Generating Multilingual Documents from a Knowledge Base: The TECHDOC Project

Dietmar Rösner
FAW Ulm
P.O. Box 2060, 89010 Ulm, Germany
roesner@faw.uni-ulm.de

Manfred Stede
(University of Toronto and) FAW Ulm
P.O. Box 2060, 89010 Ulm, Germany
stede@faw.uni-ulm.de

Abstract

TECHDOC is an implemented system demonstrating the feasibility of generating multilingual technical documents on the basis of a language-independent knowledge base. Its application domain is user and maintenance instructions, which are produced from underlying plan structures representing the activities, the participating objects with their properties, relations, and so on. This paper gives a brief outline of the system architecture and discusses some recent developments in the project: the addition of actual event simulation in the KB, steps towards a document authoring tool, and a multimodal user interface.

1 Overview

1.1 Project idea

The availability of technical documents in multiple languages is a problem of increasing significance. Not only do consumers demand adequate documentation in their mother tongue; there are also legal requirements, e.g., with respect to the upcoming European common market: the product reliability act forces merchants to offer complete technical documentation in the consumer's native language. The need to provide such a massive amount of multilingual material is likely to exceed both the capacities of human translators as well as those of machine translation technology currently available. Our work in the TECHDOC project is motivated by the feeling that this situation calls for investigating a potential alternative: *to exploit natural language generation technology in order to help overcome the documentation problem.*

TECHDOC operates in the domain of *technical manuals*, which was selected for two principal reasons. On the one hand, they represent "real-world" texts that are actually useful: the domain is *practical* instead of a "toy world". On the other hand, the language that is used in such manuals tends to be relatively simple; one mostly finds straightforward instructions that have been written with the intention to produce text that can be readily understood by a person who is executing some maintenance activity. Moreover, as our initial analyses in the first phase of TECHDOC had shown, the *structure* of manual sections is largely uniform and amenable to formalization.

1.2 Outline of the generation process

TECHDOC produces maintenance instructions in English, German and French. The system is based on a KB encoding technical domain knowledge as well as schematic text structure in LOOM, a KL-ONE dialect [LOOM, 1991]. The *macrostructure* of a manual section is captured by schemas saying that (if appropriate) one first talks about the location of the object to be repaired/maintained, then about possible replacement parts/substances; next, the activities are described, which fall into the three general categories of checking some attribute (e.g., a fluid level), adding a substance and replacing a part/substance. These actions are represented as plans in the traditional AI

sense, i.e. with pre- and postconditions, and with recursive structure (steps can be elaborated through complete refinement plans).

These representations are mapped onto a language-independent document representation that also captures its *microstructure* by means of RST relations [Mann and Thompson, 1987] with a number of specific annotations (e.g., a proposition is to be expressed as an instruction, giving rise to imperative mood). This document representation is successively transformed into a sequence of sentence plans (together with formatting instructions in a selectable target format; SGML, LaTeX, Zmacs and — for screen output — slightly formatted ASCII are currently supported), which are handed over to sentence generators. For English, we use 'Penman' and its sentence planning language (SPL) as input terms. To produce German and French text, we have implemented a German version of Penman's grammar (NIGEL), which is enhanced by a morphology module, and a fragment of a French grammar in the same way.

For a more detailled description of the system architecture see [Rösner and Stede, 1992b].

2 The Knowledge Base

The Knowledge Base is encoded in LOOM. In addition to the standard KL-ONE functionality (structured inheritance, separation of terminological and assertional knowledge), LOOM supports object-oriented and also rule-based programming.

In addition to the 'Upper Model' of the Penman generator (a basic ontology that reflects semantic distinctions made by language, [Bateman, 1990]) more than 1000 concepts and instances constitute the TECHDOC KB. They encode the technical knowledge as well as the plan structures that serve as input to the generation process. The domains currently modeled are end consumer activities in car maintenance and some technical procedures from an aircraft maintenance manual.

One of the central aims in the design philosophy of the TECHDOC knowledge base is the separation of domain-independent technical knowledge and specific concepts pertaining to the particular domain: the portability of general technical knowledge has been a concern from the beginning. For instance, knowledge about various types of *tanks* (with or without imprinted scales, dipsticks, drain bolts) is encoded on an abstract level in the inheritance network (the 'middle model'), and the particular tanks found in the engine domain are attached at the lower end. Similarly, we have an abstract model of *connections* (plugs, bolts, etc.), their properties, and the actions pertaining to them (plug-in connections can be merely connected or disconnected, screw connections can be tightly or loosely connected, or disconnected). Objects with the functionality of connections (e.g., spark plugs) appear at the bottom of the hierarchy. Thus, when the system is transferred to a different technical domain — as experienced recently when we moved to aircraft manuals —, large parts of the abstract representation levels are re-usable.

3 Document Representation Using RST

The first task undertaken in TECHDOC was a thorough analysis of a corpus of pages from multilingual manuals in terms of *content* as well as *structure* of the sections. A text representation level was sought that captured the commonalities of the correponding sections of the German, English and French texts, i.e. that was not tailored towards one of the specific languages (for a discussion of representation levels in multilingual generation, see [Grote *et al.*, 1993]). Rhetorical Structure Theory (RST) turned out to be a useful formalism: for almost every section we investigated, the RST trees for the different language versions were identical.

Our work with RST gave rise to a number of new discourse relations that we found useful in analyzing our texts. Also, we discovered several general problems with the theory, regarding the status of minimal units for the analysis and the requirement that the

text representation be a tree structure all the time (instead of a general graph). These and other experiences with RST are reported in [Rösner and Stede, 1992a].

4 Recent Developments

4.1 Event simulation in the knowledge base

We developed a detailled representation of knowledge about actions. Together with an action concept, preconditions and postconditions can be defined in a declarative way. The preconditions can be checked against the current state of the knowledge base (via LOOM's ASK queries). If the preconditions hold, the action can be performed and the postconditions are communicated to the knowledge base (with the TELL facility of LOOM). This typically leads to reclassification of certain technical objects involved. With the help of LOOM's production rule mechanism, additional actions either in the knowledge base or on an output medium (e.g., for visualization) can be triggered. In this mode, instruction generation is a by-product of simulating the actions that the instructions pertain to.

Being able to take the current state of a technical device into account, as in this simulation mode, is a prerequisite for upcoming interactive applications of instruction generation: devices equipped with adequate sensory instruments produce raw data that can be fed directly into the knowledge base. Thereby, the specific situation of the device, e.g., the car, drives the instruction generation process, so that only the truly relevant information is given to the user.

4.2 Towards a document authoring tool

A first version of an authoring tool has been designed and implemented and tested with a number of users. The authoring tool allows to interactively build up knowledge base instances of maintenance plans, including the actions and objects involved, and to convert them immediately into documents in the selected languages. At any time, the tool takes the current state of the knowledge base into account: all menus offering selections dynamically construct their selection lists, so that only options of applicable types are offered.

4.3 From text generation to a multimodal information system

The generated texts are now displayed with words, groups and phrases and whole sentences being mouse-sensitive and — when selected — offering menus with applicable queries to be directed to the underlying knowledge base instances. This allows for a number of tasks to be performed on the generated surface texts, for example:

- pronouns can be asked about their antecedent referent,
- linguistic items in the output for one language can be asked about their corresponding items in the other languages output,
- objects can be asked about their location, answered by a suitable graphic illustration,
- actions can be asked for more detailled instructions on how to perform them, answered by a short video sequence.

In essence, these facilities have paved the way to move from static, inactive strings as output to an active and dynamic interface for the associated knowledge sources and their various presentation modalities. The key is that all information types (lexemes in various languages, images and object's location therein, and video sequences) are associated with the underlying KB instances, which are in turn linked to their referents in the mouse-sensitive output text. Figure 1 shows a sample screen, where the user has just asked for additional "location" information about the dipstick, by clicking on the word in one of the text output windows.

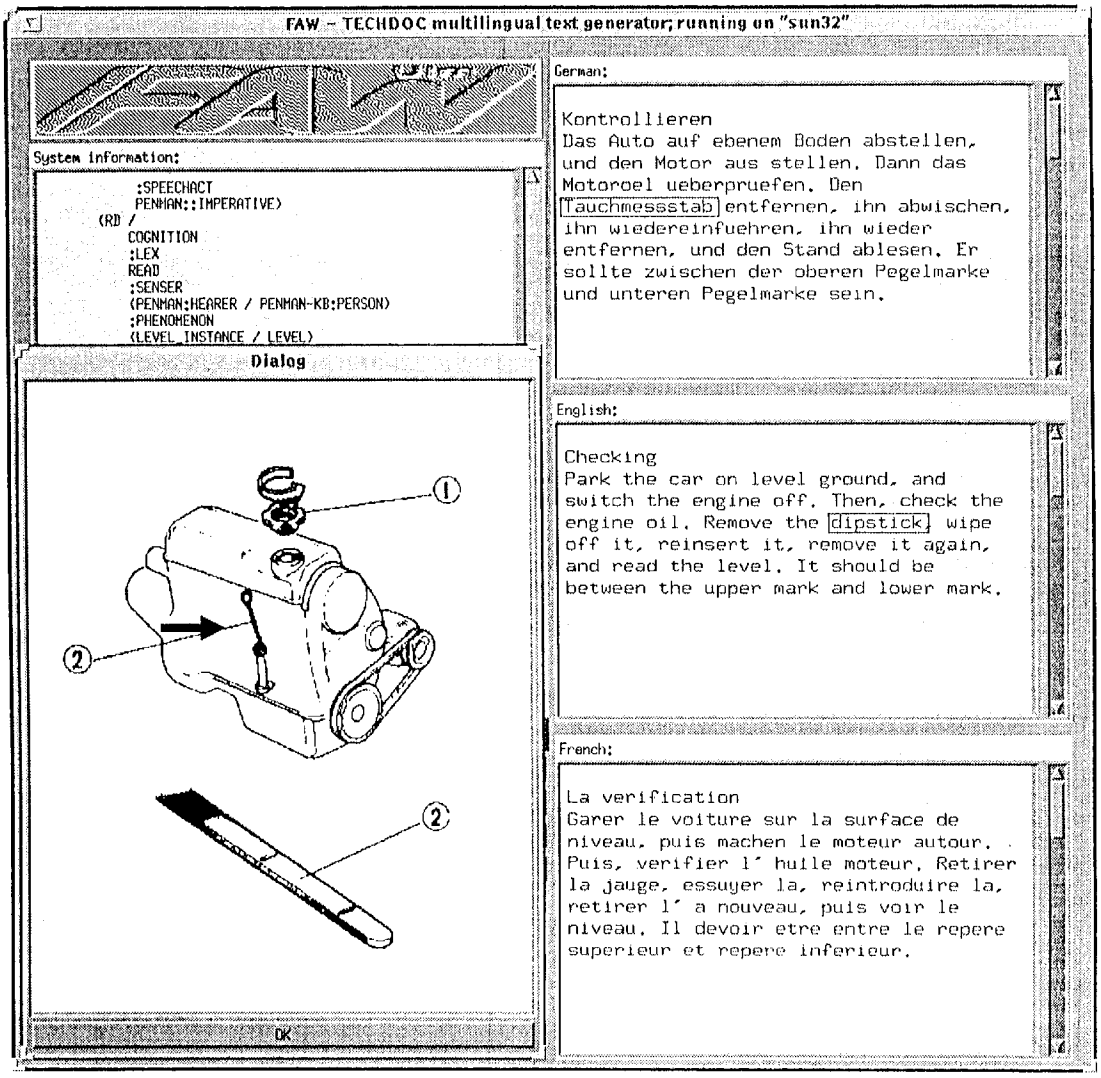

Figure 1: Trilingual output and interactive graphic support

4 RHETORICAL STRUCTURE EXTRACTION

The rhetorical structure represents logical relations between sentences or blocks of sentences of each section of the document. A rhetorical structure analysis determines logical relations between sentences based on linguistic clues, such as connectives, anaphoric expressions, and idiomatic expressions in the input text, and then recognizes an argumentative chunk of sentences.

Rhetorical structure extraction consists of six major sub–processes:

(1) **Sentence analysis** accomplishes morphological and syntactic analysis for each sentence.

(2) **Rhetorical relation extraction** detects rhetorical relations and constructs the sequence of sentence identifiers and relations.

(3) **Segmentation** detects rhetorical expressions between distant sentences which define rhetorical structure. They are added onto the sequence produced in step 2, and form restrictions for generating structures in step 4. For example, expressions like "...3 reasons. First, ... Second, ... Third, ...", and "... Of course, ...But, ..." are extracted and the structural constraint is added onto the sequence so as to form a chunk between the expressions.

(4) **Candidate generation** generates all possible rhetorical structures described by binary trees which do not violate segmentation restrictions.

(5) **Preference judgement** selects the structure candidate with the lowest penalty score, a value determined based on preference rules on every two neighboring relations in the candidate. This process selects the structure candidate with the lowest penalty score, a value determined based on preference rules on every two neighboring relations in the candidate. A preference rule used in this process represents a heuristic local preference on consecutive rhetorical relations between sentences. Consider the sequence `[P <EG> Q <SR> R]`, where `P`, `Q`, `R` are arbitrary (blocks of) sentences. The premise of `R` is obvously not only `Q` but both `P` and `Q`. Since the discussion in `P` and `Q` is considered to close locally, structure `[[P <EG> Q] <SR> R]` is preferable to `[P <EG> [Q <SR> R]]`. Penalty scores are imposed on the structure candidates violating the preference rules. For example, for the text in Fig. 1, the structure candidates which contain the substructure `[3 <EG> [[4 <EX> 5] <SR> 6]]`, which says sentence six is the entailment of sentence four and five only, are penalized. The authors have investigated all pairs of rhetorical relations and derived those preference rules.

The system analyzes inter–paragraph structures after the analysis of intra–paragraph structures. While the system uses the rhetorical relations of the first sentence of each paragraph for this analysis, it executes the same steps as it does for the intra–paragraph analysis.

5 ABSTRACT GENERATION

The system generates the abstract of each section of the document by examining its rhetorical structure. The process consists of the following 2 stages.

(1) Sentence evaluation

(2) Structure reduction

In the *sentence evaluation* stage, the system calculate the importance of each sentence in the original text based on the relative importance of rhetorical relations. They are categorized into three types as shown in Table 2. For the relations categorized into *RightNucleus*, the right node is more important, from the point of view of abstract generation, than the left node. In the case of the *LeftNucleus* relations, the situation is vice versa. And both nodes of the *BothNucleus* relations are equivalent in their importance. For example, since the right node of the serial relation (e.g., *yotte* (thus)) is the conclusion of the left node, the relation is categorized into *RightNucleus*, and the right node is more important than the left node.

The Actual sentence evaluation is carried out in a demerit marking way. In order to determine important text segments, the system imposes penalties on both nodes for each rhetorical relation according to its relative importance. The system imposes a penalty on the left node for the *RightNucleus* relation, and also on the right node for the *LeftNucleus* relation. It adds penalties from the root node to the terminal nodes in turn, to calculate the penalties of all nodes.

Then, in the *structure reduction* stage, the system recursively cuts out the nodes, from the terminal nodes, which are imposed the highest penalty. The list of terminal nodes of the final structure becomes an abstract for the original document. Suppose that the abstract is longer than the expected length. In