IJST, Transactions of Electrical Engineering, Vol. 38, No. E1, pp 59-72
Printed in The Islamic Republic of Iran, 2014

A NOVEL APPROACH TO MACHINE TRANSLATION: A PROPOSED LANGUAGE-INDEPENDENT SYSTEM BASED ON DEDUCTIVE SCHEMES*

S. M. FAKHRAHMAD**, A. R. REZAPOUR, M. H. SADREDDINI AND M. ZOLGHADRI JAHROMI

Dept. of Computer Science & Eng., School of Electrical and Computer Eng., Shiraz University, I. R. of Iran
Email: fakhrahmad@shirazu.ac.ir

Abstract– Compared to corpora-based machine translation methods, rule-based methods have deficiencies, which make them unattractive for the researchers of this field. The first problem is that these methods are language dependent. Rule-based methods require the syntactic information about source and target languages. On the other hand, in many cases, especially for proverbs and specific expressions, syntactic rules are no longer useful. In such cases, the use of example-based approaches is inevitable. In this work, we propose and integrate a set of novel schemes to introduce a new translation system, called BORNA. First a grammar induction method based on the Expectation Maximization (EM) algorithm is proposed. After representing the extracted knowledge in the form of a set of nested finite automata, a recursive model is proposed, which uses a combination of rule and example based techniques. In the translation phase, through a hierarchical chunking process, the input sentence is divided into a set of phrases. Each phrase is searched in the corpus of examples. If the phrase is found, it will not be chunked anymore. Otherwise, the phrase is divided into smaller sub-phrases. The simulation results show that BORNA outperforms its counterparts, significantly. Compared to PARS, Frengly and Google translators, BORNA receives the highest Bleu scores for its translations, while it results in the minimum values for different error measures, including PER, TER and WER.

Keywords– Machine translation, example-based, rule-based, corpora-based, finite automata, grammar induction

1. INTRODUCTION

Machine Translation (MT) is one of the most attractive and applied fields in natural language processing (NLP). Machine translation is the process of automatically analyzing a text in a source language and producing the equivalent text in a target language. Machine translation has met with limited success, up to now. Conventional machine translation systems are used to adopt *rule-based* (RBMT) methods, in which grammatical and linguistic restrictions are applied for translation. In recent studies, rule-based techniques have rarely been employed, because these methods are now believed to have many shortcomings. The major issues include ambiguity resolution and meaning interpretation. Rule-based systems suffer from inability to select the most suitable equivalent translation in many cases. Moreover, the rule-based systems are language-dependent since they are designed such that they can just be used for a specific pair of languages (source and target languages) [1-3].

In recent years, the mostly attended models of MT have been data-driven or corpora-based which is in sharp contrast to the dominant framework of the previous decades, i.e., RBMT. There are two corpora-based categories of translation methods namely, *example-based* (EBMT) and *statistics-based* (SMT) approaches proposed to overcome the shortcomings of rule-based methods [4]. In both cases the corpora comprise bilingual texts (original texts coupled with their translations) [5-8].

*Received by the editors October 26, 2012; Accepted May 25, 2014.
**Corresponding author

The EBMT approach is based on the extraction and a combination of phrases (or other short segments of texts). In EBMT methods, a large set of translation samples (i.e., pairs of source text and its translation) are stored and used for similar translations. Example-based methods are mostly used in order to detect and translate expressions. The origin of EBMT can be dated precisely to a conference paper in 1984 by Makoto Nagao [9].

EBMT systems use segments (word sequences and not individual words) of source language texts extracted from a text corpus to build texts in a target language with the same meaning. The basic units for EBMT are sequences of words (phrases, or 'fragments'), and the basic techniques are the matching of input phrases against sample source language phrases in the database, the extraction of corresponding target language phrases and the recombination of the segments as acceptable target language sentences.

The SMT approach was first proposed by Warren Weaver in 1949 [10]. It was then re-introduced in more detail by researchers of IBM's Thomas J. Watson Research Center [11]. This approach is primarily based on the study of frequencies of various linguistic units, including words, lexemes, morphemes, letters, etc., in a sample corpus to calculate a set of probabilities, so that various linguistic problems such as sense ambiguity can be solved. In other words, translation is based on statistical or probabilistic models whose parameters are extracted from the analysis of a bilingual corpus. Recently, SMT methods have widely been studied and have attracted the attention of many other researchers in the field of machine translation [12-23].

Although EBMT and SMT techniques outperform rule-based methods in terms of translation accuracy, they still have their own problems. For example, both methods require a huge bilingual corpus containing all possible word combinations, which is hardly assured to be available. Moreover, RBMT methods are usually much faster than corpora-based methods, since they rarely need to perform interpretation and deduction tasks.

Indeed, in some cases, where we aim to have a more successful translation, making use of both RBMT and corpora-based techniques is inevitable. Another challenge is that there is really no efficient algorithm to extract knowledge from a large-scale corpus, which is required for ambiguity resolution and other related problems.

In this paper, a new translation method called BORNA is proposed, which can be considered as a hybrid of rule-based and corpora-based techniques. The rule-based part of the system is not language-dependent, since the grammatical rules are automatically generated from a bi-lingual corpus. It should be noticed that this corpus is not required to be very large. It should just include samples of every grammatical rule. A set of novel schemes for knowledge representation as well as new methods of translation components (including hierarchical part-of-speech (POS) tagging, Automata construction, Automata traversing, etc.) will be presented in this work. The major scheme used in the proposed system includes a set of Deterministic Finite Automata (DFA) used in order to represent the induced grammatical rules. The interesting advantages of DFAs motivated us to construct our translation engine based on this structure. Most of the existing rule-based translation systems include a program that contains a large number of if-then rules. In such systems, the program is language-dependent, i.e., to change the source or the target language, the program has to be changed completely. Some other rule-based systems insert if-then rules in a database to be used by the main translation engine. The shortcoming of such systems is the search time they need to find the grammatical rule that is related to the input sentence. Making use of DFAs can solve both of the mentioned problems. The first advantage of this scheme is the separation of the knowledge and the translation engine. In order to change the source or the target language of the translator, just DFAs have to be changed or replaced and no change to the translation engine is needed. The other advantage of DFA is that it does not need to search for the correct grammatical rule. The input sentence enters the automaton word by word (or phrase by phrase), changing the state of the automaton, the grammatical rule is finally found after it stops at a final state. The rest of the paper is organized as

follows. In Section 2, the whole structure of the proposed system will be presented. In the first part of this section, we illustrate the method we use to induce grammatical rules from a corpus and introduce some novel schemes for representation of the extracted knowledge. The rest of this section is devoted to introduction of the translation engine. In Section 3, some standard metrics are used to evaluate the system's accuracy and compare it with one of the well-known English-to-Persian translators.

2. THE PROPOSED SYSTEM (BORNA)

In this section, the proposed MT system (BORNA) is introduced. BORNA is composed of two main parts. The first part of our system performs grammar induction. Two main tasks are carried out in this part, namely syntactic structure annotation and rule extraction. In this part, the grammatical rules of the source language and the translation order of the sentence chunks are induced from a large bi-lingual corpus. The discovered rules are represented in an automaton structure, which will then be used and traced by the translation engine. Translation engine is the main part of BORNA. It uses a hybrid of RBMT and EBMT methods and uses a dictionary, a bilingual corpus and the set of extracted rules to translate and combine chunks of a sentence. In addition to the translation engine, a set of operations are required to complete and enhance the quality of the translation. They include chunking, stemming, sense disambiguation, discovery of the tense of the sentence (needed for the verb construction in the target language), etc. Figure 1 presents the block diagram of the proposed system.

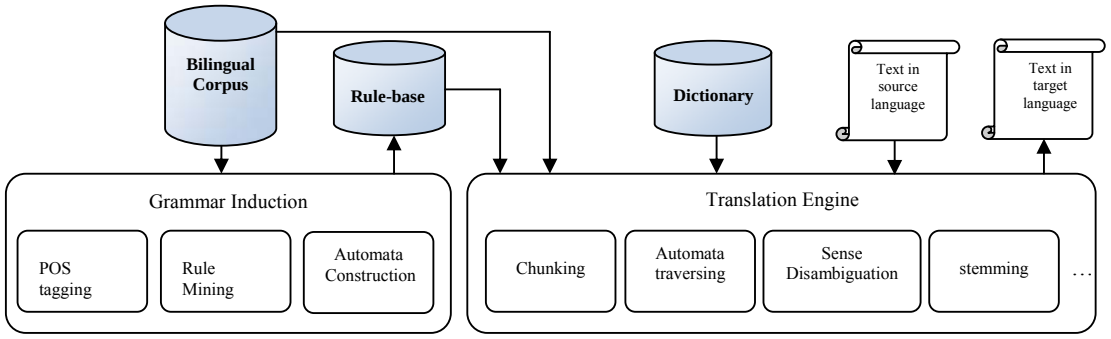


Fig. 1. A high-level view of the proposed MT system

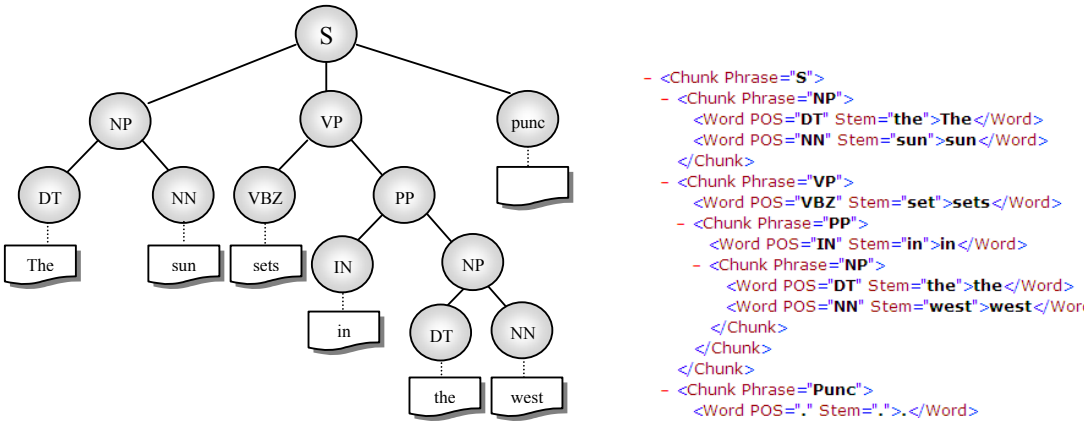
a) Grammar induction

This part of the system aims to discover and extract all possible syntactic structures of the source language by processing a large bi-lingual corpus. As will be discussed, two different structures, namely Finite automata and treebanks are simultaneously used to present and handle the syntactic schemes. The induced grammatical rules are presented in nested finite automaton structures, while the treebank structure is used to present syntactic annotated contexts. A treebank is a parsed corpus in which each sentence has been parsed and annotated with syntactic structure. Since the syntactic structure is represented as a tree, it is called a treebank. The alternative term Parsed Corpus is sometimes used for treebank. There are two main categories of treebanks: treebanks that annotate phrase structure (such as Penn Treebank [24]) and those that annotate dependency structure (such as the Quranic Arabic Dependency Treebank [25]).

In order to build a treebank, each sentence has to be annotated with syntactic structure. This can be carried out manually by linguists or semi-automatically, where a parser performs the annotation task. In the second case, linguists usually have to check and correct the result, which can be very labor intensive depending on the level of annotation details we want to present.

In our approach, a POS tagging process is first performed on the sentences of the corpus. Then, based on EM algorithm [26, 27], the treebank is created on top of the tagged corpus, using the Penn treebank as

the training corpus. Penn treebank is the first large-scale treebank published. We present the parsed corpus in XML scheme and enhance it with some other linguistic information such as word stems (obtained separately through a stemming process). Figure 2 shows the annotated scheme as well as its XML representation, for the example sentence "The sun sets in the west". Table 1 introduces all the notations used in the parsed schemes.



a) The tree structure b) The XML representation

Fig. 2. The tree structure and XML representation for the sentence "The sun sets in the west"

Table 1. Description of all notations used in the parsed corpora

Tag	Description
CC	Coordinating conjunction
CD	Cardinal number
DT	Determiner
EX	Existential there
FW	Foreign word
IN	Preposition or subordinating
JJ	conjunction
JJR	Adjective
JJS	Adjective, comparative
LS	Adjective, superlative
MD	List item marker
NN	Modal
NNS	Noun, singular or mass
NNP	Noun, plural
NNPS	Proper noun, singular
PDT	Proper noun, plural
POS	Pre-determiner
PRP	Possessive ending
PRP$	Personal pronoun
RB	Possessive pronoun
RBR	Adverb
RBS	Adverb, comparative
RP	Adverb, superlative
SYM	Particle
TO	Symbol
UH	To
VB	Interjection
VBD	Verb, base form
VBG	Verb, past tense
VBN	Verb, gerund or present participle
VBP	Verb, past participle
VBZ	Verb, non-3rd person singular
WDT	present
WP	Verb, 3rd person singular present
WP$	Wh-determiner
WRB	Wh-pronoun
	Possessive wh-pronoun
	Wh-adverb

After performing syntactic structure annotation on all contexts, each sentence is divided into a set of phrases, where each phrase is composed of a set of words or smaller phrases. By processing and comparing the set of all phrase sequences using Fast ARM which is an efficient frequent pattern mining method proposed in [28], we can find the sequences that are frequent (i.e., have occurred more than a specific threshold). Indeed, we are interested in such sequences, since each is likely to represent a grammar rule in the source language. Thus, they should all be extracted and recorded for further use. Since the order of parts of a discovered sequence will often be changed after translation (in the target language), a word alignment process has to be carried out as a complementary process on the bilingual corpus. That is, the translation order of each item of a discovered sequence is recorded so as to be used for further translation purposes.

Table 2 shows some of the phrase sequences discovered from the corpus under investigation. The first column shows different possible structures of a sentence in English, while the other columns indicate the discovered se-quences (i.e., possible schemes) for various phrases. As will be discussed in the next section, every phrase sequence shown in Table 2 will be then considered as a regular expression so as to construct a set of finite automaton struc-tures all together.

Table 2. The set of possible sequences for the major phrase structures, discovered from the corpus

S	VP		NP		ADJP	ADVP	PP	SQ	SBAR
NP+VP	VBZ+PP	VBZ+NP	DT+NN	JJR+NN	JJ	RB	IN+NP	MD+NP+VP	WHNP+S
PP+NP+VP	VB+S	VBD+NP+P	NNP	NP+NN	RBR+JJ+PP	NP+RB	TO+NP+NP-	VBP+NP+VP	WHADVP+S
VP	TO+VP	VBZ+ADJP+SBAR	PRP	NNP+POS	VBN+SBAR	DT	TMP	VBD+NP+VP	IN+S
NP+ADVP+VP	VP+CC+VP	VBP+RB+VP	PRP$+NN+NNS	NNP+CC+NNP	JJ+PP	RB+RB	TO+NP		S
SBAR+NP+VP	VB+PRT+NP	VB+PP+ADVP	PRP$+NN	PDT+PRP$+NNS	JJ+S		RB+IN+NP		IN+NN+S
S+NP+VP	VB+ADJP	VBD+NP	JJS+NNS	NNP+NNP	JJ+SBAR				
S+CC+S	VBZ+S	VBG+NP	DT+NNS	NP+SBAR	RB+JJ+CC+JJ+S				
S+VP	VB+NP+ADVP+NP-	VBP+ADJP	JJ+NNS	DT+NN+POS	RBR+JJ				
NP+ADJP	TMP	VBZ+VP	NN	DT+JJ+JJ+NN	JJR				
ADVP+VP	VBP+S	VBN+NP+PP	PRP$+JJS+NN	DT+NN+NNS	RB+JJ				
S+CC+IN+S	VB	VBP+VP	CD+NNS	PRP$+JJ+NN					
SQ	VB+SBAR	VBN+PP	NNS	DT+JJ+NNS					
	VB+NP	VBD+RB+VP	NP+PP	NN+NNS					
	VB+ADVP+PP	VB+VP	NN+JJ+NN	PRP$+NN+POS					
	VBD+S	VB+PRT+NP+PP	NN+NN	JJ+NN+NNS					
	VB+PP	VBG+VP	PRP$+NN+NN	JJR+NNS+CC+NNS					
	VBD+SBAR	VBN	DT+NN+NN	NNP+NNP+POS					
	VBD+VP	VB+PRT	DT+JJ+NN+NN	EX					
	VBN+NP	VBN+ADV	CD	NP+JJ+NN					
	MD+VP	VBD+ADJP	PRP$+JJ+JJ+NN	PRP$+NNS+POS					
	VBP+SBAR		DT+NN+S	DT+JJR+NN					
	VBP		PRP$+NNS	DT+NN+JJ					
			NNS+CC+NNS	CD+TO+C					
			NP+CONJP+NP	JJ+NN					

b) Constructing finite automata

In this stage, for each of the main phrases (i.e., S, VP, NP, ADJP, etc.) all the phrase sequences discovered in the previous section (shown in Table 1) are integrated to constitute a finite automaton. Finite automaton is one of the major components used in the translation engine. Every sequence discovered before is represented as a path in automata which connects the start state of the automaton to a final state (maybe passing through a set of middle states). Different POS tags shown in Table 1 can be seen as the labels of transitions within the automata. Each transition label is coupled with its translation order (in target language) which had been obtained through the alignment process in the last part. The input of the automaton (when being used for translation) is a parsed and POS tagged sentence which enters phrase (or word) by phrase (or word). Each phrase changes the current state of the automaton. Thus, the input sentence traverses a path through the automaton, which will specify the translation pattern.

Figure 3 shows the structure of the finite automaton built to represent different possible structures of a sentence. However, it is a non-deterministic finite automaton (NFA), since some states involve

ambiguity. This problem will be handled in the next subsection. For all of other structures (i.e., VP, NP, etc.) similar automata will be built and held all together (See Fig. 4).

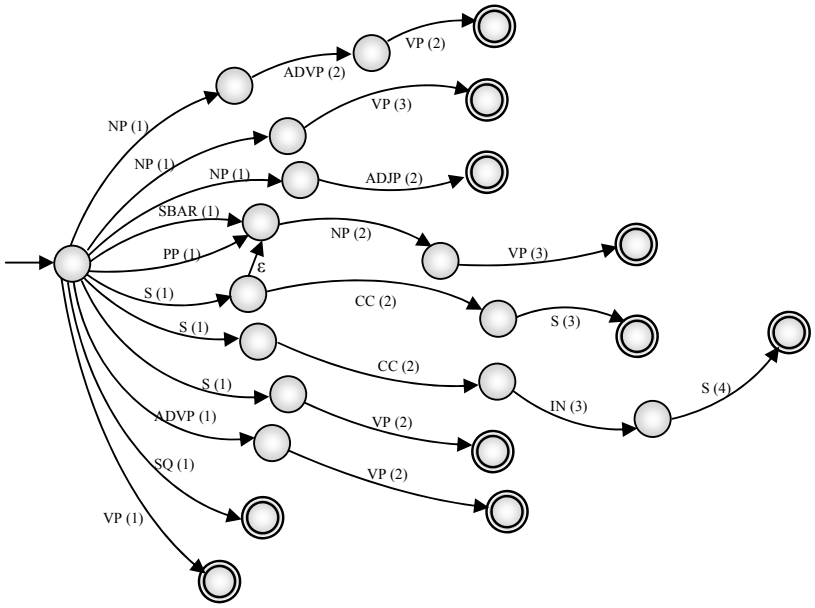


Fig. 3. The finite automaton structure (NFA) representing possible structures for a sentence in English

Although for each structure a separate automaton is constructed, the automaton structures are not independent. Indeed, the set of finite automaton structures are tightly coupled in a nested scheme. For example, in the structure of the automaton built for S (sentence), phrase tags (such as VP) can be observed. When a sentence traversing the S automaton meets a phrase transition such as VP, the tracing of the S automaton will be stopped temporarily and the execution control is switched to the VP automaton. When the traversing of VP automaton is completed, the execution control returns to the S automaton and continues from the last break point.

As mentioned at the beginning of this section, we construct a simple automaton for the sentence structures and a set of distinct automata for different phrases, rather than considering a single automaton for the whole language structures. The advantage of this strategy is that we do not have to consider an automaton path for every possible sentence in the source language, which is not possible in practice. Instead, we consider different possible structures for each phrase, separately. Hence, all possible combinations of the phrases (i.e., any typical sentence) can be represented by the set of automata.

For further illustration, consider Fig. 3. In this figure, a number of VP transitions can be seen in various states of the automaton. If we did not have a separate automaton for VP, then for each case, we would have to repeatedly consider all possible structures of a VP instead of the currently used single transition. This could lead to a huge complex automaton which is insufficient and really impossible to be implemented.

1. Conversion of NFA to DFA: As pointed out in the previous part, there is a problem with the structure of the constructed automata. The automata built according to the extracted phrase sequences are Non-deterministic Finite Automata (NFA). NFA structures suffer from two challenges: 1- Transition Ambiguity: In some states, there is more than one transition with the same label going to different states. It leads to ambiguity in finding the correct path through the automata. 2- State ambiguity: Existence of ε transitions leads to state ambiguity. For example, if two states say A and B, are connected with an ε transition, when attending A, the current state of the automaton will be both A and B, simultaneously!

In order to solve the mentioned problems, NFA structures are converted to DFA (Definite Finite Automata) in a complementary stage. Figure 4 shows the equivalent DFA for the NFA of Fig. 3.

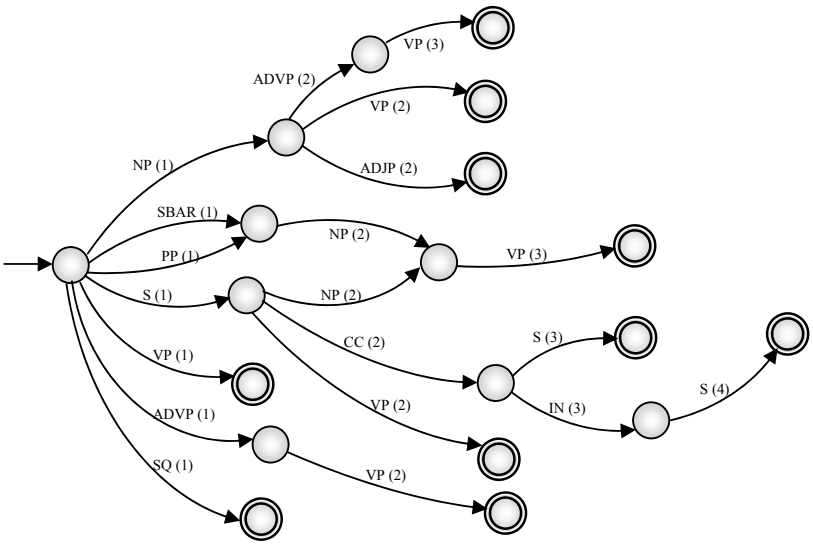
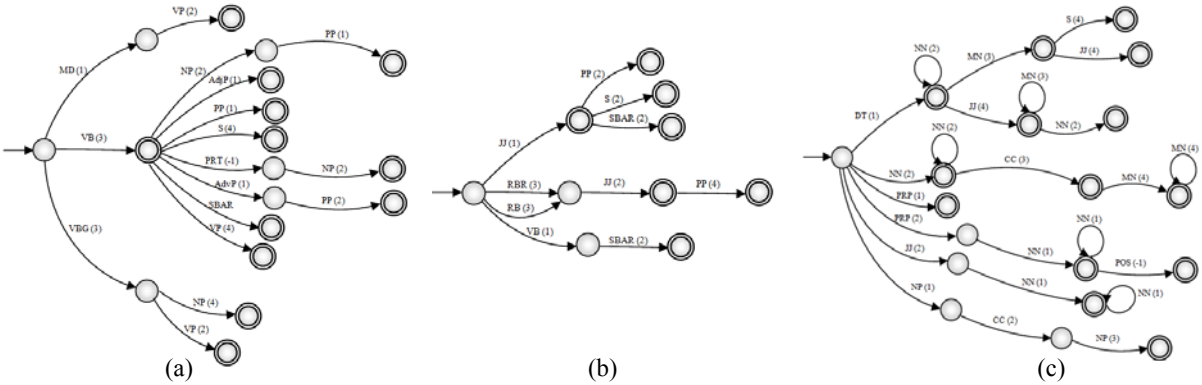


Fig. 4. The equivalent DFA for the NFA of Fig. 3

As mentioned, other than the whole sentence structure, an automaton is built for each of the main phrases. Figure 5 shows some of the automata constructed for VP, ADJP and NP, respectively.



(a) (b) (c)

Fig. 5. The finite automata (DFA) structures representing possible structures for: (a) Vp, (b) ADJP and (c) NP

c) Translation engine

The next part of BORNA is the translation engine. As shown in Fig. 1, the translation engine uses the parsed bi-lingual corpus as well as the set of automata constructed in the grammar induction phase. The translation method we use in this part includes both rule-based and corpora-based strategies. In the proposed method, each sentence or phrase is first searched through the set of examples in the corpus. If it is found in the corpus, its aligned meaning will directly be used in the translation. Otherwise, the sentence or phrase will iteratively be divided into smaller parts using the same chunking process performed in the previous part. The chunks are then fed to the related automaton to find the translation rule. As discussed in Section 2-2, if one of the chunks is a phrase (rather than a simple word), its own automaton has to be traversed. Thus, in such cases, traversing of the current automaton is stopped temporarily until the nested automaton is finished (just the same as procedure calls in software programs).

The best case in this system occurs when a sentence is found in the corpus of examples. In this case, no chunking process is required and the translation type is completely example-based. On the other hand, the worst case occurs when neither the whole sentence nor its parts could be found in the corpus. In this case, the chunking process continues hierarchically until the sentence is decomposed into to a set of words. In this case, the translation will be fully rule-based. Thus, the order of translation of the words has to be found from the set of previously built automata. Since through the chunking process, the words are placed in different branches of a tree in different depths, the order found for each part is recorded in a form that can represent its situation in the tree, too. Figure 6 presents the main function of the translation engine. The function *Translate* receives a sentence (or a part of a sentence) as an XML node. The output is a sorted list of items in target language which represents the primary translation of the input sentence. In order to translate nested phrases, the function is invoked recursively whenever a phrase is met. The *current-state* parameter shows the state of the automaton which is being traversed. For each call of the Translate function, *current-state* is initialized to 'Start'. Each part of the input text changes the value of *current-state* as it is visited.

```
Function Translate
inputs:   text: XmlNode,   current-state: String
Output:   meaning-list: List of the chunks' meanings
{
For each XmlNode such as node in child-nodes of text
   If type of node is 'chunk'
      Look in corpus to see if the chunk exists in the corpus      // corpus of bilingual examples
      If the chunk was found in corpus
         Compute its translation order according to DFA            // DFA is the automata containing grammatical paths
         Add the chunk and its translation to the meaning-list
      Else
         Translate(node, 'Start');
   Else if type of node is 'word'
      Find the meaning of the word from dictionary with respect to the values of the attributes Stem and POS
   Change the current-state of DFA according to the value of the Transition attribute of node;
If type of current-state is 'Final'
   Sort items in meaning-list according to their translation order values
}
```

Fig. 6. The main procedure of the translation engine

As an example, consider the sentence "Tomatoes are less expensive at the rural farm stand.", which is aimed to be translated. First of all, the sentence has to be parsed and POS tags should be annotated. The method used in the training phase (to construct the treebank) is exactly used in this step. We represent the parsed sentence in XML format and enrich it with some extra information such as word stems.

Since the whole sentence cannot be found in the corpus of examples, it is divided into two main parts, NP ("Tomatoes") and VP ("are less expensive at the rural farm stand"). The path of NP+VP in Sentence's automaton is then traced in order to find and label the translation order of each phrase (1 for NP and 2 for VP).

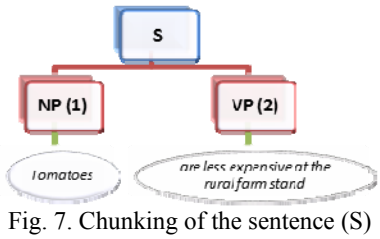


Fig. 7. Chunking of the sentence (S)

The NP phrase in this example ("Tomatoes") is just a simple word, which cannot be divided into sub-phrases. Hence, it can be translated according to its stem and its POS, using a dictionary. The meaning of

the word coupled with its translation order is added to the meaning list. On the other hand, the VP part is first searched through the corpus and if it is not found there, it has to be chunked again. The resulting phrases will be VBP ("are") and ADJP ("less expensive at the rural farm stand"). The path of VBP+ADJP in the automaton of VP is then traced and in order to find the translation order of each part. Assume that the translation orders of VBP and ADJP are 3 and 2 (according to the automaton), respectively. Since in the hierarchical chunking tree, the parent node of these two phrases is VP (with the translation order of 2), the following labels are used as the translation orders: 2-3 for VBP and 2-2 for ADJP.

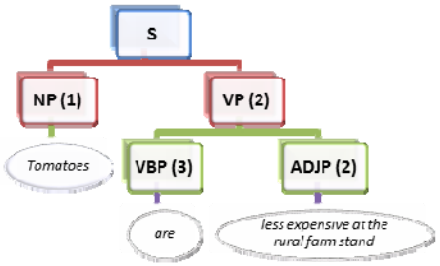


Fig. 8. Chunking of the verb phrase (VP)

The ADJP part has to be divided into three parts, RBR ("less"), JJ("expensive") and PP("at the rural farm stand"), since it cannot be found in the corpus. The translation order of each part is then found after tracing the automaton of ADJP. However, the translation of VBP ("are") is easily found and added to the meaning list.

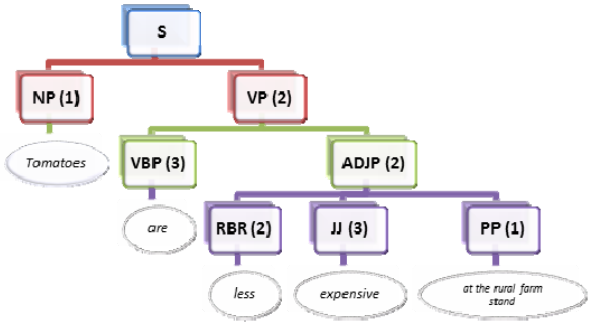


Fig. 9. Chunking of the Adjective phrase (ADJP)

In the next step, the PP phrase is divided into two parts, IN ("at") and NP ("the rural farm stand"). The translation orders of these two parts can be found by tracing the PP's automaton. Assume that the NP part exists in the corpus of examples. Hence, it does not need to be chunked again and its meaning can be taken from the corpus and added to the meaning list.

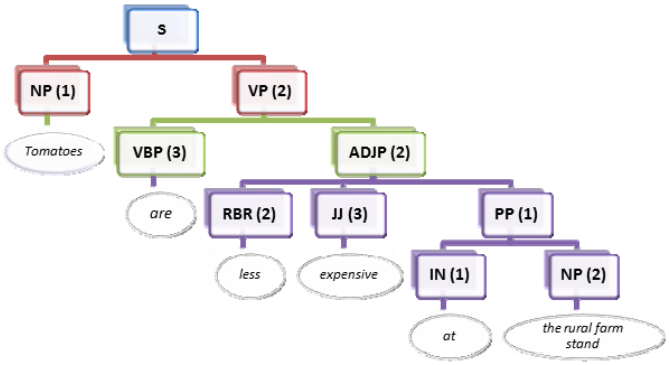


Fig. 10. Chunking of the preposition phrase (PP)

Figures 7 to 10 show the step-by-step process of chunking followed in this example. The continents of the meaning list (including the sentence's chunks and the translation orders) will finally be as follows: "*Tomatoes*" (1), "*Are*" (2-3), "*Less*" (2-2-2), "*Expensive*" (2-2-3), "*at*" (2-2-1-1), "*the rural farm stand*" (2-2-2).

Indeed, it is a primary translation. The final translation can be achieved after complementary operations such as tense identification, sense disambiguation, etc.

3. EVALUATION

Evaluation of machine translation output is a challenging problem. Automatic evaluation is simply defined as the comparison of the actual translation performed by the system (denoted as *candidate translation*) and the desired translation (denoted as *reference translation*). However, in most cases, there is not just a single correct translation for an input sentence. In the extreme case, a pair of translations for the same input can be perfectly valid, while they have different structures and include completely different words. There are many evaluation metrics already proposed, all of which assign a score to the translation output. In this work, we consider some of these evaluation metrics, which are most commonly used and most suitable to perform the experiments.

a) Common evaluation measures

The measures which are employed to evaluate the output of a machine translation system represent the quality of the output. An evaluation measure must assign quality scores such that they correlate with human judgment of quality. Various evaluation measures have already been introduced, most of which have tried to have high correlation with human judgments of quality. Before presenting the experiments, in this section, we first give a brief description on some of the most well-known evaluation measures.

1. Word Error Rate (WER) [29]: This metric represents the dissimilarity or distance of a pair of translations via the Edit distance measure. The Edit distance is defined as the minimum number of word insertions, substitutions and deletions required to convert the candidate translation into the reference translation. All three operations are assumed to have the same costs.

The value of WER is measured by dividing the number of edit operations by the number of words in the reference translation. If the candidate translation is longer than the reference, the value of WER will be greater than 1. Thus, WER has a bias towards shorter hypotheses.

When there is more than one reference translation, the reported error (WER) for a candidate translation is the minimum error over all references.

2. Position-independent Word Error Rate (PER) [30]: Unlike WER that requires exactly the same order of the words in candidate and reference translations, PER neglects word order, absolutely. It measures the difference of the words occurring in candidate and reference translations. The resulting number is then divided by the number of words in the reference translation.

3. Translation Edit Rate (TER) [31]: TER is another error measure that counts the number of edits required to convert a system output into one of the given references. This metric can measure the amount of human work that would be required to post-edit the translations proposed by the system and convert to the reference translation. In contrast to WER, movements of blocks are permitted and counted as one edit with equal costs to other legal operations, i.e., insertions, deletions and substitutions of single words.

The value of TER is obtained by dividing the number of edit operations by the average number of reference words.

4. BLEU [32]: BLEU is one of the most well-known metrics which is frequently used in evaluation of translation systems. In contrast to other metrics defined above, BLEU is a precision (or similarity) metric. It measures the similarity of n-gram vectors in the reference translations and the candidate translation. In other words, it represents the rate of n-grams of the candidate translation, which can also be found in the reference translation. If more than one reference exists, the counts are gathered for all translations.
Since BLEU is a precision measure, higher values indicate better results. If no n-gram of maximum length matches between candidate and reference translations, the BLEU score will be zero.

5. NIST [33]: NIST is another precision measure which is considered as an improved version of BLEU. When using this measure, n-gram occurrences are weighted by their importance. The importance of an n-gram is specified according to the frequency of the n-gram in the reference translations.
NIST considers less important values for frequently occurring n-grams in comparison with rare ones.

b) Evaluation results

In the first part of the experiment, we used the BLEU score in order to evaluate the translation precision and compare it to some of the well-known translators including PARS [34], Frengly [35] and Google translate [36]. For this purpose, we used a fraction of our bi-lingual corpus including 100 pairs of sentences. In order to obtain more reliable results, we divided the set of sentences into 5 blocks of 20 sentences each, and computed the BLEU metric considering 4-grams on these blocks individually. We thus have 5 samples of the BLEU metric for each system. We computed the means and variances, which are shown in Table 3.

In this experiment, the BLEU score was measured in two ways. The First case was the usual case where we considered the own words included in n-grams in order to measure the BLEU scores. In the second case, we used the POS tags of the words instead of the own words. The results of these two cases as well as the average value are given in Table 4.

Table 3. The evaluation results (mean and variance) for translation systems on 5 blocks of the test corpus

	PARS	Google	Frengly	BORNA
BLEU score (Mean)	0.378	0.493	0.437	0.712
Standard Deviation	0.023	0.02	0.017	0.013

Table 4. BLEU scores received by translation systems in two cases

	PARS	Google	Frengly	BORNA
BLEU Score (considering the words)	0.378	0.493	0.437	0.712
BLEU Score (considering the POS tags)	0.614	0.717	0.627	0.896
Average	0.496	0.605	0.532	0.804

In the second part of this experiment, we used two other evaluation metrics, namely WER and PER on the same test corpus, in order to evaluate the systems' performance from the error rate point of view.

The results for the four translation systems including Google, Frengly, BORNA and PARS are presented in Table 5.

Table 5. Comparison of various translation systems in terms of WER, PER and TER metrics

		WER	PER	TER
Considering Words	PARS	0.782	0.526	0.733
	Google	0.668	0.509	0.699
	Frengly	0.643	0.522	0.74
	BORNA	0.416	0.403	0.197
Considering POS tags	PARS	0.493	0.33	0.511
	Google	0.517	0.497	0.324
	Frengly	0.477	0.288	0.416
	BORNA	0.221	0.194	0.032

4. CONCLUSION

In this paper, we first proposed a grammar induction method based on Expectation Maximization (EM) algorithm. After representing the extracted knowledge in the form of nested finite automata, a recursive model was proposed, which used a combination of rule and example based techniques. In the translation phase, through a hierarchical chunking process, the input sentence is divided into a set of phrases. Each phrase is searched in the corpus of examples. If the phrase is found, it will not be chunked anymore. Otherwise, the phrase is divided into smaller sub-phrases. The worst case occurs when none of the phrases and sub-phrases can be found in the corpus. In this case, we will finally have a set of simple words and the translation procedure will completely be rule-based. In other cases both approaches are applied. The accuracy of the system in translating from English to Persian was evaluated through a set of experiments using various metrics. The simulation results showed the promising accuracy and efficiency of the proposed system compared to PARS (as the most well-known English-to-Persian translation system) and other translators, including Frengly and Google Translate.

The evaluations performed in the current study were all devoted to English-Persian pairs of sentences. Since BORNA claims to be a language-free MT system, we should prepare suitable bilingual corpora for other languages, so as to be employed by BORNA. Moreover, as a future work, we intend to improve the matching method we use in the example-based part of the proposed system, such that it will be able to match two sentences which seem to be different but are semantically equal, e.g., active and passive equivalent sentences.

REFERENCES

1. Sinha, R.M.K. & Jain, A. (2003). *AnglaHindi: An English to Hindi machine translation system*. MT Summit IX, New Orleans, USA, Sept. 23-27, pp. 112-119.
2. Dave, S., Parikh, J. & Bhattacharyaa, P. (2001). Interlinguabased English-Hindi machine translation and language divergence. *Machine Translation*, Vol. 16, No. 4, pp. 251-304.
3. Hettige, B. & Karunananda, A. S. (2012). Computational model of grammar for English to Sinhala machine translation. The International Conference on Advances in ICT for Emerging Regions, pp. 026-031.
4. Navigli, R. & Velardi, P. (2005). Structural semantic interconnections: a knowledge-based approach to word sense disambiguation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, Vol. 27, No. 7, 2005, pp. 1063–1074.
5. Lavecchia, C., Smaili, K. & Langlois, D. (2007). Building parallel corpora from movies. *Proceedings of the 4th International Workshop on Natural Language Processing and Cognitive Science*, (Funchal, Madeira, 12-13 June 2007), no pagination.
6. Carl, M. & Way, A. (2007). Introduction to special issue on example-based machine translation. *Machine Translation*, Vol. 19, (3-4), pp. 193-195
7. Hutchins, J. (2005). *Example-based machine translation*: a review and commentary. *Machine Translation*, Vol. 19, pp. 197-211.
8. Sumita, E. & Iida, H. (1991). Experiments and prospects of Example-Based Machine Translation. *ACL '91 Proceedings of the 29th annual meeting on Association for Computational Linguistics*, pp. 185-192.
9. Nagao, M., Nishida, T. & Tsujii, J. (1984). Dealing with incompleteness of linguistic knowledge in language translation – transfer and generation stage of Mu machine translation project. *Coling84: 10th International Conference on Computational Linguistics & 22nd Annual Meeting of the Association for Computational Linguistics, Stanford University, California. Proceedings*, pp. 420-427.

10. Weaver, W. (1949). *Translation. Repr. in: Locke, W.N. and Booth, A.D.* (eds.). Machine translation of languages: fourteen essays (Cambridge, Mass.: Technology Press of the Massachusetts Institute of Technology, 1955), pp. 15-23.
11. http://www.watson.ibm.com, Accessed September 8 (2013).
12. Abdul-Rauf, S. & Schwenk, H. (2009). On the use of comparable corpora to improve SMT performance. *In Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pp. 16-23.
13. Bojar, O. & Tamchyna, A. (2011). Forms wanted: training SMT on monolingual data. *Abstract at Machine Translation and Morphologically-Rich Languages*. Research Workshop of the Israel Science Foundation University of Haifa, Israel.
14. Chiang, D. (2011). Hierarchical phrase-based translation. *Computational Linguistics*, Vol. 33, No. 2, pp. 201–228.
15. Habash, N. (2008). Four techniques for online handling of out-of-vocabulary words in Arabic-English statistical machine translation. *In ACL 08*, pp. 77-86.
16. Johnson, H., Martin, J., Foster, G. & Kuhn, R. (2007). Improving translation quality by discarding most of the phrasetable. *In Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, Prague, Czech Republic, pp. 967-975.
17. Callison-Burch, C., Koehn, P., Monz, C. & Zaidan, O. (2011). Findings of the 2011 workshop on statistical machine translation. *In Proceedings of the Sixth Workshop on Statistical Machine Translation*, Edinburgh, Scotland, pp. 22–64.
18. Gispert, A., Blackwood, G., Iglesias, G. & Byrne, W. (2013). N-gram posterior probability confidence measures for statistical machine translation: an empirical study. *Machine Translation*, Vol. 27, pp. 85–114.
19. Eidelman, V., Wu, K., Ture, F., Resnik, P. & Lin, J. (2013). Towards efficient large-scale featurerich statistical machine translation. *In Proceedings of the Eighth Workshop on Statistical Machine Translation*, Sofia, Bulgaria, pp. 126–131.
20. Bicici, E. (2013). Feature decay algorithms for fast deployment of accurate statistical machine translation systems. *In Proceedings of the Eighth Workshop on Statistical Machine Translation*, Sofia, Bulgaria, pp. 76–82.
21. Schwenk, H. (2008). Investigations on largescale lightly-supervised training for statistical machine translation. In IWSLT, pp. 182–189.
22. Banko, M. & Moore, R. C. (2004). Part of speech tagging in context. *In Proceedings of the Inter-national Conference on Computational Linguistics (COLING)*, pp. 164-170.
23. Goldwater & Griffiths, T. L. (2007). A fully Bayesian approach to unsupervised part-of-speech tagging. *In Proceedings of the ACL*, pp.187-192.
24. www.ldc.upenn.edu/ldc/online/treebank, Accessed September 8 (2013).
25. Dukes, K. & Buckwalter, T. (2010). A dependency Treebank of the Quran using traditional Arabic grammar. *Informatics and Systems (INFOS)*, pp. 1-7.
26. McLachlan, G. & Krishnan, T. (1996). The EM Algorithm and Extensions. John Wiley & Sons, New York, 1996.
27. Goldberg, M., Adler & Elhadad, M. (2008). EM can find pretty good HMM POS-taggers (when given a good start). *In Proceedings of the ACL*.
28. Fakhrahmad, S.M. & Dastghaiby Fard, G. H. (2009). An efficient frequent pattern mining method and its parallelization in transactional databases. *Journal of Information Science and Engineering*.
29. Chiorboli, G., De Salvo, B., Franco, G. & Morandi, C. (1998). Some thoughts on the word error rate measurement of A/D converters. *IEEE International Conference on Electronics, Circuits and Systems*, Vol. 3, pp. 453 – 456.

30. Tillmann, C., Vogel, S., Ney, H., Zubiaga, A. & Sawaf, H. (1997). Accelerated DP based search for statistical translation. *In Fifth European Conf. on Speech Communication and Technology*, Rhodos, Greece, September, pp. 2667–2670.
31. Snover, M., Dorr, B., Schwartz, R., Micciulla, L. & Makhoul, J. (2006). A study of translation edit rate with targeted human annotation. *In Proceedings of Association for Machine Translation in the Americas*.
32. Papineni, K., Roukos, S., Ward, T. & Zhu, W. J. (2002). BLEU: a method for automatic evaluation of machine translation. *In Proc. of the 40th AnnualMeeting of the Association for Computational Linguistics (ACL)*, Philadelphia, PA, pp. 311–318.
33. Doddington, G. (2002). Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. *In Proc. ARPAWorkshop on Human Language Technology*.
34. *www.parstranslator.com*, Accessed September 8 (2013).
35. *www.Frengly.com*, Accessed September 8 (2013).
36. *translate.google.com*, Accessed September 8 (2013).