelberg, Germany, September 11–13 2000. Springer-Verlag.

[6] E. Gurari. *An Introduction to the Theory of Computation*. Computer Science Press, Rockville:MD, USA, 1989.

[7] A.V. Aho and J.D. Ullman. *The theory of parsing, translation, and compiling*. Prentice Hall, Englewood Cliffs:NJ, USA, 1972.

[8] K.S. Fu. *Syntactic pattern recognition and applications*. Advances in computing science and technology series. Prentice Hall, Englewood Cliffs:NJ, USA, 1982.

[9] L. G. Valiant. A theory of the learnable. *Communications of the Association for Computing Machinery*, 27(11):1134–1142, 1984.

[10] J.C. Amengual, A. Castaño, A. Castellanos, V.M. Jiménez, D. Llorens, A. Marzal, F. Prat, J.M. Vilar, J.M. Benedi, F. Casacuberta, M. Pastor, and E. Vidal. The eutrans-i spoken language translation system. *Machine Translation*, 15(1):75–103, 2000.

[11] E. Matusov, S. Kanthak, and H. Ney. Efficient statistical machine translation with constrained reordering. In *European Association for Machine Translation (EAMT) 10th Annual Conference; Budapest, Hungary*, pages 181–188, 2005.

[12] E. Vidal and F. Casacuberta. Learning finite-state models for machine translation. In Paliouras and Sakakibara [13], pages 3–15.

[13] Georgios Paliouras and Yasubumi Sakakibara, editors. *Grammatical Inference: Algorithms and Applications: Seventh International Colloquium, (ICGI); Athens, Greece*, volume 3264 of *Lecture Notes in AI*, Berlin Heidelberg, Germany, October 11–13 2004. Springer-Verlag.