Specification and Evaluation of Machine Translation Toy Systems - Criteria for laboratory assignments –

Cristina Vertan, Walther v. Hahn

University of Hamburg, Natural Language Systems Division

Hamburg, Germany
{cri,vhahn}@nats.informatik.uni-hamburg.de

Abstract

Implementation of machine translation "toy" systems is a good practical exercise especially for computer science students. Our aim in a series of courses on MT in 2002 was to make students familiar both with typical problems of Machine Translation in particular and natural language processing in general, as well as with software implementation. In order to simulate a software implementation process as realistic as possible, we introduced more than 20 evaluation criteria to be filled by the students when they evaluated their own products. The criteria go far beyond such "toy" systems, but they should demonstrate the students, what a real software evaluation means, and which are the particularities of Machine Translation Evaluation.

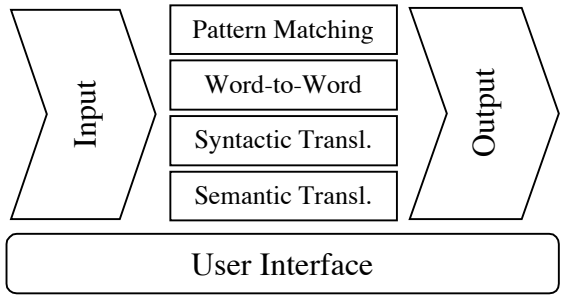
1 Introduction

Machine Translation (MT) is an important subfield both of Computational Linguistics and Natural Language Processing. Therefore academic education in MT addresses students in linguistics and computer science. Usually, according to the background of the students, courses are given separately to these two groups but with different methodologies: theoretical aspects and demonstration of tools for the linguists (Somers, 2001) on one hand, implementation of clearly defined algorithms for the computer science students on the other hand.

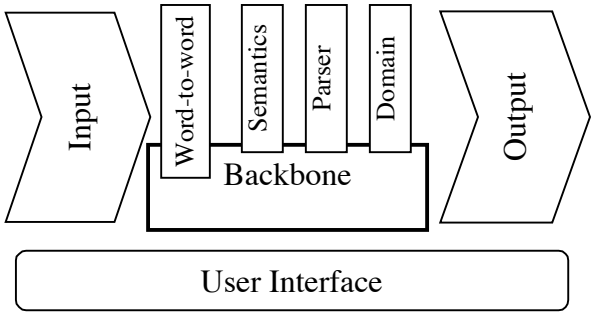
The alternative, to implement a realistic MT-system in one course is not feasible, due to the lack of time and missing background knowledge by the students. Very often they are facing the field for the first time.

A solution in between may be the implementation of one or several "toy systems", with rather limited language resources and limited functionality. In (v. Hahn and Vertan, 2002) the reader will find detailed examples of such toy systems, which have been developed mainly by courses for computer science students, but also including students from linguistics.

In one these courses the students had to implement (in small groups) parallel small systems based on either pattern matching, or word-to-word translation, or syntactic translation, or semantic translation. These sub-systems processed a corpus of app. 100 sentences, each, being controlled by a common user interface. The aim was to get a realistic idea of the possible contribution of each module in a real MT system.



In another course the students had to implement a classical centralised integrated system with a word-to-word pre-processor, syntactic and semantic modules, domain knowledge with an ontology and a user interface.



The aim of designing such systems is not only

- to offer to the students some interesting programming exercise, but
- *to make them conscious of what the implementation of a real system means.*

The remaining pages of this paper will explain details of this last topic of the courses, where we made the students reflect on questions like:

- "How will the system react with huge amounts of data or maximal throughput?",
- "How well are changes of the domain or changes of the languages supported?",
- "What about maintenance by users?"
- Or, more complicated: "What type of architecture is optimal for my required functionality?",
- "What kind of grammar is more suitable?"
- "How much from the original plans did we realise?"
- Which is the intuition of a possible user?"

In order to give the students the opportunity to reflect on all these types of questions, we asked the students, as integrated part of the laboratory assignment, to check 17 criteria for software quality evaluation (among them maintainability, system work-flow integration, or efficiency) and 7 criteria for the linguistic functionality, like lexical coverage, syntactic coverage or compatibility with (European) standards and formats.

The aim was to familiarise students with the evaluations of software projects. For some criteria, however, like maintanability or domain coverage, there was apparently no reasonable answer when working with a "toy" system, but the idea was to expose the students early enough to all aspects of software evaluation in general and machine translation in particular.

To make the whole implementation process more realistic, we also prepared criteria for the specification of the software, which had to be fulfilled before the implementation process. Both software evaluation and specification criteria follow general software engineering theory (Somerville, 1990)

In the following we will explain in detail each of the specification and evaluation criteria (both software specific and linguistic), and we will report about what the students learned from following such schemata.

2 Specification Criteria

a) Functional requirements: mainly the behaviour of input and output, both at system and module level

- On the system level the specification has to define the type and form of the input (e.g. speech or text, format of the input, e.g., file format and which other formats are supported, restrictions of text input from keyboard, menu-selection, audio formats, etc.) as well as the type and form of the output. To demonstrate the generality of the criteria we included such cases, where the output of a natural language processing system may be something else than a natural language utterance again.: A record from a databank or an action (in case of a robot system), or simply specific terms or web-links.
- The other part of the functional requirements concerns the module interfaces of the system. Specifying formats for input and output of each module right from the beginning, makes it much easier to work afterwards independently among the teams of this course. Each group can develop and test their modules with simulated input-data without waiting for completed work of the others.

b) Performance requirements

The students were asked to estimate which time behaviour has or is required for their system, and which resources it will need.

c) Usage requirements

Among students, this criterion usually is the most neglected parameter. They tend to assume that if input and output are in natural language, no special attention has to be paid to a user interface. A potential user, in their view, needs only a text field, where to type the input, and another part of the screen for the presentation of the output. We tried to raise their awareness for more specific requirements, esp. in MT systems: Dialogue windows for unknown words and errors in the input or the proper selection of labels and controls, facilities for reading in files, for pre-processing etc.

d) Embedding requirements

Under this heading the students are asked to specify, which hardware is needed for their system and which operating systems will be supported. A realistic scenario for the application has also to be discussed.

3 General Evaluation Criteria

The first group of criteria evaluates the usage of software, seen from the perspective of the user, under i) there follow criteria for the software product itself and under j) process quality critera

a) Adequacy

This point has to be assessed in reference to 2.c, i.e. how much from the specified user requirements are fulfilled, how user-friendly the system interface is, etc.. The students have to give precise examples, of situations where their system reacts adequate, and cases where improvements seem necessary.

b) Transparency

This evaluation criterion includes reasonable user estimations about processing errors, the plausibility of the system's behaviour in general or reasonable help facilities. Example: In a translation tool the user (ideally) has to be informed, whether a non-translated term is a word "out of dictionary", a proper name or simply an input error by the user. Of course at the level of toy systems we can not expect from the students (especially under time constraints) to tackle such problems, but they must be aware of their existence.

c) work-flow integration

As mentioned under paragraph 2.d, a possible scenario has to be specified initially for the system. In the evaluation statements the students are requested to explain to what extent their system would fit into an assumed work flow and with which additional time and costs their product can be adapted to other scenarios or work flow environments. Further, how flexible it is to functional extensions, because in a course such toy systems are designed exactly for the given or defined scenario. By including such a criterion we force the students to reflect about the difficulties of building a system, which is general enough to cover different scenarios and different work flow environments.

d) Specifications match

This requires a detailed comparison with all specification criteria, which of them are met by the implementation, what is still missing, and more, what is not conform with the specification criteria at all and why. The students must provide reasons why for example input and output formats were changed, or not supported, in the given form.

e) Reliability

The deterministic behaviour of the product, and its components has to be evaluated. As there is, e.g. no additional sensor input, translation systems must be deterministic.

f) Robustness

This is again an issue, where students have to learn a lot about real software behaviour and to make a very detailed evaluation. From our experience, they assume that their system works with all input data in the form that they require, and that the system runs in a "similar" way as with their test sentences. Their specifications usually cope only with the positive functionality "What to do" , not with functions to avoid certain behaviour. What happens, if the user forgets to specify parameters, if the user makes none of necessary actions, enters corrupted data etc. What to do about faulty input?

g) *Failure safety*

This criterion is mentioned only to familiarise the students with large scale evaluation procedures. For a prototype toy system it is not assumed that the implementation will include restart facilities, or that there are backup copies, but such aspects are important for real systems.

h) Efficiency

The efficiency of the program has to be estimated in terms of hardware requirements and consumption of resources, as well as the time required to perform certain operations.

i) Product quality

Under this title the students will correctly understand to briefly explain whether the program execution is correct, i.e. the expected behaviour is delivered. To refine the discussion we introduced the following sub-criteria:

- correctness (e.g. correct processing, complete correspondence to specifications)
- comprehensibility (e.g. structure of programs, choice of designators and names in the code)
- testability
- maintainability
- changeability
    - structural changes
    - functional changes
    - problem-type changes

The criteria mentioned so far are valid for any software product. In our case of toy translation tools, the problem of correctness is much more complicated due to the translation specific features. In contrast to classical software products where to any input a unique correct output must correspond, translation translation theory say clearly, that there

is more than one correct translation of the same input sentence. Moreover, the assessment "correct" for a translation is relative. For example, in the case of the Verbmobil system evaluation (Tessiore and v. Hahn, 2000), a lot of users were prepared to classify the output as "correct" already, if they could understand the meaning and pragmatics of the translation.

Usually the existent evaluation methodologies of machine translation, require the existence of a reference translation. A set of metrics are defined in the literature (Dabbadie and al., 2002) starting from the output and the reference translation. Our experience proved, that the existence of a reference translation can be even misleading for the students. For at least two of the toy systems, which were developed in our courses, we provided the students with a test corpus, consisting of about 70-100 sentences and their reference translations. At least three problems were encountered:

1. The students had a strong fixation on our reference translation: either they tried to tune their system artificially to deliver exactly the given reference, or they classified all translations as incorrect, which did not met perfectly the reference.
2. The construction of the (bilingual) lexicons is done strictly according to the reference translation: Only the morphology, meanings etc. encountered in the test corpus are included. As a consequence, the students did face the problem of disambiguation only in those cases, where we included it intentionally.
3. The development of the system was done strictly to cover the test corpus. Any additional sentence, would fail.

The scenario which we are applying now after this experience is rather different: At the beginning the students get no test corpus. Their first task together with the requirements in the specification task is to estimate what kind of sentences can occur in the given domain, and to design subsequently a lexicon which covers such situations. After the design phase and during evaluation we provide a test corpus, but only with sentences in the source language. This test corpus prevents the students from choosing only very simple cases, e.g., no anaphora and ellipses, no sub clauses or defective sentences

j. Process quality (e.g. quality of the implementation process, certification, quality of specification)

In contrast to the evaluation of the product, which addresses only the results delivered by the system and it's overall behaviour, process evaluation means the evaluation of the conditions, under which the software was produced. This covers the methodology for compiling the specifications, security measures, the design of tests, and the cooperation among the groups and with the customer. Here, the students have the opportunity to reflect about the quality of their production process and about the results of , e.g., underestimating time resources etc. Obviously, under time constraints, the code is not always documented, not always, explicit enough.

The aim of this "professional" software evaluation is not to over-criticize the results of the students but to show them what requirements are expected at a commercial level even for tasks, which are, by nature, not completely and formally defined and, by nature, vague, because this is the nature of language.

4. Criteria for Linguistic quality evaluation

In section 3 we presented evaluation criteria, which are valid for all software products. In the following we will concentrate on specific criteria for linguistic processing, in particular for translation tools.

a) Coverage

- Lexicon
- Syntax
- Semantics

As explained in section 2 .i), in a toy system the students will implement a reduced lexicon, a grammar which covers only part of the language and will deal only with restricted semantic problems. In our opinion, however, it is important that the student can define exactly the amount of linguistic features that they cover. Therefore they are asked to indicate:

- how many entries the lexicon has and to give examples of important word, which may occur in the given domain, but were not included,
- the annotations in the lexicon, the choice of lexicon type (stem lexicon versus full-form lexicon) and correspondingly, their morphological processing,
- types of sentences that can be processed, and types of realistic sentences which will fail,

- semantic phenomena, which are tackled and solved

b) Pragmatics

Here the students have to evaluate to which extent their software covers pragmatics aspects of the languages. Good examples are common directive speech acts like "The course is given in the city centre" (≠ Das Seminar wird in der Stadtmitte abgehalten", = in the university main building, not in CS building).

c) Compatibility

Translation tools make use of lots of resources (corpora, lexicons, grammars, etc.). Their development is time consuming, and therefore standardization efforts have been made since many years. The aim is to provide reusable resources. Therefore the students are asked to discuss, whether

- the format of their data, e.g., to what extent these meet existing standards and formalisms. If not, is the lexicon encoded in a reusable format (at least some XML version)?
- the grammar follows a well-known formalism (HPSG, functional grammar, etc.) and, on which basis the choice was done.

Concerning the languages, we usually define right from the beginning what is the source and what is the target language for the translation process. The students, however, must discuss if their program:

- can it be (easily) reversed to translate backwards: from target to source
- can it be adapted to new language pairs, and with which amount of work,. Here the general translation paradigm (transfer versus interlingua) can be addressed

Especially the linguistic evaluation can be a starting point for a broader discussion in the seminar about rather difficult issues in NLP:

- how much does a change of the lexicon design influences the design and the functionality of the whole system,
- is the lexicon part of the grammar (transition networks), then changes have influence on the whole grammar and the parser,
- how do technical ad-hoc decisions (easy implementation, time constraints, programming languages) restrict the whole system design and inhibit reasonable linguistic solutions.

Similar discussions can be triggered concerning the change of the domain. The change of the domain involves major re-implementations of at least the lexical resources and the pragmatic processes.

5 Conclusions

In this paper we presented criteria for the specification and evaluation of toy machine translation systems. to asses their quality The criteria can be grouped in two classes: general software evaluation criteria and specific linguistic ones. Both are used by the students to evaluate their own programmming. It is quite clear, that many of these criteria are by far too complex for such toy systems. The main aim is to familiarize computer science and linguistics students with real evaluation methodology. From our experience, the students had real difficulties to asses each point of the criteria list. However, at the end of the evaluation, they got some general ideas about why perhaps some of the methods, although locally successful, are not general enough, from which issues the success of an implementation depends and, last but not least, why the implementation of a machine translation system is not a trivial task.

6 Bibliographical References

Marianne Dabbadie and Anthony Hartley and Margeret King and Keith J. Miller and Mustafa El Hadi and Andrei Popescu-Belis and Florence Reeder and Michelle Vanni. 2002. A Hands-On Study of the Reliability and Coherence of Evaluation Metrics. In *Proceedings of the Workshop Machine Translation Evaluation – Human Evaluators meet Automated Metrics*, Third International Conference on Language Resources and Evaluation LREC 2002, pp. 8-16

Walther v. Hahn and Cristina Vertan. 2002. Architectures of "toy" systems for teaching Machine Translation. *In Proceedings of the 6th EAMT Workshop on "Teaching Machine Translation"*, Manchester, pp. 69-78.

Harald Somers. 2001. Three Perspectives on MT in the Classroom. *In Proceedings of the Workshop on Teaching Machine Translation* VIIIth MT Summit, Santiago de Compostella

Ian Somerville. 1990. *"Software Engineering"*, third edition. Addison-Wesley Publishing Company, Massachusetts

Lorenzo Tessiore and Walther v. Hahn. 2000. Functional Validation of a Machine Interpretation System:Verbmobil. In *Verbmobil: Foundations of Speech-to-speech Translation*, W. Wahlster ed., Springer Verlag, Berlin, pp. 611-634.