On the notion of a Language Products Technology

Gerhard Heyer

Universität Leipzig, Institut für Informatik

Augustusplatz 10-11, 04109 Leipzig

E-mail: heyer@informatik.uni-leipzig.de

Summary

The main task of language products technology is to find solutions to technical problems in all areas of natural language processing, and to implement them in an optimal way given particular linguistic, computational, and fincancial constraints. The key factors that play a role in the most efficient design solutions of language products comprise the aspects *software engineering*, *software ergonomics*, and *linguistic theory*. In addition, individual programmes need to be optimized with respect to at least the main processing parameters *depth* of linguistic processing, *accuracy* in the sense of errors allowed for the recognition process, and *speed* of the recognition process. In order to arrive at the design of a language product, language products technology offers various methods and tools.

1. Language Products Technology

Since it has become increasingly clear during the past years that results from main-stream computational linguistics cannot without further measures be turned into commercially relevant practical applications, the notion of a language products technology as an engineering approach to the design and implementation of language products has received considerable broader attention.

In what follows I shall try to substantiate this notion by proposing and discussing what can be called an *engineering approach* to this problem of a methodology for the construction of natural language processing programmes.

The idea of language products technology as based on the engineering approach may take a as starting point a standard definition of engineering science in general:

"Wesentliche Aufgabe eines Ingenieurs ist es, für technische Probleme mit Hilfe naturwissenschaftlicher Erkenntnisse Lösungen zu finden und sie unter den jeweils gegebenen Einschränkungen stofflicher, technologischer und wirtschaftlicher Art in optimaler Weise zu verwirklichen." (Pahl/Beitz, Konstruktionslehre 1986:1)

When applying this definition to language products, we can conceive of language products technology as a linguistic engineering that uses results from theoretical linguistics and computer science in order to find specific solutions to technical problems in all areas of natural language processing, and implements them in an optimal way given particular linguistic, computational, and financial constraints.

The present day cognitive paradigm in computational linguistics is only of limited use to this engineering approach. The problem, in my view, is *not* that of a transfer from science to technology, but the fact that *different theoretical foundations* are needed for applied computational linguistics than are provided by a computational linguistics oriented towards cognitive linguistics, despite some overlappings in certain areas. The two approaches to natural language processing differ both in methodology and objectives: While cognitive linguistics strives for general solutions based on well established theoretical paradigms with the aim of extending a paradigm to other classes of linguistic phenomena, language products technology is problem driven, and tries to find optimally engineered solutions to problems arising in specific application tasks. In general, language products technology aims at an optimization of theoretical costs and practical benefits, while any such optimization is alien to a purely theory driven concern with

language. In practice, the main task of language products technology is to manage efficiently linguistic complexity in terms of breadth of data and depth of processing required, while from a point of view interested in linguistic competence only, one can rest content when such complexities have been sufficiently described.

Although there appears to be a growing awareness in the scientific community about the fact that small-scale laboratory prototypes cannot simply be turned into language products without additional engineering efforts, there does not appear to be a clear understanding yet of the principle factors that are constitutive of language products technology. The paper is intended to contribute to this discussion about the applicability of a cognitively oriented computational linguistics, and to present a brief sketch of elements of a natural language processing technology.

2. Specific Solutions

The point also holds for the efficiency of the linguistic processing modules themselves, neglecting other hardware and software constraints for the moment. Assuming an architecture where at well defined points during the natural language processing first a syntactic structure, then a semantic structure, and finally a knowledge structure (depending on the application) is created, optimization on the complexity of each structure will not only have draw-backs on its computability (cf. Habel 1988:208), but will also jeopardize the efficiency of the whole system, as structures from one module may be passed on to the next without any provision of adequate further processing. Thus, of all those sentences that can syntactically be parsed in depth, only a limited number can be assigned an equally complex semantics, given the present state of computational semantics, and for all those sentences for which we have, say, an intensional second order predicate logic semantics, we will then have again only a limited number of sentences for which we can provide an adequate knowledge processing. In effect, we leave information generated by other modules wholly, or partially, unused, as it is too complex to be further processed. At any rate, the whole system's behaviour will be less than optimal: Either it will be able to cope in depth with only a very limited set of senteces (determined, in fact, by the

knowledge processing capacities), or it will be able to process its input data in breadth, but in doing so produces a substantial amount of redundant information (Heyer 1990:39).

The notion of efficiency also is central to the development philosophy of natural language processing software. While a holistic design reduces interface problems and increases efficiency, it generally leads to software that is not easily ported, both to other platforms and to other natural languages or applications. In order to increase implementor productivity, the nowadays preferred methodology in computational linguistics is a modular design of natural language processing programmes. (The preferred modular approach also supports nicely the view that a computational linguist only needs to know how to do linguistics on a computer, but does not need to know anything about the way how his formalizations can be efficiently implemented).

What the modular design gains in portability, however, it looses in efficiency, because it assumes, in its extreme form, a level of general and all purpose natural language processing software that is to be used in all kinds of applications. But any such general linguistic processor, quite like Newell and Simon's general problem solver, will always process the data in a specific application much less efficiently than a system taylored to the specific application and based on a holistic design. From an engineering point of view, therefore, neither approach is optimal.

In order to provide a cost efficient basis for all kinds of natural language processing programmes, and as input for tools by which holistic natural language processing solutions can be compiled, I rather imagine an approach that results in multi-functional, reusable linguistic software on all levels of linguistic knowledge, viz. lexica, grammars, and meaning definitions. Let us call this methodology for the construction of natural language processing programmes the *compilation approach* (see figure 1).

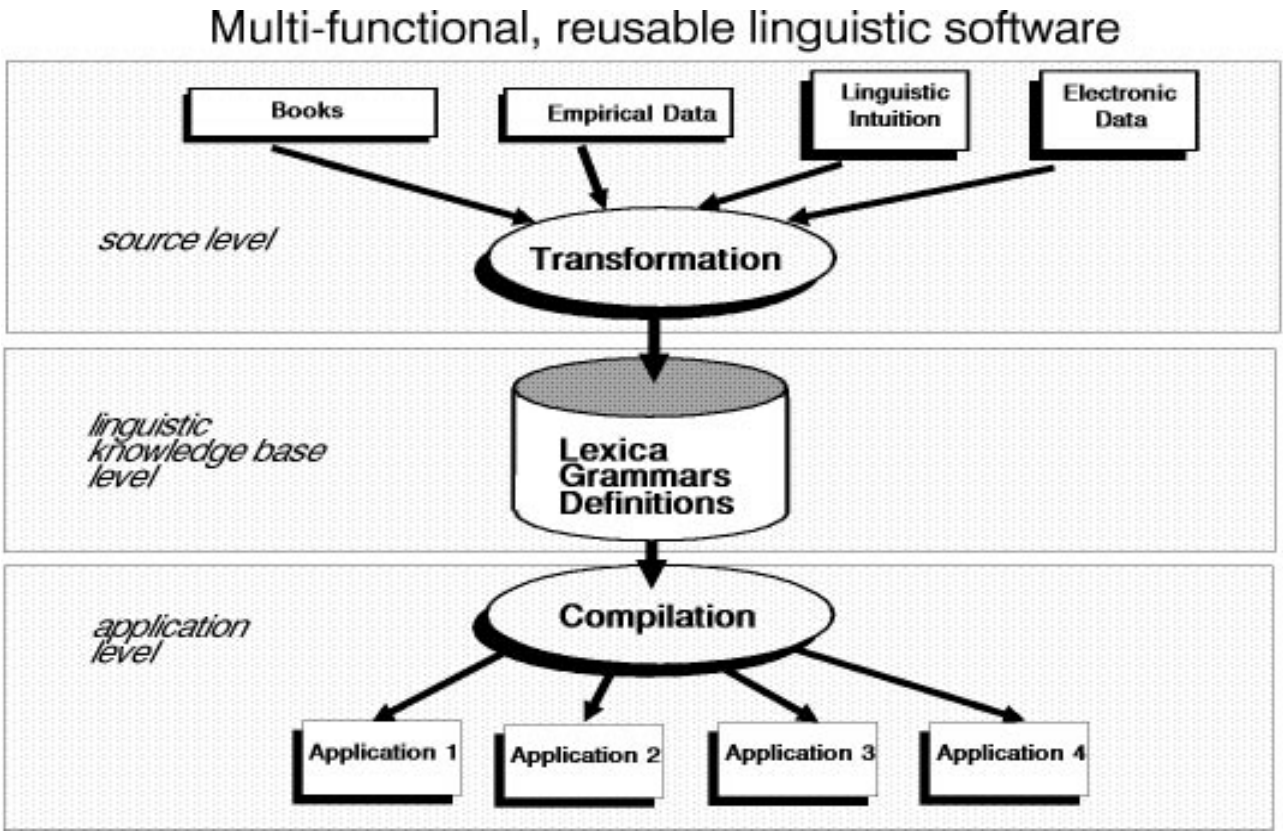

Figure 1

The intuitive idea of the compilation approach is to construct highly efficient and holistically designed natural language applications on the basis of linguistic knowledge bases that contain basic and uncontroversial linguistic data on dictionary entries, grammar rules, and meaning definitions independently from specific applications, data structures, formalizations, and theories.

To the extent that linguistics is a more than two thousand years old science, there is ample theoretical and empirical material of the required kind available in form of written texts and studies on the source level of linguistic knowledge, or can be (and is in fact) produced by competent linguists. However, very little of this knowledge is also already available on electronic media. Thus, the very first task of language products technology, as Helmut Schnelle has recently put it (Schnelle 1991, see this volume), must be the transformation of available linguistic data from passive media into active electronic media, here called lexica, grammars, and definitions on the linguistic knowledge base level. In terms of implementation, such media will mainly be relational, object-oriented, or hypermedia data bases capable of managing very large amounts of data. Clearly,

in order to be successful, any such transformation also requires formalisms to be used on the side of the linguists for adequately encoding linguistic data, the linguistic structures assigned to them, and the theories employed for deriving these structures. Moreover, within linguistics, we need to arrive at standards for each such level of formalization. In actual detail, therefore, the first task of language products technology is quite a substantial one that can only succeed, if the goal of making linguistic knowledge technologically available is allowed to have an impact on ongoing research in linguistics by focussing research on formalisms and standards that can efficiently be processed on a computer.

Now, to complete the picture, assuming that such a linguistic knowledge base is available, individual applications are to be constructed, adapted, or modified on the basis of this linguistic knowledge base by selectively extracting only that kind of information that is needed for building the specific application, and by compiling and integrating it into the application specific data structures. Details, coverage, and the compiled representation of the linguistic information depend, of course, on the individual applications. The second task of language products technology, then, consists in providing a general methodology for such a selection of the required linguistic knowledge, and the definition of its optimal data structure representation.

Although the percentage of required recognition rate may vary from application to application, and can be expected to be much higher for some translation tasks, for example, users apparently do not expect in general to communicate with a computer via a natural language interface as error-free and fault tolerant as is presumed to be the case in human-to-human communication. The finding can be explained, I think, by the fact that natural language human-computer interaction cannot be modelled along the paradigm of linguistic communication among humans, as is often assumed (e.g. Kanngießer 1989), but must be understood as a communication *sui generis*. As Krause has recently shown on the basis of substantial empirical investigations (Krause 1991, cf. also this volume), human-computer communication can be said to differ from ordinary human-human communication at least in terms of the envoked sublanguages and language registers. But if *computer-talk*, a language register comparable to baby-talk or foreigner-talk, leads to simplified and systematically distorted user input, it is plausible to assume that it also leads to a higher tolerance on the user's side towards incomplete or erroneous processing of language data on the side of the computer. From the language products technology point of view, this finding will

have the important consequence that if users do not require a 100% recognition rate of natural language input for a particular application, accuracy of the natural language recognition may be traded against broader linguistic coverage, less memory consumption, increased robustness, or increased speed, provided that the natural language processing modules have been designed in such a way as to optimally support any such trade-off.

4. Linguistic theory and linguistic functionality of language products

Linguistic theory oriented towards the cognitive paradigm assumes as its subject matter the linguistic competence of an "ideal speaker-listener, in a completely homogenous speech-community, who knows his language perfectly and is unaffected by such grammatically irrelevant conditions as memory limitations, distractions, shifts of attention and interest, and errors (random or characteristic) in applying his knowledge of the language in actual performance" (Chomsky 1965:3). This linguistic competence, it is furthermore assumed, can be adequately modelled on functionalist premises within the paradigm of symbolic representation and symbol manipulation as it originated in research on artificial intelligence (for an excellent exposition of the approach see Habel 1986:6 ff.). To the extent, however, that within this orientation the scientific interest in a computational model of language is linguistic competence, and thus more broadly, the human mind and human understanding, it is also assumed as a matter of course that the natural language processing modules draw on a representation of linguistic knowledge that represents linguistic knowledge as general as possible.

From a language products technology point of view, the primary goal is not a general, or presumedly cognitively adequate, representation of linguistic knowledge, but rather an optimally engineered solution to specific, empirically validated, problems in the area of interface or autonomous tasks systems. Here, of course, generality of the linguistic descriptions also is an issue in order to avoid *ad hoc* representations. But although there may be a software engineering level of general representation of linguistic knowledge as indicated above, the main problem for an application oriented linguistic theory is rather to provide a theoretical basis for optimizing trade-offs between different linguistic and non-

linguistic requirements for a particular application (cf. Obermeier 1989: 235 and 100).

In detail, one problem is that users may not require 100% recognition, but instead expect performance improvements with respect to other software quality criteria, as was discussed in the context of ergonomic design of language products.

Thus, from the language products technology point of view, we clearly have an indication here of a reasonable trade-off between theoretical costs and practical benefits, where for the purposes of a particular application the kind and number of errors dealt with could be changed, if there also were an additional classification of errors with respect to importance to the user.

In order to satisfy these requirements on linguistic theory, what is needed, I propose, are grammar formalism that are *gradable* with respect to the depth and accuracy of the linguistic processing. Thus, given a language L, a degree of permissible errors E, a specification of the required depth of analysis D, and a measure of (polynomial) time T, what is needed is a grammar formalism G such that a deterministic automaton can be constructed on the basis of G that decides within the given set of parameters E, D, and T, whether or not any arbitrary expression is a sentence of L, and that does so in such a way that a relaxation of E or D leads to a calculable reduction of T. While this requirement may be strange to some in the context of computational linguistics, it is in fact very natural, and there are many examples in other fields. In telecommunications engineering, for example, there obviously is a trade-off here between the quality of the transmitted signals, i.e. the reduction of noise, and the amount of information (in Shannon's sense) that can possibly be transmitted simultanously. Based on empirical findings, the band-width of frequencies to be transmitted over a telephone line nowadays is restricted in most cases to a range between 1500 and 4500 Hz, although the human ear is capable of perceiving a much broader range. Clearly, engineering requirements of the above kind naturally encourage accepting empirically validated restrictions. In contrast to computational linguistics, however, the theoretical foundations of telecommunications engineering allow for an adequate representation of the kind of trade-offs as discussed above.

Trade-offs between accuracy and efficiency are, of course, also a commonplace in computer science (see Liberherr-Specker 1981 as an example concerning the complexity of partial satisfaction). Surprisingly, however, I do not know of any

theory that explicitly adresses the issue in mainstream computational linguistics, although some frameworks, such as Left-Associative Grammars (Hausser 1991), in view of their complexity and general behaviour appear to be better suited for representing that trade-off than others. Present discussion on efficiency always seems to assume from the start a high degree of grammatical accuracy, and to the extent that it is recognized that a more efficient processing of simple cases may require the relaxation of some general grammatical constraints, it is not clear how the particular processing effort for the simpler cases is in fact reduced.

Without a linguistic theory thus suitable for real applications, language products technology can provide us for the time being only with *heuristics* of how to construct natural language products. On the one hand, grammars and parsers need to be evaluated and classified using parameters such as efficiency, expressiveness, completeness, and decidability of the respective formalisms (cf. Wahlster 1989:216). On the other hand, language technolgoy heuristics will also have to include well-founded advice on the design of the whole system, given specific application tasks, ergonomic considerations, hardware and lingware restrictions. Given that such advice will generally be based on experience, real progress will only be made, once language products technology can build on a theory of natural language processing that conceives of natural language processing applications not as a transfer from theory to practice, but in its foundations and consequences intrinsically supports the engineering approach to natural language processing.

5. Elements of a natural language processing technology

The very idea of language products technology in its present state, no doubt, is still faint and needs to be spelled out in detail. Nevertheless, I think, the problems are getting clearer, and misconceptions can be resolved. By way of conclusion, let us summarize the main elements of a natural language processing technology as it has been discussed above.

The main task of language products technology is to find solutions to technical problems in all areas of natural language processing, and to implement them in an optimal way given particular linguistic, computational, and fincancial constraints. The key factors that play a role in the most efficient design solutions of language

products comprise the aspects *software engineering*, *software ergonomics*, and *linguistic theory*. In addition, individual programmes need to be optimized with respect to at least the main processing parameters *depth* of linguistic processing, *accuracy* in the sense of errors allowed for the recognition process, and *speed* of the recognition process. In order to arrive at the design of a language product, language products technology offers various methods and tools. In my estimation, the key element is a careful empirical evaluation of the ergonomic environment and the required linguistic functionality. While language products technology at present offers at best a number of heuristics on the combination of different processing strategies at various levels, the interdependence between various parameters may also be provided a theoretical basis, and thus could be exploited for a more precise calculation of trade-offs in the design of natural language products.

In contrast to a computational linguistics mainly oriented towards cognitive linguistics, language products technology assumes a different attitude on the side of the user towards the processing of natural language by a computer than by a human. The key difference is the assumption that when human users communicate with a computer, they change to a different language register, so-called *computer-talk*. The attitude of computer-talk concerns, on the one hand, the users own linguistic behaviour, and, on the other, the tolerance by which they judge the computer's natural language processing. Thus, depending on the application, human users may be prepared to accept a computer natural language processing which does not deal with all linguistic phenomena, or a recognition rate that is significantly below 100%.

Economic and technological development crucially depends on a public appreciation of the technical problems to be solved. Let us not forget, finally, that the markets for natural language products are just developing in that sense: Various kinds of natural language products for the first time gain wider acceptance, and by doing so make users of these products more aware of the linguistic problems involved. In the very next future we will see, I expect, an increasing demand for a higher quality of natural language products that, in turn, will stimulate the development of language products technology. However, as natural language products require a very high degree of linguistic literacy and appreciation, in the long run the field of natural language processing applications will be successful only, if the markets show a still higher degree of awareness for linguistic problems and their solutions.

6. References

[1] Carbonell, "Requirements for Robust Natural Language Interfaces: The Language Craft™ and XCALIBUR experiences, Coling: Bonn 1986
[2] Engelien and McBryde, Natural Language Markets: Commercial Strategies, Ovum Ltd: London 1991
[3] Hausser, Complexity in Left-Associative Grammar, to appear in Theoretical Computer Science, Vol. 103
[4] Heid, "Study Eurotra-7: An early outline of the results and conclusions of the Study", Stuttgart 1991
[5] Hellwig/Bläsi, "Theoretical Basis for Efficient Grammar Chechers, Part II: Automatic Error Detection on Correction", ESPRIT Project 2315, Translator's Workbench, Deliverable 1.5: Heidelberg 1990
[6] Heyer, "Probleme und Aufgaben einer angewandten Computerlinguistik", KI 1/90
[7] Heyer/Kese/Oemig/Dudda, "Knowledge Representation and Semantics in a Complex Domain: The UNIX Natural Language Help System GOETHE, Coling: Helsinki 1990, p. 361-363
[8] Heyer/Khatchadourian/Waldhör, Motivation, Goals and Milestones of ESPRIT II Project MULTILEX, Genie Linguistique 1991, Vol 1: Versailles 1991
[9] Krause, Computer Talk, Manuskript, Regensburg 1991
[10] Liebeherr and Specker, Complexity of Partial Satisfaction, Journal of the Association for Computing Machinery Vol. 28 No. 2, 1981
[11] Möller, "GOETHE user study: An empirical evaluation of ergonomic aspects of a natural language user interface", TA TRIUMPH-ADLER AG, Report No. RR-P-91-06: Nürnberg 1991
[12] Obermeier, Natural language processing technologies in artificial intelligence - The science and industry perspective, Ellis Horwood: Chichester 1989
[13] Pahl/Beitz, Konstruktionslehre, Springer: Berlin, Heidelberg, New York 1986
[14] Schnelle, "Beurteilung von Sprachtechnologie und Sprachindustrie", Colloquium Sprachtechnologie und Praxis der maschinellen Sprachverarbeitung, Reimers-Stiftung: Bad Homburg 1991
[15] Wahlster, "Zum Fortschritt in der Computerlinguistik", in: Bátori, Hahn, Pinkal, Wahlster (Hg.), Computerlinguistik und ihre theoretischen Grundlagen, Springer: Berlin, Heidelberg, New York 1989