TIGERRegistry Manual

Wolfgang Lezius
Hannes Biesinger
Ciprian Gerstenberger

IMS, University of Stuttgart

April 5, 2002

Contents

Chapter 1

About this manual

Searching on syntactically annotated corpora is much more time-consuming than searching on corpora annotated on the word level (word, lemma, part-of-speech, morphological description etc.). Thus, an index structure which enables fast access to the results of many partial searches can greatly improve efficiency. For this reason we have decided to use an index-based architecture for the design of the TIGERSearch software suite.

The TIGERSearch tool is the corpus query processor. You can process queries, view the results with the GraphViewer and export your favourite matches. The TIGERSearch tool is described in the TIGERSearch manual ([4]). The query language of the TIGERSearch tool is described in detail in two internal reports (cf. [1] and [2]).

The import and indexing of corpora is realized in the TIGERRegistry tool. For the import of treebanks, we have developed the TIGER-XML format. New corpora have to be converted into this format first. TIGERRegistry comprises import filters (i.e. converters to TIGER-XML) for all popular treebank formats. The TIGER-XML format is described in detail in the TIGER-XML Quick Reference Guide (cf. [3]).

This manual deals with the TIGERRegistry tool. It first explains the general concept of corpus administration within the TIGERSearch software suite (cf. Chap. 2). The following chapters describe how to start the TIGERRegistry tool (cf. Chap. 3) and how to index a corpus (cf. Chap. 4). The next chapter deals with the individual corpus configuration (cf. Chap. 5). The final chapter describes how to modify the configuration of the TIGERSearch software suite (e.g. corpus base directory).

Chapter 2

General introduction

2.1 Corpus administration

The index-based approach splits working with the TIGERSearch software suite into two parts: The first part deals with corpus indexing (TIGERRegistry), the second part with corpus query processing (TIGERSearch). The corpus query processor can only process corpora that have already been indexed.

The indexed corpora are organized in a hierarchical file system: Each corpus is stored in a directory (in this manual called folder), but corpora can be grouped in a common folder as well. The following example illustrates the physical content of a corpus directory and its graphical tree representation in the corpus loading window of the TIGERSearch tool (Fig. 2.1).

```
DemoCorpora/
DemoCorpora/BROWNSampler/
DemoCorpora/CHRISSampler/
DemoCorpora/NEGRASampler/
DemoCorpora/SUESampler/
DemoCorpora/SWITCHBOARDSampler/
DemoCorpora/TIGERSampler/
DemoCorpora/VerbMobil/
DemoCorpora/VerbMobil/VMSampler-DE/
DemoCorpora/VerbMobil/VMSampler-EN/
DemoCorpora/VerbMobil/VMSampler-JAP/
```

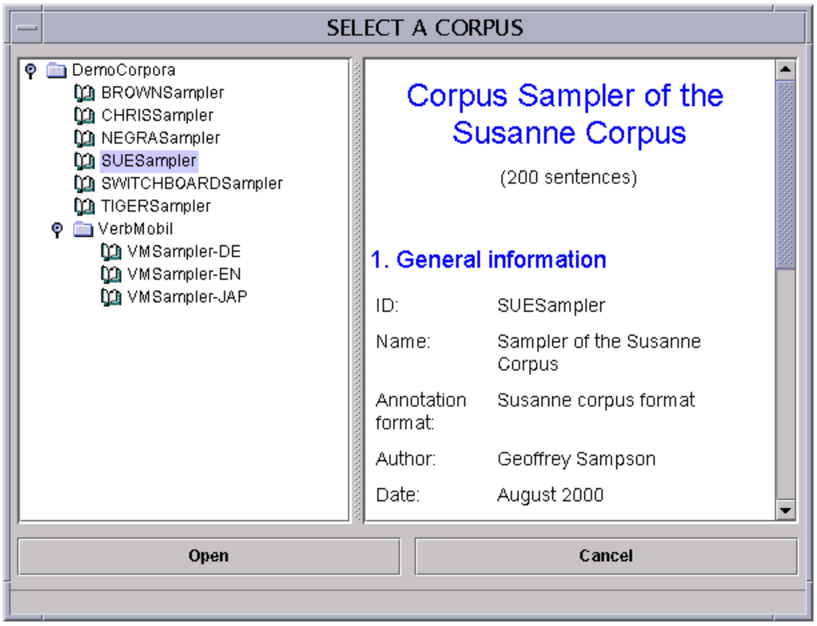

Figure 2.1: Corpus loading window

2.2 Corpus formats

As there are various formats for the representation of linguistic corpora, there are also a number of formats for the encoding of syntactically annotated corpora or treebanks: Penn Treebank, Susanne, Negra, Tipster, and several other formats plus parser output formats. Since applications such as TIGERSearch cannot support all existing formats, a better solution is to define one format as the tool's import and export format. Since the data model supported by TIGERSearch is more general and expandable than the data models of the formats mentioned, we have developed the TIGER-XML format. This format maps the supported data model to XML.

Corpora to be indexed with the TIGERRegistry tool must be encoded in the TIGER-XML format. To support as many treebank formats as possible, we have also implemented import filters (i.e. converters to TIGER-XML) for many popular formats such as Penn Treebank format or Negra format, and some parser output formats.

Indexing of TIGER-XML corpora is explained in Sec. 4.1, indexing of corpora with an import filter is described in Sec. 4.2. The TIGER-XML format is described in the TIGER-XML Quick Reference Guide (cf. [3]).

Chapter 3

Starting TIGERRegistry

The way you can start the TIGERRegistry tool depends on your operating system. On Windows machines, a program group called TIGERSearch has been created during the installation - so you just have to select the TIGERRegistry program in the start menu. On Unix machines, symbolic links have been created. If your general path is set properly, you may just need to type in `TIGERRegistry`. However, the TIGERRegistry start program can always be found in the TIGERSearch installation path: /path/to/your/installation/bin/TIGERRegistry.

Please note that relative paths specified by the user are evaluated with regard to the working directory. This directory is defined as the starting directory on Unix machines and the user's home directory on Mac and Windows machines.

3.1 Corpus administration window

When you start the TIGERRegistry tool, it first checks whether you have the permissions to read, write, and create files in the corpus directory. If you do not have the required permissions, an information window pops up. The TIGERRegistry tool will not be started. If the permission check has been successful, the TIGERRegistry main window appears (cf. Fig. 3.1). Position and size of the window are saved when leaving the tool. So the arrangement of your windows will be restored in the next TIGERRegistry session.

Now you can browse through the corpus tree (left hand side of the window) and have a look at the corpus properties (right hand side). All operations on this corpus tree (insert a new corpus, delete a corpus etc.) can be activated by

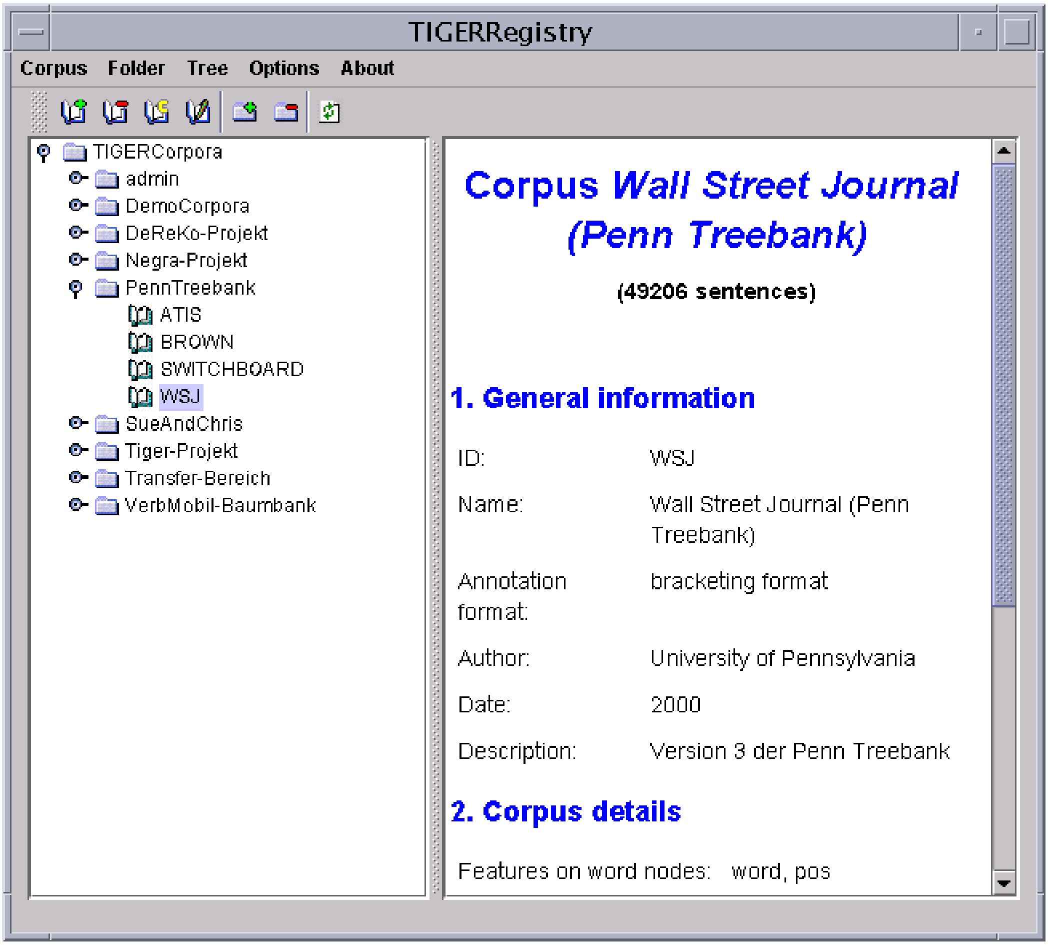

Figure 3.1: TIGERRegistry main window

pressing the appropriate button in the toolbar, or selecting the appropriate item in the popup-menu (activated by clicking the right mouse button), or selecting an item in the menu bar.

3.2 Folders

To insert a new folder (which can contain corpora and other folders) first mark the parent folder of the new folder (in Fig. 3.2 the folder *TIGERCorpora* has been marked). Please note that a corpus folder cannot be a parent folder. Now click the *Insert folder* button in the button toolbar. A new window pops up. Please type in the name of your folder (here: *Testfolder*) and press the *Save* button. Fig. 3.3 shows the corpus tree after inserting the new folder.

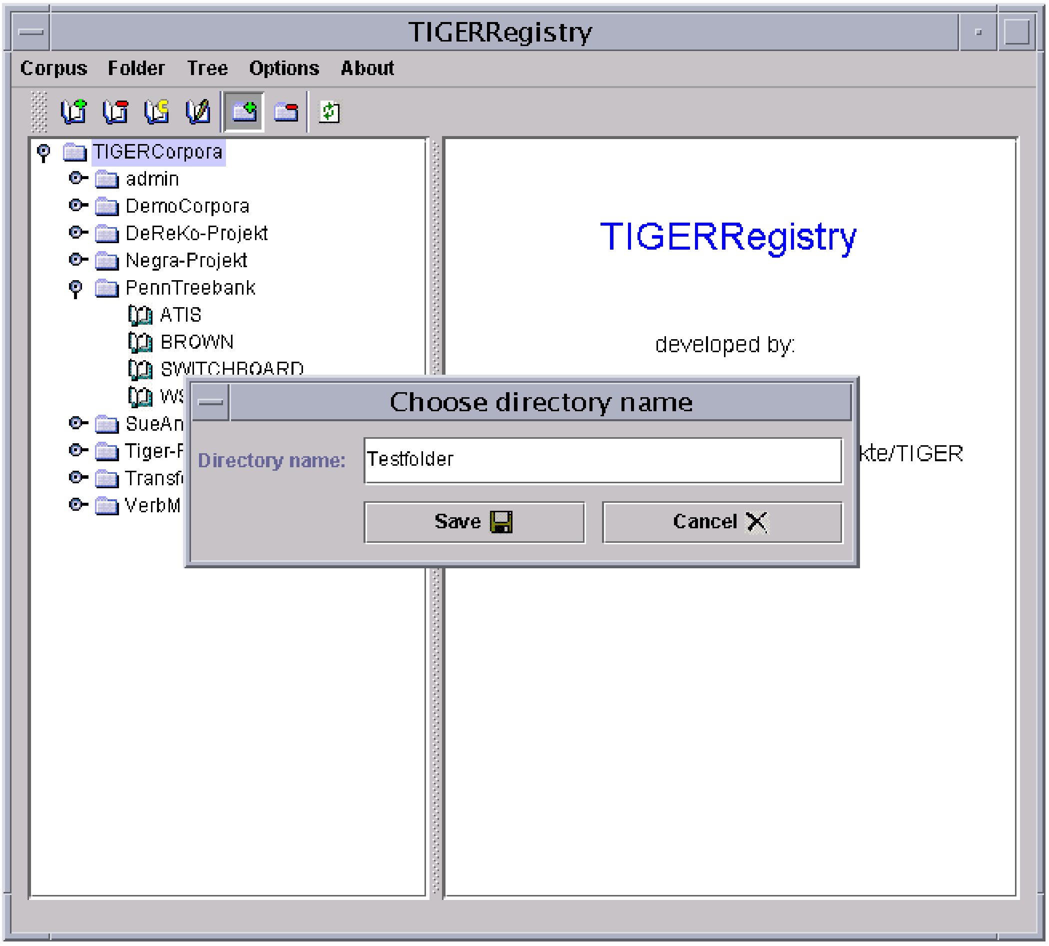

Figure 3.2: Selecting the parent folder

In order to delete a folder mark it and press the *Delete folder* button in the button toolbar. Please note that only empty folders can be deleted.

3.3 Corpora

Please read Chap. 4 for details about inserting a corpus. To delete a corpus first mark the corpus symbol in the corpus administration tree. Now click the *Delete corpus* symbol in the button toolbar. Please note that you always have to confirm a corpus deletion.

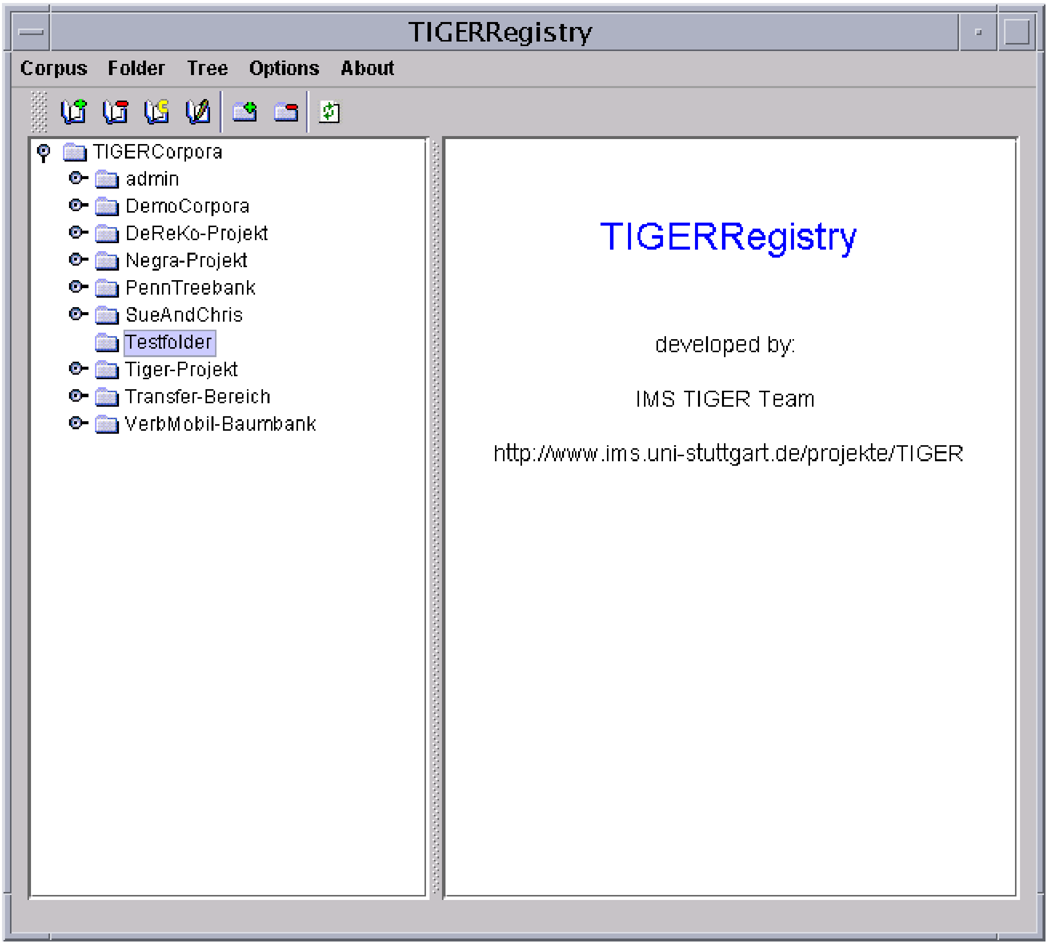

Figure 3.3: A folder has been inserted

Chapter 4

How to index a corpus

The indexing of corpora is based on the TIGER-XML format; corpora encoded in different formats have to be converted to TIGER-XML first. So if your corpus source file is encoded in a different format, you will have to use one of the already implemented corpus format filters (i.e. converters to TIGER-XML) or convert your corpus to TIGER-XML on your own.

To index a corpus first mark the parent folder of the new corpus. Now click the *Insert corpus* button in the button toolbar (cf. Fig. 4.1). The corpus indexing window pops up. First of all, you have to specify the corpus format: *TIGER-XML format* or *other format*. The indexing of TIGER-XML corpora is explained in Sec. 4.1, the indexing of other corpora is described in Sec. 4.2. A list of all implemented corpus filters can be found in Sec. 4.3.

4.1 TIGER-XML indexing

If your corpus source is encoded in TIGER-XML format, please mark *Corpus is in TIGER-XML format* at the top of the window (cf. Fig. 4.2).

Now you have to specify the following parameters:

Corpus ID
The corpus ID is used by the TIGERSearch software to realize corpus-dependent configurations. The corpus ID must be unique with regard to all indexed corpora. The uniqueness is checked before the indexing process is initiated. Please note that the ID should start with an alphabetic character.

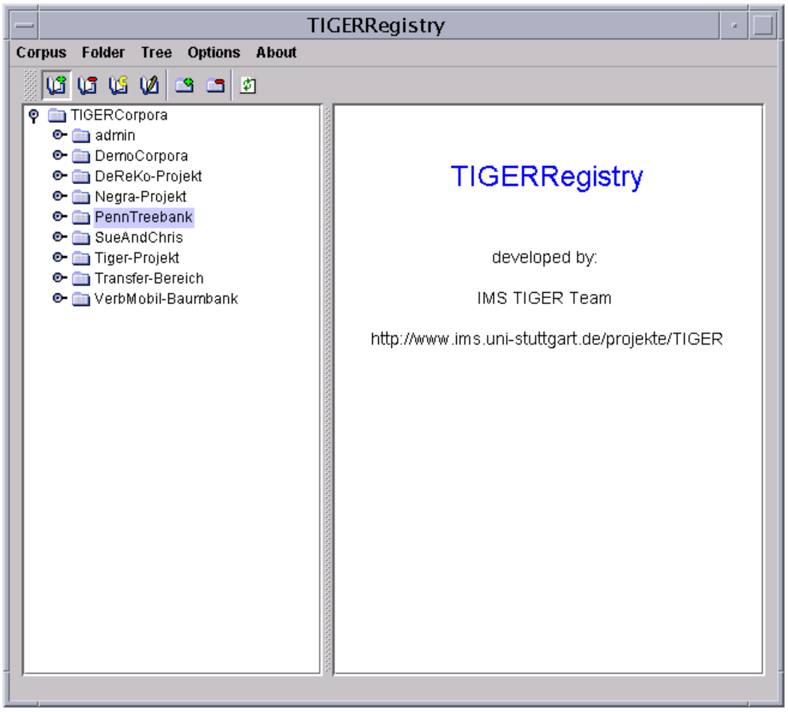

Figure 4.1: Inserting a corpus

TIGER-XML file
The source file (relative paths are evaluated with regard to the working directory) can be either an uncompressed `.xml` file, or a compressed `.xml.gz` file, or a `.zip` file that contains one source file only. Compressed files are automatically decompressed during the indexing process.

To start the indexing process please press the *Start* button. The corpus indexing can be stopped at any time. The current progress of the indexing is displayed by the indexing progress window (cf. Fig. 4.3).

4.2 Indexing with import filters

Indexing with import filters consists of two steps: First of all, the corpus is converted to TIGER-XML. Afterwards the generated TIGER-XML corpus is indexed. Thus, you have to specify the following parameters (cf. Fig. 4.4):

Corpus ID
The corpus ID is used by the TIGERSearch software to realize corpus-dependent configurations. The corpus ID must be unique with regard to all indexed cor-

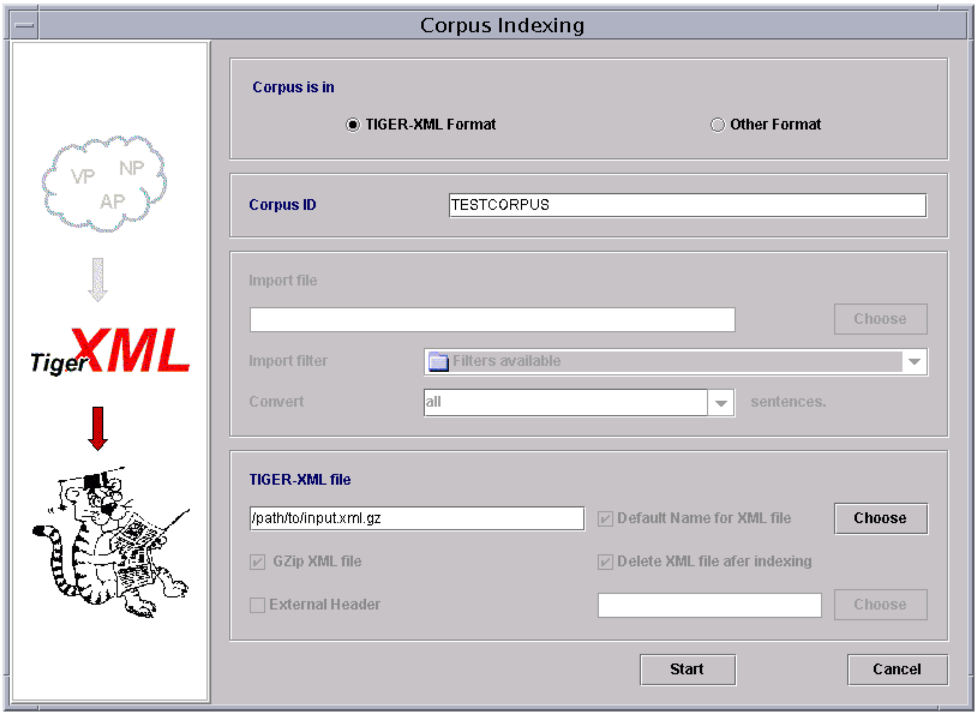

Figure 4.2: Indexing of a TIGER-XML corpus

pora. The uniqueness is checked before the indexing process is initiated. Please note that the ID should start with an alphabetic character.

Import source file
The source file can either be an uncompressed file, or a compressed `.gz file`, or a `.zip` file that contains one source file only (relative paths are evaluated with regard to the working directory). Compressed files are automatically decompressed during the indexing process.

Import filter
Select one of the corpus format filters (cf. Fig. 4.6). For a list of all implemented

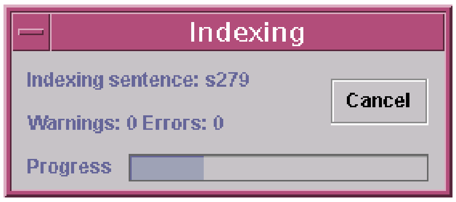

Figure 4.3: Indexing progress windows (TIGER-XML indexing)

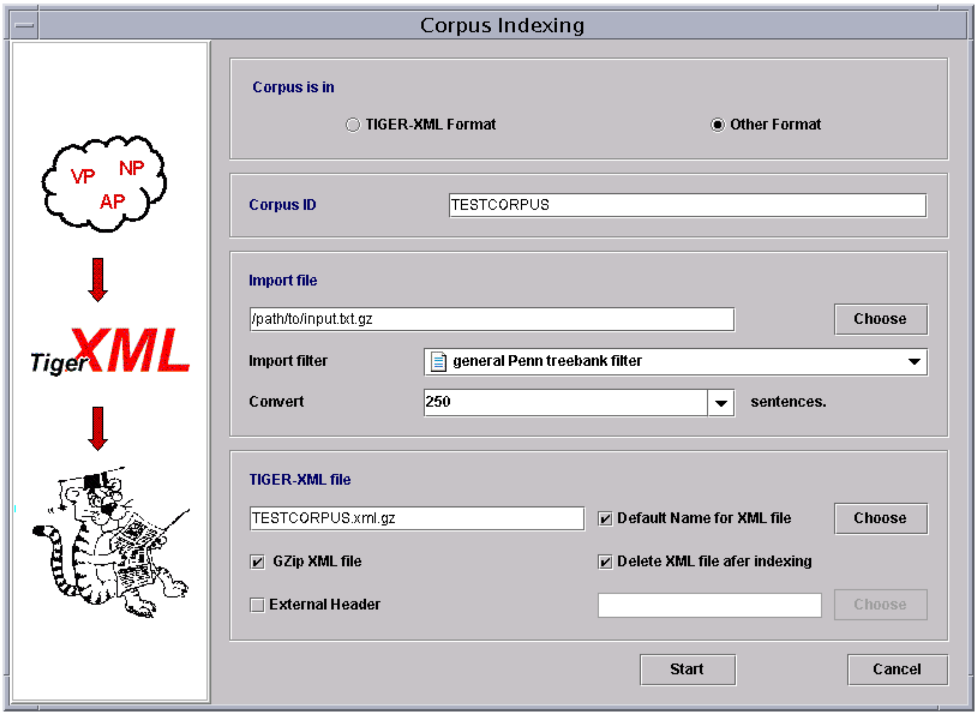

Figure 4.4: Indexing with an import filter

filters cf. Sec. 4.3.

Convert n sentences
You can either index the whole corpus or the first n sentences of the corpus.

Temporary TIGER-XML file
The first step of the indexing is the conversion to TIGER-XML. Thus, a temporary TIGER-XML file is needed. When typing in the corpus ID, a file name is automatically generated by the system (cf. checkbox *Default name for XML file*). Of course, you can also specify a different file name. Please note that relative paths are evaluated with regard to the working directory.

TIGER-XML file parameters
As the TIGER-XML file is a temporary file, it makes sense to compress it. You can enforce GZIP compression by checking the *GZip XML file* box. When the indexing progress is finished, the temporary file is automatically deleted. If you want to save the file (e.g. for debugging purposes) just uncheck the *Delete after indexing* box.
You can also make use of a so-called external header, i.e. a TIGER-XML document header which is stored in a separate file. To use this external header check the *External header* box and type in the path of the header file (relative paths

are evaluated with regard to the working directory).

To start the indexing process please click the *Start* button. The corpus indexing can be stopped at any time. The current progress of the indexing is displayed by the Conversion & Indexing progress window (cf. Fig. 4.5).

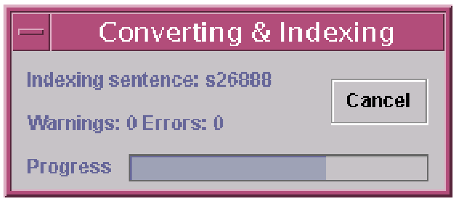

Figure 4.5: Indexing progress windows (Indexing with an import filter)

4.3 List of implemented import filters

The following import filters have been implemented (cf. Fig. 4.6):

general bracketing format

- general () filter: should work with bracketing-style corpora that use () for structuring; generates `cat`, `pos` and `word` features.
- general [] filter: should work with bracketing-style corpora that use [] for structuring; generates `cat`, `pos` and `word` features.

PennTreebank

- general PennTreebank filter: should work with UPenn-style corpora; functions are modelled as edge labels, traces are modelled as secondary edges.
- ATIS corpus filter: a special filter for the ATIS corpus format **only**; handles the different `pos/word` notation and other corpus-specific differences.
- SWITCHBOARD corpus filter: a special filter for the SWITCHBOARD corpus format **only**; skips code and interjection sections.

Susanne and Christine

- Susanne corpus filter: a special filter for the Susanne corpus format **only**.

- Christine corpus filter: a special filter for the Christine corpus format **only**.

Negra format

- general Negra format filter: should work with corpora encoded according to the Negra format, version 3.
- TIGER sampler filter: a special filter for the TIGER sample corpus **only**.

XML formats

- DEREKO format filter: should work with corpora encoded according to the DEREKO corpus format.

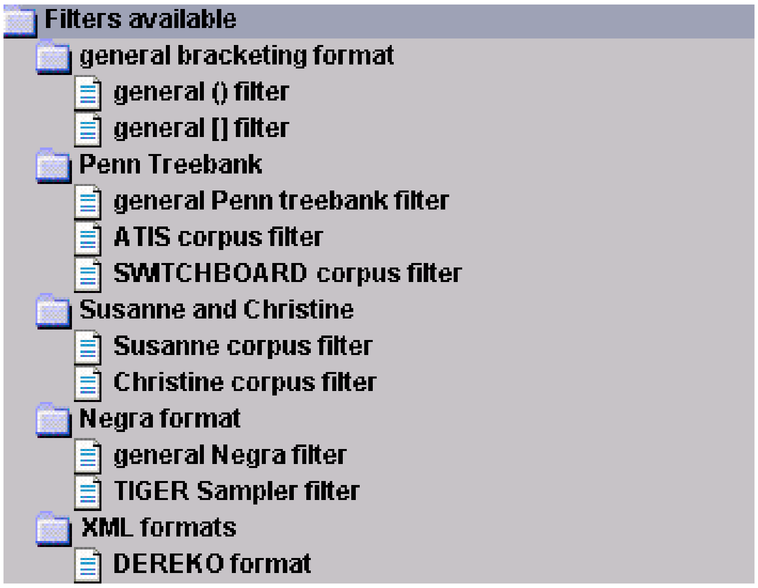

Figure 4.6: Implemented corpus filters

4.4 Corpus conversion

If you just want to convert a corpus to the TIGER-XML format, you can use the corpus conversion feature. Click on the *Corpus conversion* button in the TIGERRegistry main window. Now the corpus conversion window pops up (cf. Fig. 4.7).

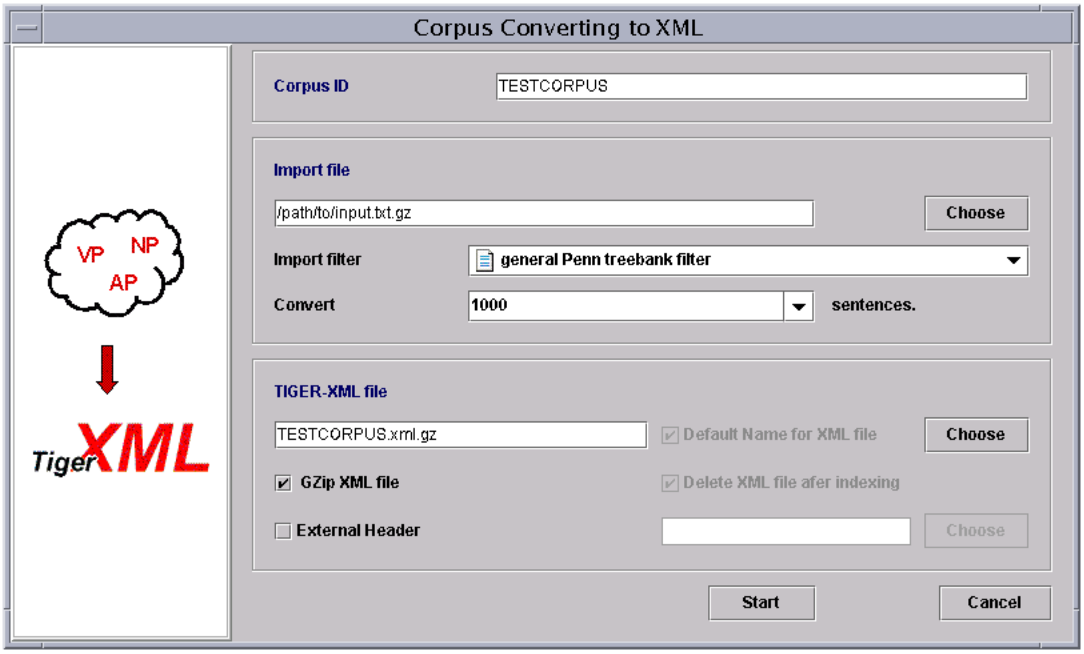

Figure 4.7: Corpus conversion window

Specify the parameters which have been explained in the previous sections and click the *Start* button to start the conversion. The conversion process can be stopped at any time.

Chapter 5

Changing corpus properties

To change the properties of a corpus, please mark the corpus symbol and press the *Corpus Properties* button in the button toolbar. The corpus properties window pops up (Fig. 5.1). Now you can change the corpus meta information (Sec. 5.1) or specify a type system (Sec. 5.2).

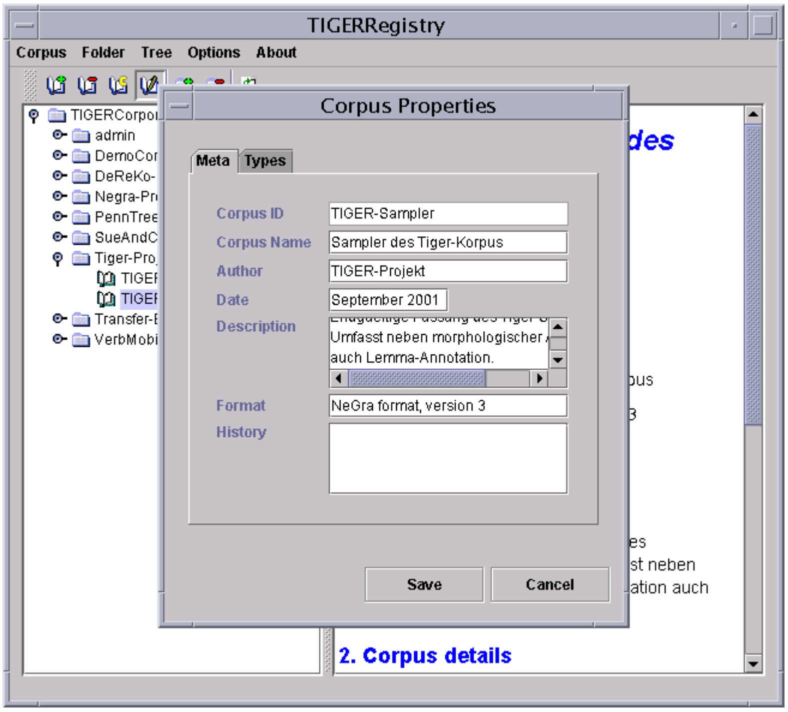

Figure 5.1: Corpus properties window

5.1 Corpus meta information

To edit the meta information of the corpus (which is displayed by the TIGERSearch and TIGERRegistry GUI), select the *Meta* tab in the corpus properties window (cf. Fig. 5.2). Now you can edit the corpus meta information except the ID which cannot be changed after corpus creation.

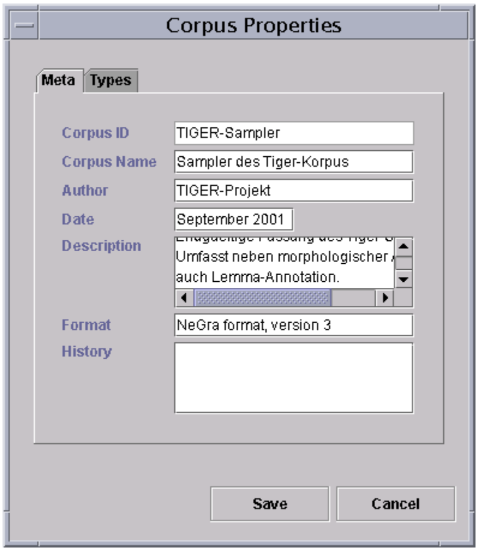

Figure 5.2: Corpus meta information

5.2 Feature types

The TIGERSearch user can make use of type hierarchies in order to formulate queries in a more adequate way:

```
[pos=nominal] .* [pos="VVFIN"]
```

The concept of type hierarchies is described in [1]. To specify a type hierarchy for a feature, you must first create an XML file comprising the hierarchy definition (see below). Afterwards you have to place a link to the XML file. Please select the *Types* tab in the corpus properties window (cf. Fig. 5.3). Now select the feature and type in the path to your file (absolute or relative to the corpus path). In the example, a relative link to the file `tigerstts.xml` in the corpus directory is specified.

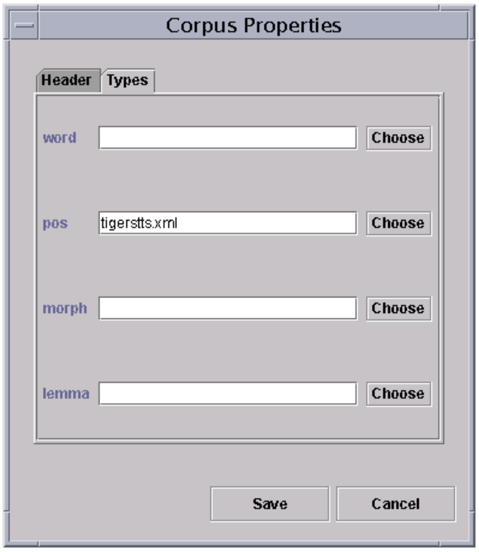

Figure 5.3: Corpus types information

The XML format for defining type hierarchies is quite simple. The following illustrates the XML encoding of an example rule.

As a starting point, we present the example rule following the Prolog-like notation that is defined in the internal report (cf. [1]) :

```
nominal := noun, "ADJA", "ADJD";
```

Now the XML representation looks like the following:

```
<type name="nominal">
   <subtype nameref="noun" />
   <constant value="ADJA" comment="[das] große [Haus]" />
   <constant value="ADJD" comment="[er fährt] schnell" />
</type>
```

A subtype is modelled as a `<subtye>` element, feature values are represented as `<constant>` elements. The feature value name is specified as the value of a *value* attribute. Optional comments are represented as *comment* attributes.

If there are any problems loading a type hierarchy (e.g. if there is a feature value in the corpus that is not used in the hierarchy, or if there is a feature value used in the hierarchy that is unknown in the corpus) all warnings are collected and will be displayed in a separate window if you have checked the *Display warnings* box in the *Corpus* menu (cf. TIGERSearch manual, [4]).

The part-of-speech type hierarchy used for the TIGER Corpus Sampler (based on a modified version version of the STTS tag set) is listed as an example below.

```
<?xml version="1.0" encoding="ISO-8859-1"?>


<!-- Type system for the TIGER corpus:
     parts of speech based on the STTS tag set -->


<typedeclaration base="pos" version="1.0">

  <!-- base type: pos -->
  <type name="pos">
     <subtype nameref="majorPOS" />
     <subtype nameref="minorPOS" />
     <subtype nameref="otherStuff" />
  </type>

  <!-- open word classes-->
  <type name="majorPOS">
     <subtype nameref="adjective" />
     <subtype nameref="adverb" />
     <subtype nameref="nominal" />
     <subtype nameref="verb" />
  </type>

  <!-- adjective -->
  <type name="adjective">
     <!-- attributive adjective -->
     <constant value="ADJA" comment="[das] große [Haus]" />
     <!-- adverbal or predicative adjective -->
     <constant value="ADJD" comment="[er fährt] schnell, [er ist] schnell" />
  </type>

  <!-- adverbs -->
  <type name="adverb">
     <!-- normal adverb -->
     <constant value="ADV" comment="schon, bald, doch" />
     <!-- interrogative adverb or adverbial relative pronoun -->
     <constant value="PWAV" comment="warum, wo, wann, worüber, wobei" />
     <!-- pronominal adverb -->
     <constant value="PROAV" comment="dafür, dabei, deswegen, trotzdem" />
  </type>

  <!-- nominals -->
  <type name="nominal">
     <!-- normal noun -->
     <constant value="NN" comment="Tisch, Herr, [das] Reisen" />
     <subtype nameref="pronoun" />
     <!-- proper noun -->
```

```
   <constant value="NE" comment="Hans, Hamburg, HSV" />
   <!-- combination of noun and proper noun -->
   <constant value="NNE" comment=" " />
</type>

<!-- pronouns -->
<type name="pronoun">
   <!-- substitutive demonstrative pronoun -->
   <constant value="PDS" comment="dieser, jener" />
   <!-- substitutive indefinite pronoun -->
   <constant value="PIS" comment="keiner, viele, man, niemand" />
   <!-- substitutive interrogative pronoun -->
   <constant value="PWS" comment="wer, was" />
   <!-- irreflexive personal pronoun -->
   <constant value="PPER" comment="ich, er, ihm, mich, dir" />
   <!-- substitutive possesive pronoun -->
   <constant value="PPOSS" comment="meins, deiner" />
   <!-- reflexive pronoun -->
   <constant value="PRF" comment="sich, einander, dich, mir" />
   <!-- substitutive relative pronoun -->
   <constant value="PRELS" comment="[der Hund,] der" />
</type>

<!-- verbals -->
<type name="verb">
   <subtype nameref="verbInflected" />
   <subtype nameref="infinitival" />
</type>

<!-- inflected verbs -->
<type name="verbInflected">
   <subtype nameref="finite" />
   <subtype nameref="imperative" />
</type>

<!-- finite verbs -->
<type name="finite">
   <!-- finite auxiliary verb -->
   <constant value="VAFIN" comment="[du] bist, [wir] werden" />
   <!-- finite full verb -->
   <constant value="VVFIN" comment="du] gehst, [wir] kommen [an]" />
   <!-- finite modal verb -->
   <constant value="VMFIN" comment="dürfen" />
</type>

<!-- imperatives -->
<type name="imperative">
   <!-- imperative, auxiliary verb -->
   <constant value="VAIMP" comment="sei [ruhig !]" />
   <!-- imperative, full verb -->
   <constant value="VVIMP" comment="komm [!]" />
```

```
</type>

<!-- infinitive forms -->
<type name="infinitival">
   <subtype nameref="infinitivalAux" />
   <subtype nameref="infinitivalFull" />
   <subtype nameref="infinitivalModal" />
</type>

<!-- inifinitive forms: auxiliaries -->
<type name="infinitivalAux">
   <!-- infinitive, auxiliary verb -->
   <constant value="VAINF" comment="werden, sein" />
   <!-- past participle, auxiliary verb -->
   <constant value="VAPP" comment="gewesen" />
</type>

<!-- infinitive forms: full verbs -->
<type name="infinitivalFull">
   <!-- inifinitive, full verb -->
   <constant value="VVINF" comment="gehen, ankommen" />
   <!-- infinitive with zu, full verb -->
   <constant value="VVIZU" comment="anzukommen, loszulassen" />
   <!-- past participle, full verb -->
   <constant value="VVPP" comment="gegangen, angekommen" />
</type>

<!-- inifinitive forms: modals -->
<type name="infinitivalModal">
   <!-- infinitive, modal verb -->
   <constant value="VMINF" comment="wollen" />
   <!-- past participle, modal verb -->
   <constant value="VMPP" comment="gekonnt, [er hat gehen] können" />
</type>

<!-- closed word classes -->
<type name="minorPOS">
   <subtype nameref="conjunction" />
   <!-- interjection -->
   <constant value="ITJ" comment="mhm, ach, tja" />
   <subtype nameref="particle" />
   <subtype nameref="prenominal" comment="all prenominals apart from adjectives" />
   <!-- cardinal number -->
   <constant value="CARD" comment="zwei [Männer], [im Jahre] 1994" />
   <subtype nameref="xposition" />
</type>

<!-- conjunctions -->
<type name="conjunction">
   <!-- comparative conjunction -->
   <constant value="KOKOM" comment="als, wie" />
```

```
    <!-- coordinating conjunction -->
    <constant value="KON" comment="und, oder, aber" />
    <subtype nameref="subordConjuntion" />
  </type>

  <!-- subordinating conjunctions -->
  <type name="subordConjuntion">
    <!-- subordinating conjunction with zu-infinitive -->
    <constant value="KOUI" comment="um [zu leben], anstatt [zu fragen]" />
    <!-- subordinating conjunction with sentence -->
    <constant value="KOUS" comment="weil, daß, damit, wenn, ob" />
  </type>

  <!-- particles -->
  <type name="particle">
    <!-- particle with adjektive or adverb -->
    <constant value="PTKA" comment="am [schönsten], zu [schnell]" />
    <!-- answer particle -->
    <constant value="PTKANT" comment="ja, nein, danke, bitte" />
    <!-- negating particle -->
    <constant value="PTKNEG" comment="nicht" />
    <!-- separated verb particle -->
    <constant value="PTKVZ" comment="er kommt] an, [er fährt] rad" />
    <!-- "zu" before infinitive -->
    <constant value="PTKZU" comment="zu [gehen]" />
  </type>

  <!-- prenominal -->
  <type name="prenominal">
    <!-- definite or indefinite determiner -->
    <constant value="ART" comment="der, die, das, ein, eine" />
    <subtype nameref="attrPronoun" />
  </type>

  <!-- attributive pronouns -->
  <type name="attrPronoun">
    <!-- attributive demonstrative pronoun -->
    <constant value="PDAT" comment="jener [Mensch]" />
    <!-- attributive indefinite pronoun -->
    <constant value="PIAT" comment="kein [Mensch], irgendein [Glas]" />
    <!-- attributive interrogative pronoun -->
    <constant value="PWAT" comment="welche [Farbe], wessen [Hut]" />
    <!-- attributive possessive pronoun -->
    <constant value="PPOSAT" comment="mein [Buch], deine [Mutter]" />
    <!-- attributive relative pronoun -->
    <constant value="PRELAT" comment="[der Mann ,] dessen [Hund]" />
  </type>

  <!-- Xposition -->
  <type name="xposition">
    <!-- preposition -->
```

```
    <constant value="APPR" comment="in [der Stadt], ohne [mich], von [jetzt an]" />
    <!-- preposition+determiner -->
    <constant value="APPRART" comment="im [Haus], zur [Sache]" />
    <!-- postposition -->
    <constant value="APPO" comment="[ihm] zufolge, [der Sache] wegen" />
    <!-- circumposition, right part -->
    <constant value="APZR" comment="[von jetzt] an" />
  </type>

  <!-- other tags -->
  <type name="otherStuff">
    <!-- foreign material -->
    <constant value="FM" comment="[Er hat das mit ‘‘] A big fish [’’ übersetzt]" />
    <!-- nonword, with special characters -->
    <constant value="XY" comment="3:7, H2O, D2XW3" />
    <subtype nameref="punctuation" />
    <!-- truncated element -->
    <constant value="TRUNC" comment="An- [und Abreise]" />
    <!-- untagged -->
    <constant value="--" />
    <!-- tagging of the token unknown -->
    <constant value="UNKNOWN" />
  </type>

  <!-- punctuation marks -->
  <type name="punctuation">
    <!-- comma -->
    <constant value="$," comment="," />
    <!-- full stop -->
    <constant value="$." comment=". ? ! ; :" />
    <!-- other punctuation marks -->
    <constant value="$(" comment="- [,]()" />
  </type>

</typedeclaration>
```

Chapter 6

Software suite preferences

Please note: The following features should only be used by the local administrator of the TIGERSearch software, usually the person who did the installation.

Whereas the TIGERRegistry tool will always work fine if you have the required 128 MB main memory on your system, the TIGERSearch tool could lack on main memory if you work with very large corpora. Unfortunately, a typical feature of Java applications is that the initial main memory size (used when starting an application) and the maximum main memory size (that can be used by an application) have to be specified before the application starting point. So if you have more than 128 MB memory on your system, this additional memory will not be used automatically.

The main memory configuration of both the TIGERSearch and TIGERRegistry tool and the corpus base directory used by the two tools can be modified in the *Software suite preferences* window. It is opened by selecting the *Software preferences* item in the *Options* menu of the TIGERRegistry tool. The preferences window is displayed in Fig. 6.1.

Memory configuration

If you want to increase the main memory used by the TIGERSearch tool, just type in the new memory size in the *max* text field. We recommend you also to increase the initial memory used when an application is started - preferably a quarter of the maximum main memory. This will increase the corpus loading speed.

The TIGERRegistry tool should work without any problems using 128 MB memory. Nevertheless you can change its memory configuration, too.

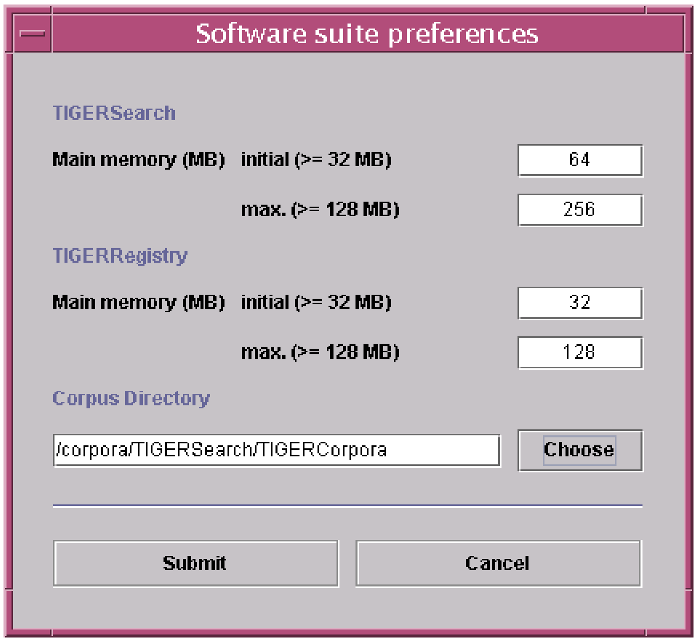

Figure 6.1: Software suite preferences

Corpus base directory

The corpus base directory used by the TIGERSearch software suite is created during the software installation. You can move the directory to another location (e.g. to another hard disc drive) as follows:

- First create the new (empty) corpus directory `/path/to/new/dir/`.
- Then open the software preferences window and specify the new directory in the *Corpus directory* text field by typing in the absolute path `/path/to/new/dir/` or using the file chooser.
- After submitting the changes (see below) move the content of the old corpus directory to the new corpus directory.

Submit modifications

If you modified the memory and corpus directory configuration, press the *Submit* button. The modifications will be checked and the software suite configuration will be changed.

Please note: The modification of the software configuration files is realized in a platform-independent way. Thus, the file permissions of the modified configuration files may change, depending on your default settings. So if you have installed the TIGERSearch software on a multi-user system, please check the file permissions of the modified configuration files. These are the two files `TIGERSearch.lax` and `TIGERRegistry.lax` in the `bin/` subdirectory of your TIGERSearch installation directory.

As the status of the sofware suite has changed, the TIGERRegistry tool will be closed automatically. To profit from the new software configuration you will also have to restart your current TIGERSearch session.

Bibliography

[1] Esther König and Wolfgang Lezius. The TIGER language - a description language for syntax graphs. Part 1: User's guidelines. Technical report, IMS, University of Stuttgart, 2001.

[2] Esther König and Wolfgang Lezius. The TIGER language - a description language for syntax graphs. Part 2: Formal definition. Technical report, IMS, University of Stuttgart, 2001.

[3] Wolfgang Lezius, Hannes Biesinger, and Ciprian Gerstenberger. *TIGER-XML Quick Reference Guide*. IMS, University of Stuttgart, 2002.

[4] Wolfgang Lezius, Hannes Biesinger, and Ciprian Gerstenberger. *TIGERSearch Manual*. IMS, University of Stuttgart, 2002.