Converting the Scania Framemaker Documents to TEI SGML

Erik F. Tjong Kim Sang
Department of Linguistics
University of Uppsala
erik.tjong@ling.uu.se

September 17, 1996

Contents

1 Introduction

The Scania corpus is a multilingual collection of truck maintenance manuals of the Swedish company Scania CV AB. The Department of Linguistics of the University of Uppsala in cooperation with Scania will use this corpus for developing translation support tools. The documents in the corpus were constructed with the word processing program Framemaker [Frame 1993]. The internal document format used by Framemaker makes linguistic processing of the Scania documents very hard. Furthermore the document format prevents exchanging these documents among the different computing platforms that are being used in our department. Therefore we have decided to convert the documents to a format which would make the exchanging and processing tasks easier.

The Text Encoding Initiative (TEI) Guidelines address the problem of exchanging documents in electronic format [SB 1994]. The proposals put forward in these guidelines are becoming a standard in corpus design. Many text corpora are currently encoded in accordance with these guidelines, for example the Multilingual Corpus I produced by ECI [Thompson 1995]. The possibilities of processing and exchanging these corpora are better than with Framemaker documents. Furthermore documents encoded in accordance with these guidelines can be processed by software that already exists for handling this type of documents. For these reasons we have decided to convert the Scania Framemaker documents to documents in the TEI format.

In this report we will describe the conversion process of Scania Framemaker documents to text structures that conform with the TEI Guidelines. We will start with an outline of the text structures that are available in the Framemaker documents. Some of these text structures are interesting for linguistic processing and we will describe what TEI tags we will use to encode these structures. After this we will present available software for the conversion process: the program MifMucker by Ken Harward. We will show what extensions had to be made to the software and what additional programs were necessary to insert extra TEI tags in the texts. Finally we will present some conclusions about this work.

2 The format of the documents

We want to perform linguistic processing of texts that are available in binary Framemaker documents [Frame 1993]. A relatively easy way for making these documents usable would be to extract only the text and ignore all formatting information. However by keeping part of the formatting information we can make the later linguistic processing of these files easier. In this chapter we will outline what formatting information we want to keep and how we will encode it according to the TEI Guidelines.

2.1 Interesting text structures in the Framemaker documents

In the first phase of the Scania corpus project we will attempt to align the corresponding sentences of the different language versions with each other. For this purpose it is necessary that every sentence boundary in the document is recognizable. Automatically finding these sentence boundaries is hard. One cannot rely on punctuation marks because these will often be omitted. Examples of places where punctuation marks are omitted are lists with small elements and headers such as for example the header of this section. We would like to use all available information for deciding where the sentence boundaries are. The formatting information in the documents provides such information.

The Scania Framemaker documents contain formatting information that is worthwhile to keep. This information can be made visible by converting the binary Framemaker documents to Maker Interchange Format (MIF), an interchange format used by Frame products. Facilities for making this conversion step are available with standard Frame software. When one examines a file in MIF format one will discover that the document contains many text formatting tags. Some tags like pages, paragraphs and tables are easily recognizable but others such as headers and lists have a less trivial encoding.

The text formatting information that we want to keep in the documents is the following:

page tags The page boundary information will be very important when the different language versions of documents will be aligned. The translations in this collection of documents has been done page by page. This makes page boundaries valuable information in the process of deciding which sentence was translated in which other sentence. The page information has been encoded in the documents explicitly and is easy to obtain.

paragraph tags Unfortunately the paragraphs in the different language versions of the documents do not correspond exactly with each other. Still we want to keep the paragraph information because it helps to structure texts within a page and because it is easy to obtain from the Framemaker documents.

header tags The headers in a document in MIF appear in separate paragraphs. It is possible to recognize a header paragraph from the format definition tag in the paragraph. An end of header in these documents should be interpreted as an end of sentence so the header boundaries are useful information. We will also keep the section level information that is encoded in the header definitions (like section headers versus subsection headers) and we will use this information for dividing the texts in different sections.

table tags Tables can easily be recognized from the MIF documents. They consist of cells which may contain words, phrases and sentences. All this information needs to be aligned and doing that without the cell boundary information will be hard. So it is a good idea to keep table formatting information.

list tags The story for lists is the same as for tables. They can easily be recognized in MIF documents. Since they may contain phrases which do not end with a punctuation mark it is a good idea to keep the list formatting information for usage in the alignment process.

One can argue about whether it is necessary to keep information about headers, lists and tables and ask if it would not be equally usable to surround the items in these structures by sentence tags. However in the later analysis of the corpus one may want to know why a particular structure was marked as a sentence. At that moment it will useful to have the header, list and table information in the corpus.

2.2 Text structures used in the TEI documents

The Text Encoding Initiative (TEI) Guidelines define an encoding scheme for documents in electronic form [SB 1994]. These guidelines propose to use a standard set of text formatting tags in order to make it easier to exchange documents between different computational platforms. The tags put forward by the guidelines are elements of the Standard Generalized Markup Language (SGML). The complete TEI Guidelines are very big. We will not need all the tags defined in these guidelines. The tags we will need in the TEI versions of the documents can all be found in the subset TEI Lite.

In the previous section we have described the necessity of keeping text structure information about pages, paragraphs, headers, tables and lists in the documents. Apart from that we will also need to add sentence boundary tags because we will perform a sentence alignment of these documents in the future. Information about sentence boundaries is not present in the Framemaker documents. This means software needs to be developed for adding this information to the documents. Finally we will also need the standard TEI document definition tags. These can be put in a header and a footer file which will be the same for all documents.

A complete list of the tags used in the TEI versions of the Scania documents can be found in Appendix C.

3 Conversion software

Some public domain software for converting binary Framemaker documents to ASCII or HTML was available. This software needed to be modified to make it generate SGML. Furthermore some extra conversion software needed to be developed for inserting sentence tags, handling abbreviations and speeding up the conversion process.

3.1 The conversion program MifMucker

MifMucker is a collection of filters which can be used to manipulate Framemaker document in Maker Interchange Format (MIF) [Harward 1994]. Among other things this software can convert Framemaker binary documents to ASCII or to HTML. It is written in the script language Perl. Mifmucker requires the program Framemaker to be able to run. It will use Framemaker for converting binary Framemaker documents to MIF documents and vice versa.

MifMucker is able to convert MIF documents to documents in HTML (Hyper-Text Markup Language), the encoding language used for documents at World Wide Web. The format of HTML tags is similar to TEI tags since both markup languages are a variant of SGML. We were able to use the code already available in the HTML conversion script and restructure this into code that generated output marked up with TEI tags. Two other parts of the software needed to be modified. The first one was the font definition in which a definition for a soft hyphen needed to be added. The second one was the table processing script. It was only able to generate ASCII tables and those needed to become TEI tables.

In the later phase of the project we discovered that there was a bug in the MifMucker software which caused problems in the order of the paragraphs in the output. For our future alignment work it is very important that the paragraph order in the TEI Lite documents is exactly the same as in the binary Framemaker documents. Therefore we have tried to find the location of the bug in the software. It turned out that the software neglected the paragraph order information that is available in MIF. MifMucker simply assumed that the order of the paragraphs in the MIF document was the same as the order in the binary Framemaker document. This assumption is not correct for the documents in our corpus. We have added extra code to the paragraph handling script and thereby removed the paragraph order error.

3.2 Additional conversion software

The modified MifMucker software abled us to extract text and formatting information from the Framemaker documents and to convert it to TEI tags. The sentence boundaries were not marked in the Framemaker document but we have modified the software in such a way that it was able to insert sentence bounday tags at obvious locations.

One of these obvious locations was behind a period at the end of a word. However a construction like this can also be caused by an abbreviation and in that case adding a sentence boundary might be wrong. We have attempted to minimalize this problem by removing the sentence boundaries after abbreviations which are infrequent in sentence final positions.

An extra problem was that the conversion of one binary Framemaker document to TEI costs about fifteen minutes of computing time at of of our IBM RS/6000 workstations running AIX 4.1.4. Converting our current corpus of 720 files would require seven and a half day. To speed up the conversion process we have developed software which uses fourteen of our workstations for performing the conversion process in parallel. This cuts down the time required for converting the corpus to approximately thirteen hours.

4 Concluding remarks

We have developed software for converting Framemaker documents to texts which are marked up with TEI Lite tags. The main software is an extension of Mifmucker by Ken Harward. This software was able to convert binary Framemaker documents to HTML. We have modified it in such a way that it managed to convert the Framemaker documents to TEI. This public domain program contained a paragraph order bug which we have fixed. The other software that we have developed for the conversion process is an abbreviation recognition script and a script that distributes the conversion process over multiple machines. The software was developed and tested on IBM RS6000 machines running AIX 4.1.4 and it requires both Framemaker and Perl 4.0 (5.0 doesn't work). No attempt has been made to make this software portable to other platforms.

References

[Clark1995] James Clark. *SP, SGML Parser.* 1995. Available from ftp://ftp.jclark.com/pub/sp/

[Frame1993] Frame Technology Corporation. *Using Framemaker, X Windows System.* September 1993.

[Harward1994] Ken Harward. *MifMucker, Application for manipulating Frame documents and books.* 1994. Available from http://www.oac.uci.edu/indiv/ehood/mifmucker.doc.html

[SB1994] C.M. Sperberg-McQueen and Lou Burnard. *Guidelines for Electronic Text Encoding and Interchange.* 1994. Available from http://etext.virginia.edu/TEI.html

[Thompson1995] Henry S. Thompson. *The ECI Multilingual Corpus I.* 1995. Available from http://www.cogsci.ed.ac.uk/elsnet/eci.html

A Determining the sentence boundaries

The Framemaker documents from Scania do not contain sentence boundary tags. One of the first tasks we will perform on the corpus is sentence alignment and for that task we need a corpus in which sentence boundaries have been marked explicitly. The corpus is too large to insert these sentence boundary tags manually. Unfortunately there is no perfect automatical method for determining the positions of sentence boundaries. The only thing we can do is to develop a set of rules that generate as few errors as possible.

Before we outline the sentence boundary rules we have used in this corpus we must define what a sentence is. In ordinary language use a sentence can be defined as a sequence of words from which the first starts with a capital character and the last ends with a punctuation mark. Apart from the fact that this definition covers structures that are larger than sentences as well it will cause some other problems when used in technical texts:

1. What about headers? These text structures do not necessarily end in a punctuation mark. This means that the naive sentence definition definition would consider them as a part of the first sentence of the next paragraph.
2. What about table elements? A table cell may contain a sentence but it may also contain a phrase, a number or a word without a punctuation mark. The sentence definition would combine all these phrases, words and numbers into long sentences.
3. What about list elements? A list element may be a sentence, a phrase, a word or a number. The sentence definition would combine all the elements without punctuation marks in one long sentence.

Some other problematic text structure are contents lines (like headers) and figure captions (like list elements). All these structures have in common that they might contain complete sentences but they can also contain phrases which neither need to start with a capital character nor need to finish with a punctuation mark. The naive sentence definition will group sequences of these structures in large sentences. This is not the behavior we are looking for.

We are fortunate enough that all these problematic text structures can easily be recognized from the Frame formats. This means that we can do something with them. A first idea would be to ignore them. We are building a corpus for linguistic analysis and phrases and separate words might not be very interesting for people that, for example, want to develop a syntactic theory. However the words in the phrases contain important lexical information so we cannot simply throw them away. We have to keep them and the only question is how to tag them.

Our approach to the sentence boundary problem has been the following. We started with extending the naive sentence definition with the constraint that a sentence cannot contain another sentence. This is no problem in this corpus of technical texts since the corpus does not contain quotations. After this we have used this definition to insert sentence boundaries into text structures which contained more than one sentence. This left us with some parts of the texts which were not embedded in a sentence. Our final step was to tag all these structures as sentences by using the nearest text structure tags as the positions for the sentence boundary tags.

The insertion of sentence boundaries requires three rules:

1. Within text structures: insert an end-of-sentence tag and a begin-sentence tag after a punc-

tuation marks (.?!) that are followed by a space.

2. Before a text structure: insert a begin-sentence tag.
3. After a text structure: insert an end-of-sentence tag.

In these rules a text structure is a sequence of zero or more words that is delimited by TEI tags but does not contain tags itself. A TEI tag is a sequence of characters that starts with <, ends with > without containing one of these two characters between the initial and the final character. The begin-sentence tag is <s> and the end-of-sentence tag is </s>.

An example: the following text has been marked up with TEI tags that were derived from the Framemaker document. It contains a paragraph and the first item of a list.

> En skruv kan inte fortsätta belastas i all evighet med fortsatt längdutvidgning och ökad förspänning som resultat. Vid en viss spänning uppnås sträckgränsen i skruvens material. Om skruven belastas ytterligare sträcks den okontrollerat, förspänningen släpper och skruven går till slut av. <list><item><label>1</label>Åtdragning med moment</item>

The sentence tags need to be added to this text. We will use the three rules for doing that. The first rule will add sentence boundary tags after *resultat.* and *material.* No tags will be added behind *av.* because there is no space behind this word. The second rule will add a begin-sentence tag before *En*, *1* and *Åtdragning*. The third rule will add an end-of-sentence tag after *av.*, *1* and *moment*. The final text with some extra newlines will look like:

>
> <s>En skruv kan inte fortsätta belastas i all evighet med fortsatt längdutvidgning och ökad förspänning som resultat.</s>
> <s>Vid en viss spänning uppnås sträckgränsen i skruvens material.</s>
> <s>Om skruven belastas ytterligare sträcks den okontrollerat, förspänningen släpper och skruven går till slut av.</s>
>
> <list>
> <item>
> <label><s>1</s></label>
> <s>Åtdragning med moment</s>
> </item>

Now all text structures consist of one or more sentences.

Applying these rules on the corpus creates two problems. The first one appears in tables that contain sentences which occupy more than one line. These sentences will be split in several sentences. An example of this can be found in the next table:

diameter	weight	comment
26 28	21 23	
30	25	A comment that is too long will be spread over several lines

In the Framemaker documents we have found many examples of table elements which should be in a cell of their own but instead share a cell with some other element, like for example the pair 26-28. In order to be able to address the elements individually we have entered line break tags in the table cell, for example between 26 and 28. This will cause items like these to appear in a sentence of their own (sentence boundary rules 2 and 3). The disadvantage of this is that long sentences in tables will be split, like in the example after *will*. We believe that this problem will be rare in this corpus so we did not try to get rid of it.

The second problem is that abbreviations that end on a period will always be recognized as sentence boundaries (sentence boundary rule 1). We will deal with this problem in the next section.

B Handling abbreviations

One of the most important tasks in the conversion process was determining the positions of the sentence boundaries and putting sentence boundary tags at those positions. One of the heuristic rules used for recognizing sentence boundaries was that a period at the end of a word marked that word as the final word of a sentence. When one applies this rule every abbreviation that ends with a period will be regarded as a sentence final word. To account for this we have developed a special script to do some post-processing after the conversion to the TEI format has been done. The script removes the sentence boundaries behind abbreviations that do not occur in sentence final position frequently.

The problem of finding abbreviations in a 1.6 million word corpus in eight languages is by no means trivial. In order to solve this problem we have assumed that abbreviations in technical manuals tend to appear in clusters on a page rather than isolated. Furthermore we have assumed that abbreviations in a text in one language occur independently of abbreviations in the translation of the text. By combining these two assumptions we concluded that we could expect that after conversion to TEI a page with abbreviations was likely to contain a different number of sentences than the corresponding page in another language.

This final conclusion made our task of finding the abbreviations easier. We only needed to check the pages of which the number of sentences was markedly different from the corresponding pages in other languages. We have counted the sentences on every page in the TEI documents (script `alignCheck`) and have checked every page and its corresponding Swedish page when their number of sentences differed four or more sentences.

The abbreviations that we have found in this way have been listed on the next page. The script will remove the sentence boundaries immediately behind these abbreviations. It has been included as standard part of the post-processing software of the conversion process.

One note for German: It seems that in German '1.' is used meaning first. However this can also be a sentence ending in one or a list label. Constructions like this one have not been handled by the post-processing software.

Dutch (27)

Afb.	Art.nr.	B.D.P.	b.v.	bladnr.	Ca.	cil.	D.w.z.
e.d.	etc.	Fig.	incl.	m.b.v.	Max.	Min.	Nr.
o.m.	pag.	Pos.	r.p.	relaispos.	resp.	sec.	st.
t.e.m.	t.o.v.	Temp.					

English (19)

cyl.	E.g.	etc.	Fig.	I.e.	ind.	km/h.	Loc.
Max.	Min.	mm.	Nm.	No.	r.p.	rpm.	Temp.
V.	V.R.	W.					

Finnish (22)

Akku.	esim.	eykk.	jykk.	Ko.	Ks.	läpim.	Maks.
man.	Min.	n.	n.k.	ns.	o.	pit.	Pos.
syl.	Sähk.	ts.	täyd.	v.	ym.		

French (11)

Cond.	d'env.	e.r.	emp.	env.	etc.	Fig.	pos.
réal.	Sousdim.	Temp.					

German (28)

Abb.	Alt.	Art.Nr.	Blattnr.	bzw.	ca.	d.h.	etc.
l.	Max	. Min.	Niedr.	Nr.	O.T.	o.ä.	Pos.
Pos.lichter.	r.	St.	Temp.	u.a.	u.U.	u.ä.	usw.
v.o.T.	z.B.	zus.	Zyl.				

Italian (63)

accel.	Ant.	Blocc.	cap.	chius./apert.	cil.	circ.	Cod.
comb.	Conf.	conn.	Cont.	Contachil.	contr.	diam.	dif.
diff.	disins.	disinser.	Distr.	ecc.	elettron.	entr.	es.
est.	ev.	Fig.	Illumin.	Impostaz.	indicat.	Ins./disins.	Limit.
longit.	Lubr.	lungh.	man.	Min.	motr.	n.	Nr.
Optic.	pag.	pagg.	parch.	part.	parz.	pf.	pneum.
Pos.	posiz./posiz.	Post.	Regol.	Regolaz.	Riscald.	sec.	sin.
Sollev.	Strum.	Teler.	Temp.	usc.	ved.	Vel.	

Spanish (75)

1er.	2o.	3er.	acel.	acopl.	acopl./desac.	Acopl./desacopl.	adv.
alim.	Alt.	Aprox.	Art.	cab.	Calent.	Cil.	circ.
col.	cond.	conm.	cst.	Cuentakil.	Cód.	del.	Dep.
der.	desacopl.	Diagr.	dif.	ej.	electr.	elev.	eléc.
embal.	estac.	etc.	Fig.	func.	incl.	ind.	Indic.
inf.	instr.	interm.	Interr.	izq.	izqu.	lim.	long.
Lámp.	marc.	Máx.	Mín.	neum.	No.	p.	parcial/Ind.
port.	pos.	post.	progr.	prop.	pág.	r.p.	reduc.
reduct.	regul.	rev.	Subdim.	sup.	Susp.	Temp.	transf.
tras.	vel.	z.r.					

Swedish (26)

alt.	bl.a.	etc.	Ev.	ex.	h.	Hast.	ind.
inkl.	Instr.	man.	Max.	Min.	min.	Nr.	r.p.
r/min.	resp.	t.ex.	t.o.m.	Underdim.	Utf.	V.	V.R.
ö.d.	Överdim.						

C The TEI tags in the corpus

The corpus has been marked up by using SGML tags from the TEI subset TEI Lite. Apart from the document header the documents in the corpus consist of three parts: a front part, a body part and a back part. This is the general structure of a TEI document. The front and back parts has been left empty in our corpus. The body part contains the actual document text. The back part will be used in the future for storing the sentence alignment information.

Within the body part we have used twelve TEI tags for encoding text structure information. These tags are:

** ** These tags divide the text in structures or divisions like sections and subsections. These tags could be derived from the Frame format easily by using the level information that was encoded in the Frame header tags. Actually there are four of these division tags: <div1>, <div2>, <div3> and <div4>. This tag contains attribute *type* because it is required by the sgml validator that we use.

<head> These tags mark headers like section headers and subsection headers. The level of the headers is equal to the level of the smallest text division that it is part of.

<p> The tag for paragraphs

<s> The tag for sentences. In our corpus this tag is also being used for marking up phrases and words which otherwise would not have been part of a sentence.

<pb> The tag for page breaks. Page breaks could easily be extracted from the Framemaker documents. They will be important when aligning the sentences of the documents since the translation of these texts has been done page by page.

<lb> The tag for line breaks. This tag was necessary in tables since the documents contained an internal table cell division which could only be made visible with line breaks. There is a discussion about this at the end of appendix A.

<table> The tag for tables. Tables are divided in rows and each row is divided in cells. The table tag contains an argument *id* which could easily be derived from the Framemaker documents but serves no purpose. It should be deleted in future.

<row> The tag for table rows. Each table row is divided in cells

<cell> The tag for cells in table rows.

<list> The tags for lists. A list is divided in items and each of these items contains a label.

<item> The tag for items of lists. Each item contains a label and some text data.

<label> The tag for the label of an item in a list.

Apart from the document text marked up with these tags the documents contain a standard header and a standard footer. The header contains one unique part which is the document name which has been encoded in the argument *n* of the tag *teiheader*. An example of the document header and footer:

```
<!DOCTYPE TEI.2 PUBLIC "-//TEI//DTD TEI Lite 1.0 //EN">
<TEI.2>
<teiheader n="000103sv.01.sgml">
<fileDesc>
<titleStmt>
<title>
</title>
<author>
</author>
</titleStmt>
<publicationStmt>
<publisher>
</publisher>
</publicationStmt>
<sourceDesc>
<bibl>
</bibl>
</sourceDesc>
</fileDesc>
</teiheader>
<text>
<front>
</front>
<body>
...
</body>
<back>
</back>
</text>
</TEI.2>
```

The document text will be put between the begin-body and the end-of-body tags. Examples of this document format can be found on World Wide Web at the url http://www.ling.uu.se/corpora/rf/ This is an overview of a project in which the documents were marked up with TEI SGML by using the same format as described in this report.

D Using the software

All the software that has been developed in this project is available on the machine strindberg.ling.uu.se in the directory /usr/users/staff/corpora/bin . One needs to have access to this machine and be member of the research group corpora at the machine in order to have access to the software.

The main conversion program is the script `fm2sgml`. This script can be used as follows:

```
/usr/users/staff/corpora/bin/fm2sgml FrameMakerFile
```

in which `FrameMakerFile` is a Framemaker file. The result of the conversion process will be stored in the file `FrameMakerFile.sgml` in the current working directory. The script will first call mifmucker

to perform the conversion process. The mifmucker program has been stored in the same directory and the scripts used by this program can be found in the subdirectory mm. The conversion process performed by mifmucker is not perfect. Thus the second step of `fm2sgml` is to run a program to perform some post-processing on mifmucker's output. The post-processing program `postSgml` performs different actions:

1. Combining words which were split by line breaks (script `sgmlHyphen`).
2. Adding the TEI header information to the document.
3. Putting each SGML tag on a separate line.
4. Handle some SGML syntax errors which were left by mifmucker.
5. Take care of the abbreviations which had been recognized as sentence boundaries (script `abbrev`, also see appendix B).

The conversion step by mifmucker takes significantly more time than the postprocessing step. Therefore we would like to propose that whenever possible modifications of the conversion process should be made in the postprocessing process rather than in the mifmucker scripts. In order to prevent having to use the mifmucker conversion step more than one time the software will save the output of this conversion step in the file `FrameMakerFile.sgml.raw`. If the postprocessing software is changed in the future then it can be applied directly to that file.

After converting the document one may wish to know if the document contains correct SGML. It is possible to get the answer to the question by running the document through an SGML validator. The validator that is available was written by James Clark [Clark 1995]. It makes use of Clark's SP software. The program can be invoked as follows:

```
sgml-ncheck SgmlFile
```

It will compare the syntax of the SGML file with the syntax defined in the TEI Lite DTD. The `sgml-ncheck` program is a modified version of the script `html-ncheck` distributed among others from http://www.math.utah.edu/~beebe/sp-notes-1.0.1.html . The output of the program is the file name and a list of syntax errors. The validator will respond with '... valid' if the file contains no TEI syntax errors.

The final useful program is the script `pool`. This program distributes the conversion process over several machines. The program can perform different predefined conversion tasks. In order to convert the Scania manuals one has to issue the command:

```
/usr/users/staff/corpora/bin/pool scaniaNew2Sgml
```

Here `scaniaNew2Sgml` is the predefined name for this task. The script will make a list of conversion commands for the files in the corpus and start a client process on fourteen machines (all available machines except for the file server). The client process will take a command from the list of commands and execute it. This results in the corpus being converted in parallel which speeds up the conversion process.

Some of the local conversion processes fail because of memory problems. The SGML files that result from such a process can be recognized by the fact that they are incomplete, that is the final tag in the file (</TEI.2>) is missing. There is a special `pool` task for handling these files: `scaniaNew2SgmlRedo`. This task will only make use of the file server which is the machine with the largest amount of memory. It is possible that even this conversion process can not handle all the files. We have experienced this problems with binary Framemaker files that were larger than one megabyte. The problems was noted to Scania and they have send us the same files without images which left about 100 kilobytes per file which was no problem for the conversion process.

The `pool` client software checks if someone is using the machine that it wants to use for a conversion process. If this is the fact then the software will wait until the computer is free. The reason for this is that the conversion process requires a lot of memory and running on a computer which is being used by someone else often caused conversion failures. To get the fastest result from the software it is best to start it when most of the machines are free.

E The structure of the corpus

The documents of the corpus have been stored in subdirectories of `/corpora/Scania1995` on the machine strindberg.ling.uu.se. The following subdirectories are available:

000readme Overview file of the contents of the directory.

Analyses Results of analyses of the files in this directory. This directory contains two subdirectories: AsciiDos for analysis results computed on a DOS platform and AsciiUnix for analysis results computed on Unix.

AsciiDos Directory containing the ascii versions for MS-DOS of the Swedish Framemaker files in CD

CD Directory containing the 1995 Scania Framemaker files as copied from the Scania compact disks.

Sgml Directory containing SGML versions of of the Framemaker files

SgmlRaw Directory containing the output of the conversion process after the mifmucker step. A modified postprocessing step can be applied to these files without having to redo the mifmucker conversion.

corrupt Contains the 48 corrupt files of the first version of the Framemaker files together with two files that were too large to convert.

tmp Directory for storing temporary files.

Some of the files in the directory have a six digit version number appended to the name. This version number indicates year, month and day of the creation of the files. Older versions are being kept because people at other platforms have used these versions and it might be necessary to reproduce them in the future.