CONFIDENCE FACTOR ASSIGNMENT TO TRANSLATION TEMPLATES

A THESIS
SUBMITTED TO THE DEPARTMENT OF COMPUTER ENGINEERING AND INFORMATION SCIENCE
AND THE INSTITUTE OF ENGINEERING AND SCIENCE
OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

By
Zeynep Orhan
September, 1998

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Asst. Prof. İlyas Çiçekli (Advisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assoc. Prof. H. Altay Güvenir

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Asst. Prof. Atilla Gürsoy

Approved for the Institute of Engineering and Sciences:

Prof. Mehmet Baray
Director of Institute of Engineering and Science

ABSTRACT

CONFIDENCE FACTOR ASSIGNMENT TO TRANSLATION TEMPLATES

Zeynep Orhan
M.S. in Computer Engineering and Information Science
Supervisor: Asst. Prof. İlyas Çiçekli
September, 1998

TTL (*Translation Template Learner*) algorithm learns lexical level correspondences between two translation examples by using analogical reasoning. The sentences used as translation examples have similar and different parts in the source language which must correspond to the similar and different parts in the target language. Therefore, these correspondences are learned as translation templates. The learned translation templates are used in the translation of other sentences. However, we need to assign confidence factors to these translation templates to order translation results with respect to previously assigned confidence factors. This thesis proposes a method for assigning confidence factors to translation templates learned by the TTL algorithm. In this process, each template is assigned a confidence factor according to the statistical information obtained from training data. Furthermore, some template combinations are also assigned confidence factors in order to eliminate certain combinations resulting bad translation.

ÖZET

ÇEVİRİ KALIPLARINA GÜVEN FAKTÖRÜ ATANMASI

Zeynep Orhan
Bilgisayar ve Enformatik Mühendisliği Bölümü, Yüksek Lisans
Tez Yöneticisi: Yrd. Doç. Dr. İlyas Çiçekli
Eylül, 1998

Çeviri Kalıpları Öğrenicisi (ÇKÖ) algoritması iki çeviri örneği arasındaki yapısal seviyedeki uygunlukları analojik muhakemeyle öğrenir. Çeviri örneğinde kullanılan cümlelerin hedef dildeki benzer ve ayrı kısımlara karşılık gelmesi gereken kaynak dilde bulunan benzer ve ayrı kısımları olmalıdır. Bu yüzden bu uygunluklar çeviri kalıpları olarak öğrenilir. Öğrenilen çeviri kalıpları diğer cümlelerin çevirisinde kullanılır. Bununla beraber, daha önce elde edilmiş olan güven faktörlerini dikkate alarak çeviri neticelerini sıralamak için bu çeviri kalıplarına güven faktörleri vermemiz gerekir. Bu tez ÇKÖ algoritması ile öğrenilen çeviri kalıplarına güven faktörleri vermek için bir algoritma ortaya koymaktadır. Bu işlemde her kalıba eğitme örneklerinden elde edilen istatistiksel bilgiye göre bir güven faktörü verilmiştir. Ayrıca, kötü çeviriye yol açacak belli kombinasyonları elemek için bazı kalıp kombinasyonlarına da güven faktörleri verilmiştir.

To My Husband and Son

Acknowledgment

I would like to express my gratitude to Asst. Prof. İlyas Çiçekli for his supervision, guidance, suggestions and invaluable encouragement throughout the development of this thesis.

I would like to thank the committee member Assoc. Prof. Halil Altay Güvenir and Asst. Prof. Attila Gürsoy for reading this thesis and their comments.

I would like to thank my husband for his patience, sacrifice and encouragement.

I would like to thank my colleague Esma Turna (Susanne Stassen) for her friendship and moral support.

I would like to acknowledge TÜBİTAK, The Scientific and Technical Council of Turkey, for the partial support under the grant EEEAG-244.

Contents

List of Figures

List of Tables

Chapter 1

Machine Translation

1.1 General Information

The term *machine translation* (**MT**) is the general name for any system which uses an electronic computer to transform a text in one language (**SL**, for source language), into some kind of text in another natural language (**TL**, for target language), ([45]). The related term *machine-aided translation* means the mechanized aids for translation. Terms like *mechanical translation* or *automatic translation* can also be seen in the literature.

Another definition for machine translation can be described by the following phrase [73]:

> *Feed a text in one language into a computer and, using a computer program, produce a text in another language such that the meaning of the* **TL** *text is the same as the meaning of the* **SL** *text.*

The description above is a very simple one which can even be easily understood by someone who does not know much about the linguistics. However, machine translation is not a simple task and many issues must be considered by related people (e.g. the customers, governments, international organizations, industrial and other corporations, designers and implementors of MT systems). These issues can be summarized as follows:

- Can the input of the system be any arbitrary text? Or in other words, should we restrict the domain of the input text?
- Is it really necessary to preserve the form of the source language text in addition to the meaning during translation? If it is, is it an attainable objective?
- Can the system be scaled? (i.e. Can the number of source and target languages be increased easily after developing it for a given pair of languages?)
- Can the system be adjusted for different subject domains after developing it for a certain subject domain?

- What are the constraints over the translation? (speech, electronic, texts, etc.)
- How can the quality of the system be determined? What are the core criteria for evaluating such systems?
- How can the economic and scientific value of the MT systems be determined?
- What should be done when an MT system fails to produce an adequate translation? Should the system signal a complete failure or will the results that are close to the target be displayed? If the latter solution is chosen, what will be the criteria in selecting the closest target?

Performance type and functionality of a machine translation system are the important selection criteria. The performance type must be considered, because the user needs vary in a wide range. User wants translations within hours for informal papers, such as working documents, notes or letters. Users will expect a translation within days for research papers or commercial reports, for instructions or other technical documents. More formal descriptions like annual reports, proceedings or official company materials should be available within weeks. Only translations for books may be allowed to extend for years. These performance types are tied to quality characteristics and acceptance criteria of users: the faster the system has to work, the more the user will accept minor quality.

Functionality of an MT system refers *what the system does for its user?* In other words whether it is a machine aided translation (MAT), machine translation (MT), or machine interpreting (MI) system. Looking more closely to the functional types of translations, we can divide MT systems into two categories: MT systems in which the quality requirements are high and low. In the former category the quality requirement is high since:

- in most cases the reader is unknown to the writer and
- the text contains exactly the information to be shared with the reader,

whereas in the latter category the amount of information to be extracted is dependent on the specific interest of the reader, which may vary between *to know what it is about* and *what exactly does the author mean?*

Machine translation is a composite field. Science and engineering, basic research and development, computer science, linguistics, artificial intelligence, and software engineering are some of the fields that are related to machine translation. Issues like grammar theory, lexicon, semantics, pragmatics, discourse, parsing, semantic interpretation and generation, and acquisition are some of the concepts that must be considered in a MT system. Additionally, operational tools, such as language analyzers and generators, are necessary in order to build functional and robust systems.

Machine translation has become a famous area for the last 50 years. The major goal is to have commercial machine translation systems, therefore, several paradigms have been developed. MT is a very promising area and successful MT systems seem to be highly profitable due to the increasing demand, so research goes on with an increasing interest for producing translation of greater accuracy.

There are many reasons for the development of the machine translation systems, since in the modern world of 21^{st} century, translation among languages becomes a vital issue from economic, sociological and political perspectives. These reasons were summarized by [45] as follows:

- People from various fields and different disciplines have to read documents and communicate in languages they do not know. In addition to this, the volume of the material that has to be translated is increasing day by day and the number of the human translators is not sufficient to cope with this.
- Many researchers have been motivated by idealism for promotion of international cooperation and peace, transferring technology to the developing or poor countries, etc.
- It is an important fact for military and intelligence contexts.
- There are pure research reasons, such as studying the basic mechanisms of language and mind, exploiting the power of the computer and finding its limitations.
- Other simple commercial and economic motives.

At certain periods, some of these reasons are more dominant than the others. The cold war between United States and Russia caused much governmental and military support for Russian-English translation. Translation among European languages became an important requirement after the unification of Europe, therefore researches on translation systems within the languages of the community were encouraged. English alone did not suffice the needs of the USA and Japan whose economies highly rely on the export markets in a large number of languages.

There are now operational systems in a number of large translation agencies. Computers are being widely used for producing readable translations for people of various fields. And there is a growing interest in machine translation within the artificial intelligence community in the United States, Japan and other parts of the world. Nowadays, the potential market of machine translation is very high and growing rapidly day by day. It seems to preserve its increasing trend in the future. There is a huge need for massive, and inexpensive translation because of the reasons explained above. So, the development of the fully automated machine translation, or machine-aided translation systems gained a big importance for many computational linguists and computer scientists.

A brief survey about the fundamental developments and the periods in machine translation history supports the hopes about the future of machine translation

1.2 Brief History of Machine Translation

Although we do not know exactly who first had the idea of machine translation, it can be accepted that the actual development of MT has begun by a conversation between Andrew D.

Booth and Warren Weaver of The Rockefeller Foundation in 1947, or more specifically we can take the starting time of MT as a note written by Weaver in 1949 to the Rockefeller Foundation which included the following two sentences:

> *I have a text in front of me which is written in Russian but I am going to pretend that it is really written in English and that has been coded in some strange symbols. All I need to do is to strip off the code in order to retrieve the information contained in the text*

He established an interesting analogy between translation and decoding. In decoding there is a one-for-one substitution process and the output is unique. However, translation is a more complex job. Thus, Weaver proposed another sophisticated view [1] ([5]). Although these ideas are strange, the note mentioned above gained a significant amount of interest and research yielding a large number of research groups in 1950s.

However, disappointments and doubts by some funding authorities were observed due to the problems faced during these researches. These doubts were emphasized in the report which the US National Academy of Sciences commissioned in 1964. They set up the Automatic Language Processing Advisory Committee (ALPAC) to report on the MT of those days. *ALPAC report*, which is very famous in machine translation history, was very pessimistic about machine translation. This pessimism caused US government to stop supporting MT. The report basically pointed out that there was no need for machine translation, and there was no prospect of successful MT. These ideas discouraged many people who were working in this field. Although MT systems could not achieve the idea of Fully Automated High Quality Machine Translation (FAHQMT), the partial successes of MT, (like the considerable amount of time gained which can be achieved by automating the draft translation stage of high volume systems), is real and could not be ignored completely.

After the ALPAC report many researches had been given up, and MT in the following ten years is supported only by small marginal sponsors. There were still some other groups working on MT systems in the world, like TAUM group who developed METEO system in Canada (see [48], [11]), groups in USSR, GETA group in Grenoble (see [10], [103]) and SUSY (see [62]) group in Saarbrücken, etc.

Towards the late 1970s MT began to gain its old fame, especially after the Commission of the European Communities (CEC) purchasing the English-French version of SYSTRAN (see [84], [111]) system, and Russian-English system which was used by USAF and NASA. CEC also supported the development of French- English and Italian-English version. In addition to these, CEC also contributed to the set up of the EUROTRA (see [55], [56], [108]) project which was an improvement of the GETA and SUSY groups. This project was really the largest project of those

[1] Weaver described an analogy of individuals in tall closed towers who communicate (badly) by shouting to each other. However, the towers have a common foundation and basement. Here communication is easy: Thus it may be true that the way to translate ... is not to attempt the direct route, shouting from tower to tower. Perhaps the way is to descend, from each language, down to the common base of human communication, the real but as yet undiscovered universal language

days in natural language processing. The other important projects that worth mentioning were the SPANAM Spanish-English MT system by Pan American Health Organization (PAHO), METAL system which was funded by United States Air Force, and the METEO system which was built on the work of TAUM group. In those days it was also observed that Japanese had an increasing interest in MT. The MT researches in Japan concentrated on building working, commercial systems and this led to the development of many MT systems in Japan which were funded by both private and public sectors.

MT has begun to gain the attention it deserved after the late 1970s and this increasing trend continued until our time despite the pessimistic ALPAC report. Now, MT is a recognized international scientific field with a worldwide community of researchers.

MT is becoming a major topic in the computer science and it seems to have an increasing trend in the future [72]. It is obvious that, the greater the internalization of commercial activity, the greater the need for MT systems will be. As Hutchins [47] foresees:

> *It is reasonable to predict that in another twenty years MT and/or MAT in various forms and packages will be normal and accepted facilities in nearly every office and laboratory.*

It can also be hoped that the theory and standards of MT will be developed by the progress in the field. Despite the less amount of progress in MT up to now, it is a promising area of the future.

In summary, it is appropriate to divide the development of MT into five evolutionary periods. The first period lies between the end of the Second World War and the mid 1950s. The second period lasted up to ALPAC report in mid-1960s. In these periods MT research was highly encouraged by US government and military. In the third period MT research had slowed down by the effect of ALPAC report and the researches had only concentrated on indirect systems. The fourth period started in mid-1970s with the interests of CEC, Soviet Union and Japan in MT. Finally the last period extends from 1980s to today. We observe a bursting interest and many activities in this period and these indicates that we can expect a promising future for MT research.

1.3 The Approaches in Machine Translation

This section identifies several strategies that are affective in current and past MT researches. Direct, transfer, interlingua or knowledge-based, and corpus-based MT strategies will be discussed in the following sections.

1.3.1 Direct MT

Direct MT systems are specifically designed for a certain SL and TL pair. The main idea behind these kinds of systems is that translation of the SL sentences can be done by a light parse (i.e. the simplest parse), replacing SL words with their TL equivalents by a single dictionary look up procedure, and then roughly rearranging their order to suit the rules of the TL. These systems depend on well-developed dictionaries, morphological analysis, and text-processing software instead of linguistic theories and syntactic structures [101]. The basics of direct MT is given in Figure 1.1. The majority of the MT systems between 1950s and 1960s were based on this approach. Typical examples developed were the ones by University of Washington and IBM, GAT of Georgetown, Harvard, Wayne State University, and etc.

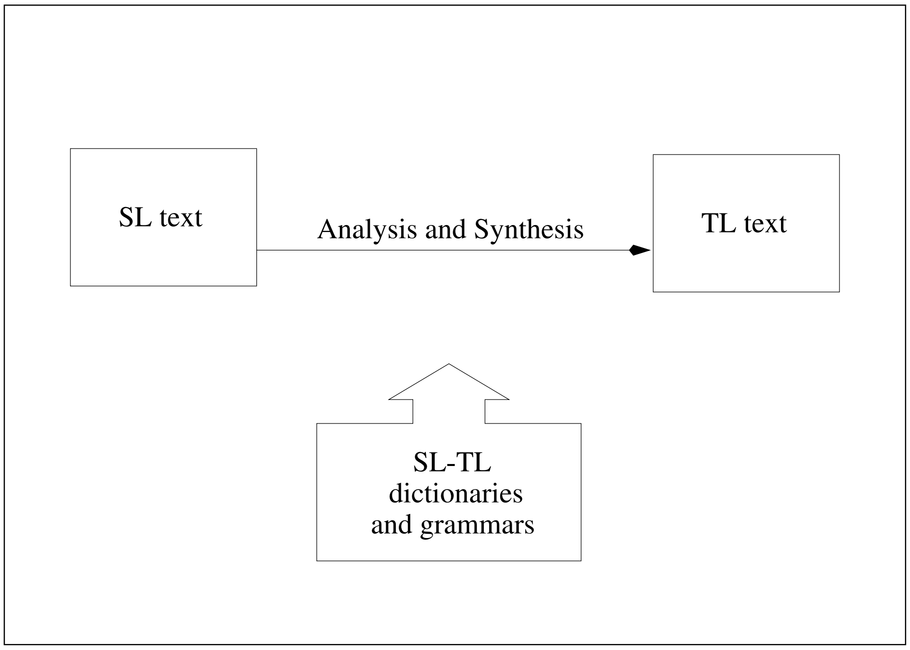

Figure 1.1: Direct Machine Translation

We can summarize some of the important design features of this approach as follows:

- Simple parsing of the input sentences for the successful operation of the transformation rules are needed. Only a few incomplete pieces of information about the structure of some of the phrases in a sentence are found, instead of getting a full and complete parse for the whole thing.
- The size of the grammar is very restricted. Therefore, it would not be able to decide for many input sentences whether it is grammatically acceptable.
- Other types of MT systems construct much more abstract and deep representations than direct MT systems
- These systems have some knowledge of the comparative grammar of two languages.
- They have no independent grammar and linguistic knowledge for TL, it uses the transformation rules rather than using a grammar for TL.

- They are highly robust especially for inputs which contain unknown words or unknown grammatical constructions.
- They can produce output that is simply unacceptable in the target language
- There are many different rules interacting in many different ways, so these systems are hard to understand, extend or modify.
- They are designed with translation in one direction and between one pair of languages. Therefore, they are not suitable for the development of multi-lingual systems

Probably the most famous example of a direct MT system is SYSTRAN [46], [106].

1.3.2 Transfer MT

Direct MT systems have many disadvantages, so they did not survive for a long time. As a consequence of the inadequateness of direct MT systems, new strategies were developed by the researchers. Transfer strategy is one of them. There are three fundamental steps in transfer systems. In the first step, SL text is converted to an internal abstract representation, then in the second step this internal structure of the SL text is transferred (both lexically and structurally) into corresponding TL representation or TL abstract and internal structure. Finally, TL text is produced from this structure in the third step. The model of this process is shown in Figure 1.2.

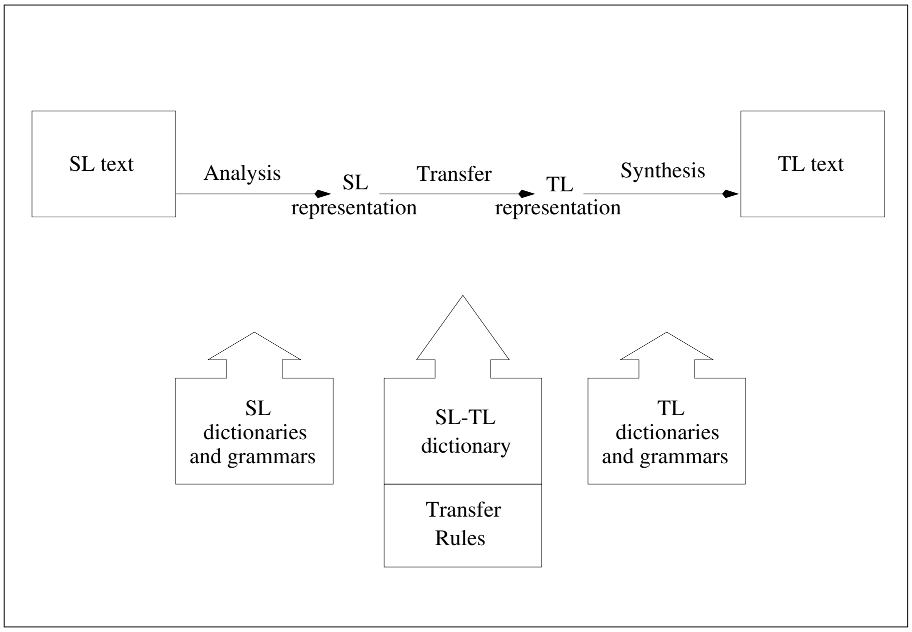

Figure 1.2: Transfer Machine Translation

The level of the transfer changes from system to system related to the representation used. In the transfer stage generally there is a bilingual component, which is specific to a certain SL-TL pair and this complicates the task for multi-lingual environments. As a consequence, the level of the transfer and relative weights of monolingual and bilingual parts play an important role in the improvement of the system for multilingual environments. The deeper the parsing towards more abstract levels of representations, the less the complexity will be. Figure 1.3 shows the level of complexity in representation. The deeper the representations used, the smaller the comparative grammar that is required to translate between two languages. As the representations become more abstract, (going through the top of the figure), there are fewer differences in SL and TL representations. This means that it will be easier to relate these representations. TRANSFER-i in Figure 1.3 shows the level of abstraction. TRANSFER-i uses more abstract representations than TRANSFER-(i+1). The level of representation where the target and source languages are identical and where no comparative grammar is needed is called **interlingua** shown as TRANSFER-0 level in Figure 1.3. In this level, the input of the target language synthesis component is the representations produced by the source language analysis component. There is no abstraction at the bottom level, and the size of the comparative grammar is maximum.

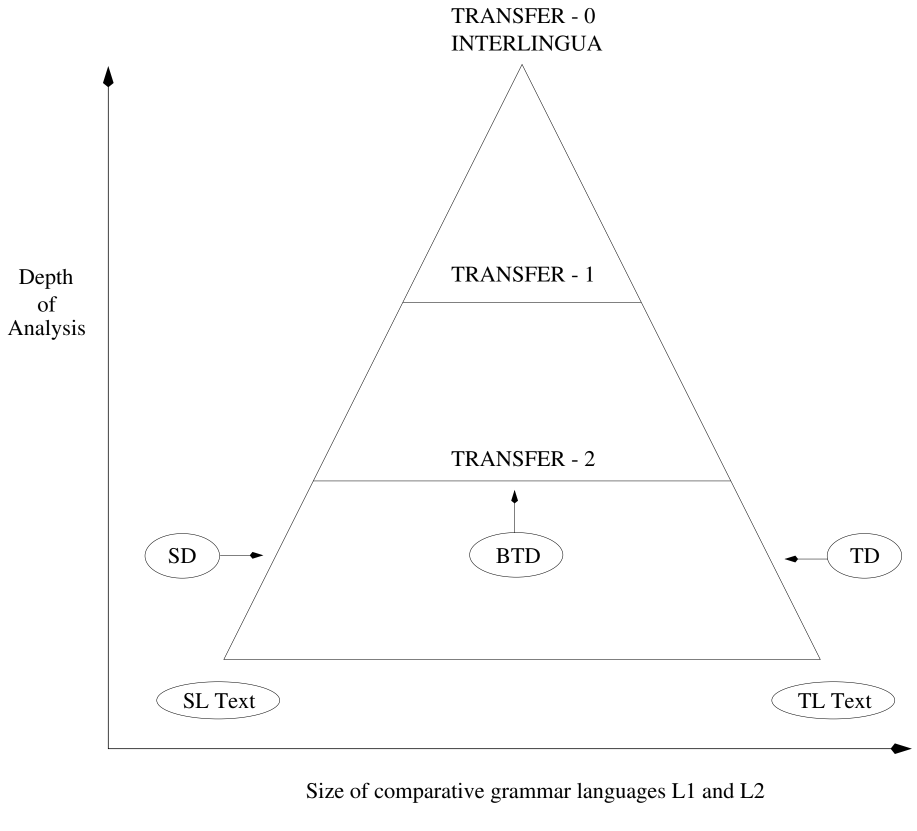

Figure 1.3: Variants of the transfer model of machine translation. SD is source dictionary, TD target dictionary, BTD is bilingual transfer dictionary.

There are various aspects of transfer MT systems which can be summarized as follows:

- **Intermediate representations in transfer:** According to the intermediate structure used, there is always a tradeoff between the level of abstractness and the complexity of the transfer. Problems of complex rule interaction as observed in the direct systems can occur. If the degree of interaction between rules are limited, then the number of facts that have to be stated will be less, and it will be both natural and economic in terms of effort involved. However, it increases the number of rules that must be applied.

- **Reversibility:** Transfer rules could be reversible in principle and this property seems to be advantageous, since the number of the transfer components will be halved. However, reversible transfer rules are not always possible, or desirable.

- **Well-formedness:** It is necessary to have well- formed TL structures as the output of the second step in transfer MT, so that we can obtain a useful and meaningful output from the synthesis of this structure. But, in order to achieve this goal, the transfer components can become rather complex.

- **Instructions for synthesis:** Some systems require extra information to be produced in the TL structure which will be used in synthesis phase, and this puts additional complexities to the transfer system.

- **Choosing between possible translations:** There may be more than one or none output produced by the transfer systems, which are not necessarily all correct, for SL inputs. Then the system must select the best answer or produce a nearly correct answer, or produce nothing. Then the system must find methods for the sequence of the rules that will be applied, or scoring the outputs in some way.

- **Declarative or procedural processing:** It will also be decided whether the system will be declarative or procedural, i.e. whether the order of the things to be done will affect the result. If it will not, then the system will be easy to understand, and modify. However, it will be more efficient to apply the most likely rules early or block some rules that will produce nonsense results.

The well-known transfer MT systems are GETA [103] in Grenoble, and SUSY [63] in Saarbrücken. Other transfer systems include the following: ARIANE by [102], MU (the Japanese National Project) [70], METAL [93], and [8], ETAP-2 [3], LMT [64], EUROTRA [6], [26], and [27], CAT-2 [92], MIMO [7], MIMO-2 [79], ELU [31]. Several of these systems are discussed in detail in [46].

1.3.3 Interlingua MT

Interlingua approach assumes that it is possible to convert an SL text into an internal structure which is common to more than one language. Its goal is to develop a universal representation which does not depend on a specific language. This goal can be achieved by using a deep analysis so that the need for transfer from SL to TL will be diminished. This means that the output of the analysis stage will be directly the input of the synthesis. The operation of these kinds of systems are shown in Figure 1.4.

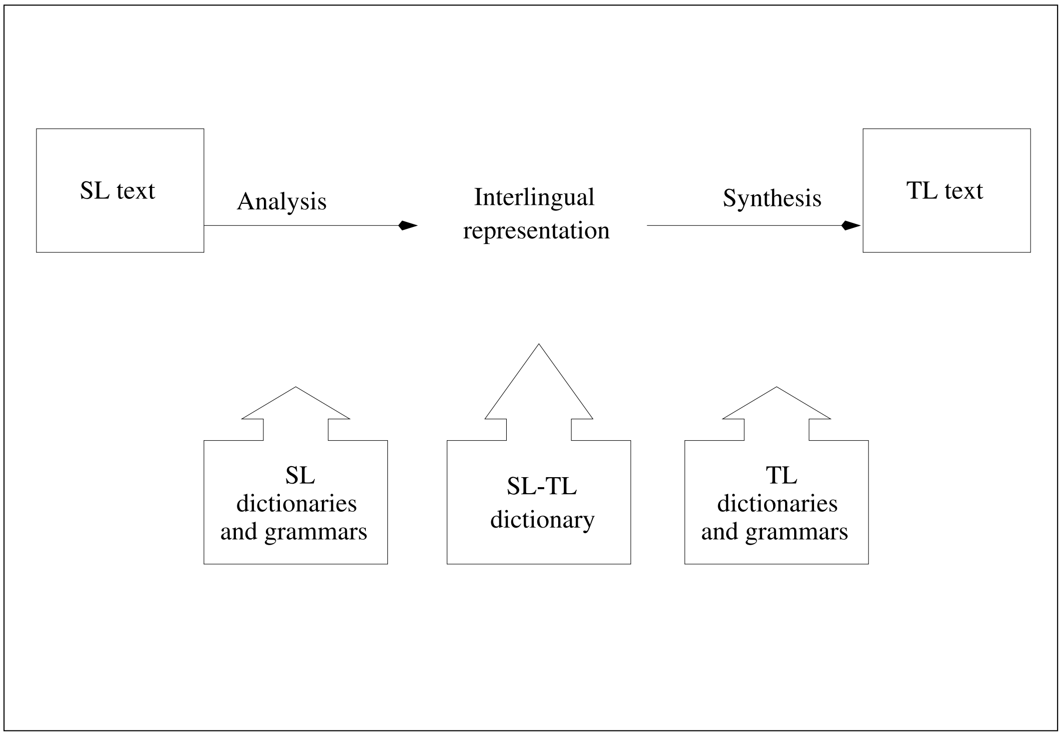

Figure 1.4: Interlingua Machine Translation

The advantage of these kinds of systems is their appropriateness for multilingual environments. In other words, it is easier to extend an interlingua MT system by adding new languages than to extend the transfer system. An empirical comparison between transfer and interlingua MT systems shows that if the number of SL and TL are O(n), we need O(n^2) transfer components for transfer approach, on the other hand we need O(n) components for interlingua systems. They are good for domains where the set of concepts and vocabulary is settled down.

However, there are many problems with interlingua MT systems. Producing a language independent representation means that we will use language independent representation of the words and the structures. Thus, it is necessary to find a way of distinction for the inputs which have the same words or structures but have different meanings or emphasis. The vocabulary selection is another important problem in these systems. Designer must consider whether an arbitrary language will be selected for conceptualization of the interlingua or all the possible distinctions among the languages will be considered. The first one may be inadequate for representation, and the latter one can be too complicated and the vocabulary size may increase too much. Other than these, interlingua may lead to extra work which is unnecessary due to the properties of the languages in the system.

Among interlingua systems , the following are noteworthy: Rosetta [60], [61], KBMT [36], [37]. Chapter 6 of [46] is recommended for an overview. One interlingua approach that has not been mentioned here is the one that uses a human language as the interlingua. The best known example of this is DLT, which uses Esperanto, see [91] and Chapter 17 of [46].

The properties of transfer and interlingua MT systems can be summarized as follows:

- The output will be more grammatical than the direct systems, since there is a partial grammar.
- Translation quality will be more reliable than for direct MT systems.
- It is relatively easy to extend the system, since the languages are separated into separate modules.
- Unusual input sentences will crash the transfer and interlingua systems due to grammar used.
- The grammars which are hand-crafted are not complete and they may not cover the complexity of the real world grammars, so there will be some complicated grammatical input sentences that the system fails to recognize.

Knowledge-Based machine translation (KBMT) is a typical derivation of interlingua approach. KBMT requires functionally (i.e. the meaning representation should be sufficient for translation, rather than sufficient for total understanding) complete understanding of the meaning of the SL text for a successful, high quality translation. It is a rule-based system in which an extensive amount of semantic and pragmatic knowledge and reasoning about the concepts of the domain exist. Therefore, the development of the domain model is the vital issue in KBMT. The main duty of the domain model is to support full disambiguation of the text. In order to achieve this, it is necessary for every event concept in the domain, to specify what restrictions are placed on the argument constituents of the object concept or the fillers of the slots in the representation.

The general architecture of a typical KBMT system is similar to the interlingua systems as shown in Figure 1.4. First, the SL text is analyzed by some tools, by the help of the knowledge recorded in the SL grammar and lexicon, then, interlingua text is produced. The interlingua expressions are defined in a specially designed unambiguous textual-meaning representation language. Interlingua texts are hierarchical structures of clause-level representation units connected through domain and textual relations from a predefined set ([73]). Then this interlingua text is passed to generator. Generator needs some additional components (e.g., text planner, lexical selection module, syntactic realizer etc.). Finally, generation is completed by using the TL grammar, lexicon and some other knowledge resources.

The advantages of KBMT systems are summarized in [73] as follows:

- KBMT systems are good for testing and devising new algorithms
- It is a comprehensive system, i.e. it is possible to test new components such as new parsers, generators etc.

- The interface, components, and computing technology of KBMT systems can be used by other applications.
- The acquisition and maintenance of large knowledge-bases can be done by the ontological and domain knowledge of KBMT system
- They can serve as a basis for other natural language processing applications

KBMT systems seem to be attractive for MT, however, given the current state of linguistic knowledge, there are serious problems in building a robust, general purpose, high quality KBMT system. These problems and the increasing availability of raw materials in the form of on-line dictionaries, term banks and corpus resources have led to a number of new developments in recent years. These developments try to minimize the linguistic knowledge engineering problem or make it more tractable at least.

Developing appropriate large-scale grammatical and lexical resources, which causes many other subproblems, is one of the serious problem for the MT approaches mentioned up to now. The first related problem is simply the problem of scaling, i.e. the numbers of linguistic rules and lexical entries needed for FAHQMT for general purpose and specialized language usage. If all such information must be manually coded, the effort that must be spent on this issue is awesome, although it can be assumed that our current state of linguistic knowledge is sophisticated enough.

Another serious issue concerns the difficulties of manipulating and managing such knowledge within a working system. It is possible to develop a wide natural language processing system by adding new rules, (that are too specific to deal with certain cases), when new problems are faced during the lifetime of the system. Then the system can solve those problems, however, it soon becomes difficult to understand, upgrade and maintain. Another disadvantage of adding these specific rules is the degradation in performance due to the interaction with other rules. In order to avoid these kinds of problems up to a certain level, it is necessary to restrict the use of special devices as much as possible. It is also very important to ensure that different grammar writers adopt essentially the same or consistent approaches and document everything they do in detail.

The quality and the level of linguistic details are other problems related to this subject. This problem shows up in a number of different areas, most notably in discriminating between different senses of a word, but also in relating pronouns to their antecedents. An extremely radical approach to this problem is to try to do away with explicitly formulated linguistic knowledge completely. This extreme form of the statistical approach to MT is found in the work carried out by MT group at IBM Yorktown Heights.

As well as the various difficulties in developing linguistic resources, there are other issues which must be addressed in the development of a working MT system.

- If a system is to be used on free text, then it must be robust. Being robust means:

  - It must have mechanisms for dealing with unknown words and ill-formed output (simply answering 'no' and refusing to proceed would not be cooperative behavior).
  - In a similar way, it must have a way of dealing with unresolved ambiguities,

- Integration of core MT engines with some additional tools is necessary:
  - Spell checkers
  - Fail-safe routines for what to do when a word in the input is not in the dictionary
  - Preference mechanisms which chose an analysis in cases of true ambiguity

1.3.4 Corpus-Based MT

In the previous section, the feasibility of the rule-based approaches had been examined, the difficulties and disadvantages of building such a system are explained. These issues mentioned above and the increasing availability of large amounts of machine readable textual material have been seen by a number of research groups. These led to different MT architectures which apply relatively low-level statistical or pattern matching techniques. In such approaches, all the linguistic knowledge that is used by the system is derived empirically. In other words, the linguistic knowledge is obtained by examination of real texts, rather than encoding manually. There are two major approaches that worth mentioning under the corpus-based MT. They are *example-based* or *analogy-based* approach, and the *statistical* approach.

1.3.4.1 Example-Based Translation

Most of the MT systems have many problems like tractability, scalability and performance, because they generally assume a model of the translation which involves explicit mapping rules of various sorts. In the *translation by analogy*, or *example-based* approach, such mapping rules are eliminated by using a procedure which involves matching against stored example translations. It is relatively a new paradigm for finding a scalable machine translation system which overcomes the problems mentioned above. This approach was first proposed by Nagao [69]. Then, various models have been proposed ([88], [58], [105], [98],[51], [95], [17], [23]) and different issues are discussed ([49]).

The idea of *translation by analogy* principle by Nagao suggests that one can translate a sentence by using translation examples of similar sentences. He claims that current MT systems tend to have increasing limitations proportional to complex information included in the system to improve performance. This fact motivated him to propose a system in which this problem is solved. Therefore, it will be suitable to use a model, so called analogical thinking, which is similar to the human translation. The system proposed is explained by Nagao himself as follows:

> Let us reflect about the mechanism of human translation of elementary sentences at the beginning of foreign language learning. A student memorizes the elementary

English sentences with the corresponding Japanese sentences. The first stage is completely a drill of memorizing lots of similar sentences and words in English, and the corresponding Japanese. Here we have no translation theory at all to give to the student. He has to get the translation mechanism through his own instinct. He has to compare several English sentences with the corresponding Japanese. He has to guess, make inferences about the structure of sentences from a lot of examples.

Along the same line as this learning process, we shall start the consideration of our machine translation system, by giving lots of example sentences with their corresponding translations. The system must be able to recognize the similarity and the difference of the given example sentences. Initially a pair of sentences are given, a simple English sentence and the corresponding Japanese sentence. The next step is to give another pair of sentences (English and Japanese) which is different from the first only by one word.

This word replacement operation is done one word at a time in the subject, object, and complement positions of a sentence with lots of different words. For each replacement someone should give the information to the system of whether the sentence is acceptable or non-acceptable. Then the system will obtain at least the following information from this experiment:

- Certain facts about the structure of a sentence
- Correspondence between English and Japanese words

It is also claimed that the human translation have three fundamental steps:

- Decomposing source language into certain fragments
- Translation of these phrases into target language by using analogy principle
- Combining the translated fragments to obtain the whole sentence

In order to adopt this model of translation, the EBMT systems have three fundamental steps:

- Finding the correspondence of units in a bilingual text
- Retrieving the best matches from previous translation examples
- Producing the translation of the given input by using these examples

The most important issue in the EBMT systems is the second step mentioned above, i.e. calculating how close the given input is to various stored example translations based on the distance of the input from the examples. This involves finding the *Most Specific Common Abstraction* for the input and the alternative translations and how *likely* the various translations are on the basis of frequency ratings for elements in the database of examples. This means it is assumed that the database of examples is representative of the texts intended to be translated. Various strategies offered to find the best matches and similarity metrics. These strategies reported are classified according to the text patterns they are applied to.

The word-based paradigms compare individual words of the two sentences or use a semantic distance ($0 \leq d \leq 1$) which is determined by most specific common abstraction (MSCA) obtained from a thesaurus abstraction hierarchy. When the word-based matching methods are concerned there are two important methods in the literature. *Dynamic Programming-matching* (DP-matching) method finds all possible matches by considering insertions, deletions, etc., and tries to find the optimum solution. On the other hand, *length first* method finds only the longest match. DP-matching by [58] uses single word as a unit of translation. However, this does not give a set of word matches equally divided along the whole sentence and can produce erroneous matches with isolated word match. The DP-matching method offered by [28] uses functional words or phrases, but the definition and the search of these do not always succeed. The length first method by [75] does not solve the problem of selecting the correct segment when multiple segments having the same length are equivalents.

The word-based methods are the most popular ones, but there are some other methods used in other systems. Watanabe [105] accepts a tree dependency structure of words in the sentence as the input, but they can fail during the construction of these structures. Sumita [99] uses syntax-rule driven method. This approach tries to find the similarity at the syntax level. This is the best approach offered as a translation proposal, especially if it is supported by lexical similarity. However, the complex task of syntactic analysis can decrease the time performance of these systems by a great amount. Sato [90] uses character-based method which can be helpful to capture certain characteristics of certain languages like Japanese. Finally, there are some other systems which use hybrid methods.

In the traditional MT systems, it is necessary to encode various facts of translations into rules, which is a very hard, error-prone, and time consuming task. On the other hand, since the main source of knowledge in the EBMT systems is the collection of translation data, the need for encoding rules manually is eliminated. This is the major fact which makes EBMT systems attractive, because writing rules is always more difficult than collecting translation examples. The quality of translation will improve incrementally as the example set becomes more complete, without the need to update and improve detailed grammatical and lexical descriptions. The accuracy of such a system increases proportional to the size of the examples, since it is easier to abstract various phenomena in translation activities into large number of examples than small number of examples. However, it is obvious that the feasibility of the approach depends strictly on the collection of good data. Moreover, the approach can be (in principle) very efficient, since in the best case there is no complex rule application to perform. All one has to do is to find the appropriate example and (sometimes) calculate distances. However, there are some complications. For example, one problem arises when one has a number of different examples each of which matches part of the string, but where the parts they match overlap, and/or do not cover the whole string. In such cases, calculating the best match can involve considering a large number of possibilities.

EBMT uses the available bilingual text resources from the previous human translations as its data. These raw data are statistically analyzed to obtain lexical and translation functions to avoid using pre-defined grammars. EBMT systems are more robust and scalable than the others. In addition to this, they are more promising for specific domains, due to their statistical grounding in past texts from that domain.

A pure example-based approach would use no grammar rules at all, only example phrases. However, one could also imagine a role for some normal linguistic analysis, producing a standard linguistic representation. If, instead of being given in simple string form, examples were stated in terms of such representations, one would expect to be able to deal with many more variations in sentence pattern, and allow for a certain amount of restructuring in generation.

When all the pros and cons of the example-based and rule-based systems are considered, the best conclusion that can be inferred is that the real challenge lies in finding the best combination of techniques from each. Here one obvious possibility is to use traditional rule-based transfer as a fall back, to be used only if there is no complete example-based translation.

The EBMT paradigm is relatively a new approach, and it has still some problems that remained unsolved before the construction of a commercial and practical MT system. These systems can be made more efficient by using massively parallel computers, and the accuracy can be increased by integrating them with the traditional MT systems.

1.3.4.2 Statistical MT

Statistical or mathematical machine translation is another corpus-based approach in MT researches. These approaches to Natural Language Processing have gained a growing interest over the last few years in the research community. In machine translation literature, the term *statistical approaches* can be understood to refer to approaches which try to avoid explicitly formulating linguistic knowledge, or in other words to denote the application of statistically or probabilistically based techniques to parts of the MT task. Here, it will be better to describe a pure statistical-based approach to MT.

In this approach, techniques which have been highly successful in *speech recognition* is applied to MT. Therefore, the details require a reasonable amount of statistical sophistication, however, the basic idea can be grasped quite simply. **Language model** and **translation model** are the two key notions involved. The language model provides the probabilities for strings of words (in fact sentences), which can be denoted by $Pr(S)$ (for a source sentence) and $Pr(T)$ (for any given target sentence). The translation model also provides other probabilities, where $Pr(T|S)$ is the conditional probability that a target sentence T will be obtained given that the input source sentence is S. The product of this and the probability of S itself, (i.e. $Pr(S) \times Pr(T|S)$ gives the probability of source-target pairs of sentences occurring. Then it is necessary to find out the probability of a source string occurring (i.e. $Pr(S)$)). This can be decomposed into the probability of the first word, multiplied by the conditional probabilities of the succeeding words, as follows:

$$Pr(S) = Pr(s1) \times Pr(s2|s1) \times Pr(s3|s1, s2), \ldots$$

The conditional probability $Pr(s2|s1)$ means the probability that s2 will occur, given that s1 has occurred. The following example clarifies how these probabilities are used. The probability that *am* and *are* occur in a text might be approximately the same, but the probability of *am* occurring after *I* is quite high, and it is a very rare case that *I* is followed by *are*. Thus the probability of the latter case is lower than the former one. Considering more than two or three words will be computationally expensive, so to have an efficient and cheap system, in calculating these conditional probabilities, it is common practice to take into account only the preceding one (that is known as **bigram model**) or two words (that is known as **trigram model**).

The requirements of such a system can be summarized as follows:

- The validity, usefulness or accuracy of the model will depend mainly on the size of the corpus, so to calculate these source language probabilities, in other words, to produce the source language model by estimating the parameters, a large amount of **monolingual data** is required.
- The parameters of the translation model can be specified by using a large **bilingual aligned corpus**.

The parameters which must be calculated from the bilingual sentence aligned corpus are then:

- The *fertility probabilities* for each source word (i.e. the likelihood of its translation as one, two, three, etc., word(s) respectively)
- The *word-pair or translation possibilities* for each word in each language
- The set of *distortion probabilities* for each source and target position

With this information (which is extracted automatically from the corpus), the translation model can, for a given S, calculate $Pr(T|S)$ (that is, the probability of T, given S). This is the essence of the approach to statistically-based MT, although the procedure is itself slightly more complicated in involving search through possible source language sentences for the one which maximizes $Pr(S) \times Pr(T|S)$, translation being essentially viewed as the problem of finding the S that is most probable given T ($Pr(S|T)$), i.e. one wants to maximize $Pr(S|T)$ (probability of the source sentence being S given that the target translation sentence is T) Given that:

$$Pr(S|T) = \frac{Pr(S) \times Pr(T|S)}{Pr(T)}$$

then one just needs to choose S that maximizes the product of $Pr(S)$ and $Pr(T|S)$.

The most popular work for statistical machine translation belongs to the researchers at IBM ([14, 12, 13]. The success of the statistics-based approaches in the speech recognition and parsing fields motivated these researchers for using similar methods in machine translation. Although, the main requirement of statistical MT systems is the huge corpus, there are rather few such resources. However, the research group at IBM had access to three million sentence pairs from the Canadian (French-English) Hansard (the official record of proceedings in the

Canadian Parliament). Using a huge corpus of text in machine readable form like this one, the probability that any word in a sentence in source language corresponds to zero, one or two words in the target language is calculated. All possible translation of every single word with its probability is stored in a glossary of word equivalences. For example *the* translates as *le* with a probability of 0.610, as *la* with a probability of 0.178, etc.

These probabilities will be helpful in making up the words of the target text by combining them in various ways and selecting the result which have the highest-scoring combination. In the next step the order of these words must be found, and this can also be done by statistical methods (i.e. bigram or trigram models).

Results to date in terms of accuracy have not been overly impressive, with a 39 per cent rate of correct translation reported on a set of 100 short test sentences. Translations which were either completely correct or preserved the meaning of the real translations were 48 per cent. Although, this performance seems to be bad, many other systems does not have a better performance than this one. A defect of this approach is that morphologically related words are treated as completely separate from each other, so that, for example, distributional information about *sees* cannot contribute to the calculation of parameters for *see* and *saw*, etc. The near-miss cases occur due to the fact that the system does not use any linguistic information. The problems occur especially when the translation of the words depends on the other ones. The researchers attempt to remedy this defect by adding low level grammatical information to their system, moving in essence towards an analysis-transfer-synthesis model of statistically-based translation. They claim that this will greatly improve the quality of the translation, but it is obvious that this will not solve all the complex problems of the natural language processing. The currently reported success rate with 100 test sentences is quite respectable, 60 per cent. A major criticism of this move is of course precisely that linguistic information is being added piecemeal, without a real view of its appropriacy or completeness. Additionally, it is questionable that how far the approach can be extended without further additions of explicit linguistic knowledge of grammar. Putting the matter more positively, it seems clear that there is a useful role for information about probabilities. However, the lower success rate for the *pure* approach without any linguistic knowledge (less than 40 per cent) suggests that the real question is how one can best combine statistical and rule-based approaches.

The advantages and the disadvantages of statistical approaches can be summarized as follows:

- In statistical approaches there is no role whatsoever for the explicit encoding of linguistic information, and thus the knowledge acquisition problem is solved.
- It is promising when used with some other approches. In other words hybrid systems which use statistical informations seem to be more successful.
- The usefulness of corpus resources depends very much on the state in which they are available to the researcher.

- The purity of the data is very effective on the system. Corpus clean-up and especially the correction of errors is a time-consuming and expensive business.
- The general applicability of the method might be doubted, since it is heavily dependent on the availability of good quality bilingual or multilingual data in very large proportions, something which is currently lacking for most languages.

Considering all MT applications and approaches the following results are offered by [43]:

- MT applications called assimilation tasks: (Such as scan translations of foreign documents and newspapers, requires lower-quality, broad domains) primarily statistical approaches
- Dissemination tasks: (Such as translations of manuals and business letters, higher-quality, limited domains) primarily example-based technology
- Narrowband communication: (Such as e-mail translation, medium-quality) highly hybridized technology.

For further reading in machine translation see the following sources: On knowledge-based MT see [37, 18, 68, 67, 73], and the special issue of the journal Machine Translation, [36]. For other types of rule-based systems see [2, 4, 52, 59, 32, 78, 100, 29, 82]. On the processing of corpora, and their use in linguistics generally, see [35], and [1]. For a review of more recent work along example-based MT see [96, 94, 33, 38, 65, 76, 77, 81, 82]. The pure statistical approach to MT is based on the work of a team at IBM, see [12, 13, 14, 15, 16, 17, 97, 30]. For general overviews and evaluations see ([41, 44, 53])

1.4 Machine Translation Today

Machine Translation at Center for Machine Translation at Carnegie Mellon University (**CMT**) is one of the important researches of today that worths mentioning. There are currently a number of active machine translation projects (see [107]) at the CMT such as:

DIPLOMAT: A speech-to-speech translation system between new language pairs, using MT techniques primarily developed during the Pangloss project. First test case: Serbo-Croatian/English.

JANUS: A Speech-to-Speech Machine Translation system in multilingual environment. Primarily using an interlingua based approach. The domain is restricted to conversations between travel agents and clients. Current set of languages includes English, German, Japanese, Korean, Italian and French by the help of other members of the C-STAR consortium. System applications include an Interactive Video Translation Station, a Portable Translator, and a Passive Dialog Interpreter.

KANT: A Knowledge-Based Machine Translation system for translating multilingual technical documents founded in 1989. Vocabulary and grammar is restricted to achieve very high accuracy in translation.

PANGLOSS: A machine translation system by a collaboration of CMT at Carnegie Mellon University, the Computing Research Laboratory (**CRL**) at New Mexico State University, and Information Sciences Institute of the University of Southern California Translation of unrestricted texts from Spanish to English or (recently) Japanese to English. High quality is achieved by human assistance. The fully-automated statistical version achieves lower quality.

Recent activities at the CRL of New Mexico State University for developing robust large-scale machine-translation are as follows:

Artwork III: A machine translation system for translation of spoken dialogues. Models of the task domain and conversational interaction are sought to provide robustness.

Corelli: Corelli tries to extend the capabilities of the Pangloss and Temple Translator's Workstation (TWS) from English and Spanish to include Arabic, Russian and Japanese. In particular, Corelli expands available on-line tools such as dictionary, and user glossary.

Pragmatics Based Machine Translation: The pragmatics of the translation process is the center of interest. Multiple translations of the same text are analyzed. The differences in the translations are used to establish a reasoning model.

There are many other MT related projects at CRL. Mikrokosmos (comprehensive semantic analysis of texts for knowledge-based machine translation), and Oleada/Cobola (user-centered multilingual language technology for teaching and machine aided human translation) are some of the MT related projects that continue at CRL.

The main goal of the machine translation research in the CLIP Laboratory in University of Maryland (**UMIACS**) is to investigate the applicability of a linguistic-based framework to the problem of large-scale and extendible interlingua machine translation. A prototype MT system (PRINCITRAN) for Arabic, English, Korean, and Spanish is currently developed. GAIJIN is recent example-based machine translation system from Dublin City University for English and German ([104]). ReVERb is another example-based machine translation system from Artificial Intelligence Lab in Trinity College Computer Science Department ([20], [21], [22], [23],[24], [25]). JAPANGLOSS developed at USC-ISI ([109]) uses statistics and linguistics. PROTEUS is an example-based machine translation system which is still under construction and it is being developed at New York University ([110]) for English and Spanish. CANDIDE is a statistical machine translation system built at IBM for English and French ([9]). Verbmobil is a long-term project of the Federal Ministry of Education, Science, Research and Technology (BMBF) [42]. National partners are about 7 industrial and 22 university institutes (including Siemens, Daimler-Benz, IBM, Philips, the German Research Center for Artificial Intelligence and the universities of Hamburg, Karlsruhe, Munich, etc.).

Chapter 2

Translation Templates

In this chapter, the development of our machine translation system will be examined in detail. Providing some background information about the progress in the project up to now will be helpful for understanding the statistical model offered in the next chapter.

Corpus-based MT systems have many advantages compared to other approaches (see Chapter 1). Therefore, our EBMT MT system for English and Turkish, which can be adaptable to any other language pairs, was designed and implemented ([40, 19]). Statistical MT techniques use statistical metrics to choose the best structures in the target language among all possible candidates. These techniques are useful for retrieving the best matches from the previous translation examples, which is a vital issue in EBMT. This fact motivated us to use statistical MT strategies in our machine translation system. This section summarizes the progress in this project before adding statistical information.

Using previous examples for learning from new examples is the main idea behind exemplar-based learning which is originally proposed by Medin and Schaffer [66]. This way of learning stores the examples in memory without any change in the representation. The characteristic examples stored in the memory are called exemplars.

In the translation process, providing the correspondences between the source and target languages is a very difficult task in EBMT. Although, manual encoding of the translation rules has been achieved by Kitano [58], when the corpus is very large, it becomes a complicated and error-prone task. Therefore, [40, 19] offered a technique in which the problem is taken as a machine learning task. Exemplars are stored in the form of templates that are generalized exemplars. The templates are learned by using translation examples and finding the correspondences between the patterns in the source and target languages. The heuristic of the translation template learning (TTL) [40, 19] algorithm can be summarized as follows: given two translation pairs, if there are some similarities (differences) in the source language, then the corresponding sentences in the target language must have similar (different) parts, and they must be translations of the similar (different) parts of the sentences in the source language. Similar parts are replaced with variables to get a template which is a generalized exemplar by this method.

2.1 The Structure of The Translation Templates

Translation examples are stored as a list of string formed by strings of root words and morphemes. In other words, the lexical level representation of the sentences are used. This representation of translation examples is suitable for learning algorithm. If we used surface level representation, the number of correspondences would be decreased and we could learn less number of generalized exemplars. For example the sentence pair **I came from school$\Leftrightarrow$ ben okuldan geldim** is stored as:

i come+p from school$\Leftrightarrow$ben okul+DAn gel+DH+m

where *i, come, from, school* denote root words and *+p* denotes the past tense morpheme in English sentence, and *ben, okul, gel* denote root words and *+DAn, +DH, +m* denote ablative, past tense and first singular person morphemes in Turkish sentence.

A template is an example translation pair where some components (e.g., words stems and morphemes) are generalized by replacing them with variables in both sentences, and establishing bindings between variables. Assume that the following is a template learned from the translations examples: **I go+p to** X^{L_1} **by bus** $\leftrightarrow$ X^{L_2}**+yA otobüs+ylA git+DH+m**

This template can be interpreted as follows: **I go+p to** X^{L_1} **by bus** in L_1 is the translation of X^{L_2}**+yA otobüs+ylA git+DH+m** in L_2 (or vice versa), if X^{L_1} in L_1 is the translation of X^{L_2} in L_2. So, if it has already been learned that *school* in L_1 is the translation of *okul* in L_2, then **I go+p to school by bus** can be translated as **okul+yA otobüs+ylA git+DH+m** Here, L_1 and L_2 denote English and Turkish respectively, but they can be used for any other language pair.

The following translation pairs given in English and Turkish illustrate how
templates are learned:

<u>I go+p to</u> school <u>by bus</u>$\leftrightarrow$ okul <u>+yA otobüs+ylA git+DH+m</u>
<u>I go+p to</u> city <u>by bus</u>$\leftrightarrow$ şehir <u>+yA otobüs+ylA git+DH+m</u>

Then the following template is learned from these examples:

I go+p to X^{L_1} by bus $\leftrightarrow$ X^{L_2}+yA otobüs+ylA git+DH+m
if $X^{L_1} \leftrightarrow X^{L_2}$

In addition to this abstract representation, it is also inferred that *school* is the translation of *okul* and *city* is the translation of *şehir*. This shows that it is possible to learn more than one template by using two translation examples.

2.2 TTL Algorithm

In this section the mechanism of the TTL algorithm will be explained. TTL algorithm learns translation templates by using the similarities and differences of the example translation pairs given to the system.

We have translation examples $E_a : E_a^1 \leftrightarrow E_a^2$ where E_a^1 and E_a^2 are the translations of each other. For two example translation pairs E_a, E_b similarities and differences between these two examples are found. Then a match sequence in the following form is generated:

$$S_0^1, D_0^1, S_1^1, D_1^1, \ldots, D_{n-1}^1, S_n^1 \leftrightarrow S_0^2, D_0^2, S_1^2, D_1^2, \ldots, D_{m-1}^2, S_m^2 for 1 \leq n, m$$

where S_k^1 denotes a similarity, and $D_k^1 : (D_{k,a}^1, D_{k,b}^1)$ denotes a difference between E_a^1 and E_b^1.

Some restrictions on these similarities and differences result in a unique or no match between examples. These restrictions are:

- For any difference D_k, $D_{k,a}$ and $D_{k,b}$ do not have any common items
- No lexical item in a similarity S_i appears in a previously formed difference D_k for $k < i$
- Any of $S_0^1, S_n^1, S_0^2, S_m^2$ can be empty, however, S_i^1 for $0 < i < n$, and S_j^2 for $0 < j < m$ must be non-empty
- At least one similarity on each side must be non-empty

These match sequences are used to learn translation templates by using two heuristics. The first heuristic replaces differences in the match sequence and produces a template which is called *similarity translation template*. The second replaces the similarities in the match sequence and produces a template which is called *difference translation template*. All pairs in the example translations are used to produce templates by the help of these two heuristics. Then these templates are stored in the memory by sorting them according to their specificities, i.e. according to the number of terminal symbols in source language side of the template.

2.2.1 Similarity Translation Template

Similarity translation template learning algorithm is given in Table 2.1.

The following examples illustrates how the algorithm produces similarity translation templates for different cases.

If our translation pairs are:

$\underline{\text{I go+p to}}$ school $\underline{\text{by bus}}$ $\leftrightarrow$ okul $\underline{\text{+yA otobüs+ylA git+DH+m}}$

$\underline{\text{I go+p to}}$ city $\underline{\text{by bus}}$ $\leftrightarrow$ şehir $\underline{\text{+yA otobüs+ylA git+DH+m}}$

The match sequence for this example is:

I go+p to (school, city) by bus $\leftrightarrow$

(okul, şehir)+yA otobüs+ylA git+DH+m

Then the similarity translation template learned is:

begin
Let's assume that the match sequence $M_{a,b}$ for the pair of translation examples E_a and E_b be:$S_0^1, D_0^1, \ldots, D_{n-1}^1, S_n^1, \leftrightarrow S_0^2, D_0^2, \ldots, D_{m-1}^2, S_m^2$
if $n = m = 1$ **then**

- learn the following rules:
  $S_0^1 X^1 S_1^1 \leftrightarrow S_0^2 X^2 S_1^2 if X^1 \leftrightarrow X^2$
  $D_{0,a}^1 \leftrightarrow D_{0,a}^2$
  $D_{0,b}^1 \leftrightarrow D_{0,b}^2$

else if $1 < n = m$ **and** $n-1$ corresponding differences can be found in $M_{a,b}$ **then**

- Assume that the unchecked corresponding differences are $((D_{k_n,a}^1, D_{k_n,b}^1), (D_{l_n,a}^2, D_{l_n,b}^2))$
- Assume that the list of corresponding differences is $(D_{k_1}^1, D_{l_1}^2) \ldots (D_{k_n}^1, D_{l_n}^2)$ including the unchecked ones.
- For each corresponding difference $(D_{k_i}^1, D_{l_i}^2)$ replace $D_{k_i}^1$ by X_i^1 and $D_{k_i}^2$ X_i^2 to get the new match sequence $M_{a,b}WithDVars$.
- Learn the following rules:
  $M_{a,b}WithDVars$ if $X_1^1 \leftrightarrow X_1^2$ and ... and $X_n^1 \leftrightarrow X_n^2$
  $D_{k_n,a}^1 \leftrightarrow D_{l_n,a}^2$
  $D_{k_n,b}^1 \leftrightarrow D_{l_n,b}^2$

end

Table 2.1: Similarity Translation Template Learning Algorithm

I go+p to X^1 by bus $\leftrightarrow$ X^2+yA otobüs+ylA git+DH+m
if $X^1 \leftrightarrow X^2$

Here, we have a single difference, i.e. n=m=1, in both sides, so they must be the translation of each other. If we generalize this case, it can be inferred that if the match sequence is:

$$S_0^1, D_0^1, S_1^1 \leftrightarrow S_0^2, D_0^2, S_1^2$$

Then the similarity translation template learned is:

$$S_0^1, X^1, S_1^1 \leftrightarrow S_0^2, X^2, S_1^2 \ \ \textbf{if} \ \ X^1 \leftrightarrow X^2$$

Two additional templates are also inferred from these example pairs:

$$D_{0,a}^1 \leftrightarrow D_{0,a}^2 D_{0,b}^1 \leftrightarrow D_{0,b}^2$$

In the following case, the number of differences is greater than 1, i.e.
$1 < n = m$. Assume that our translation pairs are:

I go+p to school by bus↔ okul +yA otobüs+ylA git+DH+m

You go+p to city by bus↔ şehir +yA otobüs+ylA git+DH+n

Then the match sequence is:

(I, you) go+p to (school, city) by bus ↔

(okul,şehir)+yA otobüs+ylA git+DH (+m, +n)

Here, we need to know correspondence for one of the differences in order to learn a similarity translation template. Assume that, we inferred that **school** corresponds to **okul** and **city** corresponds to **şehir**.

Then it is possible to infer the following similarity translation template:

X_1^1 go+p to X_1^2 by bus ↔ X_2^1+yA otobüs+ylA git+DH+ X_2^2

if $X_1^1 \leftrightarrow X_2^2$ and $X_1^2 \leftrightarrow X_2^1$

Additionally, the following templates are inferred:

I ↔ +m

you ↔ +n

In general, if the number of differences in both sides of a match sequence is greater than or equal to 1, i.e. $1 \leq n = m$, it is possible to learn new similarity translation templates iff at least n-1 differences have already been learned from the previous example pairs. Then all differences are replaced by variables to obtain new template. In addition to this template, after finding n-1 corresponding differences, the remaining difference pairs are also learned as new templates.

2.2.2 Difference Translation Template

The difference translation template learning algorithm is given in Table 2.2. The idea is similar to the similarity translation template learning algorithm, but in this algorithm similarities are replaced by variables instead of differences.

The following examples illustrate how the algorithm produces difference translation templates for different cases.

If our translation pairs are:

I drink+p tea ↔ Çay iç+DH+m

You drink+p coffee ↔ Kahve iç+DH+n

Here, we have a single non-empty similarity in both sides, so they must be the translation of each other.

The match sequence for this example is:

(I, you) drink+p (tea,coffee) ↔ (Çay, Kahve) iç+DH (+m, +n)

Then, the difference translation templates learned are:

---

begin
if $numofsim(M^1_{a,b}) = numofsim(M^2_{a,b}) = n \geq 1$
and $n-1$ corresponding similarities can be found in $M_{a,b}$ **then**

- Assume that the unchecked similarities are $(S^1_{k_n}, S^2_{l_n})$.
- Assume that the list of corresponding similarities is $(S^1_{k_1}, S^2_{l_1}) \ldots (S^1_{k_n}, S^2_{l_n})$ including the unchecked ones.
- For each corresponding difference $(S^1_{k_i}, S^2_{l_i})$ replace $S^1_{k_i}$ with X^1_i and $S^2_{k_i}$ with X^2_i to get the new match sequence $M_{a,b}WithSVars$.
- Divide $M_{a,b}WithSVars$ into $M_aWithSVars$ and $M_bWithSVars$ by separating differences.
- Learn the following rules:
  $M_aWithDVars$ if $X^1_1 \leftrightarrow X^2_1$ and ... and $X^1_n \leftrightarrow X^2_n$
  $M_bWithDVars$ if $X^1_1 \leftrightarrow X^2_1$ and ... and $X^1_n \leftrightarrow X^2_n$
  $S^1_{k_n} \leftrightarrow S^2_{l_n}$

end

Table 2.2: Difference Translation Template Learning Algorithm

I X^1 tea $\leftrightarrow$ Çay X^2 +m
if $X^1 \leftrightarrow X^2$
You X^1 coffee $\leftrightarrow$ Kahve X^2 +n
if $X^1 \leftrightarrow X^2$

Additionally, it is inferred that:
drink+p $\leftrightarrow$ iç+DH

In the following case, the number of similarities is greater than 1.

Assume that our translation pairs are:

I read+p the book to children$\leftrightarrow$ Çocuk+lAr+yA kitap +yH oku+DH+m
I read+p the newspaper to children$\leftrightarrow$ Çocuk+lAr+yA gazete +yH oku+DH+m

Then the match sequence is:
I read+p the (book, newspaper) to children $\leftrightarrow$ Çocuk+lAr+yA (kitap, gazete) +yH oku+DH+m

Here, we need to know correspondence for one of the similarities in order to learn a difference translation template. Assume that, we inferred that **I read+p the** corresponds to **+yH oku+DH+m**.

Then it is possible to infer the following difference translation templates:

X^1_1book $X^2_1 \leftrightarrow X^1_2$ kitap X^2_2
if $X^1_1 \leftrightarrow X^2_2$ and $X^2_1 \leftrightarrow X^1_2$
X^1_1 newspaper $X^2_1 \leftrightarrow X^1_2$ gazete X^2_2
if $X^1_1 \leftrightarrow X^2_2$ and $X^2_1 \leftrightarrow X^1_2$

Additionally, the following template is inferred:
to children $\leftrightarrow$ Çocuk+lAr+yA

In general, if the number of similarities in both sides of a match sequence is greater than or equal to 1, it is possible to learn new difference translation templates iff at least n-1 similarities have already been learned from the previous example pairs. Then, all similarities are replaced by variables to obtain new templates. In addition to these templates, after finding n-1 corresponding similarities, the remaining similarity pairs are also learned as a new template.

2.2.3 Inequalities in the Number of Similarities or Differences in the Match Sequences

The STTL (similarity translation template learner) and DTTL (difference translation template learner) algorithms can produce new translation templates if the number of differences or similarities are equal on both sides of a match sequence respectively. However, it is possible to learn new templates from these kinds of match sequences if the number of differences or similarities can be equated. For this purpose, the similarities or differences in the match sequence can be separated before producing the translation template.

The following example is a simple case where STTL algorithm can not produce a translation template:

I read $\underline{\text{+pres}}$ $\leftrightarrow$ oku $\underline{\text{+Hr}}$ +Hm
You sleep $\underline{\text{+pres}}$ $\leftrightarrow$ uyu $\underline{\text{+Hr}}$ +sHn

Then, the match sequence is:

(I read, You sleep) +pres $\leftrightarrow$ (oku, uyu) +Hr (+Hm, +sHn)

Here, we have one difference on the left side and two on the right side. So, we can equate the number of differences by separating the difference of the left side from morpheme boundaries.

So the match sequence becomes:

(I, you) (read, sleep) +pres $\leftrightarrow$ (oku, uyu) +Hr (+Hm, +sHn)

Now, the number of differences on both sides are the same and assume that it is previously learned that (**read, sleep**) corresponds to (**oku, uyu**).

Then, we can learn the following translation templates:

X_1^1 X_2^1 +pres $\leftrightarrow$ X_1^2 +Hr X_2^2
if $X_1^1 \leftrightarrow X_2^2$ and $X_2^1 \leftrightarrow X_1^2$
I $\leftrightarrow$ +Hm
You $\leftrightarrow$ +sHn

It is also possible that separating differences or similarities will be necessary, even if the number of the differences or similarities are equal. Another interesting case occurs, when the match sequence in which a difference with an empty constituent occurs. We apply only the DTTL algorithm to these kinds of match sequences to avoid producing a template whose one side is empty.

The following is a typical example of this case:

$\underline{\text{I read+p}}$ the $\underline{\text{book}}$ $\leftrightarrow$ $\underline{\text{kitap}}$+yH $\underline{\text{oku+DH+m}}$

$\underline{\text{I read+p}}$ a $\underline{\text{book}}$ $\leftrightarrow$ bir $\underline{\text{kitap}}$ $\underline{\text{oku+DH+m}}$

Then, the match sequence is:

I read+p (the, a) book $\leftrightarrow$ (ϵ, bir) kitap (+yH, ϵ) oku+DH+m

Then, we can learn the following translation templates if it is known that **book** corresponds to **kitap**:

$X_1^1 the X_2^1 \leftrightarrow X_1^2 + yH X_2^2$ if $X_1^1 \leftrightarrow X_2^2$ and $X_2^1 \leftrightarrow X_1^2$

$X_1^1 a X_2^1 \leftrightarrow bir X_1^2 X_2^2$ if $X_1^1 \leftrightarrow X_2^2$ and $X_2^1 \leftrightarrow X_1^2$

I read+p $\leftrightarrow$ oku+DH+m

2.3 Translation

Templates produced by STTL and DTTL are ordered according to the number of terminals in the source language. The translation is a bidirectional process, so templates are ordered according to both languages.

For example, some of the templates that are learned from a set of examples are:

he X_1^1+p $\leftrightarrow$ X_1^2+DH if $X_1^1 \leftrightarrow X_1^2$

ali X_1^1+p $\leftrightarrow$ ali X_1^2+DH if $X_1^1 \leftrightarrow X_1^2$

read $\leftrightarrow$ oku

Then, they will be ordered in the following way for translation from English to Turkish:

he X_1^1+p $\leftrightarrow$ X_1^2+DH if $X_1^1 \leftrightarrow X_1^2$

ali X_1^1+p $\leftrightarrow$ ali X_1^2+DH if $X_1^1 \leftrightarrow X_1^2$

read $\leftrightarrow$ oku

And they are ordered as follows if translation is from Turkish to English:

ali X_1^1+p $\leftrightarrow$ ali X_1^2+DH if $X_1^1 \leftrightarrow X_1^2$

he X_1^1+p $\leftrightarrow$ X_1^2+DH if $X_1^1 \leftrightarrow X_1^2$

read $\leftrightarrow$ oku

The translation process is straightforward after the translation templates are produced. It is summarized in Table 2.3.

If the input sentence is *i went to school*, then the translation will take place according to the steps in Table 2.3 as follows:

- Lexical level representation is *i go+p to school*
- Assume the template similar to this input is: I go+p to $X_1 \leftrightarrow X_2$ +yA git+DH+m **if** $X_1 \leftrightarrow X_2$
- It is also found that there is a template school $\leftrightarrow$ okul Then the target sentence is obtained as *okul+yA git+DH+m*

- The lexical level representation is derived for the input sentence
- Templates that are similar to the input are collected
- The variables of the selected templates are bounded by the values of the input sentence. Then, templates for these bounded values are sought. The values of these templates are replaced in the target sentence if they are found among the translation templates.
- Surface form of the target sentence is generated

Table 2.3: Translation Without Confidence Factors

- The surface form is generated as *okula gittim*

2.4 Examples

The operations of the STTL and DTTL algorithms will be explained by the following small example. Assume that the training examples are:

1. i must come ↔ gel+mAlH+yHm
2. i must go ↔ git+mAlH+yHm
3. i go+p ↔ git+DH+m
4. they come+p ↔ gel+DH+lAr

For Pairs 1 and 2:
Match Sequence:

i must (come, go) ↔ (gel, git)+mAlH+yHm

Templates produced by STTL:

i must X ↔ Y+mAlH+yHm **if** X ↔ Y

come ↔ gel

go ↔ git

Templates produced by DTTL:

X come ↔ gel Y **if** X ↔ Y

X go ↔ git Y **if** X ↔ Y

i must ↔ +mAlH+yHm

For Pairs 1 and 3:
Match Sequence:

i (must come, go+p) $\leftrightarrow$ (gel+mAlH+yHm, git+DH+m)

Since we have a single difference on the left side, nothing can be inferred from these example pairs.

For Pairs 1 and 4:

Match Sequence:

(i must, they) come (ϵ, +p) $\leftrightarrow$ gel (+mAlH+yHm, +DH+lAr)

Since we have empty constituent in the match sequence, we can only apply DTTL algorithm. Templates produced by DTTL:

i must X $\leftrightarrow$ Y+mAlH+yHm **if** X $\leftrightarrow$ Y

they X+p $\leftrightarrow$ Y+DH+lAr **if** X $\leftrightarrow$ Y

come $\leftrightarrow$ gel

For Pairs 2 and 3:

Match Sequence:

i (must, ϵ) go (ϵ, +p) $\leftrightarrow$ git (+mAlH+yHm, +DH+m)

Since we have empty constituent in the match sequence we can only apply DTTL algorithm. However, we have a single similarity on the left side, so nothing can be inferred from these example pairs.

For Pairs 2 and 4:

Match Sequence:

Match sequence is empty, i.e. these examples do not have any similarities, so nothing can be inferred from these example pairs.

For Pairs 3 and 4:

Match Sequence:

(i go, they come) +p $\leftrightarrow$ (git, gel), +DH (+m,+lAr)

We have a single difference on the right hand side, however, there are two differences on the left hand side. STTL can learn something useful, if the difference on the right is decomposed into two. Then match sequence becomes

(i, they) (go, come) +p $\leftrightarrow$ (git, gel), +DH (+m,+lAr)

Templates produced by STTL:

X Z+p $\leftrightarrow$ Y+DH W **if** X $\leftrightarrow$ W **and** Z $\leftrightarrow$ Y

i $\leftrightarrow$ +m

they $\leftrightarrow$ +lAr

Templates produced by DTTL:

i go X ↔ git Y+m **if** X ↔ Y

they come X ↔ gel Y+lAr **if** X ↔ Y

+p ↔ +DH

There are some templates, that are learned more than once, after eliminating the duplicates all the templates learned from these examples are:

1. i must X ↔ Y+mAlH+yHm **if** X ↔ Y
2. come ↔ gel
3. go ↔ git
4. X come ↔ gel Y **if** X ↔ Y
5. X go ↔ git Y **if** X ↔ Y
6. they X+p ↔ Y+DH+lAr
7. X Z+p ↔ Y+DH W
   if X ↔ W **and** Z ↔ Y
8. i ↔ +m
9. they ↔ +lAr
10. i go X ↔ git Y+m **if** X ↔ Y
11. they come X ↔ gel Y+lAr **if** X ↔ Y
12. +p ↔ +DH

Assume that after learning these templates, we are given the input sentence *i came* to be translated. Its lexical form is *i come+p*. In the translation of this sentence it matches template 7, then **X** is instantiated as **i** and **Z** is instantiated as **come**. In the next step it is found that **+m** corresponds to **i** from template 9, and **gel** corresponds to **come** from template 2. So **Y** is instantiated as **gel** and **W** is instantiated as **+m** giving *gel+DH+m* whose surface form is *geldim* as the result of the translation.

Chapter 3

Methods for Assigning Confidence Factors

The ordering of the translation templates is an important issue which effects the results of the translation process. Changing the order of their application may change the outputs significantly. The aim of translation is not finding the correct results somewhere in the hundreds of results, but finding them as the topmost ones. Therefore, an ordering mechanism for these translation templates becomes a vital issue. This mechanism will enable us to use the templates in such a way that they will produce the most correct results first.

The translation templates are ordered according to the number of terminal symbols of the templates in the previous version of TTL algorithm [40]. However, this criteria is not sufficient for large systems, and we need another method where a statistical method is a powerful candidate, in order to improve the soundness of the translation process. Therefore, in the new version of the TTL algorithm, learning translation templates is followed by a confidence factor assignment process in which each rule and some rule combinations are assigned weights. Our main resource for assigning confidence factor is the training data that is used in the learning of translation templates. The idea is simply the following: If we learned a template from a set of translations examples, then this left side of the template matches the left side of a translation example, then the right side of it must match the right side of that example if it is a correct template. If this is not the case, then the template can either be a totally incorrect one, or there is more then one correspondence for the given language pair. For the former case the template must be assigned a zero or a very small confidence factor, for the latter one the template must be assigned a frequency. For example if we have a template *hundred*↔*yüz*, then the confidence factor of it must be the frequency of *yüz* in the training set corresponding to *hundred*, since *yüz* has other meanings in English, like *face, swim, etc.*

This process has three fundamental parts: Confidence factor assignment to facts (i.e. specific templates without variables), rules (i.e. generalized templates in which the similarities are replaced with variables) and rule combinations. These three parts are explained in detail in the following sections.

Our translation process is bidirectional. In other words, it is possible to give an input sentence in language L_1 and obtain a translation in language L_2 and vice versa. Therefore we have templates that will be used for translation from L_1 to L_2, (left to right) and from L_2 to L_1(right to left). In fact, these templates are equal, but their order is different.

3.1 Method for Assigning Confidence Factors to Facts

In this section confidence factor assignment to facts, which are the simplest case of this process, are discussed. We do not need to consider any other rule during this process and we use only the translation examples.

Consider the case that, $rule_k$ is a fact which will be used for left to right translation. Assume that, it is in the form of $X \Leftrightarrow Y$ and we have training pairs in the form of $trainpair(X_i, Y_i)$ then the confidence factor of $rule_k$ for left to right translation is evaluated as follows:

- $N1$ denotes the number of training pairs where X is a substring of X_i and Y is a substring of Y_i
- $N2$ denotes the number of training pairs where X is a substring of X_i and Y is not a substring of Y_i

$$confidencefactor_{rule_k} = \frac{N1}{N1+N2}$$

If $rule_k$ is a fact which will be used for right to left translation, everything will be the same except definition of $N2$

- $N2$ denotes the number of training pairs where X is not a substring of X_i and Y is a substring of Y_i

Consider the following example for the illustration:

If $rule_k$ is **+s→Hr**, (i.e. it is a fact and it will be used in left to right translation), and our training pairs are the followings:

he come+s⇔gel+Hr
he go+s⇔git+Hr
book+s⇔kitap+lAr
pen+s⇔kalem+lAr

Then we will find the confidence factor of $rule_k$ as:
$N1 = 2$, from pairs 1 and 2
$N2 = 2$, from pairs 3 and 4

$$confidencefactor_{rule_k} = \frac{N1}{N1+N2} = \frac{2}{2+2} = 0.5$$

If $rule_k$ is **+s**$\leftarrow$ **+Hr**, (i.e. it is a fact and it will be used in right to left translation)
Then we will find confidence factor of $rule_k$ as:

$N1 = 2$ from pairs 1 and 2

$N2 = 0$ no such pair

$$confidencefactor_{rule_k} = \frac{N1}{N1+N2} = \frac{2}{2+0} = 1.0$$

It is possible to have the same confidence factor for left to right and right to left usage of the same rule, but it is more probable to have different values. For example, in the following example we have the same confidence factors in both direction.

1) if $rule_k$ is **come**$\rightarrow$ **gel** (i.e. it is a fact and it will be used in left to right translation), and our translation examples are the same as the previous example.
Then we will find confidence factor of $rule_k$ as:

$N1 = 1$, from pair 1

$N2 = 0$, no such pair

$$confidencefactor_{rule_k} = \frac{N1}{N1+N2} = \frac{1}{1+0} = 1.0$$

2) if $rule_k$ is **come**$\leftarrow$ **gel**, (i.e. it is a fact and it will be used in right to left translation)
Then we will find confidence factor of $rule_k$ as:

$N1 = 1$, from pair 1

$N2 = 0$, no such pair

$$confidencefactor_{rule_k} = \frac{N1}{N1+N2} = \frac{1}{1+0} = 1.0$$

3.2 Method for Assigning Confidence Factors to Rules

Assigning confidence factor to a rule, (templates that has variables in it) is a more complicated task if we try to find the confidence factor of that rule completely. Therefore, if $rule_k$ has variables which will be unified with other rules in the translation phase then we will assign a partial confidence factor to this rule by considering the parts which do not include variables according to the confidence factor formula used in the previous section. In the translation process, the variables are bound using some other rules or facts, and we find the whole confidence factor of this rule by multiplying the confidence factors of all rules which are used to bind the variables. The following is an example for this:

If $rule_k$ is X^{L_1}**+s**$\Leftrightarrow$ X^{L_2}**+Hr** if $X^{L_1} \Leftrightarrow X^{L_2}$
and our training pairs are the same with the previous example. Since X^{L_1}**+s** can be a substring of left sides of all pairs and $\mathbf{X}^{L_2}$**+Hr** can be a substring of right sides of pairs 1 and 2 by assuming that the variables can match one or more tokens of the string, (i.e. variables can not match empty string), we will get the following confidence factor for left to right usage:

$N1 = 2$, from pairs 1 and 2

$N2 = 2$, from pairs 3 and 4

$partialconfidencefactor_{rule_k} = \frac{N1}{N1+N2} = \frac{2}{2+2} = 0.5$

Since X^{L_2}**+Hr** can be a substring of right sides of pairs 1 and 2 and $\mathbf{X}^{L_1}$**+s** can be a substring of left sides of all pairs by assuming that the variables can match one or more tokens of a string, we will find the following confidence factor for right to left usage:

$N1 = 2$, from pairs 1 and 2

$N2 = 0$, no such pair

$partialconfidencefactor_{rule_k} = \frac{N1}{N1+N2} = \frac{2}{2+0} = 1.0$

In the translation phase, these partial confidence factors are multiplied by the confidence factors of the rules replacing variables to calculate the real confidence factor of that translation output. Assume that we are given the input sentence *ali comes*. Its represented as *ali come+s* and matches the above template. Then, we seek for the corresponding translation of *ali*. It is found that there is a rule that says that *ali* corresponds to *ali* and its confidence factor is 1.0. Then, the confidence factor of the output *ali gel+Hr* is calculated as $0.5 * 1.0 = 0.5$

3.3 Method for Assigning Confidence Factors to Rule Combinations

The most complicated task of the procedure is the assignment of confidence factors to rule combinations. The reason for considering these rule combinations is the following: although some rules or facts are assigned high confidence factors when they are considered as single rules or facts, they may have a very low confidence factor when they are used with other rules or facts. The algorithm of this assignment process is given in Table 3.1. The algorithm in Table 3.1 is used only for left to right translation. This algorithm is repeated for right to left translation by replacing X^{L_1} with X^{L_2}.

At this point, calculation of the minimum distance between a translation result, T_i, and a part of training pair, $X_j \in \mathbf{Xs}^{L_2}$, needs more explanation. First of all, X_j and T_i are assumed to be points whose coordinates are (Length of X_j, 0) and (Length of Similarities between X_j and T_i, Length of Differences between X_j and T_i) in a two-dimensional space, respectively. Then the distance is calculated by using the Euclidean formula for calculating the distance between two points:

$$distance = \sqrt{(LengthofX_j - LengthofSimilarities)^2 + (LengthofDifferences)^2}$$

Assume that we have X^{L_1}=`you come+p` and we obtained

$\mathbf{Xs}^{L_2}$= {`gel+DH+n, siz gel+DH+nHz`}

Ts={`gel+Hr+DH+n, gel+DH+nHz`}

For each training pair $X^{L_1} \Leftrightarrow X^{L_2}$
  Find all corresponding $\mathbf{Xs}^{L_2}$ for X^{L_1} from training pairs
  Find all translations (**Ts**) with their proofs (**Ps**) of X^{L_1} from translation templates where proofs show the rules used in the translation
  For each $T_i \in \mathbf{Ts}$ do the following steps
    If $T_i \in \mathbf{Ts}$ is the same as $X_j \in \mathbf{Xs}^{L_2}$
      Assign confidence factor of the rule combination $P_i \in \mathbf{Ps}$ as 1
    Else
      Find distances between T_i and each $X_j \in \mathbf{Xs}^{L_2}$
      Choose the minimum distance **d** among these distances
      Assign confidence factor of this rule combination $\mathrm{P}_i \in \mathbf{Ps}$ as

$$confidencefactor_{P_i} = \frac{1}{1+d}$$

Table 3.1: Algorithm for Assigning Confidence Factor to Rule Combinations

then confidence factors for rule combinations used to find translations in **Ts** are computed as follows:

1) For T_1=`gel+Hr+DH+n`:

If T_1 is found by using n rules $i_1, \cdots, i_n$

Similarities between T_1 and X_1 are gel, +DH, +n and the length of similarities is 3.

Differences between T_1 and X_1 are, +Hr and the length of differences is 1, since length of +Hr is 1.

d1$=\sqrt{(3-3)^2+(1)^2} = 1$

Similarities between T_1 and X_2 are gel,+DH and the length of similarities is 2.

Differences between T_1 and X_2 are (siz,),(,+Hr),(+nHz,+n) and the length of differences is 3, since length of siz is 1, length of +Hr is 1 and length of +nHz or +n is 1, giving a total of 3.

d2$=\sqrt{(4-2)^2+(3)^2} = \sqrt{13}$

min(d1,d2)=d1=1 and confidence factor$_{i_1,\cdots,i_n}$=0.5

2) For T_2=`gel+DH+nHz`:

If T_2 is found by using m rules $j_1, \cdots, j_m$

Similarities between T_2 and X_1 are gel,+DH and the length of similarities is 2.

Differences between T_2 and X_1 are (+n,+nHz) and the length of differences is 1, since length of +n or +nHz is 1.

d1$=\sqrt{(3-2)^2+(1)^2} = \sqrt{2}$

Similarities between T_2 and X_2 are gel,+DH,+nHz and length of similarities is 3.

Differences between T_2 and X_2 are (siz,) and the length of differences is 1, since length of siz is 1.

d2$=\sqrt{(4-3)^2+(1)^2} = \sqrt{2}$

min(d1,d2)=d1 or **d2** and confidence factor$_{j_1,\cdots,j_m} = \frac{1}{1+\sqrt{2}}$

Note that, the length of differences is calculated by choosing the maximum of lengths in difference pairs.

These rule combinations are represented as tree structures. For example if $rule_i$ has two variables that are bound to $rule_j$ and $rule_k$, then the root of the tree is assumed to be $rule_i$ and its children are $rule_j$ and $rule_k$. If $rule_j$ or $rule_k$ has variables then they become the root of that subtree and their children become the numbers of the rules that are used in the binding of their variables. This tree structure is formed recursively. The tree structure will be helpful during the translation process and its usage will be explained in the next section.

3.4 Translation Process by Using Confidence Factors

Translation process can be summarized by the four steps given in Table 3.2. We find all possible translations by using the templates obtained in our learning phase. Then these results are evaluated according to their weights. These weights come either directly from the weights of the rules or rule combinations. After the evaluation of the results, the ones that have the highest weights are given as the output, and the ones with lowest weights are eliminated. Therefore the correct output is ensured to be among these selected outputs, and hopefully will be on the top of the selected outputs.

The second step of the algorithm is the most important part of the translation process. As it seems, finding the confidence factors of these results is not as simple. We need both the confidence factors of the rules and rule combinations which are calculated in the learning process. The details of these calculations are given in Table 3.3. The rules that are pertaining to the result are found and a tree structure is obtained from these rules as explained in Section 2.3. Then this tree structure is used for comparison. If the result does not match a rule combination that is assigned a weight in the learning phase, then the comparison continues among the subtrees.

Assume that, we want to translate the sentence *amcaları geldi*. This sentence can be analyzed in three ways:

amca+lAr+sH gel+DH

amca+lAr+yH gel+DH

amca+lArH gel+DH

Then, translations for these three analysis are sought. Assume that we found the following results:

their uncle come+p by using rules 1 and 2 denoted as [1,[2]]

his uncle+s come+p by using rules 3 and 4 denoted as [3,[4,6]]

their uncle+s come+p by using rule 5 [5]

And we have the following confidence factors denoted as rw(RL,W), where RL is the rule list used to obtain that translation, and W is the confidence factor of it.

rw([1],1.0)

- Find all the analysis of the words
- Find all combinations of the analysis
- Find the lexical form of the input sentence
- Find all possible translations and their proofs
- Find confidence factors of these results by using the confidence factors assigned in the confidence factor assignment process.
- Sort results according to the calculated confidence factors in descending order by using a sort algorithm
- Generate surface forms of the outputs

Table 3.2: Translation Algorithm

Find the translation output's confidence factor by using the previously calculated confidence factors of rule combinations

Find the set of rule combinations (**R**) which are assigned confidence factors

If $rp = R_i \in \mathbf{R}$ then cf_{result} =cf_{R_i} where rp is the resulting proof

else cf_{result} =$\mathrm{cf}_{rp_{root}} * \mathrm{cf}_{rp_{child_1}} * \mathrm{cf}_{rp_{child_2}} * \cdots * \mathrm{cf}_{rp_{child_n}}$

if child_k is a fact, fact_m, then cf_{child_m} =confidence factor_{fact_m}

else calculate recursively cf_{child_k} as a tree

Table 3.3: Algorithm for Calculating Confidence Factors of the Translations

rw([2],1.0)

rw([1,[2]],0.33)

rw([3],1.0)

rw([4],1.0)

rw([5],0.33)

rw([6],0.8)

Then, the confidence factors for the outputs will be calculated as:

their uncle come+p=0.33, since we search for the rule combinations first.

his uncle+s come+p=0.8. First, we seek for a rule combination of 3,4, and 6, but since there is no confidence factor assigned to this combination, we seek for the confidence factors of 4 and 6. It is found that their confidence factors are 0.8 and 1.0. Then we multiply these values (confidence factors of children) by 1.0, confidence factor of rule 3 (root of the tree).

their uncle+s come+p=0.33 which is the confidence factor of rule 5.

The results ordered according to the confidence factors are:

his uncle+s come+p

their uncle come+p

their uncle+s come+p

Then the surface forms are generated as:

his uncles came

their uncle came

their uncles came

Chapter 4

System Architecture

Our system consists of three fundamental parts:

- Interface components
- Learning components
- Translation components

Interface components are necessary for analysis of the translation examples prior to the learning phase. Generation of the surface forms of the outputs are also obtained by the help of interface components. Learning component produces templates and their confidence factors by using translation examples. Translation components finds possible target sentences for the given source sentences by using the templates and their confidence factors. General architecture of the system is given in Figure 4.1. The components will be explained in details in the following sections.

4.1 Interface Components

The interface components are used for analysis and generation purposes. In the analysis phase the surface form of the given English and Turkish sentences are converted to their lexical level representations. Then these lexical level representations are used to obtain the training data. First, the training sentences are written to a file whose lines are in the form of E=T, where E denotes the surface form English sentence, and T denotes the corresponding surface form of Turkish sentence (i.e. translation of the English sentence). It is assumed that both sentences are grammatically correct, and they are the exact translation of each other. Then the interface component uses this file as the input and produces another file whose lines are in the form of trainpair([**E**],[**T**]).Here **E** and **T** are the lexical level representations of E and T, respectively. For example if we have a line in the first file such as:

I went to school=okula gittim

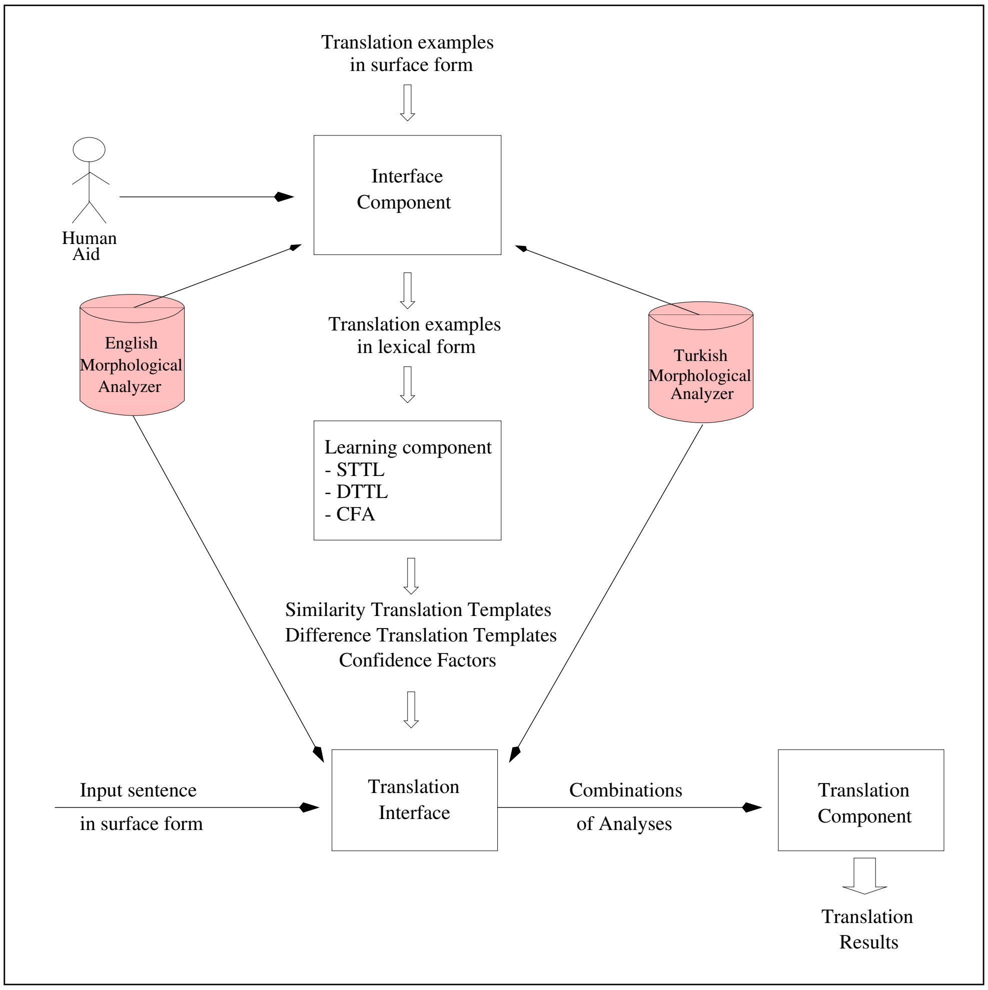

Figure 4.1: General Architecture

then the second file has a line such as:

trainpair([I,go,+p,to,school],[okul,+yA,git,+DH,+m]).

The interface uses two morphological analyzers, one for English and one for Turkish. For the time being, we could not find a reliable, high-coverage and commercially available morphological analyzer for English, so one is written by the author. For this purpose, I created a database by using a text file [1] which consists of words and their analysis. The database is indexed by a hash function which is provided by *SICSTUS PROLOG*. Indexing is done in two ways: One uses the real word and the other uses the root of the word as the key. A typical database entry looks like:

$$\mathrm{m}(\mathrm{Word}_i,\mathrm{Root\ of\ Word}_i,+\mathrm{morpheme}_1,\ \ldots,\ +\mathrm{morpheme}_m)$$

[1] The file is a dictionary in text format. It is obtained from XTAG project:http://www.upenn.edu/ xtag

For example, the word *went* has a corresponding entry in the database as: *m(went,go,+p)*. The word *went* and *go* are the keys of this entry for analysis and generation, respectively. All the interface components for English morphological analysis is written in Sicstus Prolog.

Turkish morphological analyzer is written by using PCKIMMO [2], The results of the analysis are gathered by an interface written in C and passed to the Prolog side.

In most cases, there is more than one analysis for a given word both in English and Turkish. Therefore, all morphological analysis are obtained for the input and the correct analysis is chosen by the user. For example, if the input file contains the following lines,

their faces=yUzleri

their notebooks=defterleri

then, the interface will respond as:

```
0==>[n(face),+(pl)]
1==>[v(face),+(s)]
Select correct analysis for word ==faces==
in the sentence ==their faces=yUzleri==
0
0==>yUz+(lAr)+(sH)
1==>yUz+(lAr)+(yH)
2==>yUz+(lArH)
Select correct analysis for word ==yUzleri==
in the sentence ==their faces=yUzleri==
2
0==>defter+(lAr)+(sH)
1==>defter+(lAr)+(yH)
2==>defter+(lArH)
Select correct analysis for word ==defterleri==
in the sentence ==their notebooks=defterleri==
2
```

So the correct analysis will be selected by a human.

The general architecture of the interface is shown in Figure 4.2.

4.2 Learning Component

Learning component consist of STTL and DTTL and confidence factor assignment procedures. The learning component uses the output of the analyzers as its input. The examples are compared with all the other examples. A match sequence is obtained from each comparison by using the similarities and differences of the translation examples. If the match sequence fits to

[2]PCKIMMO:http://www.sil.org/pckimmo

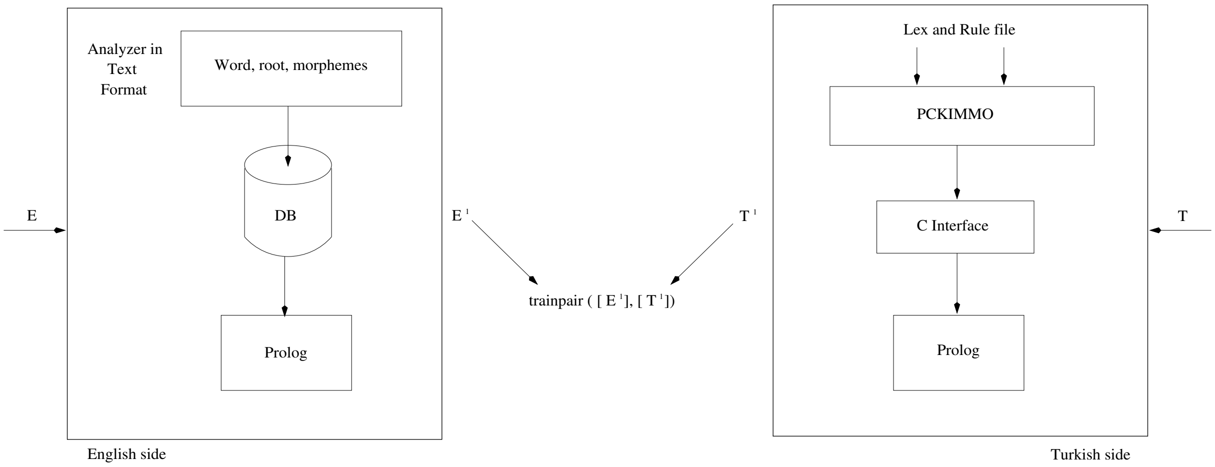

Figure 4.2: Interface Architecture

the requirements of the TTL, then STTL and DTTL algorithms produce translation templates by using the match sequences.

Confidence factors for each and some combinations of these translation templates are obtained in addition to the templates at the end of the learning process. In Figure 4.3 the inputs and outputs are shown.

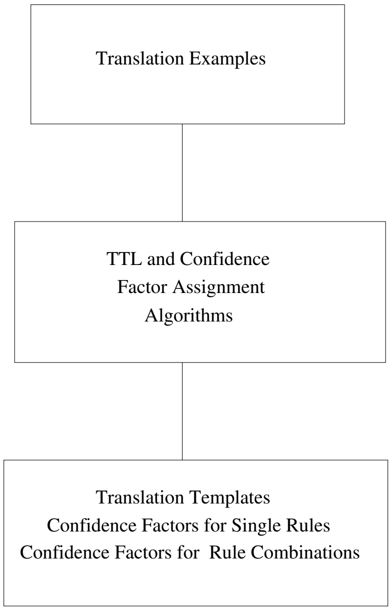

Figure 4.3: Learning Component

The translation component takes templates, confidence factors, analyzers of English and Turkish, and the input sentence. The input sentence is taken in surface form and converted to its lexical form by using the analyzers. Generally, there are more than one analysis of the words of the input sentence, since we do not know the correct analysis and no human assistance is involved at this stage all possible analyses are tried.

Translation templates are used to obtain all translations for the given input. Finally, the confidence factors are used to evaluate the translations. Then, all the translations both correct and incorrect are provided along their confidence factors and the rules they are obtained from. They are displayed in two ways: sorted according to the confidence factor and unsorted (i.e. in the order they are obtained) as shown in Figure 4.4.

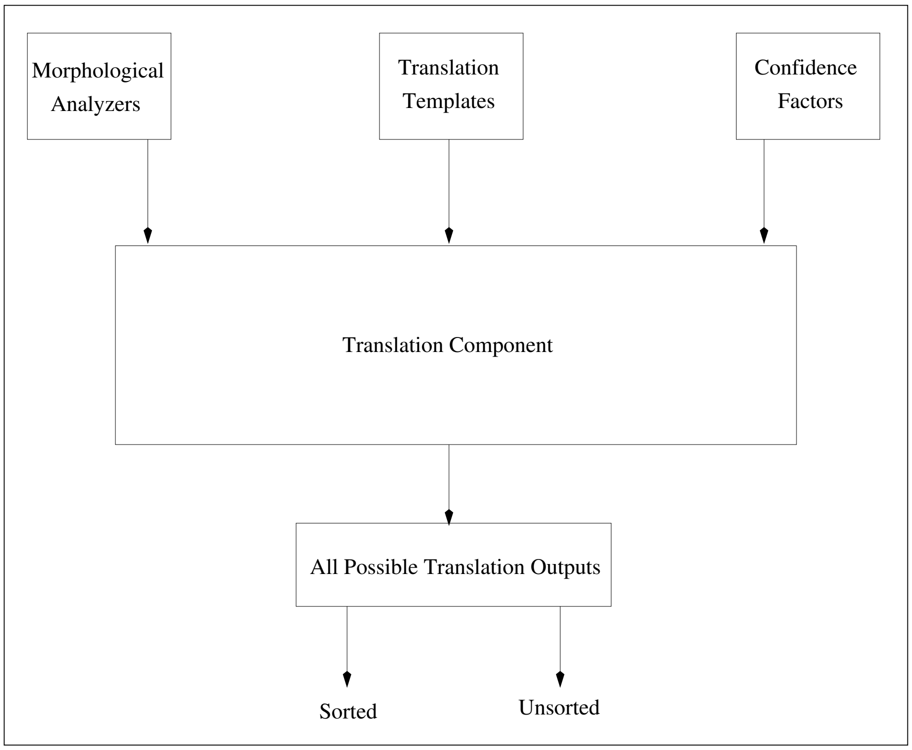

Figure 4.4: Translation Component

Chapter 5

Performance Results

5.1 Time Complexity Analysis

In this section, the time complexities of all the fundamental procedures, (i.e. learning, confidence factor assignment to rules and rule combinations, and translation will be derived. Assume that we have N translation examples, and the number of translation templates produced by TTL algorithm is denoted by NT.

In the learning phase, for the example translation E_i, where $1 \leq i \leq N$, the algorithm compares E_i with the remaining pairs E_j, where $i < j \leq N$. Then the total number of comparison is:

$$\sum_{i=1}^{N} n - i = \frac{(n-1) * n}{2} = O(N^2)$$

Each of the STTL and DTTL algorithms can infer three templates at most from an example translation pair. For E_i $6 * (N - i)$ templates can be obtained. Therefore for each pass of the learning phase:

$$\sum_{i=1}^{N} 6 * (N - i) = \frac{6 * (N-1) * N}{2} = O(N^2)$$

The examples can be passed over more than once to obtain new templates. Since the algorithm is sensitive to the order of examples, using these examples more than once will decrease the level of this sensitivity. The number of the passes can be a constant p where $p \ll N$. Thus, the number of the translation templates produced is $O(N^2)$.

The confidence factor assignment to single templates is $O(N^3)$, since we can have $O(N^2)$ templates at most, and we search N examples for each of these templates.

Assigning confidence factors to rule combinations is the most time consuming process of all the procedures. In this process, translations of all N pairs are found by searching all the templates. When an example is matched with a template, and if that template has variables in it, then all the templates are searched again to bound those variables. If the length of that sentence is l, then it can match templates $2^l - 1$ way in the worst case (a variable can match single words, double words, etc.). So the number of comparisons can be:

$$N * N^2 * (2^l - 1) = O(N^3 2^l) = O(N^3)$$

In the translation process, the dominat factor in translation time is finding all possible translations. So the time complexity of translation will be the following by the same reasoning above:

$$N^2 * (2^l - 1) = O(N^2 2^l) = O(N^2)$$

The cost of this comparison is high, therefore the number of translations can be limited to reduce this cost. The results on some small corpora by limiting the number of translations to be found is given in Table 5.1. Times are given in seconds and obtained by using an Ultra Sparc station. CFA in columns two and three denotes confedence factor assignment

N	Learning Time	CFA to Single Rules Time	CFA to Rule Combinations Time	Number of Templates
10	0.040	0.1	0.224	47
20	0.1	0.26	0.62	123
30	0.27	0.69	1.19	251
40	0.62	1.0	1.62	310
50	1.15	1.680	2.420	428
650	1347.030	539.510	3500.900	12950

Table 5.1: Analysis of Procedure

5.2 Evaluation Criteria

There are many different evaluation criteria of machine translation systems offered in the literature ([50], [80], [83], [57], [54],[71],[85]).

The evaluation criteria of [50] are:

1. Providing necessary functionality (i.e. matching domain, language pair, text type, etc.)
2. The financial stability of the vendor or research group to continue their work and support user
3. System appropriateness to the current and the future environment of the user (i.e. matching the computer environment and the work style of the user)
4. Upgrade and maintenance costs (i.e. modular implementation, readily extensible, strong theoretical foundation)
5. Increase throughput (i.e. increase user productivity, wide linguistic and text formatting coverage, and appropriate output quality)

This system provides necessary functionality, since if the system is given training data which is large enough for the translation domain, then it is possible to have translations for any text of that domain. The second and third items above are not so important for the time being. The system is based on the ideas of EBMT and Statistical MT which are two important and nearly well-formed paradigms of MT. The system can easily be customized for new domains, language pairs and text types only by paying for the training data. The system can be useful for manual type translations and it is suitable for users who need these kinds of translation.

Evaluation of MT systems are considered from different perspectives in [80], First of all the issue is examined according to the *completeness, correctness, and stylistics* criteria.

Completeness is defined as assigning some output string to every input string during the translation. According to this definition completeness of our system depends on the training examples.

Correctness is defined as assigning a correct output string to every input string during the translation. Although our system is given grammatically correct training examples, some of the translation templates produced are not hundred per cent correct or they may even be totally incorrect. Translation outputs produced by using these templates can result in incorrect translations for some of the inputs. However, in general the system produces at least one correct output for the given input that is to be translated. Since our goal is to find a correct answer, producing incorrect results can be tolerated as far as the correct outputs are given before the incorrect ones.

The *stylistics* criterion says that the chosen output must not only be correct but it must be the most appropriate output among the other candidates. This criteria is tried to be matched by using statistical information in our case.

The following section provides test results of the system and examine the system mostly from correctness criterion.

5.3 Test Results

In this section, the results of the simulations are summarized. The results on a small corpora are as follows: A training set of examples has contained 488 sentences. Total number of the translation templates that are learned in the learning phase is 4723. In the confidence factor assignment process 4723 templates for left to right usage (from English to Turkish), and 4723 templates for right to left usage (from Turkish to English) are assigned confidence factors. 55845 rule combinations for left to right usage and 53676 rule combinations for right to left usage are assigned confidence factors. Therefore, we obtained a total of 118967 confidence factor assignments.

The results on a relatively large corpora are as follows: A training set of examples has contained 650 sentences. Total number of the translation templates that are learned in the learning phase is 12950. In the confidence factor assignment process 12950 templates for left to

Type of data	Percentage of correct results in translations	Percentage of incorrect results in translations	Percentage of correct results in top 5 without weights	Percentage of correct results in top 5 with weights
Sentences selected from training data	42.0	58.0	44.0	80.0
New sentences not appearing in training data	33.0	67.0	40.0	60.0

Table 5.2: Performance Results on a Small Corpora

right usage (from English to Turkish), and 12950 templates for right to left usage (from Turkish to English) are assigned confidence factors. 162456 rule combinations for left to right usage and 153678 rule combinations for right to left usage are assigned confidence factors. Therefore, we obtained a total of 342034 confidence factor assignments.

Type of data	Percentage of correct results in translations	Percentage of incorrect results in translations	Percentage of correct results in top 5 without weights	Percentage of correct results in top 5 with weights
Sentences selected from training data	35.0	65.0	32.0	69.7
New sentences not appearing in training data	29.5	70.5	29.8	52.5

Table 5.3: Performance Results on a Large Corpora

In the translation process, we used two groups of sentences to evaluate the performance of the results. The first group of sentences are randomly selected from training data and the second group of sentences are the new sentences which do not occur in the training data. The results are obtained by using the previously assigned weights and they are sorted in ascending order according to these weights. We also produced the outputs without using the weights of the templates for comparison purposes. Then they are sent to the generator to obtain surface forms from the lexical forms. In Table 5.2 and Table 5.3 the results for the small and large corpora with weights and without weights are summarized, respectively. The columns denote the percentage of the correct translations among all the results, percentage of the incorrect translations, and percentage of the correct translations seen in the top five results, respectively.

5.4 Causes For Incorrect Translations and Failures

The performance results indicate that the system produces some incorrect results. Some of these incorrect results are eliminated (i.e. they are shifted towards the bottom of the translation output list) by using confidence factors, however this is not always the case. In other words, it is not always possible to get rid of the incorrect translation output. Then, it is necessary to find the sources of these incorrect translations and to cope with them. In addition to the incorrect translations, the system fails to find a translation for some inputs. The following is a brief list of the possible causes for incorrect translations and failures:

Analyzers: The training examples are stored as strings of root words, and morphemes. This representation is useful to have more abstraction and to learn more templates by eliminating the effect of issues like vowel harmony, etc. The idea is to have a unique representation for each of the morphemes. For example, *came* and *went* do not exhibit any similarity if they are denoted in this way, however if they are denoted as *come,+PAST* and *go,+PAST* then the past tense morpheme is perceived as a similarity. This the representation required for the system. Another aspect is to have large coverage analyzers. However, we could not find morphological analyzers, both for English and Turkish, which exactly do these tasks. The words that can be analyzed by these analyzers are not large enough to serve for a real MT system, but they can be used for experimental purposes.

Some of the representations for the same types of morphemes are handled in different manners. For example *gelirim (I come)* and *giderim (I go)* must be handled in the same way. They must be represented as *gel,+PRES+Hm* and *git,+PRES+Hm*, however, they are represented as *gel,+Hr+Hm* and *git,+Ar+Hm*. Therefore, these kinds of distinctions may lead to production of incorrect templates by effecting the number of similarities, or no template can be learned.

Generally, words have more than one morphological analysis both in English and Turkish. In the learning phase, input sentences are assumed to be analyzed correctly and the disambiguated by a human assistant, However, in the translation process there is no human support. Therefore, the procedure has to produce all analysis of all the words in a given input sentence and every combinations of these analysis have to be translated. Trying all these possible combinations also yield some incorrect translations.

Order of the translation example pairs: Learning algorithm is sensitive to the order of the translation examples, therefore, sometimes it is necessary to pass over these examples more than once. The number of passes is proportional to the number of the example pairs and the similarities they exhibited.

Incorrect confidence factor assignment: Confidence factor assignment is a nice solution for choosing the best result among all possible candidate. However, finding the best confidence factor assignment procedure is not an easy task. The method offered here may not always assign the best values, although it is a very simple approach and it increases the accuracy in a significant way. There are infrequent cases where the assigned values can effect system performance in a negative way. For example, if one of the rules learned is $i \leftrightarrow +n$ which is absolutely incorrect, can obtain a high confidence factor if we have translation examples such as *i come+p to school and you go+p to cinema*$\leftrightarrow$ *okul+yA gel+DH+m ve sinema+yA git+DH+n* where i and $+n$ are seen together but they are not related with each other, then the confidence factor assignment algorithm give a wrong value since it is not aware of this fact. In other words, the method can perform negatively for the overlapping templates.

Chapter 6

CONCLUSION AND FUTURE WORK

In this thesis, I have presented a statistical model for assigning confidence factors to the translation templates learned by the translation model offered in Çicekli [40, 19]. This translation model learns general translation patterns from the given translation examples by using analogy principle.

In the early versions of the algorithm, translation templates are sorted according to their specificities (i.e., the number of terminals in templates). Although this way of sorting gives correct results, the accuracy was not high enough. The major contribution of this thesis is assigning confidence factors to templates in order to improve the accuracy. Assigning confidence factor to these rules depends on the statistical data collected from translation examples which are assumed to be grammatically correct. As mentioned before, in the translation process, the output translations which have the highest weights are selected among all possibilities. Thus, it is ensured that the correct answer will be among these selected output and at the top of the list.

The algorithm is tested on Turkish and English for illustration purposes, but it is applicable to any pair of languages. On a small set of data, learning and translation times are reasonable enough. The accuracy of the results are promising. We need to test it on larger corpora. Although, there are large amount of electronic data available for many language pairs, we do not have any bilingual corpus for English and Turkish. Thus, we are trying to form a large corpus for this purpose.

The training examples are being formed manually by the following assumptions:

- The input bilingual text is aligned correctly prior to the learning,
- The text is morphologically analyzed and disambiguated correctly,
- Necessary clean up is done.

- It is large enough to reflect all possible structures that can be encountered in that domain.
- Grammatical correctness is ensured.

The learning process on a large corpus takes a considerable amount of time, but it can be tolerated since it will be done only once and it will increase the translation accuracy. The percentage of the correct outputs during translation seems to be low, however, they are obtained among the top results. Therefore, large number of incorrect results can be tolerated, as far as the correct results are provided before them.

In the future, the system accuracy can be increased by using a human assistance for the verification of the templates, morphological analysis etc. However, in order to fully automate the system, it will be better to use some additional reliable tools for parallel text alignment, disambiguation, etc.

The most time consuming part is assigning confidence factors to rule combinations. Finding all possible translations is an expensive task. The number of translations that is to be found can be bounded by a certain value for efficiency purposes. In addition to this, parallel processing of the data can also be involved to increase performance. Although, learning process is not suitable for parallel processing, the tasks that assign confidence factors to rules and rule combinations are good candidates of parallel processing, since they are independent from other tasks. In the translation process, finding all the analysis of each word and seeking for the translation of all the combinations of these analysis decreases output accuracy and correctness, and increases response time. So, disambiguation becomes an important task to increase the efficiency. Additionally, templates that are assigned very low confidence factors can be directly eliminated in the learning phase, so that the number of incorrect results that can be produced from these templates will be decreased during translation. This will also decrease the response time, since the search space will shrink.

Using some information about the words (i.e. whether they are noun, verb, adjective, etc.) can be useful for restricting the applicability of the templates.

Furthermore, the current system fails to find a solution for some inputs in which there is an unseen word or phrase. It is possible to provide a few answers that are close to these kinds of inputs in future.

Bibliography

[1] K. Aijmer, B. Altenberg English Corpus Linguistics *Longman*, London, 1991

[2] L. Appelo A Compositional Approach to the Translation of Temporal Expessions in the Rosetta Systems *Proceedings of Eleventh International Conference on Computational Linguistics*, Bonn, Germany, pp: 313-318, 1986

[3] J. Apresian, I. Boguslavskij, L. Iomdin, A. Lazurskij, V. Sannikov, L. Tsinman ETAP-2: The linguistics of a Machine Translation System *META*, Volume:37, No:4, pp:97112, 1992

[4] D.J. Arnold, K. Steven, T. Louis, S. Louisa Relaxed Compositionality in Machine Translation *Proceedings of the Second International Conference on Theoretical and Methodological Issues in Machine Translation of Natural Languages*, Cornegie Mellon University, Pittsburgh, PA

[5] D.J. Arnold, L. Balkan, R. L. Humphreys, S. Meijer, L. Sadler Machine Translation: An Introductory Guide *NCC Blackwell*, Manchester, Oxford, 1994

[6] D.J. Arnold, L. Tombe Basic Theory and Methodology in Eurotra *Machine Translation: Theoretical and Methodological Issues*, pp:114-135, Cambridge University Press, Cambridge, 1987

[7] D.J. Arnold, L. G. Sadler The Theoretical Basis of Mimo *Machine translation*, Volume:5, pp:195-222, 1990

[8] W.S. Bennett, J. Slocum METAL: The LRC Machine Translation System *Machine Translations Systems*, pp:111-140, CUP, Cambridge, 1988

[9] A.L. Berger, P.F. Brown, S.A. Della Pietra, V.J. Della Pietra, J.R. Gillett, J.D. Lafferty, R.L. Mercer, H. Printz, L. Ures The Candide System for Machine Translation *Proceedings of the ARPA Human Language Technology Workshop* Plainsbourgh, New Jersey, pp:157-162, 1994

[10] C. Boitet, R. Gerber Expert Systems and Other New Techniques in MT Systems *COLING-84*, Stanford University, 1984

[11] L. Bourbeau Linguistic Documentation of Computerized Translation Chain of the TAUM-AVIATION System *Practical Experience of Machine Translation*, Amsterdam, North Holland, pp:129-133, 1981

[12] P.F. Brown, J. Cocke, S.A. Della Pietra, V.J. Della Pietra, J. Jellinek Lafferty, R.L. Mercer, P.S. Rossin A Statistical Approach to French/English Translation *Proceedings, Second International Conference on Theoretical and Methodological Issues in Machine Translation of Natural Languages*, Carnegie Mellon University, Pittsburgh, Pennsylvania, 1988

[13] P.F. Brown, J.Cocke, S.A. Della Pietra, V.J. Della Pietra, J. Jellinek Lafferty, R.L. Mercer, P.S. Rossin A Statistical Approach to Language Translation *Proceedings of the 12th International Conference on Computational Linguistics*, D. Vargha (ed.) COLING Budapest, John von Neumann Society for Computing Sciences, pp:71-76, 1988

[14] P.F. Brown, J. Cocke, S.A. Della Pietra, V.J. Della Pietra, J. Jellinek Lafferty, R.L. Mercer, P.S. Rossin A Statistical Approach to Machine Translation *Computational Linguistics*, Vol:16, pp:79-85, 1990

[15] P.F. Brown Aligning Sentences in Parallel Corpora *Proc. of the 29th Annual Meeting of the ACL*, pp:169-176, 1991

[16] P.F. Brown The Mathematics of Statistical Machine Translation: Parameter Estimation *Computational Linguistics*, pp:233-311, 1993

[17] R.D. Brown, R.E. Frederking Applying Statistical Language Modelling to Symbolic Machine Translation *Proceedings of the Sixth International Conference on the Theoretical and Methodological Issues in Machine Translation*, Vol:II, pp:354-372, Leuven, Belgium, 1995

[18] J.G. Carbonell, M. Tomita Knowledge-Based Machine Translation: The CMU Approach *Machine Translation: Theoretical and Methodological Issues*, Sergei Nirengurg (ed.), Cambridge University Press, Cambridge, England, pp:68-89, 1987

[19] I. Cicekli, H.A. Guvenir Learning Translation Rules From a Bilingual Corpus *Proceedings of the 2nd International Conference on New Methods in Language Processing (NeMLaP-2)*, Ankara, Turkey, September, pp:90-97, 1996

[20] B. Collins, P. Cunningham Abstraction and Adaptation in EBMT *Proceedings of CSNLP-95*, DCU, Ireland, 1995

[21] B. Collins, P. Cunningham Translating Software Documentation by Example *International ECAI Workshop: Multilinguality in the Software Industry*, Budapest, 1996

[22] B. Collins, P. Cunningham An Example-Based Approach to Machine Translation *Proceedings of the 2nd Conference of the Association for Machine Translation in the Americas*, Quebec, 1996

[23] B. Collins, P. Cunningham, T. Veale Adaptation Guided Retrieval in EBMT *I. Smith, B. Faltings (eds.) Lecture Notes in Artificial Intelligence 1168*, Springer

[24] B. Collins, P. Cunningham Adaptation Guided Retrieval: Approaching EBMT with Caution *Proceedings of the 7th International Conference on Theoretical and Methodological Issues in Machine Translation*, Santa Fe, 1997

[25] B. Collins, P. Cunningham A Methodology for Example-Based Machine Translation *Trinity College*, Dublin, 1995

[26] C. Copeland, J, Durand, S. Krauwer, B. Maegaard The Eurotra Formal Specifications *Studies in Machine Translations and Natural Language Processing*, Volume:2, Office for Official Publications of the Commission of the European Community, Luxembourg, 1991

[27] C. Copeland, J. Durand, S. Krauwer, B. Maegaard The Eurotra Linguistic Specifications *Studies in Machine Translations and Natural Language Processing*, Volume:1, Office for Official Publications of the Commission of the European Community, Luxembourg, 1991

[28] L. Cranias, H. Papageorgiou, S. Piperidis A matching Technique in Example-Based Machine Translation *Institute for Language and Speech Processing, Greece*, Paper presented to *Computtion and Language: http://xxx.yukawa.kyoto-u.ac.jp/archive/cmp-lg*

[29] L. Danlos, P. Samvelian Translation of the Predicative Element of a Sentence: Category Switching, Aspect and Diathesis *Proceedings of the Fourth International Conference on Theoretical and Methodological Issues in Machine Translation of Natural Languages*, Montreal, Canada, pp:21-34, 1992

[30] S. Doi, M. Kazunori Translation Ambiguity Resolution Based on Text Corpora of Source and Target Languages *Proceedings of Fourteenth International Conference on Computational Linguistics*, Nantes, France, pp:525-531, 1992

[31] D. Estival, A. Ballim, G. Russell, S. Warwick A Syntax and Semantics for Feature-Structure Transfer *Proceedings of the 3rd International Conference on Theoretical and methodological Issues in Machine Translation of Natural Language*, pp:131-143, Linguistics Research Centre, University of Texas at Austin, USA, 1990

[32] Y. Fujii, K. Suzuki, F. Maruyama, T. Dasai Analysis of Long Sentence in Japanese-English Machine Translation System *Proceedings of Information Processing Society of Japan*, 1990

[33] O. Furuse, H. Iida An Example-Based Method for Transfer-Driven Machine Translation *Proceedings of the Fourth International Conference on Theoretical and Methodological Issues in Machine Translation of Natural Languages*, Montreal, Canada, pp:139-150, 1992

[34] W.A. Gale, K.W. Church A Program for Aligning Sentences in Bilingual Corpora *Proc. of the 29th Annual Meeting of the ACL*, pp:177-184, 1991

[35] R. Garside, G. Leech, G. Sampson The Computational Analysis of English: A Corpus-Based Approach *Longman*, London, 1987

[36] K. Goodman Special Issues on Knowledge-Based MT *Machine Translation*, Volume:4, No:1 and 2, Parts:I and II, 1989

[37] K. Goodman, S. Nirenburg The KBMT Project: A Case Study in Knowledge-Based Machine Translation *Morgan Kaufmann*, San Mateo, California, 1991

[38] R. Grishman, M. Kosaka Combining Rationalist and Ampiricist Approaches to Machine Translation *Proceedings of the Fourth International Conference on Theoretical and Methodological Issues in Machine Translation of Natural Languages*, Montreal, Canada, pp:263-274, 1992

[39] H.A. Guvenir, A. Tunc Corpus-Based Learning of Generalized Parse Tree Rules for Translation Gord McCalla (Ed) *New Directions in Artificial Intelligence: Proceedings of the 11th Biennial Conference of the Canadian Society for Computational Studies of Intelligence* Springer-Verlag, LNCS 1081, Toronto, Ontario, Canada, pp:121-131, May 1996

[40] H.A. Guvenir, I. Cicekli Learning Translation Templates from Examples *Proceedings of the 6th Annual Workshop on Information Technologies and Systems (WITS'96)*, Cleveland, Ohio, USA, pp:112-123, December 1996

[41] W. Hahn, G. Angelova, Providing Factual Information in MAT *Proceedings of MT - 10 Years On*, Cranfield, pp:11-1 to 11.16, 1994

[42] W. Hahn The Project Verbmobil. Mobile Speech-to-Speech translation *In Proceedings IJCAI*, Chambery, 1993

[43] E. H. Hovy Deepening Wisdom or Compromised Principles? The Hybridization of Statistical and Symbolic MT *IEEE Expert*, pp:16-18, April 1996

[44] W. J. Hutchins Research Methods and System Designs in Machine Translation: A Ten-Year Review 1984-1994 *Proceedings of Machine Translation Ten Years On*, Cranfield, U.K., 1994

[45] W.J. Hutchins Machine Translation: Past, Present, Future *Ellis Horwood Series in Computers and Their Applications*, Chichester, England, 1986

[46] W.J. Hutchins, H.L. Somers An introduction to Machine Translation *Academic Press*, London, 1992

[47] W.J. Hutchins Future Perspectives in Translation Technologies *Vaconcellos (ed.) Technology as Translation Strategy*, American Translators Association, Scholarly Monograph Series, Vol:II, Binghamton, New York, State University of New York, pp:223-240, 1988

[48] P. Isabelle Machine Translation at the TAUM Group *in Tutorial on Machine Translation*, Lugano, April, 1984

[49] D. Jones Analogical Natural Language Processing *UCL Press*, London, 1996

[50] P. W. Jordan, B. J. Dorr, J. W. Benoit A First-Pass Approach for Evaluating Machine Translation Systems *Journal of Machine Translation*, Vol:8, No:1-2, pp:49-58, 1993

[51] H. Kaji, Y. Kida, Y. Morimoto Learning Translation Templates from Bilingual Text *Proc. of Coling*, pp:672-678, 1992

[52] R. M. Kaplan, K. Netter (eds.) Translation by Structural Correspondences *Proceedings of Fourth Conference of the European Chapter of the Association for Computational Linguistics*, Manchester, pp:272-281,1989

[53] M. Kay : The Proper Place of Men and Machine in Langauge Translation *Report Xerox CSL-80-11*, Palo Alto, October, 1980.

[54] M. King Panel on Evaluation: MT Summit IV. Introduction *Proceedings of MT Summit IV*, Kobe, Japan, 1993

[55] M. King EUROTRA: A European System for Machine Translation *Lebende Sprachen*, Vol:26, pp:12-14, 1982

[56] M. King, S. Perschke EUROTRA and Its Objectives *Multilingua*, Vol:1, No:1, pp:27-32, 1982

[57] M. King, K. Falkedal Using Test Suites in Evaluation of Machine Translation Systems *Proceedings of the Fourteenth International Conference on Computational Linguistics*, Vol:2, pp:211-216, 1990

[58] H. Kitano A Comprehensive and Practical Model of Memory-Based Machine Translation. In Ruzena Bajcsy (Ed.) *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, Morgan Kaufmann Volume:2, pp:1276-1282, 1993

[59] J. Landsbergen, J. Odijk, A. Schenk The Power of Compositional Translation *Literary and Linguistic Computing*, 4:3, 1989

[60] J. Landsbergen Isomorphic Grammars and Their Use in the Rosetta Translation System *Machine Translation Today: The State of the Art, Proceedings of the Third Lugano Tutorial*, Edinburgh University Press, Edinburgh, pp:351-372, 1987

[61] J. Landsbergen Montague Grammar and Machine Translation *Linguistic Theory and Applications*, Academic Press, London, pp:113-147, 1987

[62] H.D. Luckhardt SUSY: Capabilities and Range of Application *Multilingua*, Vol:1, No:4, pp:213-219, 1982

[63] H.D. Maas The MT System SUSY *Machine Translation Today: The State of the Art, Proceedings of the Third Lugano Tutorial*, Edinburgh University Press, Edinburgh, pp:209-246, 1987

[64] M.C. McCord Design of LMT: A Prolog-Based Machine Translation System *Computational Linguistics*, Volume:15, No:1, pp:33-52, 1989

[65] C.I. McLean Example-Based Machine Translation Using Connectionist Matching *Proceedings of the Fourth International Conference on Theoretical and Methodological Issues in Machine Translation of Natural Languages*, Montreal, Canada, pp:35-43, 1992

[66] D.L. Medin and M.M. Schaffer Context Theory of Classification learning *Psychological Review*, 85, pp:207-238, 1978

[67] T. Mitamura, E. Nyberg, J. Carbonell An Efficient Interlingua Translation System for Multilingual Document Production *Proceedings of Machine Translation Summit*, Washington, DC, pp:55-61, 1991

[68] M.A. Nagao Dependency Analyzer: A Knowledge-Based Approach to Structural Disambiguation *Proceedings of Thirteenth International Conference on Computational Linguistics*, Helsinki, Finland, pp:282-287, 1990

[69] M.A. Nagao Framework of a Mechanical Translation between Japanese and English by Analogy Principle *Artificial and Human Intelligence*, A. Elithorn and R. Banerji (eds.), NATO Publications, 1984.

[70] M.A. Nagao Machine Translation: How Far Can It Go? *Oxford University Press*, Oxford, Translated by Norman Cook, 1986

[71] M.A. Nagao Evaluation of the Quality of Machine-Translated Sentences and the Control of Language *Journal of Information Processing Society of Japan*, Vol:26, No:10, pp:1197-1202, 1985

[72] J. Newton Computers in Translation: A Practical Appraisal *Routledge*, New Fetter Lane, London, 1992

[73] S. Nirenburg, J. Carbonnell, M. Tomita, K. Goodman Machine Translation: A Knowledge-Based Approach *Morgan Kaufmann*, San Mateo, California, 1992

[74] S. Nirenburg Machine Translation: Theoretical and Methodological Issues *Cambridge University Press*, 1987

[75] S. Nirenburg Two Approaches to Matching in Example-Based Machine Translation *proceedings of TMI93*, Japan, 1993

[76] H. Nomiyama Lexical Selection Mechanism Using Target Language Knowledge and Its Learning Ability *NL86-8, IPSJ-WG*, 1991

[77] H. Nomiyama Machine Translation by Case Generalization *Proceedings of Fourteenth International Conference on Computational Linguistics*, Nantes, France, pp:714-720, 1992

[78] G. Noord Reversible Unification-Based Machine Translation *Proceedings of Thirteenth International Conference on Computational Linguistics*, Helsinki, Finland, pp:299-304, 1990

[79] G. Noord, J. Dorrepaal, P. va der Eijk, m. Florenza, L. des Tombe The MiMo2 Research System *Third International Conference on Theoretical and Methodological Issues in Machine Translation*, Linguistics Research Centre, Austin, Texas, pp:213-224, 1990

[80] E.H. Nyberg, T. Mitamura, J.G. Carbonell Evaluation Metrics for Knowledge-Based Machine Translation *Proceedings of the Sixteenth International Conference on Computational Linguistics*, 1994

[81] H. Maruyama, H. Watanabe Tree Cover Search Algorithm for Example-Based Machine Translation *Proceedings of the Fourth International Conference on Theoretical and Methodological Issues in Machine Translation of Natural Languages*, Montreal, Canada, pp:173-184, 1992

[82] A. Okumura, M. Kazunori, Y. Kiyoshi A Pattern-Learning Based, Hybrid Model for the Syntactic Analysis of Structural Relationships among Japanese Clauses *Proceedings of the Fourth International Conference on Theoretical and Methodological Issues in Machine Translation of Natural Languages*, Montreal, Canada, pp:45-54, 1992

[83] M. Palmer, T. Finin Workshop on the Evaluation of Natural Language Processing Systems *Computational Linguistics*, Vol:16, No:3, 1990

[84] I.M. Pigott Systran: A Key to Overcoming Language Barriers in Europe *Multilingua*, Vol:2, No:3, pp:149-156, 1983

[85] A. Rinsche Towards a MT Evaluation Methodology *Proceedings of the Fifth International Conference on Theoretical and Methodological Issues in Machine Translation*, Kyoto, Japan, pp:14-16, 1993

[86] V. Sadler Working with Analogical Semantics: Disambiguation Techniques in DLT *Foris Publications*, Dodrecht, Netherlands, 1989

[87] V. Sadler, R. Vendelmans Pilot Implementation of a Bilingual Knowledge Bank *Proc. of Coling*, pp:449-451, 1990

[88] S. Sato, M.A. Nagao Toward Memory-Based Machine Translation *In Proceedings of COLING-90*, pp:247-252, 1990

[89] S. Sato MBT2: A Method for Combining Fragments of Examples in Example-Based Translation *Artificial Intelligence*, Vol:75, pp:31-49, 1995

[90] S. Sato CTM: An Example-Based Translation Aid System *Proceedings of COLING*, pp:1259-1263, 1992

[91] K. Schubert Esperanto as an Intermediate Language for Machine Translation *Computers in Translation: A Practical Appraisal*, Routledge, London, pp:78-95, 1992

[92] R. Sharp CAT-2: Implementing a Formalism for Multi-lingual MT *Proceedings of the 2nd International Conference on Theoretical and Methodological Issues in Machine Translation of Natural Languages*, Carnegie Mellon university, Center for Machine Translation, Pittsburgh, USA, pp:76-87, 1988

[93] J. Slocum, W.S. Bennett, J. Bear, M. Morgan, R. Root Metal: The LRC Machine Translation System *Machine Translation Today*, Edinburgh University Press, Edinburgh, pp:319-350, 1987

[94] H. Somers Interactive Multilingual Text Generation for a Monolingual User *Proceedings of the Fourth International Conference on Theoretical and Methodological Issues in Machine Translation of Natural Languages*, Montreal, Canada, pp:151-161, 1992

[95] H.L. Somers, I. McLean, D. Jones Experiments in Multilingual Example-Based Generation *In the Proceedings of the 3rd International Conference on Cognitive Science of Natural Language Processing*, Dublin, Ireland, 1994

[96] H.L. Somers Current Research in Machine Translation *Computers in Translation: A Practical Appraisal*, John Newton eds, Routledge, London, 1992

[97] K-Y. Su, J-S. Chang Why Corpus-Based Statistics-Oriented Machine Translation *Proceedings of the Fourth International Conference on Theoretical and Methodological Issues in Machine Translation of Natural Languages*, Montreal, Canada, pp:249-262, 1992

[98] E. Sumita, H. Iida, H. Kohyama Translating with Examples: A New Approach to Machine Translation *Proceedings Third International Conference on Theoretical and Methodological Issues in Machine Translation of Natural Language*, 1990

[99] E. Sumita, Y. Tsutsumi A Translation Aid System Using Flexible Text Retrieval Based on Syntax Matching *TRL Res. Report TR-87-1019*, Tokyo Research Laboratory, IBM, Tokyo, Japan, 1988

[100] G. Thurmair Complex Lexical Transfer in METAL *Proceedings of the Third International Conference on Theoretical and Methodological Issues in Machine Translation of Natural Languages, Linguistic Research Center*, The University of Texas, TX, pp:91-107, 1990

[101] A. B. Tucker Current Strategies in Machine Translation Research and Development *Machine Translation: Theoretical and Methodological Issues* Cambridge University Press, pp:22-42, 1987

[102] B. Vauquois, C. Boitet Automated Translation at Grenoble University *Computational Linguistics*, Vol:11, No:1, pp:28-36, 1985

[103] B. Vauquois, C. Boitet Automated Translation at GETA *Grenoble: GETA*, 1984

[104] T. Veale, A. Way Gaijin: A Template-Driven Bootstrapping Approach to Example-Based Machine Translation *Proceedings of NeMNLP97: New Methods in Natural Language Processing*, Sofia, Bulgaria, 1997

[105] H. Watanabe A Model of a Bi-Directional Transfer Mechanism Using Rule Combinations *Machine Translation*, Vol:10, No:4, pp:269-291, 1995

[106] Y. Wilks Systran: It Obviously Works But How Much Can It be Improved? *Computers in Translation: A Practical Appraisal*, Routledge, London, pp:166-188, 1992

[107] *CMT:http://www.mt.cs.cmu.edu/cmt/CMT-home.html*

[108] *EUROTRA:http://www.ccl.kuleuven.ac.be/about/EUROTRA.html*

[109] *ISI:http://www.isi.edu/natural-language/nlp-at-isi.html*

[110] *PROTEUS:http://cs.nyu.edu/cs/faculty/grishman/proteus.html*

[111] *SYSTRAN:http://systranmt.com*

Appendix A

Example Run

In this section a simple run of the system will be given for illustration purposes. here, the translation examples are given in the form as they are kept by the system, so they differ slightly from the examples given before. We have some additional information coming from the English analyzer about the types of the words. However, this does not have any effect on the procedures. They are put for future use. Training examples contain negative sentences for some tenses. There are thirty translation examples which are kept as the following Prolog fact:

trainpair([English Sentence],[Turkish Sentence]).

Here, both sentences are in lexical level representation. Thirty example sentences are:

```
trainpair([pron(he),v(do),+(s),adv(not),v(come)],[gel,+(mA),+(z)]).
trainpair([propN(aySe),v(do),+(s),adv(not),v(come)],[aySe,gel,+(mA),+(z)]).
trainpair([propN(ali),v(do),+(s),adv(not),v(come)],[ali,gel,+(mA),+(z)]).
trainpair([pron(we),v(do),adv(not),v(come)],[gel,+(mA),+(yHz)]).
trainpair([pron2pl(you),v(do),adv(not),v(come)],[gel,+(mA),+(zsHnHz)]).
trainpair([pron(they),v(do),adv(not),v(come)],[gel,+(mA),+(zlAr)]).
trainpair([pron('I'),v(am),adv(not),v(come),+('PROG')],[gel,+(mA),+('Hyor'),+(yHm)]).
trainpair([pron(you),v(are),adv(not),v(come),+('PROG')],[gel,+(mA),+('Hyor'),+('ZHn')]).
trainpair([pron(he),v(is),adv(not),v(come),+('PROG')],[gel,+(mA),+('Hyor')]).
trainpair([propN(aySe),v(is),adv(not),v(come),+('PROG')],[aySe,gel,+(mA),+('Hyor')]).
trainpair([propN(ali),v(is),adv(not),v(come),+('PROG')],[ali,gel,+(mA),+('Hyor')]).
trainpair([pron(we),v(are),adv(not),v(come),+('PROG')],[gel,+(mA),+('Hyor'),+(yHz)]).
trainpair([pron2pl(you),v(are),adv(not),v(come),+('PROG')],[gel,+(mA),+('Hyor'),+('ZHnHz')]).
trainpair([pron(they),v(are),adv(not),v(come),+('PROG')],[gel,+(mA),+('Hyor'),+(lAr)]).
trainpair([pron('I'),v(do),+('PAST'),adv(not),v(come)],[gel,+(mA),+('DH'),+(m)]).
trainpair([pron(you),v(do),+('PAST'),adv(not),v(come)],[gel,+(mA),+('DH'),+(n)]).
trainpair([pron(he),v(do),+('PAST'),adv(not),v(come)],[gel,+(mA),+('DH')]).
trainpair([propN(aySe),v(do),+('PAST'),adv(not),v(come)],[aySe,gel,+(mA),+('DH')]).
```

```
trainpair([propN(ali),v(do),+('PAST'),adv(not),v(come)],[ali,gel,+(mA),+('DH')]).
trainpair([pron(we),v(do),+('PAST'),adv(not),v(come)],[gel,+(mA),+('DH'),+(k)]).
trainpair([pron(you),v(do),+('PAST'),adv(not),v(come)],[gel,+(mA),+('DH'),+(nHz)]).
trainpair([pron(they),v(do),+('PAST'),adv(not),v(come)],[gel,+(mA),+('DH'),+(lAr)]).
trainpair([pron('I'),v(will),adv(not),v(come)],[gel,+(mA),+(yAcAk),+(yHm)]).
trainpair([pron(you),v(will),adv(not),v(come)],[gel,+(mA),+(yAcAk),+('ZHn')]).
trainpair([pron(he),v(will),adv(not),v(come)],[gel,+(mA),+(yAcAk)]).
trainpair([propN(aySe),v(will),adv(not),v(come)],[aySe,gel,+(mA),+(yAcAk)]).
trainpair([propN(ali),v(will),adv(not),v(come)],[ali,gel,+(mA),+(yAcAk)]).
trainpair([pron(we),v(will),adv(not),v(come)],[gel,+(mA),+(yAcAk),+(yHz)]).
```

The TTL algorithm uses these examples yielding 251 similarity and difference translation templates. The templates that are shown as *facts* are simple rules and they do not have variables. The templates that are shown as *rules* have variables which are denoted as integers. Two integers at the end of each fact and rule denote the template number and confidence factor, respectively.

```
fact([pron(he),v(do),+(s),adv(not),v(come)],[gel,+(mA),+(z)],233,1.0)
fact([propN(aySe),v(do),+(s),adv(not),v(come)],[aySe,gel,+(mA),+(z)],234,1.0)
fact([propN(ali),v(do),+(s),adv(not),v(come)],[ali,gel,+(mA),+(z)],235,1.0)
fact([pron(I),v(am),adv(not),v(come),+(PROG)],[gel,+(mA),+(Hyor),+(yHm)],236,1.0)
fact([pron(you),v(are),adv(not),v(come),+(PROG)],[gel,+(mA),+(Hyor),+(ZHn)],237,1.0)
fact([pron(he),v(is),adv(not),v(come),+(PROG)],[gel,+(mA),+(Hyor)],238,1.0)
fact([propN(aySe),v(is),adv(not),v(come),+(PROG)],[aySe,gel,+(mA),+(Hyor)],239,1.0)
fact([propN(ali),v(is),adv(not),v(come),+(PROG)],[ali,gel,+(mA),+(Hyor)],240,1.0)
fact([pron(we),v(are),adv(not),v(come),+(PROG)],[gel,+(mA),+(Hyor),+(yHz)],241,1.0)
fact([pron2pl(you),v(are),adv(not),v(come),+(PROG)],[gel,+(mA),+(Hyor),+(ZHnHz)],242,1.0)
fact([pron(they),v(are),adv(not),v(come),+(PROG)],[gel,+(mA),+(Hyor),+(lAr)],243,1.0)
fact([pron(I),v(do),+(PAST),adv(not),v(come)],[gel,+(mA),+(DH),+(m)],244,1.0)
fact([pron(you),v(do),+(PAST),adv(not),v(come)],[gel,+(mA),+(DH),+(n)],245,0.5)
fact([pron(he),v(do),+(PAST),adv(not),v(come)],[gel,+(mA),+(DH)],246,1.0)
fact([propN(aySe),v(do),+(PAST),adv(not),v(come)],[aySe,gel,+(mA),+(DH)],247,1.0)
fact([propN(ali),v(do),+(PAST),adv(not),v(come)],[ali,gel,+(mA),+(DH)],248,1.0)
fact([pron(we),v(do),+(PAST),adv(not),v(come)],[gel,+(mA),+(DH),+(k)],249,1.0)
fact([pron(you),v(do),+(PAST),adv(not),v(come)],[gel,+(mA),+(DH),+(nHz)],250,0.5)
fact([pron(they),v(do),+(PAST),adv(not),v(come)],[gel,+(mA),+(DH),+(lAr)],251,1.0)
fact([pron(I),v(do),adv(not),v(come)],[gel,+(mA),+(Hm)],210,0.5)
fact([pron(you),v(do),adv(not),v(come)],[gel,+(mA),+(zsHn)],211,0.3333333333333333)
fact([pron(we),v(do),adv(not),v(come)],[gel,+(mA),+(yHz)],212,0.5)
fact([pron2pl(you),v(do),adv(not),v(come)],[gel,+(mA),+(zsHnHz)],213,1.0)
fact([pron(they),v(do),adv(not),v(come)],[gel,+(mA),+(zlAr)],214,0.5)
```

```
fact([pron(I),v(will),adv(not),v(come)],[gel,+(mA),+(yAcAk),+(yHm)],215,1.0)
fact([pron(you),v(will),adv(not),v(come)],[gel,+(mA),+(yAcAk),+(ZHn)],216,1.0)
fact([pron(he),v(will),adv(not),v(come)],[gel,+(mA),+(yAcAk)],217,1.0)
fact([propN(aySe),v(will),adv(not),v(come)],[aySe,gel,+(mA),+(yAcAk)],218,1.0)
fact([propN(ali),v(will),adv(not),v(come)],[ali,gel,+(mA),+(yAcAk)],219,1.0)
fact([pron(we),v(will),adv(not),v(come)],[gel,+(mA),+(yAcAk),+(yHz)],220,1.0)
rule([[pron(he),v(do)],1,[adv(not),v(come)]],[[gel,+(mA)],1],221,1.0)
rule([1,[v(do),+(s),adv(not),v(come)]],[1,[gel,+(mA),+(z)]],222,0.6666666666666666)
fact([v(do),+(s),adv(not),v(come)],[gel,+(mA),+(z)],223,1.0)
rule([[propN(aySe),v(do)],1,[adv(not),v(come)]],[[aySe,gel,+(mA)],1],224,1.0)
rule([[propN(ali),v(do)],1,[adv(not),v(come)]],[[ali,gel,+(mA)],1],225,1.0)
rule([1,[v(are),adv(not),v(come),+(PROG)]],[[gel,+(mA),+(Hyor)],1],226,1.0)
fact([v(are),adv(not),v(come),+(PROG)],[gel,+(mA),+(Hyor)],227,1.0)
rule([1,[v(is),adv(not),v(come),+(PROG)]],[1,[gel,+(mA),+(Hyor)]],228,0.6666666666666666)
fact([v(is),adv(not),v(come),+(PROG)],[gel,+(mA),+(Hyor)],229,1.0)
rule([1,[v(do),+(PAST),adv(not),v(come)]],[[gel,+(mA),+(DH)],1],230,0.625)
fact([v(do),+(PAST),adv(not),v(come)],[gel,+(mA),+(DH)],231,1.0)
rule([1,[v(do),+(PAST),adv(not),v(come)]],[1,[gel,+(mA),+(DH)]],232,0.25)
rule([1,[v(do),adv(not),v(come)]],[[gel,+(mA)],1],139,1.0)
fact([v(do),adv(not),v(come)],[gel,+(mA)],140,1.0)
rule([[pron(you),v(are)],1,[+(PROG)]],[1,[+(Hyor),+(ZHn)]],141,1.0)
rule([[pron(he),v(is)],1,[+(PROG)]],[1,[+(Hyor)]],142,1.0)
rule([[propN(aySe),v(is)],1,[+(PROG)]],[[aySe],1,[+(Hyor)]],143,1.0)
rule([[propN(ali),v(is)],1,[+(PROG)]],[[ali],1,[+(Hyor)]],144,1.0)
rule([[pron(we),v(are)],1,[+(PROG)]],[1,[+(Hyor),+(yHz)]],145,1.0)
rule([[pron2pl(you),v(are)],1,[+(PROG)]],[1,[+(Hyor),+(ZHnHz)]],146,1.0)
rule([[pron(they),v(are)],1,[+(PROG)]],[1,[+(Hyor),+(lAr)]],147,1.0)
rule([[pron(I)],1,[adv(not),v(come)]],[[gel,+(mA)],1],148,1.0)
rule([[pron(I),v(am)],1,[+(PROG)]],[1,[+(Hyor),+(yHm)]],149,1.0)
rule([[pron(you)],1,[adv(not),v(come)]],[[gel,+(mA)],1],150,1.0)
rule([[pron(he),v(do),+(s)],1,[]],[1,[+(z)]],151,1.0)
rule([[pron(he),v(do),+(s)],1,[]],[[],1,[+(z)]],152,1.0)
fact([pron(he),v(do),+(s)],[+(z)],153,1.0)
rule([[pron(he),v(do),+(s)],1],[1,[+(z)]],154,1.0)
rule([[pron(he)],1,[adv(not),v(come)]],[[gel,+(mA)],1],155,1.0)
rule([[pron(he),v(do),+(s)],1],[[],1,[+(z)]],156,1.0)
rule([[propN(aySe),v(do),+(s)],1,[]],[[aySe],1,[+(z)]],157,1.0)
rule([[pron(I),v(am)],1,[+(PROG)]],[[],1,[+(Hyor),+(yHm)]],158,1.0)
rule([[pron(you),v(are)],1,[+(PROG)]],[[],1,[+(Hyor),+(ZHn)]],159,1.0)
rule([[pron(he),v(is)],1,[+(PROG)]],[[],1,[+(Hyor)]],160,1.0)
```

```
rule([[pron(we),v(are)],1,[+(PROG)]],[[],1,[+(Hyor),+(yHz)]],161,1.0)
rule([[pron2pl(you),v(are)],1,[+(PROG)]],[[],1,[+(Hyor),+(ZHnHz)]],162,1.0)
rule([[pron(they),v(are)],1,[+(PROG)]],[[],1,[+(Hyor),+(lAr)]],163,1.0)
rule([1,[v(do)],2,[adv(not),v(come)]],[1,[gel,+(mA)],2],164,0.36363636363636365)
rule([[propN(aySe),v(do),+(s)],1],[[aySe],1,[+(z)]],165,1.0)
rule([[propN(aySe)],1,[adv(not),v(come)]],[[aySe,gel,+(mA)],1],166,1.0)
rule([[propN(ali),v(do),+(s)],1,[]],[[ali],1,[+(z)]],167,1.0)
rule([[propN(ali),v(do),+(s)],1],[[ali],1,[+(z)]],168,1.0)
rule([[propN(ali)],1,[adv(not),v(come)]],[[ali,gel,+(mA)],1],169,1.0)
rule([1,[adv(not),v(come),+(PROG)]],[[gel,+(mA),+(Hyor)],1],170,0.625)
fact([adv(not),v(come),+(PROG)],[gel,+(mA),+(Hyor)],171,1.0)
rule([[pron(you),v(do),+(PAST)],1,[]],[1,[+(DH),+(n)]],172,0.5)
rule([[pron(he),v(do),+(PAST)],1,[]],[1,[+(DH)]],173,1.0)
rule([[propN(aySe),v(do),+(PAST)],1,[]],[[aySe],1,[+(DH)]],174,1.0)
rule([[propN(ali),v(do),+(PAST)],1,[]],[[ali],1,[+(DH)]],175,1.0)
rule([[pron(we),v(do),+(PAST)],1,[]],[1,[+(DH),+(k)]],176,1.0)
rule([[pron(you),v(do),+(PAST)],1,[]],[1,[+(DH),+(nHz)]],177,0.5)
rule([[pron(they),v(do),+(PAST)],1,[]],[1,[+(DH),+(lAr)]],178,1.0)
rule([[pron(I),v(do),+(PAST)],1,[]],[1,[+(DH),+(m)]],179,1.0)
rule([[pron(I),v(do),+(PAST)],1,[]],[[],1,[+(DH),+(m)]],180,1.0)
rule([[pron(you),v(do),+(PAST)],1,[]],[[],1,[+(DH),+(n)]],181,0.5)
rule([[pron(he),v(do),+(PAST)],1,[]],[[],1,[+(DH)]],182,1.0)
rule([[pron(we),v(do),+(PAST)],1,[]],[[],1,[+(DH),+(k)]],183,1.0)
rule([[pron(you),v(do),+(PAST)],1,[]],[[],1,[+(DH),+(nHz)]],184,0.5)
rule([[pron(they),v(do),+(PAST)],1,[]],[[],1,[+(DH),+(lAr)]],185,1.0)
fact([pron(I),v(do),+(PAST)],[+(DH),+(m)],186,1.0)
rule([[pron(I),v(do),+(PAST)],1],[1,[+(DH),+(m)]],187,1.0)
rule([[pron(I),v(do),+(PAST)],1],[[],1,[+(DH),+(m)]],188,1.0)
fact([pron(you),v(do),+(PAST)],[+(DH),+(n)],189,0.5)
rule([[pron(you),v(do),+(PAST)],1],[1,[+(DH),+(n)]],190,0.5)
rule([[pron(you),v(do),+(PAST)],1],[[],1,[+(DH),+(n)]],191,0.5)
fact([pron(he),v(do),+(PAST)],[+(DH)],192,1.0)
rule([[pron(he),v(do),+(PAST)],1],[1,[+(DH)]],193,1.0)
rule([[pron(he),v(do),+(PAST)],1],[[],1,[+(DH)]],194,1.0)
rule([[propN(aySe),v(do),+(PAST)],1],[[aySe],1,[+(DH)]],195,1.0)
rule([[propN(ali),v(do),+(PAST)],1],[[ali],1,[+(DH)]],196,1.0)
fact([pron(we),v(do),+(PAST)],[+(DH),+(k)],197,1.0)
rule([[pron(we),v(do),+(PAST)],1],[1,[+(DH),+(k)]],198,1.0)
rule([[pron(we),v(do),+(PAST)],1],[[],1,[+(DH),+(k)]],199,1.0)
rule([[pron(we)],1,[adv(not),v(come)]],[[gel,+(mA)],1],200,1.0)
```

```
fact([pron(you),v(do),+(PAST)],[+(DH),+(nHz)],201,0.5)
rule([[pron(you),v(do),+(PAST)],1],[1,[+(DH),+(nHz)]],202,0.5)
rule([[pron(you),v(do),+(PAST)],1],[[],1,[+(DH),+(nHz)]],203,0.5)
fact([pron(they),v(do),+(PAST)],[+(DH),+(lAr)],204,1.0)
rule([[pron(they),v(do),+(PAST)],1],[1,[+(DH),+(lAr)]],205,1.0)
rule([[pron(they),v(do),+(PAST)],1],[[],1,[+(DH),+(lAr)]],206,1.0)
rule([1,[v(will),adv(not),v(come)]],[[gel,+(mA),+(yAcAk)],1],207,0.5)
fact([v(will),adv(not),v(come)],[gel,+(mA),+(yAcAk)],208,1.0)
rule([1,[v(will),adv(not),v(come)]],[1,[gel,+(mA),+(yAcAk)]],209,0.3333333333333333)
fact([adv(not),v(come)],[gel,+(mA)],54,1.0)
rule([[pron(I),v(do)],1,[]],[1,[+(Hm)]],55,0.5)
rule([[pron(I),v(do)],1,[]],[[],1,[+(Hm)]],56,0.5)
fact([pron(I),v(do)],[+(Hm)],57,0.5)
fact([pron(you),v(will)],[+(yAcAk),+(ZHn)],58,1.0)
rule([1,[adv(not),v(come)]],[[gel,+(mA)],1],59,1.0)
rule([[pron(I),v(do)],1],[1,[+(Hm)]],60,0.5)
rule([[pron(you),v(will)],1],[1,[+(yAcAk),+(ZHn)]],61,1.0)
fact([pron(he),v(will)],[+(yAcAk)],62,1.0)
rule([[pron(he),v(will)],1],[1,[+(yAcAk)]],63,1.0)
rule([[pron(I),v(do)],1],[[],1,[+(Hm)]],64,0.5)
rule([[propN(aySe),v(will)],1],[[aySe],1,[+(yAcAk)]],65,1.0)
rule([[propN(ali),v(will)],1],[[ali],1,[+(yAcAk)]],66,1.0)
fact([pron(we),v(will)],[+(yAcAk),+(yHz)],67,1.0)
rule([[pron(we),v(will)],1],[1,[+(yAcAk),+(yHz)]],68,1.0)
rule([[pron(you),v(do)],1,[]],[1,[+(zsHn)]],69,0.3333333333333333)
rule([[pron(you),v(do)],1,[]],[[],1,[+(zsHn)]],70,0.3333333333333333)
fact([pron(you),v(do)],[+(zsHn)],71,0.3333333333333333)
fact([pron(I),v(will)],[+(yAcAk),+(yHm)],72,1.0)
rule([[pron(you),v(do)],1],[1,[+(zsHn)]],73,0.3333333333333333)
rule([[pron(I),v(will)],1],[1,[+(yAcAk),+(yHm)]],74,1.0)
rule([[pron(you),v(do)],1],[[],1,[+(zsHn)]],75,0.3333333333333333)
fact([v(do),+(s)],[+(z)],76,1.0)
rule([1,[v(do),+(s)],2,[]],[1,2,[+(z)]],77,1.0)
rule([1,[v(is)],2,[+(PROG)]],[1,2,[+(Hyor)]],78,1.0)
fact([propN(aySe),v(do)],[aySe],79,1.0)
rule([[pron(I),v(will)],1],[[],1,[+(yAcAk),+(yHm)]],80,1.0)
rule([[pron(you),v(will)],1],[[],1,[+(yAcAk),+(ZHn)]],81,1.0)
rule([[pron(he),v(will)],1],[[],1,[+(yAcAk)]],82,1.0)
rule([1,[v(do),+(s)],2],[1,2,[+(z)]],83,1.0)
rule([1,2,[adv(not),v(come)]],[1,[gel,+(mA)],2],84,0.2666666666666666)
```

```
rule([[pron(we),v(will)],1],[[],1,[+(yAcAk),+(yHz)]],85,1.0)
fact([propN(ali),v(do)],[ali],86,1.0)
rule([[pron(we),v(do)],1,[]],[1,[+(yHz)]],87,0.5)
rule([[pron(we),v(do)],1,[]],[[],1,[+(yHz)]],88,0.5)
fact([pron(we),v(do)],[+(yHz)],89,0.5)
rule([[pron(we),v(do)],1],[1,[+(yHz)]],90,0.5)
rule([[pron(we),v(do)],1],[[],1,[+(yHz)]],91,0.5)
rule([[pron2pl(you),v(do)],1,[]],[1,[+(zsHnHz)]],92,1.0)
rule([[pron2pl(you),v(do)],1,[]],[[],1,[+(zsHnHz)]],93,1.0)
fact([pron2pl(you),v(do)],[+(zsHnHz)],94,1.0)
rule([[pron2pl(you),v(do)],1],[1,[+(zsHnHz)]],95,1.0)
rule([[pron2pl(you),v(do)],1],[[],1,[+(zsHnHz)]],96,1.0)
rule([[pron(they),v(do)],1,[]],[1,[+(zlAr)]],97,0.5)
rule([[pron(they),v(do)],1,[]],[[],1,[+(zlAr)]],98,0.5)
fact([pron(they),v(do)],[+(zlAr)],99,0.5)
rule([[pron(they),v(do)],1],[1,[+(zlAr)]],100,0.5)
rule([[pron(they),v(do)],1],[[],1,[+(zlAr)]],101,0.5)
fact([pron(I),v(am)],[+(yHm)],102,1.0)
fact([pron(you),v(are)],[+(ZHn)],103,1.0)
rule([[pron(I),v(am)],1],[1,[+(yHm)]],104,1.0)
rule([[pron(you),v(are)],1],[1,[+(ZHn)]],105,1.0)
rule([[pron(I),v(am)],1],[[],1,[+(yHm)]],106,1.0)
rule([[propN(aySe),v(is)],1],[[aySe],1,[]],107,1.0)
rule([[propN(ali),v(is)],1],[[ali],1,[]],108,1.0)
fact([pron(we),v(are)],[+(yHz)],109,1.0)
rule([[pron(we),v(are)],1],[1,[+(yHz)]],110,1.0)
fact([pron2pl(you),v(are)],[+(ZHnHz)],111,1.0)
rule([[pron2pl(you),v(are)],1],[1,[+(ZHnHz)]],112,1.0)
fact([pron(they),v(are)],[+(lAr)],113,1.0)
rule([[pron(they),v(are)],1],[1,[+(lAr)]],114,1.0)
rule([1,[v(am)],2,[+(PROG)]],[1,[+(Hyor)],2],115,1.0)
rule([[pron(you),v(will)],1,[]],[1,[+(yAcAk),+(ZHn)]],116,1.0)
rule([[pron(he),v(will)],1,[]],[1,[+(yAcAk)]],117,1.0)
rule([[propN(aySe),v(will)],1,[]],[[aySe],1,[+(yAcAk)]],118,1.0)
rule([[propN(ali),v(will)],1,[]],[[ali],1,[+(yAcAk)]],119,1.0)
rule([[pron(we),v(will)],1,[]],[1,[+(yAcAk),+(yHz)]],120,1.0)
rule([[pron(you),v(are)],1],[[],1,[+(ZHn)]],121,1.0)
rule([[pron(I),v(will)],1,[]],[1,[+(yAcAk),+(yHm)]],122,1.0)
rule([1,[v(are)],2,[+(PROG)]],[1,[+(Hyor)],2],123,1.0)
rule([[pron(we),v(are)],1],[[],1,[+(yHz)]],124,1.0)
```

```
rule([[pron2pl(you),v(are)],1],[[],1,[+(ZHnHz)]],125,1.0)
rule([[pron(they),v(are)],1],[[],1,[+(lAr)]],126,1.0)
rule([1,[v(do),+(PAST)],2,[]],[1,2,[+(DH)]],127,1.0)
rule([[pron(I),v(will)],1,[]],[[],1,[+(yAcAk),+(yHm)]],128,1.0)
rule([[pron(you),v(will)],1,[]],[[],1,[+(yAcAk),+(ZHn)]],129,1.0)
rule([[pron(he),v(will)],1,[]],[[],1,[+(yAcAk)]],130,1.0)
rule([[pron(we),v(will)],1,[]],[[],1,[+(yAcAk),+(yHz)]],131,1.0)
rule([1,[v(do),+(PAST)],2,[]],[1,[+(DH)],2],132,0.625)
fact([v(do),+(PAST)],[+(DH),+(m)],133,0.125)
fact([v(do),+(PAST)],[+(DH),+(n)],134,0.125)
fact([v(do),+(PAST)],[+(DH)],135,1.0)
rule([1,[v(do),+(PAST)],2],[1,2,[+(DH)]],136,1.0)
fact([v(do),+(PAST)],[+(DH),+(k)],137,0.125)
fact([v(do),+(PAST)],[+(DH),+(nHz)],138,0.125)
fact([pron(I)],[+(Hm)],1,0.25)
fact([pron(you)],[+(zsHn)],2,0.2)
rule([[pron(I)],1],[1,[+(Hm)]],3,0.25)
rule([[pron(you)],1],[1,[+(zsHn)]],4,0.2)
fact([pron(we)],[+(yHz)],5,0.75)
rule([[pron(we)],1],[1,[+(yHz)]],6,0.75)
fact([pron2pl(you)],[+(zsHnHz)],7,0.5)
rule([[pron2pl(you)],1],[1,[+(zsHnHz)]],8,0.5)
fact([pron(they)],[+(zlAr)],9,0.3333333333333333)
rule([[pron(they)],1],[1,[+(zlAr)]],10,0.3333333333333333)
fact([v(do)],[+(Hm)],11,0.0625)
fact([v(will)],[+(yAcAk),+(yHm)],12,0.16666666666666666)
fact([v(do)],[+(zsHn)],13,0.0625)
fact([v(will)],[+(yAcAk),+(ZHn)],14,0.16666666666666666)
fact([+(s)],[+(z)],15,1.0)
fact([+(PAST)],[+(DH)],16,1.0)
fact([v(will)],[+(yAcAk)],17,1.0)
fact([propN(aySe)],[aySe],18,1.0)
fact([propN(ali)],[ali],19,1.0)
rule([[propN(aySe)],1],[[aySe],1],20,1.0)
rule([[propN(ali)],1],[[ali],1],21,1.0)
rule([1,[+(s)],2],[1,2,[+(z)]],22,1.0)
rule([1,[+(PAST)],2],[1,2,[+(DH)]],23,1.0)
rule([1,[v(will)],2],[1,2,[+(yAcAk)]],24,1.0)
fact([pron(I)],[+(yHm)],25,0.5)
rule([1,[v(will)],2,[]],[1,[+(yAcAk)],2],26,0.5)
```

```
fact([pron(you)],[+(ZHn)],27,0.4)
rule([[pron(you)],1],[1,[+(ZHn)]],28,0.4)
fact([pron2pl(you)],[+(ZHnHz)],29,0.5)
rule([[pron2pl(you)],1],[1,[+(ZHnHz)]],30,0.5)
fact([pron(they)],[+(lAr)],31,0.6666666666666666)
rule([[pron(they)],1],[1,[+(lAr)]],32,0.6666666666666666)
rule([1,[v(will)],2,[]],[1,2,[+(yAcAk)]],33,1.0)
fact([pron(I)],[+(m)],34,0.25)
fact([pron(you)],[+(n)],35,0.2)
rule([[pron(I)],1],[1,[+(m)]],36,0.25)
rule([[pron(you)],1],[1,[+(n)]],37,0.2)
rule([[pron(I)],1],[[],1,[+(m)]],38,0.25)
rule([[propN(aySe)],1],[[aySe],1,[]],39,1.0)
rule([[propN(ali)],1],[[ali],1,[]],40,1.0)
fact([pron(we)],[+(k)],41,0.25)
rule([[pron(we)],1],[1,[+(k)]],42,0.25)
fact([pron(you)],[+(nHz)],43,0.2)
rule([[pron(you)],1],[1,[+(nHz)]],44,0.2)
rule([[pron(you)],1],[[],1,[+(n)]],45,0.2)
rule([[pron(we)],1],[[],1,[+(k)]],46,0.25)
rule([[pron(you)],1],[[],1,[+(nHz)]],47,0.2)
rule([[pron(they)],1],[[],1,[+(lAr)]],48,0.6666666666666666)
fact([v(will)],[+(yAcAk),+(yHz)],49,0.16666666666666666)
rule([[pron(I)],1],[1,[+(yHm)]],50,0.5)
rule([[pron(I)],1],[[],1,[+(yHm)]],51,0.5)
rule([[pron(you)],1],[[],1,[+(ZHn)]],52,0.4)
rule([[pron(we)],1],[[],1,[+(yHz)]],53,0.75)
```

Templates are ordered according to the number of terminals in the source language. Therefore, we need to order these templates in two ways, one for English and one for Turkish, since the translation process is bidirectional. The above templates are ordered according to English. The templates below are ordered according to Turkish:

```
fact([propN(aySe),v(do),+(s),adv(not),v(come)],[aySe,gel,+(mA),+(z)],231,1.0)
fact([propN(ali),v(do),+(s),adv(not),v(come)],[ali,gel,+(mA),+(z)],232,1.0)
fact([pron(I),v(am),adv(not),v(come),+(PROG)],[gel,+(mA),+(Hyor),+(yHm)],233,1.0)
fact([pron(you),v(are),adv(not),v(come),+(PROG)],[gel,+(mA),+(Hyor),+(ZHn)],234,1.0)
fact([propN(aySe),v(is),adv(not),v(come),+(PROG)],[aySe,gel,+(mA),+(Hyor)],235,1.0)
fact([propN(ali),v(is),adv(not),v(come),+(PROG)],[ali,gel,+(mA),+(Hyor)],236,1.0)
fact([pron(we),v(are),adv(not),v(come),+(PROG)],[gel,+(mA),+(Hyor),+(yHz)],237,1.0)
fact([pron2pl(you),v(are),adv(not),v(come),+(PROG)],[gel,+(mA),+(Hyor),+(ZHnHz)],238,1.0)
```

```
fact([pron(they),v(are),adv(not),v(come),+(PROG)],[gel,+(mA),+(Hyor),+(lAr)],239,1.0)
fact([pron(I),v(do),+(PAST),adv(not),v(come)],[gel,+(mA),+(DH),+(m)],240,1.0)
fact([pron(you),v(do),+(PAST),adv(not),v(come)],[gel,+(mA),+(DH),+(n)],241,1.0)
fact([propN(aySe),v(do),+(PAST),adv(not),v(come)],[aySe,gel,+(mA),+(DH)],242,1.0)
fact([propN(ali),v(do),+(PAST),adv(not),v(come)],[ali,gel,+(mA),+(DH)],243,1.0)
fact([pron(we),v(do),+(PAST),adv(not),v(come)],[gel,+(mA),+(DH),+(k)],244,1.0)
fact([pron(you),v(do),+(PAST),adv(not),v(come)],[gel,+(mA),+(DH),+(nHz)],245,1.0)
fact([pron(they),v(do),+(PAST),adv(not),v(come)],[gel,+(mA),+(DH),+(lAr)],246,1.0)
fact([pron(I),v(will),adv(not),v(come)],[gel,+(mA),+(yAcAk),+(yHm)],247,1.0)
fact([pron(you),v(will),adv(not),v(come)],[gel,+(mA),+(yAcAk),+(ZHn)],248,1.0)
fact([propN(aySe),v(will),adv(not),v(come)],[aySe,gel,+(mA),+(yAcAk)],249,1.0)
fact([propN(ali),v(will),adv(not),v(come)],[ali,gel,+(mA),+(yAcAk)],250,1.0)
fact([pron(we),v(will),adv(not),v(come)],[gel,+(mA),+(yAcAk),+(yHz)],251,1.0)
fact([pron(I),v(do),adv(not),v(come)],[gel,+(mA),+(Hm)],204,1.0)
fact([pron(you),v(do),adv(not),v(come)],[gel,+(mA),+(zsHn)],205,1.0)
fact([pron(he),v(do),+(s),adv(not),v(come)],[gel,+(mA),+(z)],206,0.3333333333333333)
fact([pron(we),v(do),adv(not),v(come)],[gel,+(mA),+(yHz)],207,0.3333333333333333)
fact([pron2pl(you),v(do),adv(not),v(come)],[gel,+(mA),+(zsHnHz)],208,1.0)
fact([pron(they),v(do),adv(not),v(come)],[gel,+(mA),+(zlAr)],209,1.0)
fact([pron(he),v(is),adv(not),v(come),+(PROG)],[gel,+(mA),+(Hyor)],210,0.125)
fact([pron(he),v(do),+(PAST),adv(not),v(come)],[gel,+(mA),+(DH)],211,0.125)
fact([pron(he),v(will),adv(not),v(come)],[gel,+(mA),+(yAcAk)],212,0.1666666666666666)
rule([1,[v(do),+(s),adv(not),v(come)]],[1,[gel,+(mA),+(z)]],213,1.0)
fact([v(do),+(s),adv(not),v(come)],[gel,+(mA),+(z)],214,1.0)
rule([[propN(aySe),v(do)],1,[adv(not),v(come)]],[[aySe,gel,+(mA)],1],215,0.5)
rule([[propN(aySe)],1,[adv(not),v(come)]],[[aySe,gel,+(mA)],1],216,1.0)
rule([[propN(ali),v(do)],1,[adv(not),v(come)]],[[ali,gel,+(mA)],1],217,0.5)
rule([[propN(ali)],1,[adv(not),v(come)]],[[ali,gel,+(mA)],1],218,1.0)
rule([1,[adv(not),v(come),+(PROG)]],[[gel,+(mA),+(Hyor)],1],219,1.0)
fact([adv(not),v(come),+(PROG)],[gel,+(mA),+(Hyor)],220,1.0)
rule([1,[v(are),adv(not),v(come),+(PROG)]],[[gel,+(mA),+(Hyor)],1],221,0.8)
fact([v(are),adv(not),v(come),+(PROG)],[gel,+(mA),+(Hyor)],222,0.5)
rule([1,[v(is),adv(not),v(come),+(PROG)]],[1,[gel,+(mA),+(Hyor)]],223,1.0)
fact([v(is),adv(not),v(come),+(PROG)],[gel,+(mA),+(Hyor)],224,0.375)
rule([1,[v(do),+(PAST),adv(not),v(come)]],[[gel,+(mA),+(DH)],1],225,1.0)
fact([v(do),+(PAST),adv(not),v(come)],[gel,+(mA),+(DH)],226,1.0)
rule([1,[v(do),+(PAST),adv(not),v(come)]],[1,[gel,+(mA),+(DH)]],227,1.0)
rule([1,[v(will),adv(not),v(come)]],[[gel,+(mA),+(yAcAk)],1],228,1.0)
fact([v(will),adv(not),v(come)],[gel,+(mA),+(yAcAk)],229,1.0)
rule([1,[v(will),adv(not),v(come)]],[1,[gel,+(mA),+(yAcAk)]],230,1.0)
```

```
rule([1,[v(do),adv(not),v(come)]],[[gel,+(mA)],1],122,0.5333333333333333)
fact([v(do),adv(not),v(come)],[gel,+(mA)],123,0.5333333333333333)
fact([adv(not),v(come)],[gel,+(mA)],124,1.0)
rule([[pron(you),v(are)],1,[+(PROG)]],[1,[+(Hyor),+(ZHn)]],125,1.0)
rule([[propN(aySe),v(is)],1,[+(PROG)]],[[aySe],1,[+(Hyor)]],126,1.0)
rule([[propN(ali),v(is)],1,[+(PROG)]],[[ali],1,[+(Hyor)]],127,1.0)
rule([[pron(we),v(are)],1,[+(PROG)]],[1,[+(Hyor),+(yHz)]],128,1.0)
rule([[pron2pl(you),v(are)],1,[+(PROG)]],[1,[+(Hyor),+(ZHnHz)]],129,1.0)
rule([[pron(they),v(are)],1,[+(PROG)]],[1,[+(Hyor),+(lAr)]],130,1.0)
fact([v(will)],[+(yAcAk),+(yHm)],131,1.0)
rule([[pron(I)],1,[adv(not),v(come)]],[[gel,+(mA)],1],132,0.1333333333333333)
fact([pron(you),v(will)],[+(yAcAk),+(ZHn)],133,1.0)
rule([1,[adv(not),v(come)]],[[gel,+(mA)],1],134,1.0)
rule([[pron(you),v(will)],1],[1,[+(yAcAk),+(ZHn)]],135,1.0)
rule([[propN(aySe),v(will)],1],[[aySe],1,[+(yAcAk)]],136,1.0)
rule([[propN(ali),v(will)],1],[[ali],1,[+(yAcAk)]],137,1.0)
fact([pron(we),v(will)],[+(yAcAk),+(yHz)],138,1.0)
rule([[pron(we),v(will)],1],[1,[+(yAcAk),+(yHz)]],139,1.0)
rule([[pron(I),v(am)],1,[+(PROG)]],[1,[+(Hyor),+(yHm)]],140,1.0)
fact([pron(I),v(will)],[+(yAcAk),+(yHm)],141,1.0)
rule([[pron(I),v(will)],1],[1,[+(yAcAk),+(yHm)]],142,1.0)
fact([v(will)],[+(yAcAk),+(ZHn)],143,1.0)
rule([[pron(you)],1,[adv(not),v(come)]],[[gel,+(mA)],1],144,0.1666666666666666)
rule([[pron(he),v(do)],1,[adv(not),v(come)]],[[gel,+(mA)],1],145,0.0666666666666667)
rule([[pron(he)],1,[adv(not),v(come)]],[[gel,+(mA)],1],146,0.1333333333333333)
rule([[propN(aySe),v(do),+(s)],1,[]],[[aySe],1,[+(z)]],147,1.0)
rule([[pron(I),v(am)],1,[+(PROG)]],[[],1,[+(Hyor),+(yHm)]],148,1.0)
rule([[pron(you),v(are)],1,[+(PROG)]],[[],1,[+(Hyor),+(ZHn)]],149,1.0)
rule([[pron(we),v(are)],1,[+(PROG)]],[[],1,[+(Hyor),+(yHz)]],150,1.0)
rule([[pron2pl(you),v(are)],1,[+(PROG)]],[[],1,[+(Hyor),+(ZHnHz)]],151,1.0)
rule([[pron(they),v(are)],1,[+(PROG)]],[[],1,[+(Hyor),+(lAr)]],152,1.0)
rule([1,[v(do)],2,[adv(not),v(come)]],[1,[gel,+(mA)],2],153,0.5)
rule([[propN(aySe),v(do),+(s)],1],[[aySe],1,[+(z)]],154,1.0)
rule([[pron(I),v(will)],1],[[],1,[+(yAcAk),+(yHm)]],155,1.0)
rule([[pron(you),v(will)],1],[[],1,[+(yAcAk),+(ZHn)]],156,1.0)
rule([1,2,[adv(not),v(come)]],[1,[gel,+(mA)],2],157,1.0)
rule([[pron(we),v(will)],1],[[],1,[+(yAcAk),+(yHz)]],158,1.0)
rule([[propN(ali),v(do),+(s)],1,[]],[[ali],1,[+(z)]],159,1.0)
rule([[propN(ali),v(do),+(s)],1],[[ali],1,[+(z)]],160,1.0)
rule([[pron(you),v(do),+(PAST)],1,[]],[1,[+(DH),+(n)]],161,1.0)
```

```
rule([[propN(aySe),v(do),+(PAST)],1,[]],[[aySe],1,[+(DH)]],162,1.0)
rule([[propN(ali),v(do),+(PAST)],1,[]],[[ali],1,[+(DH)]],163,1.0)
rule([[pron(we),v(do),+(PAST)],1,[]],[1,[+(DH),+(k)]],164,1.0)
rule([[pron(you),v(do),+(PAST)],1,[]],[1,[+(DH),+(nHz)]],165,1.0)
rule([[pron(they),v(do),+(PAST)],1,[]],[1,[+(DH),+(lAr)]],166,1.0)
rule([[pron(you),v(will)],1,[]],[1,[+(yAcAk),+(ZHn)]],167,1.0)
rule([[propN(aySe),v(will)],1,[]],[[aySe],1,[+(yAcAk)]],168,1.0)
rule([[propN(ali),v(will)],1,[]],[[ali],1,[+(yAcAk)]],169,1.0)
rule([[pron(we),v(will)],1,[]],[1,[+(yAcAk),+(yHz)]],170,1.0)
rule([[pron(I),v(do),+(PAST)],1,[]],[1,[+(DH),+(m)]],171,1.0)
rule([[pron(I),v(will)],1,[]],[1,[+(yAcAk),+(yHm)]],172,1.0)
rule([[pron(I),v(do),+(PAST)],1,[]],[[],1,[+(DH),+(m)]],173,1.0)
rule([[pron(you),v(do),+(PAST)],1,[]],[[],1,[+(DH),+(n)]],174,1.0)
rule([[pron(we),v(do),+(PAST)],1,[]],[[],1,[+(DH),+(k)]],175,1.0)
rule([[pron(you),v(do),+(PAST)],1,[]],[[],1,[+(DH),+(nHz)]],176,1.0)
rule([[pron(they),v(do),+(PAST)],1,[]],[[],1,[+(DH),+(lAr)]],177,1.0)
rule([[pron(I),v(will)],1,[]],[[],1,[+(yAcAk),+(yHm)]],178,1.0)
rule([[pron(you),v(will)],1,[]],[[],1,[+(yAcAk),+(ZHn)]],179,1.0)
rule([[pron(we),v(will)],1,[]],[[],1,[+(yAcAk),+(yHz)]],180,1.0)
fact([v(do),+(PAST)],[+(DH),+(m)],181,1.0)
fact([pron(I),v(do),+(PAST)],[+(DH),+(m)],182,1.0)
rule([[pron(I),v(do),+(PAST)],1],[1,[+(DH),+(m)]],183,1.0)
rule([[pron(I),v(do),+(PAST)],1],[[],1,[+(DH),+(m)]],184,1.0)
fact([pron(you),v(do),+(PAST)],[+(DH),+(n)],185,1.0)
rule([[pron(you),v(do),+(PAST)],1],[1,[+(DH),+(n)]],186,1.0)
fact([v(do),+(PAST)],[+(DH),+(n)],187,1.0)
rule([[pron(you),v(do),+(PAST)],1],[[],1,[+(DH),+(n)]],188,1.0)
rule([[propN(aySe),v(do),+(PAST)],1],[[aySe],1,[+(DH)]],189,1.0)
rule([[propN(ali),v(do),+(PAST)],1],[[ali],1,[+(DH)]],190,1.0)
fact([pron(we),v(do),+(PAST)],[+(DH),+(k)],191,1.0)
rule([[pron(we),v(do),+(PAST)],1],[1,[+(DH),+(k)]],192,1.0)
rule([[pron(we),v(do),+(PAST)],1],[[],1,[+(DH),+(k)]],193,1.0)
fact([v(do),+(PAST)],[+(DH),+(k)],194,1.0)
fact([v(will)],[+(yAcAk),+(yHz)],195,1.0)
rule([[pron(we)],1,[adv(not),v(come)]],[[gel,+(mA)],1],196,0.13333333333333333)
fact([pron(you),v(do),+(PAST)],[+(DH),+(nHz)],197,1.0)
rule([[pron(you),v(do),+(PAST)],1],[1,[+(DH),+(nHz)]],198,1.0)
fact([v(do),+(PAST)],[+(DH),+(nHz)],199,1.0)
rule([[pron(you),v(do),+(PAST)],1],[[],1,[+(DH),+(nHz)]],200,1.0)
fact([pron(they),v(do),+(PAST)],[+(DH),+(lAr)],201,1.0)
```

```
rule([[pron(they),v(do),+(PAST)],1],[1,[+(DH),+(lAr)]],202,1.0)
rule([[pron(they),v(do),+(PAST)],1],[[],1,[+(DH),+(lAr)]],203,1.0)
fact([pron(I)],[+(Hm)],1,1.0)
fact([pron(you)],[+(zsHn)],2,1.0)
rule([[pron(I)],1],[1,[+(Hm)]],3,1.0)
rule([[pron(you)],1],[1,[+(zsHn)]],4,1.0)
fact([pron(we)],[+(yHz)],5,1.0)
rule([[pron(we)],1],[1,[+(yHz)]],6,1.0)
fact([pron2pl(you)],[+(zsHnHz)],7,1.0)
rule([[pron2pl(you)],1],[1,[+(zsHnHz)]],8,1.0)
fact([pron(they)],[+(zlAr)],9,1.0)
rule([[pron(they)],1],[1,[+(zlAr)]],10,1.0)
rule([[pron(I),v(do)],1,[]],[1,[+(Hm)]],11,1.0)
rule([[pron(he),v(is)],1,[+(PROG)]],[1,[+(Hyor)]],12,0.125)
rule([[pron(I),v(do)],1,[]],[[],1,[+(Hm)]],13,1.0)
fact([v(do)],[+(Hm)],14,1.0)
fact([pron(I),v(do)],[+(Hm)],15,1.0)
rule([[pron(I),v(do)],1],[1,[+(Hm)]],16,1.0)
fact([pron(he),v(will)],[+(yAcAk)],17,0.16666666666666666)
rule([[pron(he),v(will)],1],[1,[+(yAcAk)]],18,0.16666666666666666)
rule([[pron(I),v(do)],1],[[],1,[+(Hm)]],19,1.0)
rule([[pron(you),v(do)],1,[]],[1,[+(zsHn)]],20,1.0)
rule([[pron(you),v(do)],1,[]],[[],1,[+(zsHn)]],21,1.0)
fact([pron(you),v(do)],[+(zsHn)],22,1.0)
rule([[pron(you),v(do)],1],[1,[+(zsHn)]],23,1.0)
fact([v(do)],[+(zsHn)],24,1.0)
rule([[pron(you),v(do)],1],[[],1,[+(zsHn)]],25,1.0)
rule([[pron(he),v(do),+(s)],1,[]],[1,[+(z)]],26,0.3333333333333333)
rule([[pron(he),v(do),+(s)],1,[]],[[],1,[+(z)]],27,0.3333333333333333)
fact([+(s)],[+(z)],28,1.0)
fact([+(PAST)],[+(DH)],29,1.0)
fact([pron(he),v(do),+(s)],[+(z)],30,0.3333333333333333)
rule([[pron(he),v(do),+(s)],1],[1,[+(z)]],31,0.3333333333333333)
fact([v(do),+(s)],[+(z)],32,1.0)
fact([v(will)],[+(yAcAk)],33,1.0)
rule([[pron(he),v(do),+(s)],1],[[],1,[+(z)]],34,0.3333333333333333)
fact([propN(aySe)],[aySe],35,1.0)
fact([propN(ali)],[ali],36,1.0)
rule([[propN(aySe)],1],[[aySe],1],37,1.0)
rule([[propN(ali)],1],[[ali],1],38,1.0)
```

```
rule([[pron(he),v(is)],1,[+(PROG)]],[[],1,[+(Hyor)]],39,0.125)
rule([1,[v(do),+(s)],2,[]],[1,2,[+(z)]],40,1.0)
rule([1,[v(is)],2,[+(PROG)]],[1,2,[+(Hyor)]],41,0.375)
fact([propN(aySe),v(do)],[aySe],42,0.5)
rule([1,[+(s)],2],[1,2,[+(z)]],43,1.0)
rule([1,[+(PAST)],2],[1,2,[+(DH)]],44,1.0)
rule([[pron(he),v(will)],1],[[],1,[+(yAcAk)]],45,0.1666666666666666)
rule([1,[v(do),+(s)],2],[1,2,[+(z)]],46,1.0)
rule([1,[v(will)],2],[1,2,[+(yAcAk)]],47,1.0)
fact([propN(ali),v(do)],[ali],48,0.5)
rule([[pron(we),v(do)],1,[]],[1,[+(yHz)]],49,0.3333333333333333)
rule([[pron(we),v(do)],1,[]],[[],1,[+(yHz)]],50,0.3333333333333333)
fact([pron(we),v(do)],[+(yHz)],51,0.3333333333333333)
rule([[pron(we),v(do)],1],[1,[+(yHz)]],52,0.3333333333333333)
rule([[pron(we),v(do)],1],[[],1,[+(yHz)]],53,0.3333333333333333)
rule([[pron2pl(you),v(do)],1,[]],[1,[+(zsHnHz)]],54,1.0)
rule([[pron2pl(you),v(do)],1,[]],[[],1,[+(zsHnHz)]],55,1.0)
fact([pron2pl(you),v(do)],[+(zsHnHz)],56,1.0)
rule([[pron2pl(you),v(do)],1],[1,[+(zsHnHz)]],57,1.0)
rule([[pron2pl(you),v(do)],1],[[],1,[+(zsHnHz)]],58,1.0)
rule([[pron(they),v(do)],1,[]],[1,[+(zlAr)]],59,1.0)
rule([[pron(they),v(do)],1,[]],[[],1,[+(zlAr)]],60,1.0)
fact([pron(they),v(do)],[+(zlAr)],61,1.0)
rule([[pron(they),v(do)],1],[1,[+(zlAr)]],62,1.0)
rule([[pron(they),v(do)],1],[[],1,[+(zlAr)]],63,1.0)
fact([pron(I),v(am)],[+(yHm)],64,0.5)
fact([pron(you),v(are)],[+(ZHn)],65,0.5)
rule([[pron(I),v(am)],1],[1,[+(yHm)]],66,0.5)
rule([[pron(you),v(are)],1],[1,[+(ZHn)]],67,0.5)
rule([[pron(I),v(am)],1],[[],1,[+(yHm)]],68,0.5)
rule([[propN(aySe),v(is)],1],[[aySe],1,[]],69,0.25)
rule([[propN(ali),v(is)],1],[[ali],1,[]],70,0.25)
fact([pron(we),v(are)],[+(yHz)],71,0.3333333333333333)
rule([[pron(we),v(are)],1],[1,[+(yHz)]],72,0.3333333333333333)
fact([pron2pl(you),v(are)],[+(ZHnHz)],73,1.0)
rule([[pron2pl(you),v(are)],1],[1,[+(ZHnHz)]],74,1.0)
fact([pron(they),v(are)],[+(lAr)],75,0.5)
rule([[pron(they),v(are)],1],[1,[+(lAr)]],76,0.5)
rule([[pron(he),v(do),+(PAST)],1,[]],[1,[+(DH)]],77,0.125)
fact([pron(I)],[+(yHm)],78,1.0)
```

```
rule([1,[v(am)],2,[+(PROG)]],[1,[+(Hyor)],2],79,0.2)
rule([1,[v(will)],2,[]],[1,[+(yAcAk)],2],80,1.0)
rule([[pron(he),v(will)],1,[]],[1,[+(yAcAk)]],81,0.16666666666666666)
rule([[pron(you),v(are)],1],[[],1,[+(ZHn)]],82,0.5)
fact([pron(you)],[+(ZHn)],83,1.0)
rule([[pron(you)],1],[1,[+(ZHn)]],84,1.0)
fact([pron2pl(you)],[+(ZHnHz)],85,1.0)
rule([[pron2pl(you)],1],[1,[+(ZHnHz)]],86,1.0)
fact([pron(they)],[+(lAr)],87,1.0)
rule([[pron(they)],1],[1,[+(lAr)]],88,1.0)
rule([1,[v(are)],2,[+(PROG)]],[1,[+(Hyor)],2],89,0.8)
rule([[pron(we),v(are)],1],[[],1,[+(yHz)]],90,0.3333333333333333)
rule([[pron2pl(you),v(are)],1],[[],1,[+(ZHnHz)]],91,1.0)
rule([[pron(they),v(are)],1],[[],1,[+(lAr)]],92,0.5)
rule([[pron(he),v(do),+(PAST)],1,[]],[[],1,[+(DH)]],93,0.125)
rule([1,[v(do),+(PAST)],2,[]],[1,2,[+(DH)]],94,1.0)
rule([[pron(he),v(will)],1,[]],[[],1,[+(yAcAk)]],95,0.16666666666666666)
rule([1,[v(will)],2,[]],[1,2,[+(yAcAk)]],96,1.0)
rule([1,[v(do),+(PAST)],2,[]],[1,[+(DH)],2],97,1.0)
fact([pron(I)],[+(m)],98,1.0)
fact([pron(you)],[+(n)],99,1.0)
rule([[pron(I)],1],[1,[+(m)]],100,1.0)
rule([[pron(you)],1],[1,[+(n)]],101,1.0)
rule([[pron(I)],1],[[],1,[+(m)]],102,1.0)
rule([[propN(aySe)],1],[[aySe],1,[]],103,1.0)
rule([[propN(ali)],1],[[ali],1,[]],104,1.0)
fact([pron(we)],[+(k)],105,1.0)
rule([[pron(we)],1],[1,[+(k)]],106,1.0)
fact([pron(you)],[+(nHz)],107,1.0)
rule([[pron(you)],1],[1,[+(nHz)]],108,1.0)
rule([[pron(you)],1],[[],1,[+(n)]],109,1.0)
fact([pron(he),v(do),+(PAST)],[+(DH)],110,0.125)
rule([[pron(he),v(do),+(PAST)],1],[1,[+(DH)]],111,0.125)
fact([v(do),+(PAST)],[+(DH)],112,1.0)
rule([[pron(he),v(do),+(PAST)],1],[[],1,[+(DH)]],113,0.125)
rule([[pron(we)],1],[[],1,[+(k)]],114,1.0)
rule([[pron(you)],1],[[],1,[+(nHz)]],115,1.0)
rule([[pron(they)],1],[[],1,[+(lAr)]],116,1.0)
rule([1,[v(do),+(PAST)],2],[1,2,[+(DH)]],117,1.0)
rule([[pron(I)],1],[1,[+(yHm)]],118,1.0)
```

```
rule([[pron(I)],1],[[],1,[+(yHm)]],119,1.0)
rule([[pron(you)],1],[[],1,[+(ZHn)]],120,1.0)
rule([[pron(we)],1],[[],1,[+(yHz)]],121,1.0)
```

The next step is assigning confidence factors to rule combinations. The confidence factors of the rule combinations are denoted as:

rw([Root,[Children]], CF, Flag).

where **[Root,[Children]]** is a tree structure mentioned before, **Root** is the root of the tree, and **Children** are its children and **CF** denotes confidence factor, and **Flag** denotes whether this confidence factor is for left to right or right to left translation. Then the some of the confidence factors for rule combinations are obtained as:

```
rw([139,[1]],1.0,lr).
rw([139,[25]],0.4142135623730951,lr).
rw([139,[34]],0.4142135623730951,lr).
rw([148,[11]],1.0,lr).
rw([148,[13]],0.4142135623730951,lr).
rw([55,[54]],1.0,lr).
rw([56,[54]],1.0,lr).
rw([59,[57]],1.0,lr).
rw([60,[54]],1.0,lr).
rw([64,[54]],1.0,lr).
rw([84,[1,11]],0.000999000999000999,lr).
rw([84,[1,13]],0.000999000999000999,lr).
rw([84,[25,11]],0.5,lr).
rw([84,[25,13]],0.3090169943749474,lr).
rw([84,[34,11]],0.5,lr).
rw([84,[34,13]],0.3090169943749474,lr).
rw([3,[140]],1.0,lr).
rw([36,[140]],0.4142135623730951,lr).
rw([38,[140]],0.4142135623730951,lr).
rw([50,[140]],0.4142135623730951,lr).
rw([51,[140]],0.4142135623730951,lr).
rw([211],1.0,lr).
rw([139,[2]],1.0,lr).
rw([139,[27]],0.4142135623730951,lr).
rw([139,[35]],0.4142135623730951,lr).
rw([139,[43]],0.4142135623730951,lr).
rw([150,[11]],0.4142135623730951,lr).
rw([150,[13]],1.0,lr).
```

```
rw([59,[71]],1.0,lr).
rw([69,[54]],1.0,lr).
rw([70,[54]],1.0,lr).
rw([73,[54]],1.0,lr).
rw([75,[54]],1.0,lr).
rw([84,[2,11]],0.000999000999000999,lr).
rw([84,[2,13]],0.000999000999000999,lr).
rw([84,[27,11]],0.3090169943749474,lr).
rw([84,[27,13]],0.5,lr).
rw([84,[35,11]],0.3090169943749474,lr).
rw([84,[35,13]],0.5,lr).
rw([84,[43,11]],0.3090169943749474,lr).
rw([84,[43,13]],0.5,lr).
rw([4,[140]],1.0,lr).
rw([28,[140]],0.4142135623730951,lr).
rw([37,[140]],0.4142135623730951,lr).
rw([44,[140]],0.4142135623730951,lr).
rw([45,[140]],0.4142135623730951,lr).
rw([47,[140]],0.4142135623730951,lr).
rw([52,[140]],0.4142135623730951,lr).
rw([233],1.0,lr).
rw([139,[29]],0.4142135623730951,lr).
rw([59,[94]],1.0,lr).
rw([84,[7,11]],0.000999000999000999,lr).
rw([84,[7,13]],0.000999000999000999,lr).
rw([84,[29,11]],0.3090169943749474,lr).
rw([84,[29,13]],0.3090169943749474,lr).
rw([92,[54]],1.0,lr).
rw([48,[140]],0.4142135623730951,lr).
rw([232,[34]],0.000999000999000999,lr).
rw([148,[134]],0.4142135623730951,lr).
rw([148,[135]],0.4142135623730951,lr).
rw([148,[137]],0.4142135623730951,lr).
rw([148,[138]],0.4142135623730951,lr).
rw([164,[1,16]],0.3090169943749474,lr).
rw([164,[25,16]],0.3090169943749474,lr).
rw([164,[34,16]],0.000999000999000999,lr).
rw([84,[1,134]],0.3090169943749474,lr).
rw([84,[1,135]],0.3090169943749474,lr).
rw([84,[1,137]],0.3090169943749474,lr).
```

```
rw([84,[1,138]],0.3090169943749474,lr).
rw([127,[34,54]],0.000999000999000999,lr).
rw([132,[1,54]],0.4142135623730951,lr).
rw([132,[25,54]],0.4142135623730951,lr).
rw([132,[34,54]],1.0,lr).
rw([150,[134]],0.5,lr).
rw([150,[135]],0.4142135623730951,lr).
rw([150,[137]],0.4142135623730951,lr).
rw([84,[43,134]],0.5,lr).
rw([84,[43,135]],0.3090169943749474,lr).
rw([84,[43,137]],0.3090169943749474,lr).
rw([84,[43,138]],0.3090169943749474,lr).
rw([84,[71,16]],0.3090169943749474,lr).
rw([174,[54]],1.0,lr).
rw([195,[54]],1.0,lr).
rw([166,[49]],0.5,lr).
rw([207,[18]],0.000999000999000999,lr).
rw([209,[18]],1.0,lr).
rw([65,[54]],1.0,lr).
rw([84,[18,12]],0.5,lr).
rw([84,[18,14]],0.5,lr).
rw([84,[18,17]],1.0,lr).
rw([84,[18,49]],0.5,lr).
rw([118,[54]],1.0,lr).
rw([20,[208]],1.0,lr).
rw([24,[18,54]],1.0,lr).
rw([26,[18,54]],0.000999000999000999,lr).
rw([33,[18,54]],1.0,lr).
rw([39,[208]],1.0,lr).
rw([219],1.0,lr).
rw([169,[12]],0.5,lr).
rw([169,[14]],0.5,lr).
rw([169,[17]],1.0,lr).
rw([169,[49]],0.5,lr).
rw([207,[19]],0.000999000999000999,lr).
rw([209,[19]],1.0,lr).
rw([66,[54]],1.0,lr).
rw([84,[19,12]],0.5,lr).
rw([84,[19,14]],0.5,lr).
rw([84,[19,17]],1.0,lr).
```

```
rw([84,[19,49]],0.5,lr).
rw([119,[54]],1.0,lr).
rw([21,[208]],1.0,lr).
rw([24,[19,54]],1.0,lr).
rw([26,[19,54]],0.000999000999000999,lr).
rw([33,[19,54]],1.0,lr).
rw([40,[208]],1.0,lr).
rw([220],1.0,lr).
rw([200,[12]],0.4142135623730951,lr).


rw([217,[28]],1.0,rl).
rw([217,[30]],0.3333333333333333,rl).
rw([217,[32]],0.5,rl).
rw([218,[28]],0.4142135623730951,rl).
rw([218,[30]],0.5,rl).
rw([218,[32]],1.0,rl).
rw([153,[36,28]],1.0,rl).
rw([153,[36,30]],0.3333333333333333,rl).
rw([153,[36,32]],0.5,rl).
rw([153,[48,28]],0.5,rl).
rw([153,[48,30]],0.25,rl).
rw([153,[48,32]],0.3333333333333333,rl).
rw([157,[36,28]],0.4142135623730951,rl).
rw([157,[36,30]],0.5,rl).
rw([157,[36,32]],1.0,rl).
rw([157,[48,28]],1.0,rl).
rw([157,[48,30]],0.3333333333333333,rl).
rw([157,[48,32]],0.5,rl).
rw([159,[123]],0.5,rl).
rw([159,[124]],1.0,rl).
rw([160,[123]],0.5,rl).
rw([134,[5]],0.4142135623730951,rl).
rw([134,[51]],1.0,rl).
rw([134,[71]],0.4142135623730951,rl).
rw([84,[224]],0.4142135623730951,rl).
rw([89,[65,123]],0.3333333333333333,rl).
rw([89,[65,124]],0.5,rl).
rw([89,[83,123]],0.5,rl).
rw([89,[83,124]],1.0,rl).
```

```
rw([79,[73,123]],0.3333333333333333,rl).
rw([79,[73,124]],0.5,rl).
rw([79,[85,123]],0.3090169943749474,rl).
rw([79,[85,124]],0.4142135623730951,rl).
rw([86,[210]],0.3090169943749474,rl).
rw([86,[220]],0.4142135623730951,rl).
rw([217,[112]],0.5,rl).
rw([218,[29]],0.4142135623730951,rl).
rw([218,[110]],0.5,rl).
rw([218,[112]],1.0,rl).
rw([227,[36]],1.0,rl).
rw([227,[48]],0.5,rl).
rw([153,[36,29]],1.0,rl).
rw([153,[36,110]],0.3333333333333333,rl).
rw([153,[36,112]],0.5,rl).
rw([153,[48,29]],0.5,rl).
rw([153,[48,110]],0.25,rl).
rw([153,[48,112]],0.3333333333333333,rl).
rw([157,[36,29]],0.4142135623730951,rl).
rw([157,[36,110]],0.5,rl).
rw([157,[36,112]],1.0,rl).
rw([157,[48,29]],1.0,rl).
rw([157,[48,110]],0.3333333333333333,rl).
rw([157,[48,112]],0.5,rl).
rw([163,[123]],0.5,rl).
rw([163,[124]],1.0,rl).
rw([190,[123]],0.5,rl).
rw([190,[124]],1.0,rl).
rw([38,[211]],0.5,rl).
rw([38,[226]],1.0,rl).
rw([44,[36,123]],0.000999000999000999,rl).
rw([44,[36,124]],0.4142135623730951,rl).
rw([44,[48,123]],0.5,rl).
rw([44,[48,124]],1.0,rl).
rw([70,[211]],0.3333333333333333,rl).
rw([70,[226]],0.5,rl).
rw([94,[36,123]],0.5,rl).
rw([94,[36,124]],1.0,rl).
rw([94,[48,123]],0.3333333333333333,rl).
rw([94,[48,124]],0.5,rl).
```

```
rw([104,[211]],0.5,rl).
rw([104,[226]],1.0,rl).
rw([117,[36,123]],0.5,rl).
rw([117,[36,124]],1.0,rl).
rw([117,[48,123]],0.3333333333333333,rl).
rw([117,[48,124]],0.5,rl).
rw([244],1.0,rl).
rw([225,[105]],1.0,rl).
rw([122,[191]],0.5,rl).
rw([122,[194]],0.3090169943749474,rl).
rw([132,[191]],0.5,rl).
rw([132,[194]],0.4142135623730951,rl).
rw([134,[191]],1.0,rl).
rw([134,[194]],0.4142135623730951,rl).
rw([144,[191]],0.5,rl).
rw([144,[194]],0.4142135623730951,rl).
rw([145,[191]],0.000999000999000999,rl).
rw([145,[194]],0.3090169943749474,rl).
rw([146,[191]],0.5,rl).
rw([146,[194]],0.4142135623730951,rl).
rw([164,[123]],0.5,rl).
rw([164,[124]],1.0,rl).
rw([175,[123]],0.5,rl).
rw([175,[124]],1.0,rl).
rw([192,[123]],0.5,rl).
rw([192,[124]],1.0,rl).
rw([193,[123]],0.5,rl).
rw([193,[124]],1.0,rl).
rw([196,[191]],0.5,rl).
rw([196,[194]],1.0,rl).
rw([97,[105,123]],0.5,rl).
rw([97,[105,124]],1.0,rl).
rw([106,[211]],0.5,rl).
rw([106,[226]],1.0,rl).
rw([114,[211]],0.5,rl).
rw([114,[226]],1.0,rl).
rw([245],1.0,rl).
rw([225,[107]],1.0,rl).
rw([122,[197]],0.5,rl).
rw([122,[199]],0.3090169943749474,rl).
```

```
rw([132,[197]],0.5,rl).
rw([132,[199]],0.4142135623730951,rl).
rw([134,[197]],1.0,rl).
rw([134,[199]],0.4142135623730951,rl).
rw([144,[197]],0.5,rl).
rw([144,[199]],1.0,rl).
rw([145,[197]],0.000999000999000999,rl).
rw([145,[199]],0.3090169943749474,rl).
rw([146,[197]],0.5,rl).
rw([146,[199]],0.4142135623730951,rl).
rw([165,[123]],0.5,rl).
rw([165,[124]],1.0,rl).
rw([176,[123]],0.5,rl).
rw([176,[124]],1.0,rl).
rw([196,[197]],0.5,rl).
rw([196,[199]],0.4142135623730951,rl).
rw([198,[123]],0.5,rl).
rw([198,[124]],1.0,rl).
rw([200,[123]],0.5,rl).
rw([200,[124]],1.0,rl).
rw([97,[107,123]],0.5,rl).
rw([97,[107,124]],1.0,rl).
rw([108,[211]],0.5,rl).
rw([108,[226]],1.0,rl).
rw([115,[211]],0.5,rl).
rw([115,[226]],1.0,rl).
rw([246],1.0,rl).
rw([225,[75]],0.5,rl).
rw([225,[87]],1.0,rl).
rw([122,[201]],0.5,rl).
rw([132,[201]],0.5,rl).
rw([134,[201]],1.0,rl).
rw([144,[201]],0.5,rl).
rw([145,[201]],0.000999000999000999,rl).
rw([146,[201]],0.5,rl).
rw([145,[133]],0.3333333333333333,rl).
rw([145,[143]],0.3090169943749474,rl).
rw([146,[133]],0.5,rl).
rw([146,[143]],0.4142135623730951,rl).
```

After learning the required templates and assigning confidence factors to them, translation process is straightforward. The following is a typical example where confidence factor assignment method outperforms the ordering according to the terminal method. Assume that the input sentence is *gelmezsin* (*you do not come*). The lexical level representation is *gel+mA+zsHn* and the translation results by using confidence factors are:

```
1==>you do not come  with probability: 1.0
 Rules used:[25,[124]]
2==>you do not come  with probability: 1.0
 Rules used:[205]
3==>you do not come  with probability: 1.0
 Rules used:[122,[2]]
4==>you do not come  with probability: 1.0
 Rules used:[134,[22]]
5==>you do not come  with probability: 1.0
 Rules used:[144,[24]]
6==>you do not come  with probability: 1.0
 Rules used:[4,[123]]
7==>you do not come  with probability: 1.0
 Rules used:[20,[124]]
8==>you do not come  with probability: 1.0
 Rules used:[21,[124]]
9==>you do not come  with probability: 1.0
 Rules used:[23,[124]]
10==>you do do not come  with probability: 0.5
 Rules used:[122,[22]]
11==>I you do not come  with probability: 0.5
 Rules used:[132,[22]]
12==>you you do not come  with probability: 0.5
 Rules used:[144,[22]]
13==>he you do not come  with probability: 0.5
 Rules used:[146,[22]]
14==>we you do not come  with probability: 0.5
 Rules used:[196,[22]]
15==>you do do not come  with probability: 0.5
 Rules used:[20,[123]]
16==>you do do not come  with probability: 0.5
 Rules used:[21,[123]]
17==>you do do not come  with probability: 0.5
 Rules used:[23,[123]]
18==>you do do not come  with probability: 0.5
```

```
 Rules used:[25,[123]]
19==>I do not come  with probability: 0.4142135623730951
 Rules used:[132,[24]]
20==>you not come  with probability: 0.4142135623730951
 Rules used:[134,[2]]
21==>do not come  with probability: 0.4142135623730951
 Rules used:[134,[24]]
22==>you you not come  with probability: 0.4142135623730951
 Rules used:[144,[2]]
23==>he do not come  with probability: 0.4142135623730951
 Rules used:[146,[24]]
24==>we do not come  with probability: 0.4142135623730951
 Rules used:[196,[24]]
25==>you not come  with probability: 0.4142135623730951
 Rules used:[4,[124]]
26==>do do not come  with probability: 0.3090169943749474
 Rules used:[122,[24]]
27==>I you not come  with probability: 0.3090169943749474
 Rules used:[132,[2]]28==>he do do not come  with probability: 0.3090169943749474
 Rules used:[145,[24]]
29==>he you not come  with probability: 0.3090169943749474
 Rules used:[146,[2]]
30==>we you not come  with probability: 0.3090169943749474
 Rules used:[196,[2]]
31==>he do you not come  with probability: 0.000999000999000999
 Rules used:[145,[2]]
32==>he do you do not come  with probability: 0.000999000999000999
 Rules used:[145,[22]]
```

The order of the outputs change when confidence factors are not used:

```
1==>you do do not come  with probability: 0.5
 Rules used:[25,[123]]
2==>you do not come  with probability: 1.0
 Rules used:[23,[124]]
3==>you do do not come  with probability: 0.5
 Rules used:[23,[123]]
4==>you do not come  with probability: 1.0
 Rules used:[21,[124]]
5==>you do do not come  with probability: 0.5
 Rules used:[21,[123]]
```

```
6==>you do not come  with probability: 1.0
 Rules used:[20,[124]]
7==>you do do not come  with probability: 0.5
 Rules used:[20,[123]]
8==>you not come  with probability: 0.4142135623730951
 Rules used:[4,[124]]
9==>you do not come  with probability: 1.0
 Rules used:[4,[123]]
10==>we do not come  with probability: 0.4142135623730951
 Rules used:[196,[24]]
11==>we you do not come  with probability: 0.5
 Rules used:[196,[22]]
12==>we you not come  with probability: 0.3090169943749474
 Rules used:[196,[2]]
13==>he do not come  with probability: 0.4142135623730951
 Rules used:[146,[24]]
14==>he you do not come  with probability: 0.5
 Rules used:[146,[22]]
15==>he you not come  with probability: 0.3090169943749474
 Rules used:[146,[2]]
16==>he do do not come  with probability: 0.3090169943749474
 Rules used:[145,[24]]
17==>he do you do not come  with probability: 0.000999000999000999
 Rules used:[145,[22]]
18==>he do you not come  with probability: 0.000999000999000999
 Rules used:[145,[2]]
19==>you do not come  with probability: 1.0
 Rules used:[144,[24]]
20==>you you do not come  with probability: 0.5
 Rules used:[144,[22]]
21==>you you not come  with probability: 0.4142135623730951
 Rules used:[144,[2]]
22==>do not come  with probability: 0.4142135623730951
 Rules used:[134,[24]]
23==>you do not come  with probability: 1.0
 Rules used:[134,[22]]
24==>you not come  with probability: 0.4142135623730951
 Rules used:[134,[2]]
25==>I do not come  with probability: 0.4142135623730951
 Rules used:[132,[24]]
```

```
26==>I you do not come  with probability: 0.5
 Rules used:[132,[22]]
27==>I you not come  with probability: 0.3090169943749474
 Rules used:[132,[2]]
28==>do do not come  with probability: 0.3090169943749474
 Rules used:[122,[24]]
29==>you do do not come  with probability: 0.5
 Rules used:[122,[22]]
30==>you do not come  with probability: 1.0
 Rules used:[122,[2]]
31==>you do not come  with probability: 1.0
 Rules used:[205]
32==>you do not come  with probability: 1.0
 Rules used:[25,[124]]
```

Translation of *gelmez* (*he does not come*, gel+mA+z) exhibits a similar behaviour as shown below:

```
1==>he does not come  with probability: 1.0
 Rules used:[34,[124]]
2==>does not come  with probability: 1.0
 Rules used:[214]
3==>he does not come  with probability: 1.0
 Rules used:[134,[30]]
4==>he does not come  with probability: 1.0
 Rules used:[145,[28]]
5==>he does not come  with probability: 1.0
 Rules used:[146,[32]]
6==>he does not come  with probability: 1.0
 Rules used:[26,[124]]
7==>he does not come  with probability: 1.0
 Rules used:[27,[124]]
8==>he does not come  with probability: 1.0
 Rules used:[31,[124]]
9==>he does do not come  with probability: 0.5
 Rules used:[122,[30]]
10==>I he does not come  with probability: 0.5
 Rules used:[132,[30]]
11==>you he does not come  with probability: 0.5
 Rules used:[144,[30]]
12==>he do does not come  with probability: 0.5
```

```
 Rules used:[145,[32]]
13==>he he does not come  with probability: 0.5
 Rules used:[146,[30]]
14==>we he does not come  with probability: 0.5
 Rules used:[196,[30]]
15==>he does do not come  with probability: 0.5
 Rules used:[26,[123]]
16==>he does do not come  with probability: 0.5
 Rules used:[27,[123]]
17==>he does do not come  with probability: 0.5
 Rules used:[31,[123]]
18==>he does do not come  with probability: 0.5
 Rules used:[34,[123]]
19==>I does not come  with probability: 0.4142135623730951
 Rules used:[132,[32]]
20==>does not come  with probability: 0.4142135623730951
 Rules used:[134,[32]]
21==>you does not come  with probability: 0.4142135623730951
 Rules used:[144,[32]]
 wrong answer  with probability: 0.4142135623730951
 Rules used for wrong answer:[146,[28]]
22==>we does not come  with probability: 0.4142135623730951
 Rules used:[196,[32]]
23==>he does not come  with probability: 0.3333333333333333
 Rules used:[206]
24==>he do he does not come  with probability: 0.3333333333333333
 Rules used:[145,[30]]
25==>does do not come  with probability: 0.3090169943749474
 Rules used:[122,[32]]
 wrong answer  with probability: 0.2612038749637414
 Rules used for wrong answer:[132,[28]]
wrong answer  with probability: 0.2612038749637414
 Rules used for wrong answer:[134,[28]]
 wrong answer  with probability: 0.2612038749637414
 Rules used for wrong answer:[144,[28]]
 wrong answer  with probability: 0.2612038749637414
 Rules used for wrong answer:[196,[28]]
 wrong answer  with probability: 0.000999000999000999
 Rules used for wrong answer:[122,[28]]
```

are the translations by using confidence factors, and translations without confidence factors are:

```
1==>he does do not come  with probability: 0.5
 Rules used:[34,[123]]
2==>he does not come  with probability: 1.0
 Rules used:[31,[124]]
3==>he does do not come  with probability: 0.5
 Rules used:[31,[123]]
4==>he does not come  with probability: 1.0
 Rules used:[27,[124]]
5==>he does do not come  with probability: 0.5
 Rules used:[27,[123]]
6==>he does not come  with probability: 1.0
 Rules used:[26,[124]]
7==>he does do not come  with probability: 0.5
 Rules used:[26,[123]]
8==>we does not come  with probability: 0.4142135623730951
 Rules used:[196,[32]]
9==>we he does not come  with probability: 0.5
 Rules used:[196,[30]]
 wrong answer  with probability: 0.2612038749637414
 Rules used for wrong answer:[196,[28]]
10==>he does not come  with probability: 1.0
 Rules used:[146,[32]]
11==>he he does not come  with probability: 0.5
 Rules used:[146,[30]]
 wrong answer  with probability: 0.4142135623730951
 Rules used for wrong answer:[146,[28]]
12==>he do does not come  with probability: 0.5
 Rules used:[145,[32]]
13==>he do he does not come  with probability: 0.3333333333333333
 Rules used:[145,[30]]
14==>he does not come  with probability: 1.0
 Rules used:[145,[28]]
15==>you does not come  with probability: 0.4142135623730951
 Rules used:[144,[32]]
16==>you he does not come  with probability: 0.5
 Rules used:[144,[30]]
 wrong answer  with probability: 0.2612038749637414
 Rules used for wrong answer:[144,[28]]
```

```
17==>does not come  with probability: 0.4142135623730951
 Rules used:[134,[32]]
18==>he does not come  with probability: 1.0
 Rules used:[134,[30]]
 wrong answer  with probability: 0.2612038749637414
 Rules used for wrong answer:[134,[28]]
19==>I does not come  with probability: 0.4142135623730951
 Rules used:[132,[32]]
20==>I he does not come  with probability: 0.5
 Rules used:[132,[30]]
 wrong answer  with probability: 0.2612038749637414
 Rules used for wrong answer:[132,[28]]
21==>does do not come  with probability: 0.3090169943749474
 Rules used:[122,[32]]
22==>he does do not come  with probability: 0.5
 Rules used:[122,[30]]
 wrong answer  with probability: 0.000999000999000999
 Rules used for wrong answer:[122,[28]]
23==>does not come  with probability: 1.0
 Rules used:[214]
24==>he does not come  with probability: 0.3333333333333333
 Rules used:[206]
25==>he does not come  with probability: 1.0
 Rules used:[34,[124]]
```

are the results without using confidence factors.

The input sentence *I will not come* reflects the effect of confidence factors in a significant way. The results by using confidence factors are:

```
ok21==>gelmeyeceGim  with probability: 1.0
 Rules used:[51,[208]]
2==>gelmeyeceGim  with probability: 1.0
 Rules used:[215]
3==>gelmeyeceGim  with probability: 1.0
 Rules used:[148,[12]]
4==>gelmeyeceGim  with probability: 1.0
 Rules used:[207,[25]]
5==>gelmeyeceGim  with probability: 1.0
 Rules used:[59,[72]]
6==>gelmeyeceGim  with probability: 1.0
 Rules used:[74,[54]]
```

```
7==>gelmeyeceGim  with probability: 1.0
 Rules used:[80,[54]]
8==>gelmeyeceGim  with probability: 1.0
 Rules used:[122,[54]]
9==>gelmeyeceGim  with probability: 1.0
 Rules used:[128,[54]]
10==>gelmeyeceGim  with probability: 1.0
 Rules used:[26,[25,54]]
11==>gelmeyeceGim  with probability: 1.0
 Rules used:[50,[208]]
12==> gelmeyeceGim  with probability: 0.5
 Rules used:[84,[1,12]]
13==>m gelmeyeceGim  with probability: 0.5
 Rules used:[84,[34,12]]
14==>gelmeyeceksin  with probability: 0.4142135623730951
 Rules used:[148,[14]]
15==>gelmeyecek  with probability: 0.4142135623730951
 Rules used:[148,[17]]
16==>gelmeyeceGiz  with probability: 0.4142135623730951
 Rules used:[148,[49]]
17==>gelmeyeceGim  with probability: 0.4142135623730951
 Rules used:[207,[1]]
18==>gelmeyecekm  with probability: 0.4142135623730951
 Rules used:[207,[34]]
19==>gelmeyeceGim  with probability: 0.4142135623730951
 Rules used:[3,[208]]
20==>gelmeyeceGim  with probability: 0.4142135623730951
 Rules used:[26,[1,54]]
21==>gelmeyecekm  with probability: 0.4142135623730951
 Rules used:[26,[34,54]]
22==>gelmeyecekm  with probability: 0.4142135623730951
 Rules used:[36,[208]]
23==>gelmeyecekm  with probability: 0.4142135623730951
 Rules used:[38,[208]]
24==> gelmeyecek  with probability: 0.3090169943749474
 Rules used:[209,[1]]
25==>m gelmeyecek  with probability: 0.3090169943749474
 Rules used:[209,[34]]
26==> gelmeyeceksin  with probability: 0.3090169943749474
 Rules used:[84,[1,14]]
```

```
27==> gelmeyecek  with probability: 0.3090169943749474
 Rules used:[84,[1,17]]
28==> gelmeyeceGiz  with probability: 0.3090169943749474
 Rules used:[84,[1,49]]
29==>m gelmeyeceksin  with probability: 0.3090169943749474
 Rules used:[84,[34,14]]
30==>m gelmeyecek  with probability: 0.3090169943749474
 Rules used:[84,[34,17]] 31==>m gelmeyeceGiz  with probability: 0.3090169943749474
 Rules used:[84,[34,49]]
32==> gelmeyecek  with probability: 0.3090169943749474
 Rules used:[24,[1,54]]
33==>m gelmeyecek  with probability: 0.3090169943749474
 Rules used:[24,[34,54]]
34==> gelmeyecek  with probability: 0.3090169943749474
 Rules used:[33,[1,54]]
35==>m gelmeyecek  with probability: 0.3090169943749474
 Rules used:[33,[34,54]]
36==> gelmeyecek  with probability: 0.000999000999000999
 Rules used:[209,[25]]
37==> gelmeyeceGim  with probability: 0.000999000999000999
 Rules used:[84,[25,12]]
38==> gelmeyeceksin  with probability: 0.000999000999000999
 Rules used:[84,[25,14]]
39==> gelmeyecek  with probability: 0.000999000999000999
 Rules used:[84,[25,17]]
40==> gelmeyeceGiz  with probability: 0.000999000999000999
 Rules used:[84,[25,49]]
41==> gelmeyecek  with probability: 0.000999000999000999
 Rules used:[24,[25,54]]
42==> gelmeyecek  with probability: 0.000999000999000999
 Rules used:[33,[25,54]]
```

The results without confidence factors are:

```
1==>gelmeyeceGim  with probability: 1.0
 Rules used:[50,[208]]
2==>gelmeyecekm  with probability: 0.4142135623730951
 Rules used:[38,[208]]
3==>gelmeyecekm  with probability: 0.4142135623730951
 Rules used:[36,[208]]
```

```
4==>m gelmeyecek  with probability: 0.3090169943749474
 Rules used:[33,[34,54]]
5==> gelmeyecek  with probability: 0.000999000999000999
 Rules used:[33,[25,54]]
6==> gelmeyecek  with probability: 0.3090169943749474
 Rules used:[33,[1,54]]
7==>gelmeyecekm  with probability: 0.4142135623730951
 Rules used:[26,[34,54]]
8==>gelmeyeceGim  with probability: 1.0
 Rules used:[26,[25,54]]
9==>gelmeyeceGim  with probability: 0.4142135623730951
 Rules used:[26,[1,54]]
10==>m gelmeyecek  with probability: 0.3090169943749474
 Rules used:[24,[34,54]]
11==> gelmeyecek  with probability: 0.000999000999000999
 Rules used:[24,[25,54]]
12==> gelmeyecek  with probability: 0.3090169943749474
 Rules used:[24,[1,54]]
13==>gelmeyeceGim  with probability: 0.4142135623730951
 Rules used:[3,[208]]
14==>gelmeyeceGim  with probability: 1.0
 Rules used:[128,[54]]
15==>gelmeyeceGim  with probability: 1.0
 Rules used:[122,[54]]
16==>m gelmeyeceGiz  with probability: 0.3090169943749474
 Rules used:[84,[34,49]]
17==>m gelmeyecek  with probability: 0.3090169943749474
 Rules used:[84,[34,17]]
18==>m gelmeyeceksin  with probability: 0.3090169943749474
 Rules used:[84,[34,14]]
19==>m gelmeyeceGim  with probability: 0.5
 Rules used:[84,[34,12]]
20==> gelmeyeceGiz  with probability: 0.000999000999000999
 Rules used:[84,[25,49]]
21==> gelmeyecek  with probability: 0.000999000999000999
 Rules used:[84,[25,17]]
22==> gelmeyeceksin  with probability: 0.000999000999000999
 Rules used:[84,[25,14]]
23==> gelmeyeceGim  with probability: 0.000999000999000999
 Rules used:[84,[25,12]]
```

```
24==> gelmeyeceGiz  with probability: 0.3090169943749474
 Rules used:[84,[1,49]]
25==> gelmeyecek  with probability: 0.3090169943749474
 Rules used:[84,[1,17]]
26==> gelmeyeceksin  with probability: 0.3090169943749474
 Rules used:[84,[1,14]]
27==> gelmeyeceGim  with probability: 0.5
 Rules used:[84,[1,12]]
28==>gelmeyeceGim  with probability: 1.0
 Rules used:[80,[54]]
29==>gelmeyeceGim  with probability: 1.0
 Rules used:[74,[54]]
30==>gelmeyeceGim  with probability: 1.0
 Rules used:[59,[72]]31==>m gelmeyecek  with probability: 0.3090169943749474
 Rules used:[209,[34]]
32==> gelmeyecek  with probability: 0.000999000999000999
 Rules used:[209,[25]]
33==> gelmeyecek  with probability: 0.3090169943749474
 Rules used:[209,[1]]
34==>gelmeyecekm  with probability: 0.4142135623730951
 Rules used:[207,[34]]
35==>gelmeyeceGim  with probability: 1.0
 Rules used:[207,[25]]
36==>gelmeyeceGim  with probability: 0.4142135623730951
 Rules used:[207,[1]]
37==>gelmeyeceGiz  with probability: 0.4142135623730951
 Rules used:[148,[49]]
38==>gelmeyecek  with probability: 0.4142135623730951
 Rules used:[148,[17]]
39==>gelmeyeceksin  with probability: 0.4142135623730951
 Rules used:[148,[14]]
40==>gelmeyeceGim  with probability: 1.0
 Rules used:[148,[12]]
41==>gelmeyeceGim  with probability: 1.0
 Rules used:[215]
42==>gelmeyeceGim  with probability: 1.0
 Rules used:[51,[208]]
```

Although, the number of correct translation remains the same for translations that use or does not use confidence factors, the translation results of the above examples indicate that the correct translations are more probable in the top results when we used the confidence factors.

In most cases, translation by using confidence factors outperforms the translation without confidence factors.