CIYA SOFT
CORPORATION

CiyaTran™ Machine Translation Whitepaper

By
Ciyasoft Technical team
1/28/2002

Revisions History:
9/15/2002
4/8/2005
10/10/2008
10/11/2010
7/4/2012
7/1/2015
9/8/2017
10/9/2017

CiyaTran™

CiyaSoft Corporation designs and develops Integrated Information Management Software Tools (IMST) for languages such as Farsi, Pashto, Dari and Arabic, as well as Machine Translation (MT) for these languages. IMST includes:

- Data Gathering (Spidering)
- File Format Conversion
- Data Classification
- Statistical Analysis of Text
- Summarization of Text
- Concept-Based Data Searching

The objective of this document is to describe Ciyasoft unique approach to MT and how it interacts with IMST. Description of different components of IMST is presented in other documents.

While this document does not emphasize on what MT is, or its limitations and its applications, focus is placed on presenting Ciyasoft research and the resulting techniques developed to solve the problems associated with MT designs. These techniques are exemplified by using English and Farsi as source and target languages. It is assumed that the reader does not know Farsi, so there are no Farsi examples. However, the examples presented equally apply to all supported languages.

Implementation details and technical information may have been intentionally omitted in some sections of this document to preserve our competitive position, while patents are pending.

Machine Translation: Definition

Machine translation (**MT**) means translation, using computers. In its broadest sense, MT can be understood to include such computer applications as compilers and compression programs, which convert a file in one computer language into a file in another computer language. However, what we are interested in here is *natural language processing* (**NLP**). One thing that MT does not mean, but which is sometimes confused with MT, is *automatic speech recognition*.

There are four basic types of translation, three of which are types of machine translation or machine-assisted translation:

Human Translation: A human translator performs all the steps in the translation process, using a computer only as a word processor.

Machine-assisted human translation (**MAHT**): The translation is performed by a human translator, but she uses the computer as a tool to improve or speed up the translation process. This is called *computer-assisted translation* (**CAT**) by people in the field of translation as opposed to the field of MT.

Human-assisted machine translation (**HAMT**): The source language (**SL**) text is modified by a human translator either before, during, or after it is translated by the computer.

Fully automatic machine translation (**FAMT**): The SL text is fed into the computer as a file, and the computer produces a translation automatically without any human intervention. This is sometimes referred to as ***batch mode***. There are two types of fully automatic machine translation: *fully automatic high-quality machine translation* (**FAHQMT**) and low-quality machine translation.

The type of MT, which people think of when they hear the word machine translation, is usually the last type (fully automatic MT). Obviously, the ultimate goal of MT is FAMT, although its results have also made it the subject of much ridicule, sometimes with good reason. Note that the distinction between HAMT and FAMT is partly conventional, since a FAMT system would be considered a HAMT system if the output is post-edited, and any translation can be checked.

MT is a part of the field of knowledge called *artificial intelligence* (**AI**). There are many different definitions about what artificial intelligence is, however, a simple way of thinking of it is the attempt to emulate human patterns of thinking and behavior using computer models. Today AI is used in robotics, pattern recognition, expert systems, simulation, as well as in MT.

Why Machine Translation?

In the last decades of the 20th century, a process began that is often referred to as globalization – the disappearance of political blocks, the fall of trade barriers and the advent of the Internet, which fosters information exchange all over the planet. Due to this process, the need to communicate with persons speaking languages other than one's own is increasing dramatically. With the rapid economic expansion in the Far East, especially in China, the role of English as a de facto lingua franca will probably diminish. At the beginning of the 1990s almost all Web content was in English. In the year 2000, its share has dropped to about 68% (http://www.emarketer.com/analysis/edemographics/20010227_edemo.html). This process is very likely to continue, and the importance of translation will rise with the ongoing linguistic diversification of the Internet.

Multinational organizations like the United Nations or the European Union (EU) produce hundreds of thousands of text pages every year. They have to employ small armies of translators and interpreters; the EU spends an estimated 40-45% of its administrative budget on translation services. Every company operating on an international level has the problem of having to translate documentation and user manuals into the world's major languages. Companies offering Internet content operate on a global scale and want to offer their services worldwide. There is a strong demand for translation services, and it is growing.

Human translation, however, has two serious downsides: it is slow and expensive. An average human translator translates about 2000 words (about 8 pages) a day. If some of this huge amount of text to be translated is processed automatically or semi-automatically, the savings in time and money can be enormous.

Apart from social and economic reasons, there are also scientific motivations. Machine Translation makes it necessary to describe language in a machine-understandable, mathematical way. This formalization makes it possible to apply and test new theories in all related disciplines: linguistics, translation theory, language philosophy, artificial intelligence (AI), computational linguistics and computer science.

History

Apparently, the first suggestions concerning MT were made by the Russian Smirnov-Troyansky and the Frenchman G.B. Artsouni in the 1930's. However, the first serious discussions were begun in 1946 by the mathematician Warren Weaver. He and many others were inspired by the success of the Allied efforts using the British Colossus computer to break the German military code produced by the Enigma machine, and the obvious similarity between the task of decoding an encoded message and the task of translation of one language into another. By 1954 there was an MT project at Georgetown University which succeeded in correctly translating several sentences from Russian into English. Soon there were MT projects at MIT, Harvard, and the University of Pennsylvania.

In 1964, after more than $20 million had been invested by the Federal Government in MT, the National Academy of Sciences commissioned the Automatic Language Processing Advisory Committee (**ALPAC**) to write a study of the status of MT. The committee, headed by John R. Pierce, wrote a now-famous report in which it expressed doubt that a fully automatic MT system could ever be produced. That report sounded the death-knell for funding of MT research, and MT was neglected for many years afterwards.

The reasons for this failure have been described many times, and come down to the fact that the analysis by humans of messages in natural language relies to some extent on information which is not present in the words which make up the message. This led the linguist Yehoshua Bar-Hillel to declare that MT was impossible. The example which he provided has since become a classic, and is now called the *Bar-Hillel paradox*:

The pen is in the box.
[i.e. the writing instrument is in the container]
The box is in the pen.
[i.e. the container is in the playpen or the pigpen]

There are two possible ways that a person could correctly infer the meaning of these sentences. First, if there is a context preceding these sentences, it could make clear which meaning of *pen* is being used in which sentence. That is, the meaning of the words and information about the context is carried over from one sentence to the next. There is now an entire branch of linguistics, called *discourse* *analysis*, devoted to the study of how context affects the meaning of words and sentences. In order to infer in this way, the correct meaning of an ambiguous sentence, computers will have to learn how to "remember" a context and make use of it to interpret the correct meaning of words and sentences within that context.

However, in the examples given above, most humans can understand the meaning correctly *without* any context. For a fully automatic MT system to translate these sentences correctly, the following information would have to be available to the computer:

pens [writing instruments] are smaller than boxes
boxes are bigger than pens [writing instruments],
  but smaller than pens [playpens, pigpens, etc.]
it is impossible for a bigger object to be inside
  a smaller object

Thus, one way or the other, whether the correct meaning of the sentences is inferred based on the context or in isolation, it is necessary for the computer to have information at its disposal, which is not included in the message itself. During the early days of MT this realization was enough to make MT seem an impossible task.

Interest in MT revived in the 1980's, following dramatic advances in computer hardware (storage capacity, speed, etc.) and software (LISP, etc.). The need to store and process tremendous amounts of real-world knowledge in order to analyze a single word in the message ceased to be an impediment to design and use of MT systems. The following table shows the evolution of MT since 1933:

Year	Event
1933	Russian Petr Smirnov- Troyanskii patents a device for transforming word-root sequences into their other-language equivalents.
1939	Bell Labs demonstrates the first electronic speech-synthesizing device at the New York World's Fair.
1949	Warren Weaver, director of the Rockefeller Foundation's natural sciences division, drafts a memorandum for peer review outlining the prospects of machine translation (MT).
1952	Yehoshua Bar-Hillel, MIT's first full-time MT researcher, organizes the

	maiden MT conference.
1954	First public demo of computer translation at Georgetown University: 49 Russian sentences are translated into English using a 250-word vocabulary and 6 grammar rules.
1960	Bar-Hillel publishes his report arguing that fully automatic and accurate translation systems are, in principle, impossible.
1964	The National Academy of Sciences creates the Automatic Language Processing Advisory Committee (Alpac) to study MT's feasibility.
1966	Alpac publishes a report on MT concluding that years of research haven't produced useful results. The outcome is a halt in federal funding for machine translation R&D.
1967	L. E. Baum and colleagues at the Institute for Defense Analyses (IDA) in Princeton, New Jersey, develop hidden Markov models, the mathematical backbone of continuous-speech recognition.
1968	Peter Toma, a former Georgetown University linguist, starts one of the first MT companies, Language Automated Translation System and Electronic Communications (Latsec).
1969	In Middletown, New York, Charles Byrne and Bernard Scott found Logos to develop MT systems.
1978	Arpa's Network Speech Compression (NSC) project transmits the first spoken words over the Internet.
1982	Janet and Jim Baker found Newton, Massachusetts-based Dragon Systems.
1983	The Automated Language Processing System (ALPS) is the first MT software for a microcomputer.
1985	Darpa launches its speech recognition program.
1986	Japan launches the ATR Interpreting Telecommunication Research Laboratories (ATR-ITL) to study multilingual speech translation.
1987	In Belgium, Jo Lernout and Pol Hauspie found Lernout & Hauspie.
1988	Researchers at IBM's Thomas J. Watson Research Center revive statistical MT methods that equate parallel texts, then calculate the probabilities that words in one version will correspond to words in another.
1990	TRC (TechnoResearch) introduces OmniTran, the first Farsi, Dari machine translation
1991	The first translator-dedicated workstations appear, including STAR's Transit, IBM's TranslationManager, Canadian Translation Services' PTT, and Eurolang's Optimizer.
1992	ATR-ITL founds the Consortium for Speech Translation Advanced Research

	(C-STAR), which gives the first public demo of phone translation between English, German, and Japanese.
1993	The German-funded Verbmobil project gets under way. Researchers focus on portable systems for face-to-face English-language business negotiations in German and Japanese.
1994	Systran machine translation is available in select CompuServe chat forums.
1997	AltaVista's Babel Fish offers real-time Systran translation on the Web.
1999	A televised newscast is automatically transcribed with 85 percent accuracy. Logos releases e.Sense Enterprise Translation, the first Web-enabled multiple translator operating from a single server. IBM releases ViaVoice for the Macintosh, the first continuous-speech-recognition Mac software.
2000	At MIT's Lincoln Laboratory, Young-Suk Lee and Clifford Weinstein demonstrate an advanced Korean-English speech-to-speech translation-system prototype. USC's ISI performs backward machine-transliterations of proper nouns, which are replaced with phonetic approximations. *Southern California* translates to "Janoub Kalyfornya" in Arabic.
2001	Carnegie Mellon University's Language Technologies Institute (LTI), led by Jaime Carbonell, constructs speech-to-speech translation for "small" languages like Croatian or Mapudungun, spoken by Mapuches in Chile.
2002	NowHear offers an agent-based newsreader device that translates articles from thousands of publications worldwide, delivering them as MP3 audio files.

Theory

There are several basic theoretical approaches to MT.

The most primitive is called the ***direct MT strategy***. This approach is always between pairs of languages and is based on good glossaries and morphological analysis.

The next most advanced system is called the ***transfer MT strategy***. This is still being used today, although it has a competitor (see below). First, the SL is parsed into an abstract internal representation. Thereafter, a 'transfer' is made into the corresponding structures in the target language (TL). Then a translation is generated. This approach is more advanced theoretically, but also translates between specific pairs of languages. Both the direct MT strategy and the transfer

MT strategy can take advantage of similarities between languages. The direct MT strategy has been criticized as theoretically inelegant, although it probably comes closest to modeling how human translators work.

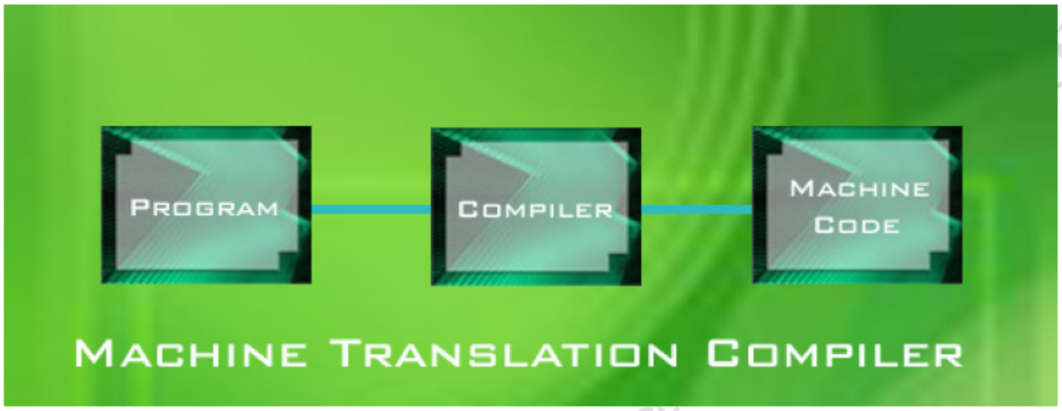
Both the direct MT strategy and the transfer MT strategy have been criticized since they require that separate translation software be written for every language combination. (Actually, from the point of view of someone with experience in the translation industry, these concerns seem trivial, since an overwhelming majority of translation work is done in less than a dozen language combinations: Ten or so packages could be written for these combinations, and other combinations could continue to be translated manually.)

The most advanced system is called the ***Medium MT Strategy*** (or ***interlingua MT strategy)***. The idea behind this approach is to create an artificial language, known as the ***interlingua***, which shares all the features and makes all the distinctions of all languages. To translate between two different languages, an ***analyzer*** is used to put the SL into the interlingua, and a ***generator*** converts the interlingua into the TL. The proponents of this system argue that it reduces the number of analyzers and generators which are required, since only one generator and one analyzer is required for each language, no matter how many other languages there are. While this is true, the proponents of the interlingua MT strategy probably underestimate how complex an interlingua would have to be in order to work as an intermediary among many or even a few unrelated languages, as opposed to among the few related European languages on which most work has been done. If the interlingua is extremely complex, this also means that the analyzers and generators will have to be extremely complex. It is hard to avoid thinking that this approach was inspired by the idea of a universal language such as *Esperanto*, which has excited certain linguists for centuries.

Artificial Languages vs. Natural Languages

A computer high-level language, such as C, is an artificial language. A program written in C is translated into machine code by a compiler. Speaking in general terms, C is defined by a limited group of words and symbols, and a fixed set of rules that define the ways in which these words and symbols can be put next to each other. A program in C can only contain this limited list of words and symbols and has to strictly follow the (grammar) rules of the language. A compiler design is, therefore, a straightforward task, because any deviation from the rules, or presence of an unrecognized word would constitute an error, resulting in compilation to stop.

Similarly, natural languages consist of a vocabulary (words) and a grammar (rules). Punctuations are equivalent of symbols in this analogy.



MT/Compiler Analogy

Natural languages, in contrast to programming languages (and artificial planned languages such as Esperanto), have more exceptions than rules. A programming language is "static". The words and rules do not change for a given compiler. But a natural language is "dynamic" in the sense that new words and phrases are being created, and patterns (new variations of rules) are being added to the body of the source and target languages by millions of writers and speakers every day. This description may make a useful MT implementation seem an impossible task. The solutions to the problems caused by this dynamism will be discussed.

Machine Translation

Machine Translation tries to automatically convert a piece of knowledge from one language into another. In this way, MT encompasses the full range of human knowledge expressed in writing and verbally. This automation is an extension of automation of thinking which is the core of artificial intelligence (AI). CiyaTran^{TM} is CiyaSoft's implementation of MT.

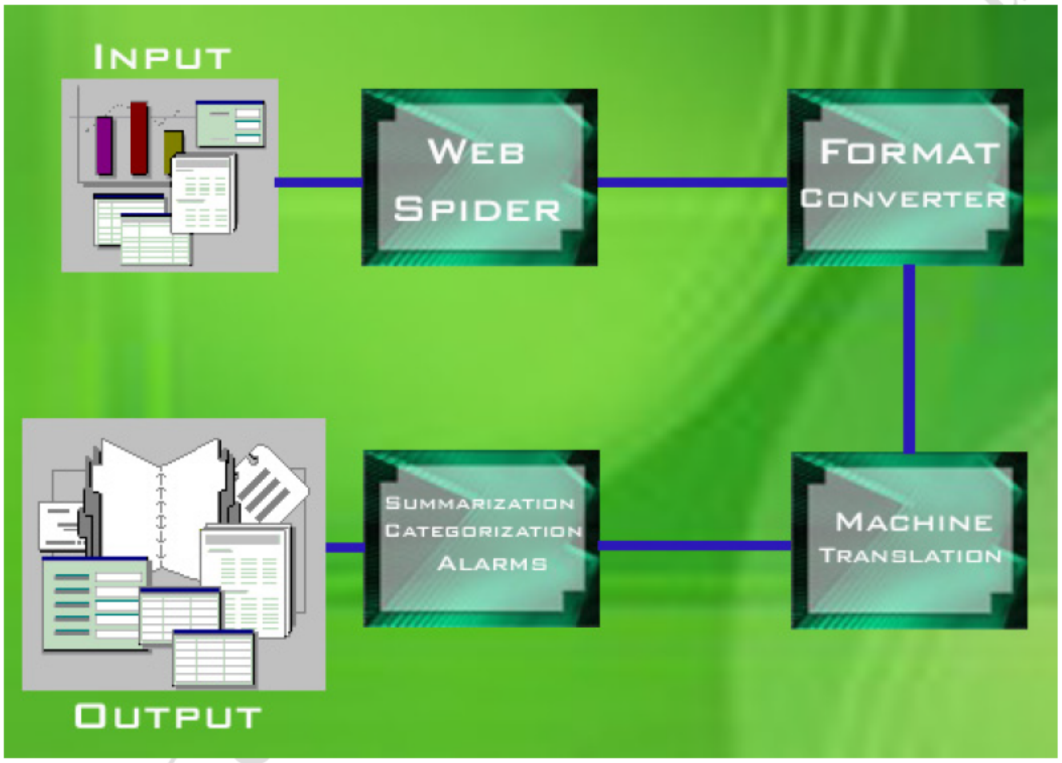
Integrated CiyaTran™ and IMST

CiyaTran output can be used by IMST for the purpose of "understanding" the input text. This automated understating is rendered in form of histograms, charts, graphs, summarized text or setting up of an alarm when a certain subject area, idea or a writing style of a person is detected in input text for which (or whom) IMST is configured to be sensitive about.

Integrated CiyaTran and IMST have several uses aside from translation applications. For the law enforcement and intelligence community, for example, it can be set to automatically and quickly go through thousands of pieces of electronic texts and mark items which should be investigated further. The integrated system may be configured to be sensitive to fragments of texts from certain sources or written by certain authors. Examples are Koran and other holy books, books or speeches or letters written by known terrorists. It is possible to determine if a given input text contains fragments from certain books (such as those quoted by terrorists), or if it contains certain subject matters (such as bomb making), or if it follows the style of writing or speaking of a certain individual, provided sufficient sample texts from the individual are available. In one recent experiment, we took all press briefing of the White House Press Secretary that are available in the White House WEB site and fed it through DGU (DGU will be explained later.) Then we gave the IMST two documents, one from a more recent news brief by the Press Secretary not yet processed by DGU and another, a speech by someone else picked up randomly. Then we set IMST to be sensitive about the Press Secretary, which IMST gave a high probability that the document was made by the Press Secretary, and the second document had a very low probability. Many other experiments by CiyaSoft have proved this hypothesis.

Just as it is possible to recognize a person from his handwriting, it is possible to detect similarities in writings of a person with his other writings. A person writes and speaks uniquely. To speak or write differently, a person has to make a special effort, as difficult as trying to change one's handwriting, because people tend to use the same group of words, phrases and even sentences they have used throughout their lives, for their education, the unique way they think and verbalize, the particular liking or taste they have developed for given words or phrases, and many other psychological reasons. It is possible, for example, to determine the education level of a writer from his writing. It is also possible to tell if he/she is a physician, an engineer or an artist. We have determined that even the state of mind can be determined. Of course, this kind of profiling takes place by forensic scientists routinely, but what we are proposing is the use of machine, because the data to process is prohibitively voluminous.

For reasons that are specific to the Eastern cultures, this detection can be done even more easily for the languages we support. The result of extensive research on this subject is described in detail in another CiyaSoft document.



Integrated CiyaTran^{TM} and IMST

Peculiarities of non-Latin languages

CiyaSoft CiyaTran Machine Translation System deals with peculiarities of the languages it supports. The peculiarities of Farsi/Dari, for example, facilitate certain aspects of MT problems; yet prevent full utilization of other techniques that can be applied to, for example, Latin languages. It is important to note that CiyaTran aims to automate, not some, but all the processes of translating to, and from English. Some of the reasons are:

- Absence of a single encoding standard, i.e. character set (ASCII equivalents, Unicode and non-standard codes)
- Absence of capitalization, and presence of special requirement whereby the shape of a character should change depending on the characters that comes before or after it
- Absence of a standard for word separation (using space between words is not a requirement)
- Absence of rigid rules for punctuation, and variation of meaning of punctuation among various writers or publishers
- Absence of rigid rules for morphology and even syntax
- Presence of incorrectly used words (Using words like "irregardless" or "theirselves", for example, when the source is English. This kind of abuse is a lot more prevalent in Farsi/Dari text.)
- Variety of input formats (PDF, HTML, RTF, ...)
- Variations in font type, size and style

CiyaSoft's MT

CiyaTran MT System takes full use of the latest advances and technique of AI, Linguistics, Computational Linguistics and Mathematical Combinatorics, wherever needed, but its strength is in utilizing a set of unique approaches researched and developed by CiyaSoft.
CiyaTran consists of three main components: The Database, the Translation Engine and the Interface. In the course of development, several other programs have been created to aid research, simulation, testing and building of databases. More time and resources have been spent on these utilities than on the Engine. These components and utilities are described later in this document and in detail in other CiyaSoft documents.

Man-Year Requirements to Increase Accuracy

Software design for MT is not like other (non-AI) applications. It is not exact and, theoretically, it can never be. An initial accuracy of about 50% for any language pair can be attained fairly quickly by taking advantage of well-established AI algorithms and Natural Language Processing (NLP) techniques. But the quick initial success can be misleading. If reaching the 50% level takes 5 man-years, for example, getting another 10% will not take one year. This is because:

- There are known problems for which there are no solutions.
- Not all problems are known. The system must be tested against many text samples and the translation manually examined to identify unknown problems to find solutions.
- The source and target languages are dynamic in both vocabulary and rules.
- Addition of a single new algorithm to the Engine, or even a new word or pattern to the database, could negatively affect the overall system. This is because the ambiguities and exceptions are interrelated in natural languages and solution for one could invalidate a solution for the other.

Time needed to increase accuracy is an exponential-like function. However, CiyaTran MT design has solutions for each of these issues reducing the exponential rate to a quadratic rate.

Accuracy in MT

Evaluation of MT is always connected to percentage of accuracy. Except for situations of 100% accuracy, any other figure is average or statistical. Accuracy has a definition when something is measured in the arena of Physical Sciences. When we try to find the speed of an aircraft, or the resistance of a resistor, we often give a range of +/- percentage, of error. In MT, we may say 50% accuracy in a given situation, but we could get 4%, or 98% for a given sentence, however, the average would be 50%. The measured accuracy can be compared to the normal distribution curve. If the curve is fatter, the accuracy is closer to 100, and vice versa.

Generalization vs. Statistical Approach

The inexactness of the definition of the problem, and the uncertainty of the solution is a characteristic of most AI-related software. To demonstrate this, suppose it is intended to design a circuit simulation software, the requirements of which can be precisely defined. Such software can be written hundreds of different ways. But no matter what approach is taken, who the programmer is, and what techniques are applied, the software should be able to provide identical output for a given input. In contrast, MT for a given language-pair is not (and cannot be) precisely and exhaustively defined. Therefore, its operation can be evaluated only statistically.

Many researchers tried to provide solutions for various aspects of MT. Thousands of papers have been written that discuss different problems in various settings and offer solutions in general terms. Almost always, upon implementation of a solution, we are faced with exceptions. The intent of any research project in a given area of MT is to find similarities to formulate a solution. By definition, exceptions cannot be formulated. The solution aims to find the generality, to the extent of attempting to model it mathematically. But exceptions cannot be modeled; they can only be studied statistically. This is the gist of our approach: Our research shows that an MT system would be more successful if the emphasis is placed on statistical methods such as Fuzzy Logic, rather than precise methods like mathematical modeling.

Modeling techniques, pattern recognition and neural networks are extensively employed by CiyaTran to translate well-behaved sentences. But Fuzzy Logic and heuristic and statistical approaches are used to deal with exceptions. In essence, we have tried to get closer to the way human brain deals with the ambiguities and exceptions of natural languages.

Database: The Starting Point

More than twenty-five years ago, we started working on English-to-Farsi/Dari MT. A practical implementation requires an elaborate design, based on research. So, we started with reviewing all the research papers already done on NLP. We looked at the problem from the point of view in which a person, who already knows one language, tries to learn a new language. One approach is to see how a child learns a language. But we assumed that the computer already would not learn like a child. If a person wants to learn a new language, he would start with leaning the basic structures, i.e. the grammar, and the words. Here is a summary of the steps we have taken that has brought us to this point: We wanted to initially design a simple machine translator from English to Farsi. We needed an English-to-Farsi database and a pair of grammar rules. We looked at the simple approach of word-for-word substitution and reordering the substitutions based on grammar rules of the target language. This simple approach would render 100 percent accuracy, if there were a word-to-word relationship between the two languages, and if there were no exceptions.

As we later learned, an MT design could not be based upon such delightfully easy methods. Any translator knows that word-to-word method doesn't produce a practical translation. Translation requires a good knowledge of the vocabulary and grammar of both languages. It also requires a general knowledge about the subject matter of the text in hand. In any case, a database is needed. So, we started with designing the database.

One approach to design the database was to enter the contents of an English-to-Farsi dictionary. We did not take that approach, because we determined that a dictionary contains many words that may not be needed in the initial stages of engine design. Also, a dictionary lacks many words and phrases that have entered the language after the dictionary has been published; and those are the more important words, because they are newer. We determined that over 80% of the contents of available dictionaries are used in less than 5% of Random Contemporary (RC) text. (By "random contemporary" text, we mean text from such sources as books, newspapers instruction manuals, research papers, mail, movie dialogues and speeches that are electronically available, are not duplicates, are randomly selected and are created no more than a hundred years ago.) The words that were useful to us were, of course, mixed up with other entries in the dictionary. There was no easy way to separate them. To solve this problem, we did a statistical analysis on a large sample of English RC text that were input or were available in electronic format. The analysis showed that about 50,000 words had the highest frequency of usage. We entered the meaning for those 50,000 words from available dictionaries. (The size of the database is now more than 2 million.) In entering the meanings, we had to follow very strict guidelines because a database used for MT is not a dictionary. The meanings, particularly for nouns, should be in a format to be readily usable within a translation. Where an English-to-English dictionary may sometimes play the role of an encyclopedia, a dictionary from English to another language is even more encyclopedic. This is because, in many cases, the equivalent of a word is not always a word, rather, the meaning is a definition. For example, the word "cassava" is defined in dictionary as "a tropical plant cultivated for its starchy roots". Since this plant does not grow in Farsi-speaking countries, it may not have a Farsi equivalent.

There are hundreds of other guidelines that must be considered during data entry and review. In addition to meanings, each entry included such other characteristics as grammatical state (GS), order, confidence, context and person. (Currently, each entry includes more than 50 characteristics.)

GT of an entry plays a central role in translation and helps identify other elements of the sentence. It should not be confused with part of speech (POS). Although similar, GS includes only those of POS categories that are meaningful in an MT application. In addition, they are redefined to suit MT requirements.

Grammatical States used by English-to-Farsi CiyaTran:

- singular noun
- regular plural noun
- irregular plural noun
- phrase
- countable
- transitive verb
- intransitive verb
- past verb
- past participle verb
- compound verb needing object
- compound verb without object
- adverb
- forward adjectives
- backward adjective
- regular adjectives
- passive adjective
- comparative adjective
- superlative adjective
- DNT (do not translate)
- noun substitution
- possessive substitution
- interrogatory
- separator
- starter
- number

- number word
- pure pronoun
- title
- colloquial
- old English
- foreign
- preposition
- negative

Returning to the simple word-for-word substitution approach, we found, after completion of the initial 50,000-word database, that only 27% of the words are one-to-one for this language pair, and 43% of them lose their one-to-oneness due to presence of phrases containing the one-to-one word. (Note: This percentage is higher for pairs of Latin languages.) The rate still holds today, even though the database is now 50 times larger. The one-to-one substitution approach, i.e. literal translation could yield a general understating of the input text, but what makes it impractical is the fact that a high percentage of the words used in a given RC text come from the non-one-to-one group. Therefore, for majority of RC text, the simple literal translation Engine does not know which meaning to choose for substitution. Nevertheless, if the output is only to be used by IMST, moderately practical results are achievable.

We also examined the exhaustive approach, which is a frequently-used technique for solving AI problems. The exhaustive approach requires that we have Farsi translation for all possible sentences in English. With ever-decreasing price of storage and increase in CPU speed, this approach was worth investigating.

We had to generate all possible sentences and have some way of automatically generating translations. From a combinatorics point of view, this is an impossible task. This is because, the number of unique and correct English sentences is a large number, so large that it is more than the total number of particles in the entire known universe. But, if a restriction is placed on the maximum number of words per sentence, this number is finite. So, our implementation had to impose restrictions. We wanted to see if we could design a heavily database-dependent and slightly engine-defendant MT system.

Let us see how feasible this would be: How many Simple Positive Present Tense sentences can be generated with a subject and a verb? Suppose the subject is any one of (I, you, he, she, it, this, that, we, they, these and those) and the verb is one of the 13,000 simple verbs. (Simple verbs constitute less than 1% of the CiyaTran database.) The answer is 11 x 13,000=143,000. This requires only 9.6MB of storage. (The average size of the sentence and its translation for this experiment would be 70 characters long.) A small hard disk and a smart binary search would do the job. We then expanded the database further: To add objects to every sentence (me, you, him, her, it, this, that, us, them, these, those), the required storage became 12 times as much, i.e. 115MB. Including the auxiliary verbs (can, must, may, could, should, might): 7 times (7 x 115MB=805MB), including all 16 tenses: 16 times (16 x 805MB=12.9GB), including negative, interrogative, and two forms of negative interrogative: 5 times (5 x 12.9GB=64.5GB), and including passive forms: 2 times (2 x 64.5GB=129GB). By this time, the database had nearly 2 billion entries and binary search required a maximum of 31 lookups for each input sentence: A translation could be found in less than a second. We then set aside a separate database for simple nouns, adjectives and adverbs and 3 simple rules for how they could be connected and rules for translation. This changed the factor for the subject, and the object from 12 x 11 to 8,100,000,000 x 8,100,000,000 x 3 and an additional 45,000 for adverbs. This means that with a storage of 150GB and a simple engine with just 3 rules, we were able to translate an astronomical 57 x 10^{30} moderately simple sentences with 100% accuracy. With most of our time devoted to research and database development, the engine itself was designed in just a few months. We were able to translate any positive, negative or interrogative sentence with or without auxiliary verbs, using one of the 13,000 simple verbs in any tense with 100% accuracy and high speed, which essentially covers all simple sentences.

We scaled down this experimental system by reducing the number of sentences, through elimination of less frequently-used verbs. Using, two 80MB hard disks and some clever dynamic decompression methods, we demonstrated this approach to the visitors in an exhibition in London in 1992. The system was able to translate thousands of sentences the visitors were able to think of in a noisy, crowded exhibition. If it did not find the exact translation, it returned an error, or rendered a word-to-word substitute. This humble, database-based system received a lot of admiration it did not deserve, as it translated each sentence with

100% accuracy. The following are examples of sentences that this simple experimental system was able to translate:

I go.
He had worked.
We cannot sleep.
Did they like him?
Will she be contemplating this?
I wouldn't have been seeing him again.

Let us examine the usefulness of this approach. What is the probability that these pre-translated simple sentences actually occur in a given input RC text or, to put it differently, what percentage of the sentences in a typical input RC text would match the sentences this system can translate? Obviously, the 57 x 10^{30} figure is astronomically more than all the sentences ever written, spoken or thought about in the history of the English language by all people, dead or alive. But, while we correctly claimed that the experimental system could translate 57 x 10^{30} sentences 100% accurately, the fact is that an RC text contains a very small number of simple sentences.

Using simulation, based on statistical analysis of millions of pages of RC text shows that nearly 100% of the sentences this system can translate never occur. For those sentences that do occur, the frequency is high, but comprises less than 15% of RC text, in whole or in fragments. If the input text is literature written for first-grade school children, this percentage will be slightly higher. We could also increase this percentage by adding more rules. But this approach is of no practical use, if used alone. To make it practical, we modified the approach by narrowing down to words, phrases and sentence fragments. Our set of data gathering utilities (DGU) were redesigned and enhanced to process large amounts of RC input text. DGU identifies and stores the sentences and dynamically updates the resulting database. Since the process is automatic, it is possible that DGU would encounter a given text more than once. It may be possible, for example that a given political speech is published in two different newspapers, or that a research paper quotes from another document, which has already been processed by DGU. DGU detects such situations and discards the duplicate input, because if the text has a spelling error in it, the misspelled word would get its frequency incremented twice, making it seem a legitimate word. The detection algorithm is so sophisticated that it would never accept an RC text that is more than 20% similar to all the text it has processed before.

A sliding window of size 31 is used in DGU to capture any word or sentence fragment. (We have determined that 31 is the optimum window size.) By analyzing a large sample of RC text, we generated a dynamic database with over 4 trillion entries. This is an on-going research and aims to help upgrading CiyaTran by finding newly created words, phrases and patterns. As more text is analyzed and added to the database, the fragments with low frequency are deleted, if they have remained in the database for too long. To show how this sliding window works, consider the following sentence that was given to DGU:

Obviously, pure sequential searching is unattractive because, searching for a word in the example above would require reading all the pages.

The following 105 words and fragments were extracted by DGU for this sentence:

Obviously
pure
sequential
searching
is
unattractive
because
searching
for
word
in
the
example
above
would
require
reading
all
the
pages

Obviously ,
, pure
pure sequential
sequential searching
searching is
is unattractive
unattractive because
because ,
, searching
searching for
for a
a word
word in
in the
the example
example above
above would
would require
require reading
reading all
all the
the pages
pages .
Obviously , pure
, pure sequential
pure sequential searching
sequential searching is
searching is unattractive
is unattractive because
unattractive because ,
because , searching
, searching for
searching for a
for a word
a word in
word in the
in the example

the example above
example above would
above would require
would require reading
require reading all
reading all the
all the pages
the pages .
Obviously , pure sequential
, pure sequential searching
pure sequential searching is
is unattractive because ,
because , searching for
, searching for a
searching for a word
for a word in
a word in the
word in the example
in the example above
the example above would
example above would require
above would require reading
would require reading all
require reading all the
reading all the pages
all the pages .
, pure sequential searching is
because , searching for a
, searching for a word
searching for a word in
for a word in the
a word in the example
word in the example above
in the example above would
the example above would require
above would require reading all
would require reading all the

require reading all the pages
reading all the pages .
because , searching for a word
, searching for a word in
searching for a word in the
for a word in the example
a word in the example above
word in the example above would
require reading all the pages .
, searching for a word in the
for a word in the example above

If the word or phrase is already present in the database, its frequency is incremented. If not, it is added with a frequency of 1. Also a time/count stamp is used to get rid of fragments that are determined not to be useful based on certain criteria. If a word, or a fragment has a high frequency and is not already in the MT database, we may need to add it. The addition is also automated when possible. If not possible, meanings have to be provided manually. When the frequency of a fragment is high but other indicators show that it is not a phrase, it may be pointing to a new rule or pattern, which may need to be added to the pattern database.

Ranking Words Based on Frequency

Our analysis showed that from over 1,300,000 simple words of the database, only about 78,000 (6%) occur most frequently, and over 95% of all RC text is from these 78,000 words (excluding proper names). (A more comprehensive paper about this analysis is presented in another CiyaSoft document.) This percentage is even higher for less dynamic languages such as Farsi and Arabic. If the words are ranked based on their usage frequency, more than 75% of an RC text is comprised of the top 30,000. This is a small percentage of the simple words in the database. This group of words is the most troublesome, and the bulk of the Engine serves to resolve problems caused by them. These are troublesome because they are the source of 90% of the ambiguity in meanings, and exceptions to the rules. Most of these words have more than one GS, and more than one distinct meaning per GS. For example, the words "man", "time", and "state" have top ranks of 42, 47 and 66, respectively:

Word	Rank	GS	# of Meanings
man	42	3	15
time	47	3	24
state	66	3	16

As an example, the word "man" has at least the following 15 meanings:

1. As a noun:
   - a person
   - a human being
   - the human race
   - mankind
   - an adult human male
   - a husband
   - a strong (virile, brave or accomplished) male
   - a piece in a game
   - the police
   - the authority
   - male of low rank(soldier)

2. As a verb
   - assign
   - provide with people for operation

3. As an adjective:
   - person
   - mankind

As we move down the list, the words with lower ranks tend to have less GS and meanings. This does not mean that a word with the rank of, for example 600 could not have 15 meanings or 3 grammatical states. For example, the word "spring" has a rank of 774, yet it has 23 different meanings and 4 GS:

1. As an intransitive verb:
   - leap
   - rise up suddenly
   - recoil

- wrap

2. As a transitive verb:
    - cause to leap
    - cause to rise
    - disclose suddenly
    - produce suddenly
    - release the spring provide bail for

3. As a noun:
    - a leap
    - the vernal season
    - the beginning
    - water rising to the surface
    - any source of supply
    - an elastic strip of steel
    - a spiral wire
    - elasticity
    - motive power
    - a split
    - a wrap

4. As an adjective:
    - spiral wire (spring bolt)
    - vernal season (spring break)
    - water rising (spring water)

The less GS and distinct meanings per GS, the easier it is to determine the meaning for a word. When there is ambiguity in a meaning, the Engine first tries to determine the GS of the word in the input sentence. This is done by examination of the structure of the sentence as a whole and, sometimes, the neighboring words. Once GS is determined, the meaning is selected by the neighboring words in the sentence (using Fuzzy Logic), or the context of the text (as selected by the user, or automatically determined by the Engine). If configured so, the user may be allowed to select the meaning, or to use a supplied meaning from the Personal Dictionary (PD). If all fails, the most frequently used meaning, based on statistical analysis is taken. The analysis has been done statically during database builds.

Utilities

Utilities are the most important elements of our MT design. They are the programs developed internally (not part of the end product) and are used for the following:

Data gathering

A set of data gathering utilities (DGU) process RC text to identify (separate) sentences and store them in Fragment Database (FD) for batch detailed analysis. The result of DGU is used to enhance the databases used by the Engine. Simulation, statistical analysis and modeling tools also use FD to enhance the Engine.

Format and code conversion

The input text for DGU needs to be of text format only. It should also use a single code (in case of English, this would mean ASCII Text.) Format and code conversion utilities convert popular formats such as PDF, HTML, RTF and DOCX into ASCII text. In case of Farsi, for example, more than 20 different code standards are supported for conversion into a uniform ASCII-like code called ISCII.

Database Development and Maintenance (DDM)

This was the first utility for CiyaSoft MT. It consists of a very elaborate group of programs capable of preventing the user from making many common mistakes. Several years were spent to design and debug DDM and so many people have used it for thousands of hours, so it is very bulletproof. It provide the primary channel for manual testing and review of the databases, and it has many built-in facilities to make data entry semi-automatic.

Software Tools to Facilitate Research

Hundreds of programs in DBMS and high-level languages have been written during the past twenty years and are used every day to generate statistical reports on the databases, or behavioral reports on Engine simulations. These reports are used to make decisions for enhancement of the databases and modification or addition of new features to the Engine.

Simulation

A small-scale dynamic engine and database are used to simulate translation without committing the main database or the Engine. Any change to the Engine requires time-consuming re-compilation and testing. Similarly, any addition to the databases would require complete rebuilding, encryption, compression and integrity assurance. A complete build process takes several days. To shortcut this lengthy process, the simulation utilities are used to test if addition of a given algorithm to the Engine, or a given word, phrase or pattern to the databases would solve a particular translation problem. The solution is examined without consideration to the overall system. If it is successful, it would be tried in subsequent phases. If not, it would be modified until a solution is found.

Error Detection and Recovery (EDR)

These programs find errors in fields of the databases. The also look for discrepancies between various fields. The errors are usually data-entry-related or are caused by errors in the source, such as dictionaries or from the output of DGU, i.e. FD. EDR also automatically makes corrections, if it can, and generates reports. EDR set of programs is very extensive and is designed separately for each language pair. They are usually run in batch mode since it takes about a month to go over the entire database.

Database building

Building is a long process and involves building the main (general) database as well as related databases such as British English database, context databases, irregular verbs, special-case phrases, etc. The databases are encrypted, sorted, indexed and compressed, and Address Tables (AT) are built to be used during compilation. To make look-up faster during dynamic decompression and decryption, the AT is hard coded into the Engine. Therefore, each build of databases and corresponding AT would only work with the compiled code, which uses that AT.

Database and Engine integrity assurance

To avoid system crashes during translation, databases and the Engine are checked for integrity using CRC. All or part of databases or the Engine are subject to corruption during installation and activation, or due to worn-out fixed disks, bad memory, malicious or accidental erasure or modification, or virus.

Testing

Testing is an important part of any software, but testing for MT is very critical, because when more words are added to the databases, or more algorithms are implemented in the Engine, the entire system is affected. Because the Engine employs Fuzzy Logic, addition of single word, or changing the definition could break down the links between the words such that certain meanings would not be properly picked up. Similarly, addition of new patterns could negatively affect patterns that are partially analogous. There are two testing strategies: Regression Testing (RT) and Simulated Testing (ST).

Our RT is an AI-based program that works with a database of sentences for which CiyaTran has provided acceptable translations at each successive build. Both the sentence and its translation are in the database along with other information to help find the cause of error. As the development progresses and new features are added, more sample sentences are created, manually tested and added to RT database. After every complete database build or compilation, RT is run to determine what areas, if any, have been affected. RT is smart enough to pinpoint the problem and often provide useful suggestions or areas of code that could have contributed to the problem. The cycle of database build, compilation and regression test should be repeated until all problems have been resolved. If some or all of the problems cannot be solved, a decision should be made to drop one of the solutions in favor of the other depending which problem is more important. The cycle can be simulated in a smaller scale using the simulation technique explained above.

Simulation Testing (ST), on the other hand, automatically generates millions of sentences by following given criteria. The criteria are stored in the ST database but every time the ST is run the same group of sentences are regenerated and are presented to CiyaTran for batch translation. The results are compared against translations that were also generated by using the last build and compilation of CiyaTran (before the changes) and any differences are reported.
For automatic testing of configurable options, CiyaTran supports a number of macro commands. If these commands appear in a text, they are used as commands to CiyaTran instead of being translated.

Static Word Building

These are utilities that build words not already in the database, and are not normally in a dictionary. Such words are those that are often created by writers by using suffixes and prefixes. Examples are adverbs that are built by adding "ly" to adjectives, or nouns that can be built by adding "ness", or "ity". The utilities build these words statically and then examine them against the Fragment Database (FD) maintained by DGU. If the word is found there with an acceptable frequency, it would be added to the general database with a manually-provided meaning. This is obviously advantageous to dynamic meaning building during translation.

Components of CiyaTran: Interface

This component is responsible for communicating with the user to get such things as input and output file names, format, etc. Also, the use can configure CiyaTran for:

- Context (if not selected, it will be determined automatically)
- Sensitivity level
- Page setup for printing
- Printer setup
- Input file format
- Output file format and font
- Enabling user intervention for resolution of ambiguity
- Activating Personal Dictionary (PD)
- Enabling editing capability for each translated sentence
- Enabling building of meanings using prefix/suffix databases

- Management of PD (add words or phrases to PD, or add/change meanings, GS or context for words and phrases)
- AI management (add new patterns, productions and other AI elements or change existing ones)

The interface consists of an MS-Word-like bilingual word processor with all needed capabilities, such as WYSIWYG, editing, font selection, cut and paste, find and replace and print, and it has additional menu items required for translation.

Databases

1. General Database (GD): The main database, which allows 6 GS and 24 meanings per English word or phrase. It includes over 2.5 million entries with the following fields for each record: Meanings, GS, Person, Popularity, Rank, Confidence, Fuzzy Logic Link, Context, Source, Error Rate, Usage, Peculiarity, Variation, Order, etc. GD is about 3 times the size of context databases:

Average word size	12.91
Average word size (phrases excluded)	9.64
Average word size (phrases)	15.61
Phrases (noun phrases, regular adjective phrases, etc.)	51.91%
Average number of words per phrase	5.7
Simple verbs	2.10%
Compound verbs	3.66%
Noun phrases	73.75%
Nouns excluding noun phrases	71.67%
Adverbs	6.53%
Regular adjectives	13.69%
Forward adjectives	0.93%
Entries with 1 GS (across the Entire database)	98.25%
with 2 GS	1.51%
with 3 GS	0.24%
Single words with 1 GS	96.36%
with 2 GS	3.12%
with 3 GS	0.52%

2. Context Databases: Over seventy databases structurally similar to GD but including words only used by one field of science or technology.

3. Proper Name Databases: Includes most popular male and female first names and last names, names of famous people, names of places, countries, cities and words or names of foreign origin.

4. Grammar Databases: Used by the Engine for implementation of various features (Mathematical Modeling, Patterns, Productions, Irregular Verbs, Abbreviations, Prefixes, Suffixes, Synonyms, Acronyms, British Words, Words from Latin languages (Spanish, German, French and Italian), etc.

Engine

Once the file is selected and "translate" command issued; or in case of batch processing, once a file is spotted for translation, it is first converted into an ASCII text format. (If the source language is not English, the file is also code-converted.) The document is then submitted to the Pre-processing Unit (PPU), where it is scanned 12 times by various sections of PPU. By completion of the last scan (syntactic parse), all words, numbers and symbols (called elements) have been separated and tagged, and the sentences and titles (headings) have been isolated into arrays of elements (AE). (If the source language is in Farsi, the task of isolating sentences is very difficult, as using punctuation does not have a standard that all writers follow. Separating words is even more difficult and would require a morphological analyzer because space is not always used to separate words. Solution to these problems is the subject of another CiyaSoft document.)

Over 300 tags are used by PPU for translation from English to the languages we support. The following are some examples from the tag list used when source is English:

1. Auxiliary Tag: can, may, must, might, could, ...
2. Be Tag: am, is, are, was, were, be, been
3. Negative Auxiliary Tag: can't, cannot, couldn't, ...
4. Preposition Tag: from, with, at, on, in, for, into, ...
5. Possessive Tag type 1: my, his, ...
6. Reflexive/Emphatic Pronoun Tag: myself, ...

7. Title Tag: Dr., Mr., Mrs., Capt., ...
8. Do Not Translate Tag
9. Date Tag
10. Question Mark Tag
11. Number Tag
12. Time Tag
13. Case Tags: All Upper, All Lower, Capitalized, Mixed
14. Word Type Tags: Person (First, Second, ...),
15. S/P (Singular, Plural), ...
16. Start/End Tags: Valid Start, Valid End, Suspicious Start or End, Invalid Start or End
17. Irregular Verb Tags: Past only (broke, arose), PP only (broken, begun), Past or PP (built, felt), Present/Past/PP (cut, put, cast), Present or PP (run, come)
18. Regular Verb Tags Past/PP: Type 1 (worked > work), Type 2 (advanced > advance), Type 3 (begged > beg), Type 4 (carried > carry), Type 5 (egged > egg)
19. Gerund Verb Tags: Type 1 (working > work), Type 2 (begging > beg), Type 3 (competing > compete), type 4 (egging > egg), type 5 (dying > die)

PPU also does the following:

1. Determination of the language, context, subject and source (informal letter, newspaper, technical journal, text book,...) of the document
2. Error assessment in spelling, grammar and punctuation
3. Detecting if document came from OCR or speech recognition
4. Detecting whether the document was written, or was derived from a speech or dialog
5. Determination of translatability of the document

If the document successfully passes through PPU, one sentence or title at a time, consisting of one AE is submitted to the Translation Unit to do the following (not necessary in the given order):

1. Look up a default set of meanings for each element based on preliminary tags
2. Identify the main verb of AE
3. Determine if AE is imperative, positive, negative or interrogative
4. Determine if AE is passive or active voice
5. Identify a compound verb, if any, using sliding windows
6. Identify the tense of AE
7. Identify the main subject, object and adverb
8. Identify breaking points
9. Apply all patterns in the pattern database to AE (this will capture over 1,000 patterns such as number patterns, date and time patterns, titled or title-less proper names, etc.)
10. Apply all productions and mathematical models from AI database to AE
11. Identify Grammatical State (GS) for each element of AE
12. Identify phrases using sliding windows
13. Re-tag the elements based on the new information
14. Re-look up set of meanings for elements with changed tags
15. Use Fuzzy Logic to select a meaning when more than one is available for the identified GS
16. Use meanings provided by the user, if personal dictionary is active
17. Use meaning from context databases if context has been selected by user or determined by Engine
18. Use new patterns and models modified or defined by the user, if any
19. Submit the AE to the target language unit which uses the reordering rules to render a translation