Rapid Development of Translation Tools: Application to Persian and Turkish

Jan W. Amtrup, Karine Megerdoomian and Rémi Zajac
Computing Research Lab
New Mexico State University
{jamtrup,karine,zajac}@crl.nmsu.edu

Abstract

The Computing Research laboratory (CRL) is developing a machine translation toolkit that allows rapid deployment of translation capabilities. This toolkit has been used to develop several machine translation systems, including a Persian-English and a Turkish-English system, which will be demonstrated. We present the architecture of these systems as well as the development methodology.

1 Introduction

At CRL, one of the major research topics is the development and deployment of machine translation systems for low-density languages in a short amount of time. As the availability of knowledge sources suitable for automatic processing in those languages (e.g. Persian, Turkish, Serbo-Croatian) is usually scarce, the systems developed have to assist the acquisition process in an incremental fashion, starting out from low-level translation on a word-for-word basis and gradually extending to the incorporation of syntactic and world knowledge.

The tasks and requirements for a machine translation environment that supports linguists with the necessary tools to develop and debug increasingly complex knowledge about a specific language include:

- The development of a bilingual dictionary that is used for initial basic translation and can further be utilized in the more complex translation system stages.
- Methods to describe and process morphologically rich languages, either by integrating already existing processors or by developing a morphological processor within the system framework.
- Glossing a text to ensure the correctness of morphological analysis and the completeness of the dictionary for a given corpus.
- Processors and grammar development tools for the syntactic analysis of the source language.
- In order to allow rapid development cycles, the translation system itself has to be reasonably fast and provide the user with a rich environment for debugging of linguistic data and knowledge.
- The system used for development must be configurable for a large variety of tasks that emerge during the development process.

We have developed a component-based translation system that meets all of the criteria mentioned above. In the next section, we will describe the architecture of the system MEAT (*M*ultilingual *E*nvironment for *A*dvanced *T*ranslations) , which is used to translate between a number of languages (Persian, Turkish, Arabic, Japanese, Korean, Russian, Serbo-Croatian and Spanish) and English. In the following sections, we will describe the general development cycle for a new language, going into more detail for two such languages, Persian and Turkish.

2 General architecture

MEAT is a publicly available environment[1] that assists a linguist in rapidly developing a machine translation system. In order to keep the overhead involved in learning and using the system as low as possible, the linguist uses use simple yet powerful basic data and control structures. These structures are oriented towards contemporary linguistic and computational linguistic theories.

In MEAT, linguistic knowledge is entirely represented using Typed Feature Structures (TFS) (Carpenter, 1992; Zajac, 1992), the most widely used representational formalism today. We developed a fast implementation of Typed Feature Structures with appropriateness, based on an abstract machine view (cf. Carpenter and Qu (1995), Wintner and Francez (1995). Bilingual dictionary entries as well as all kinds of rules (morphology, syntax, transfer, generation) are expressed as feature structures of specific types, so only one description language has to be mastered. This usually leads to a rapid familiarity with the system, yielding a high productivity almost from the start.

[1] http://crl.nmsu.edu/~jamtrup/Meat

Runtime linguistic objects (words, syntactic structures etc.) are stored in a central data structure. We use an extension of the well-known concept of a chart (Kay, 1973) to hold all temporary and final results. As multiple components have to process different origins of data, the chart is equipped with several different layers, each of which denotes a specific aspect of processing. Thus, at every point during the runtime of the system, the contents of the chart reflect what operations have been performed so far (Amtrup, 1999). The complete chart is available to the user using a graphical interface. This chart browser can be used to exactly trace why a specific solution was produced, a significant aid in developing and debugging grammars.

MEAT addresses the necessity of carrying out several different tasks by providing a component-based architecture. The core of the system consists of the formalism and the chart data representation. All processing components are implemented in the form of plug-ins, components that obey a small interface to communicate with the main application. The choice of which components to apply to an actual input, the order in which the components are processed, and individual parameters for components can be specified by the user, allowing for a highly flexible way of configuring a machine translation (or word-lookup, of glossing etc.) system (cf. Amtrup et al. (2000) for a more detailed description of the architecture).

The MEAT system is completely implemented in C++, resulting in a relatively fast mode of operation. The implementation of the TFS formalism supports between 3000 and 4500 unifications per second, depending on the application it is used in. Translating an average length sentence (20-25 words) takes about 3.5 seconds on a Pentium PII400 (in non-optimized debug mode). The system supports Unix (tested on Solaris and Linux) and Windows95/98/NT. We use Unicode to represent character data, as we face translations of several different, non-European languages with a variety of scripts.

3 Development cycle

One of the main requirement facilitating the deployment of a new language with possibly scarce preexisting resources is the ability to incrementally develop knowledge sources and translation capability (the incremental approach to MT development is described in (Zajac, 1999)). In the case of translation systems at our laboratory, we mostly translate into English. Thus, a complete set of English resources is already available (dictionary, generation grammars and morphological generation) and does not need to be developed.

The first step in bootstrapping a running system is to build a bilingual dictionary. The work on the dictionary usually continues throughout the development process of higher level knowledge sources. We use dictionaries where entries are encoded as flat feature-value pairs, as shown in Figure 1.

```
$Headword @J|yb hftg|nh
$Category Noun
$Number Plural
$Regular False
$English Seven Wonder<nc plur>;
$$
```

Figure 1: A Persian-English dictionary entry.

While this is already enough information to facilitate a basic word-for-word translation, in general a morphological analyzer for the source language is needed to translate real-world text. For MEAT, one can either import the results of an existing morphological analyzer, or use the native description language, based on a finite-state transducer using characters as left projections and typed feature structures as right projections (Zajac, 1998). After completing the morphological analysis of the source language and specifying the mapping of lexical features to English, glossing is available. The Glosser is an MEAT application consisting of morphological analysis of source language words, followed by dictionary lookup for single words and compounds, and the translation into English inflected word forms. An example of the interface for the glosser is shown in Figure 2.

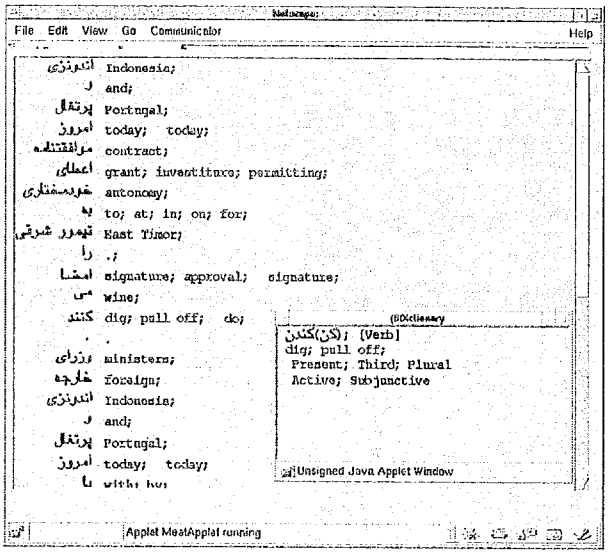

Figure 2: The Glosser interface

The next step in developing a medium-quality, broad coverage translation system is to develop

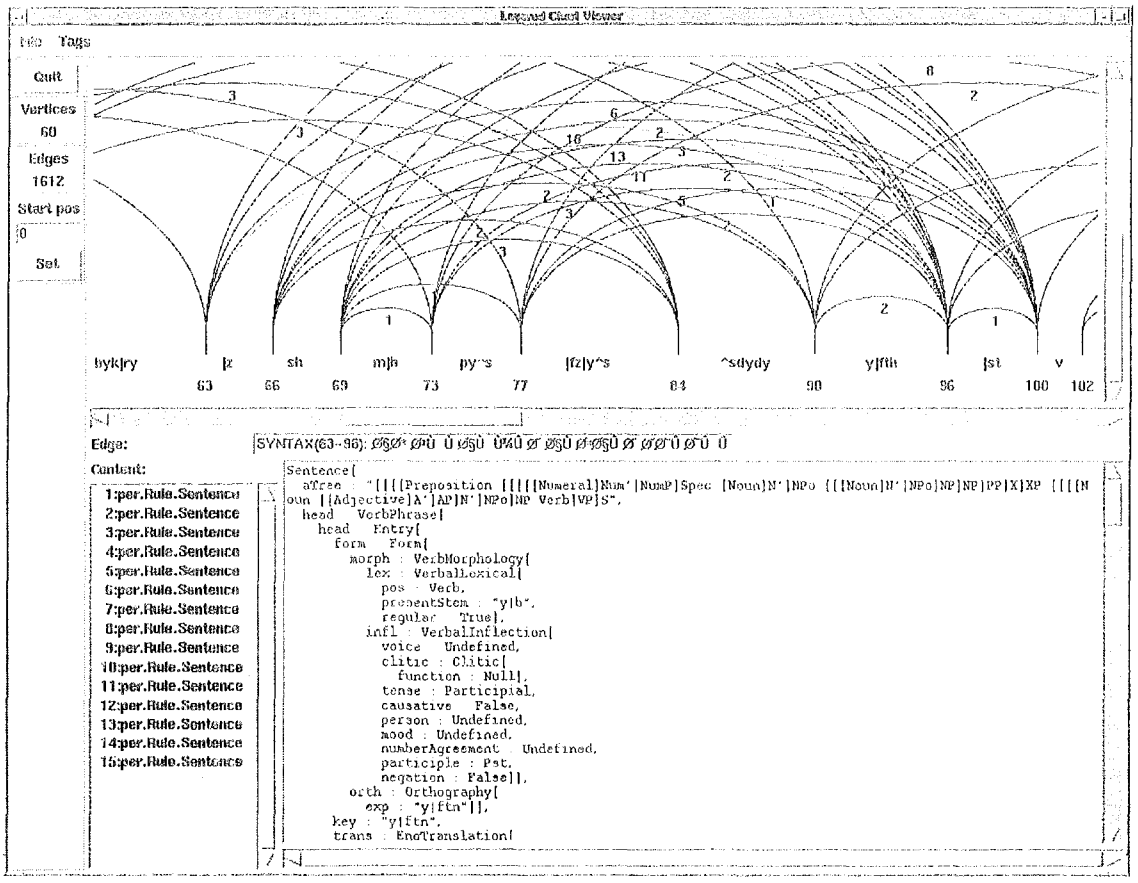

Figure 3: Viewing a complex analysis

knowledge sources for the structural analysis of input sentences. MEAT supports the use of modular unification grammars, which facilitates development and debugging. Each grammar module can be developed and tested in isolation, the final system applying each grammar in a linear fashion (Zajac and Amtrup, 2000). The main component used is a bidirectional island-parser (cf. Stock et al. (1988)) for unification-based grammars. The grammar rules are usually written in the style of context-free rules with associated unification constraints as shown in Figure 4. The rules allow for the specification of the right-hand side as a regular-expression of feature structures. We plan to add more restricted types of grammars (e.g. based on finite-state transducers) to give the linguist a richer choice of syntactic processes to choose from.

For the time being, the transfer capabilities of the system are restricted to lexical transfer, as we have not finished the implementation of a complex transfer module. Thus, the grammar developer either needs to create structural descriptions that match the English generation, or the English generation grammar has to be modified for each language.

At each point during the development of a trans-

```
Adj' = tur.Rule[
  lhs: tur.AdjBar[
         lexHead: #adj,
         spec: #adv],
  rhs: <:  #adv= tur.AdvEntry "?"
           #adj= tur.AdjEntry
       :>
  ];
```

Figure 4: A Turkish syntax rule

lation system, we consider it essential to be able not only to see the results, but also to monitor the processing history of a result. Thus, the chart that leads to the construction of an English output can be viewed in order to examine all intermediate constructions. In the MEAT system, each module records various steps of computations in the chart which can be inspected statically after processing. A unified data interface for all modules in the system allows both the inspection of recorded internal data structures for each module (when it makes sense,

such as in a chart parser), and the inspection of the input/output of all modules. The graphical interface used to view complex analyses is shown in Figure 3.

4 Applications

In this section, we give an overview of the capabilities of MEAT using two current examples from work at our laboratory. In the Shiraz project[2] (Amtrup et al., 2000), we developed a machine translation system from Farsi to English, for which no previous knowledge sources were available. We mainly target news material and the translation of web pages. The Turkish-English system has been developed with the Expedition project[3], an enterprise for the rapid development of MT systems for low-density languages.

Both systems use a common user interface for access to MEAT, which is shown in Figure 5. The MT systems are targeted to the translation of newspaper text and other sources available online (e.g. web pages). The emphasis is therefore put on extensive coverage rather than very-high quality translation. Currently, we reach for both systems a level of quality that allows to assess in detail the content of source texts, at the expense of some unfelicitous English.

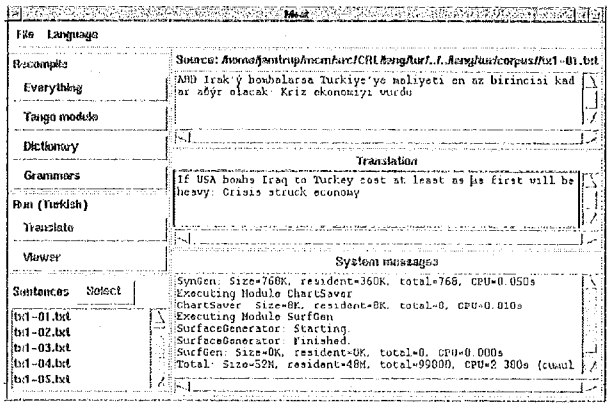

Figure 5: The MEAT user interface

4.1 Persian-English MT

The input for the Persian-English system is usually taken from web pages (on-line news articles), although plain text can be handled as well. The internal encoding is Unicode, and various codeset converters are available; we also developed an ASCII-based transliteration to facilitate the easy acquisition of dictionaries and grammars (see Figure 1). The dictionary consists of approximately 50,000 entries, single words as well as multi-word compounds. Additionally, we utilize a multi-lingual onomasticon maintained locally to identify proper names.

[2]http://crl.nmsu.edu/shiraz

[3]http://crl.nmsu.edu/expedition

We developed a morphological grammar for Persian using a unification-based formalism (Zajac, 1998). A sample rule is shown in Figure 6 (cf. Megerdoomian (2000) for a more thorough description of the Persian morphological analyzer).

```
PresentStem = <
  RegularPresentStem
  < < <"|n" "y"?>
      per.Verbal[infl.causative: True]> |
    < per.Verbal[infl.causative: False]>
  >>;
```

Figure 6: A Persian morphological rule

The knowledge sources for syntax were (manually) developed using a corpus of 3,000 tagged and bracketed sentences extracted from a 10MB corpus of Persian news articles. We use three grammars, responsible for the attachment of auxiliaries to main verbs, the recognition and processing of light verb phenomena, and phrasal and sentential syntax, respectively. The combined size is about 110 rules. The development of the Persian resources took several months, primarily due to the fact that the translation system was developed in parallel to the linguistic knowledge. The Persian resources were developed by a team of one computational linguist (morphological and syntactic grammars, overall supervision for language resources), and 3 lexicographers (dictionary and corpus annotation).

4.2 Turkish-English MT

Within yet another project (Expedition), we developed a machine translation system for Turkish. This application functioned as a benchmark on how much effort the building of a medium-quality system requires, given that an appropriate framework is already available.

For Turkish, we use a pre-existing morphological analyzer (Oflazer, 1994). Turkish shows a rich derivational and inflectional morphology, which accounts for most of the system development work that was necessary to build a wrapper for integrating the Turkish morphological analyzer in the system (approximatly 60 person-hours)[4]. The development of the Turkish syntactic grammars took around 100 person-hours, resulting in 85 unification-based phrase structure rules describing the basics of Turkish syntax. The development of the bilingual Turkish-English dictionary had been going on for

[4]The main problem during the adaptation was the treatment of derivational information, where the morphologically analyzed input does not conform exactly to the contents of the dictionary

some time prior to the application of MEAT, and currently contains approximately 43,000 headwords.

5 Conclusion

The development of automatic machine translation systems for several languages is a complicated task, even more so if it has to be done in a short amount of time. We have shown how the availability of a machine translation environment based on contemporary computational linguistic theories and using a sound system design aids a linguist in building a complete system, starting out with relatively simple tasks as word-for-word translation and incrementally increasing the complexity and capabilities of the system. Using the current library of modules, it is possible to achieve a level of quality on par with the best transfer-based MT systems. Some of the strong points of the system are: (1) The system can be used by a linguist with reasonable knowledge of computational linguistics and does not require specific programming skills. (2) It can be easily configured for building a variety of applications, including a complete MT system, by reusing a library of generic modular components. (3) It supports an incremental development methodology which allows to develop a system in a step-wise fashion and enables to deliver running systems early in the development cycle. (4) Based on the experiences in building the MT systems mentioned in the paper, we estimate that a team of one linguist and three lexicographers can build a basic transfer-based MT system with medium-size coverage (dictionary of 50,000 headwords, all most frequent syntactic constructions in the language) in less than a year.

One major improvement of the system would be a more integrated test and debug cycle linked to a corpus and to a database of test items. Although existing testing methodologies and tools could be used (Dauphin E., 1996; Judith Klein and Wegst, 1998), building test sets is a rather time-consuming task and some new approach to testing supporting rapid development of MT systems with an emphasis on wide coverage would be needed.

References

Jan W. Amtrup, Hamid Mansouri Rad, Karine Megerdoomian, and Rémi Zajac. 2000. Persian-English Machine Translation: An Overview of the Shiraz Project. Memoranda in Computer and Cognitive Science MCCS-00-319, Computing Research Laboratory, New Mexico State University, Las Cruces, NM, April.

Jan W. Amtrup. 1999. *Incremental Speech Translation.* Number 1735 in Lecture Notes in Artificial Intelligence. Springer Verlag, Berlin, Heidelberg, New York.

Bob Carpenter and Yan Qu. 1995. An Abstract Machine for Attribute-Value Logics. In *Proceedings of the 4^{th} International Workshop on Parsing Technologies (IWPT95)*, pages 59–70, Prague. Charles University.

Bob Carpenter. 1992. *The Logic of Typed Feature Structures.* Tracts in Theoretical Computer Science. Cambridge University Press, Cambridge.

Lux V. Dauphin E. 1996. Corpus-Based annotated Test Set for Machine Translation Evaluation by an Industrial User. In *Proceedings of the 16th International Conference on Computational Linguistics, COLING-96*, pages 1061–1065, Center for Sprogteknologi, Copenhagen, Denmark, August 5-9.

Klaus Netter Judith Klein, Sabine Lehmann and Tillman Wegst. 1998. DIET in the context of MT evaluation. In *Proceedings of KONVENS-98.*

Martin Kay. 1973. The MIND System. In R. Rustin, editor, *Natural Language Processing*, pages 155–188. Algorithmic Press, New York.

Karine Megerdoomian. 2000. Unification-Based Persian Morphology. In *Proceedings of the CICling 2000*, Mexico City, Mexico, February.

Kemal Oflazer. 1994. Two-level Description of Turkish Morphology. *Literary and Linguistic Computing*, 9(2).

Oliviero Stock, Rino Falcone, and Patrizia Insinnamo. 1988. Island Parsing and Bidirectional Charts. In *Proc. of the 12 th COLING*, pages 636–641, Budapest, Hungary, August.

Shuly Wintner and Nissim Francez. 1995. Parsing with Typed Feature Structures. In *Proceedings of the 4^{th} International Workshop on Parsing Technologies (IWPT95)*, pages 273–287, Prague, September. Charles University.

Rémi Zajac and Jan W. Amtrup. 2000. Modular Unification-Based Parsers. In *Proc. Sixth International Workshop on Parsing Technologies*, trento, Italy, February.

Rémi Zajac. 1992. Inheritance and Constraint-Based Grammar Formalisms. *Computational Linguistics*, 18(2):159–182.

Rémi Zajac. 1998. Feature Structures, Unification and Finite-State Transducers. In *FSMNLP'98, International Workshop on Finite State Methods in Natural Language Processing*, Ankara, Turkey, June.

Remi Zajac. 1999. A Multilevel Framework for Incremental Development of MT Systems. In *Machine Translation Summit VII*, pages 131–157, University of Singapore, September 13-17.