Natural language interface to legal databases

Rafal Piotrowski

June 7, 2005
Master's Thesis in Computing Science, 20 credits
Supervisor at CS-UmU: Michael Minock
Examiner: Per Lindström

Umeå University
Department of Computing Science
SE-901 87 UMEÅ
SWEDEN

Abstract

Browsing big legal databases has become a part of daily work activity in an almost every modern law office. Legal regulations change more often than it was in the past and to remain up-to-date, one must use electronic media collections for help. The problem is that most popular databases take queries in formal query languages (e.g. SQL), which can be difficult for the casual database user. This project tries to determine, how the lawyers can be helped by providing natural language interface to legal knowledge databases. It can possibly separate a lawyer or any other user that looks for a legal information, from complexity and understandability problems, that formal query languages involve.

Contents

1 Introduction **1**

2 Background **3**

2.1 Advantages of using the natural language interface 3
2.2 AI and law 3
2.3 Interfaces to legal databases 4
2.4 Natural language interface for legal databases 4
2.5 Phrasal lexicons 5
2.6 Introduction to STEP 5
2.7 Problem Statement 5
2.8 Goals 6
2.9 Methods 6

3 Approach **7**

3.1 Preliminaries 7
3.1.1 Interaction between the STEP system and the user 7
3.1.2 Internal architecture of the STEP system 7
3.1.3 Components of a application in STEP 8
3.2 Task steps 9
3.2.1 Analyzing data model 9
3.2.2 Preeliminary database schema 9
3.2.3 Collecting data 9
3.2.4 Final database schema 9
3.2.5 ODBC interface for PostgreSQL 9
3.2.6 Populating the database 10
3.2.7 Development of the phrasal lexicon and the sample queries . . . 10
3.2.8 Initial tests of the application 10
3.2.9 Refinements of the phrasal lexicon, and updating the sample queries 10
3.2.10 Final evaluation 10

4 Results **11**
4.1 Database schema 11
4.1.1 Entities and relationships 11
4.2 Physical representation 14
4.2.1 Representation of phrases in STEP 14
4.3 Evaluation of the application 15
4.3.1 Cognitive Walkthrough 15
4.3.2 Testing of the legal application using Cognitive Walkthrough . . 15
4.3.3 Results of the evaluation 16

5 Conclusions **21**
5.1 Limitations 21
5.2 Future work 21

6 Acknowledgements **23**

References **23**

A Source Code of the ODBC interface **27**

B Database specification **33**

C Phrasal lexicon **35**

List of Figures

3.1 Architecture of the STEP system. 8

4.1 EER diagram. 12
4.2 Categories of the Intellectual Property legal domain. 12
4.3 Tables. 14
4.4 Welcome page of the legal application. 16
4.5 Legal regulations in a specified country. 17
4.6 statues that match the specified keyword. 17
4.7 Answer to the query about available statues. 18
4.8 statues that pertain to the specified category in a specified country. . . . 18
4.9 Available categories in the database. 19
4.10 Statues in country enacted after specified date. 19

Chapter 1

Introduction

The following project has the goal to evaluate the usefulness of a natural language interface to the legal databases. First, main questions of the natural language interfaces are discussed. They are more suitable for some situations and for some users, and have advantages over other interfaces. However, they also involve various challenges, to be dealt with.

Next, intersections of Artificial Intelligence and law are described. Many systems already exist that set a goal to support various legal tasks, for instance, legislation process or drafting of legal documents with help of achievements of the Artificial Intelligence. However, the situation is still far from creating a "Virtual Lawyer", who provides the consistent, complete and fair law, or who automatically solves legal cases basing on the case description and applying legal regulations that are suitable for the given situation.

As stated above, a modern lawyer often uses legal databases. Databases that contain up-to-date legal regulations or precedence cases, facilitate lawyers daily activities in a big extent and are a must-have in larger law firms. The problem is that people looking for legal information have to face is that queries have to be provided in a formal language like SQL, which they are usually not skilled in. They experience so called conceptual gap between the information they want to retrieve and the method they have to use in order to get this information.

The problem mentioned above, can be bridged by using the natural language interface to legal databases. The natural language is the part of every lawyers daily work and would definitely be more appropriate for many casual database users. Although, natural language processing has many open problems.

STEP is a natural language processing system that answers questions of the user over a specified knowledge base that consists of the database and a phrasal lexicon. Provided a knowledge database as well a phrasal lexicon of a specified domain, STEP takes queries in a natural language interface format and returns answers to the user also as sentences build in the natural language. Detailed description of the interaction between the system and the user during a sample session follows, as well as the description of the internal architecture of the system. Components of each STEP application are also discussed.

In order to evaluate the usefulness of the natural language concept for a legal database, a legal application has been developed. It consists of a database that contains legal regulations that are related to the Intellectual Property legal domain from 83 countries and from the phrasal lexicon from the legal domain. Challenges to be dealt with and approaches that have been adopted to the project are described.

As a last part of the project, the final usability evaluation follows, that involves the Cognitive Evaluation usability testing method, which is one of most helpful ones when testing task - based systems. Results of the usability evaluations are presented.

Chapter 2

Background

2.1 Advantages of using the natural language interface

The main method to store big amount of data nowadays is to store it in databases. Databases usually process queries in a formal query language, like SQL[17]. Queries using natural languages, in contrast to those using a formal database query language like SQL, do not require the use of artificial syntax of the query[11]. This is particularly helpful to a set of people who do not want or do not have time to learn or use SQL. Constructing correct complex queries in a formal language involves problems especially for casual database users. Learning an artificial language might be a serious problem for managers with no technical skills, or who use a computer for different tasks than querying databases. Even for technical specialists, learning different query languages when working on different systems can be difficult. Therefore, natural language interfaces can sometimes be helpful because they do not require training or involve forgetting [13] [6]. Query in a formal language might be more complex than the corresponding question in a natural language. Natural language interfaces are more convenient, and interaction with the system can proceed in terms that the user really needs, provided that a robust language processing system shields the user from details of the implementation.[5]

2.2 AI and law

While computers and data processing systems have found their way to the legal profession, they have mainly been used to support office tasks, such as billing, word processing, and other administrative tasks. As Artificial Intelligence technology develops, the creation of systems that can perform reasoning with the legal situation and legal case description to support determining legal solutions becomes feasible[12].

Expert systems might in the future, for instance, be able to predict the probability of winning or loosing a case with a reasonable level of accuracy. An important feature of such expert systems is that can explain what the calculation of the probability is based on [3]. System assigns numerical "weights" in relation to case facts. These facts are matched with similar cases in the knowledge base of the system, and outcome of the calculation is presented to the user.

Besides, a possibility exists that AI will succeed in expert systems that could help

judges produce fair judgments; since the machines are sometimes more capable in considering many circumstances at the same time [14]. This feature might be helpful, for instance, by judging a fair punishment for a criminal offense, considering attributes of the crime, mitigating and aggravating circumstances, and the personal characteristics of accused individual.

Nowadays, expert systems of such complexity are still not available. There are some systems that support lawyers in drafting complex legal documents. [2] These systems often provide strategic considerations, practical tips and natural language constructing tips, like synonyms. One of such systems supports filling in of patent applications.

2.3 Interfaces to legal databases

Lawyers nowadays face the problem of ever growing quantities of information which they have to consider in solving legal problems. The volume of statutes and treaties is growing rapidly despite efforts towards deregulation. Also the body of recent case law is expanding because people tend to request legal decisions more often. The rise of information technology (IT) came with great promises to help ordering and processing these quantities of information. Many written sources of legal information have been stored in text databases which add new possibility for information retrieval and can replace paper equivalences to a large extent, since the electronic documents can be searched in a more complex way, and do not need much physical space to be stored [8]. If a lawyer has legal documents stored in the electronic database, the computer can quickly scan a huge amount of documents and select those that match the query given by the user. Apart from storing legal texts, the legal rules which are expressed by them, can be brought into legal expert systems which can perform automatically reasoning. It may improve a quality of the legal regulations or to help users with no legal knowledge to retrieve information about normative content of stored legal acts.

As written above, expert systems can be very useful in the legal knowledge domain. Although, theoretical possibilities of these systems are not exploited to the full extent. A possible cause of these problems is that the user interface does not always shield the users of these systems from the internal representation of the data and from processing algorithms [15]. In other words, non technical users have to know many technical details of the implementation of the system, in order to be able to use the system.

2.4 Natural language interface for legal databases

As stated above, the user interface of the computer system can be used to separate the lawyer from the technical database concepts. If the interface is supplied with knowledge of both the structure of the information in the system and the legal problem solving process, it can act as an intelligent proxy, which lawyers can exploit easily using the specific legal terminology which is a part of their daily work. The lawyer should be supported in his usual activities. The system should facilitate these activities and automatically present the relevant information the user might need. Natural language interface to legal databases can be particularly useful. Lawyers use written and spoken language in their daily activates, therefore natural language interface appears especially appropriate for them [4]. Lawyers usually do not have any knowledge about formal query language, like SQL. Additionally, keyword - based search is not always sufficient

to address their needs, since it can not express relations between concepts that are more complex than Boolean expressions.

2.5 Phrasal lexicons

Natural language processing, lexicons and lexicography was always subject of great interest of researchers people working in the fields of cognitive science, linguistics, psycholinguistics, computational linguistics, artificial intelligence, natural language processing, and information technology[18].

Development of large lexical knowledge bases emerged as one of the most expensive and time-consuming task in the linguistics, computational linguistics and artificial intelligence [1]. Any list of words which has been gathered and arranged according to any kind of structure which serves a particular purpose to be a lexicon[16]. Thesauruses and dictionaries are particular kinds of lexicons. A dictionary is a lexicon which defines particular words in terms of other words in the same language, or in words from a different language. Indices of books are lexicons which represent the content of a book in terms of the significant words, phrases, and concepts arranged in alphabetical order, which facilitates the process of quickly finding relevant information in the volume. The table of contents of a book is also a lexicon in that it is a hierarchically ordered list of words and phrases which represents the content of the book.

2.6 Introduction to STEP

STEP is a system dedicated to providing conversational access to relational databases.[7] The user can connect to the specified knowledge database and ask questions in the natural language. System parses the given query and tries to answer the question in the form of a sentence in the natural language. User is supplied with sample queries that can be sent to the application, that are helpful to understand what kind of information can be retrieved, as well as how the syntax of the asked question might give the best results. User can also turn on the tracing, which shows the expression in the first order logic, to which the question is transformed into. The user can also see the SQL query that is sent to the database in order to retrieve the results for the asked question. If the system can parse the question the user has asked, it sends an appropriate query to the database, otherwise it tries to negotiate with user, what he was asking for, or sends a message that the given question cannot be parsed.

STEP uses first order logic to represent the knowledge base. It sends satisfiability queries to the SPASS first - order theorem prover. It makes use of the WordNet lexicon [1] to generate synonyms so that the number of understandable user queries can be bigger.

STEP enables an easy integration with new applications. A detailed explanation of the interaction between the user and the STEP system, as well as architecture of the system, will follow in the next sections.

2.7 Problem Statement

STEP currently runs over the geography database. The main aim of this project is to develope a legal application which will integrate with STEP and use its features: taking

[1] http://wordnet.princeton.edu/

queries in the natural language and also generating answers as sentences in the natural language. Usability testing of the natural language approach to the legal knowledge follows. At the initial stage of the project STEP was interfacing with PostgreSQL databases only. As part of my work on STEP I created ODBC interface. This was neccessary to enable interfacing of STEP with a MS Access database.

2.8 Goals

As explained above, STEP enables an easy integration with new applications. In this project, this system will be used as a base for a legal application. The user should be able to ask questions about legal regulations related to the intellectual property. Information about relations of the regulation with some legal subcategories and keywords that this regulation match, will be also made available. The knowledge base will be stored in a relational database in a DBMS that can be accessible using the SQL language. Next task is to provide an appropriate phrasal lexicon, in order to enable the user asking questions in the syntax he might expect. This task can involve some challenges, since the legal language is very specific and can cause problem to casual users. Sample queries are also to be prepared to point the user on the right way of asking questions to the application.

Main task of the project is to test a usefulness of the natural language interface to legal databases. Thus, a usability evaluation is to be performed. Whether the application does what is expected, it means whether the user gets the information he expects (and might expect).

At the beginning stage of the project, STEP was using the proprietary interface in LISP for PostgreSQL DBMS in LISP. A secondary objective of the project is development of the ODBC interface for the STEP system, which will enable using any of the DBMS that provide ODBC drivers. For the source code of the ODBC interface, see Appendix A.

2.9 Methods

At different stages of the project, different tools and approaches were applied. For creating the conceptual model, the CASE tool DIA was used. For collecting data, primary MS Access DMBS was used. In the later stage transferring data to the MySQL appeared necessary to perform basic efficiency tests with concurrent connections, since the STEP system works as a server application. A phrasal lexicon was developed in a system specific syntax, similar to the notation used in the first order logic. Cognitive Walkthrough usability testing method was used to evaluate the usability of the system and to refine the phrasal lexicon. The ODBC interface was written using Common LISP implementation. The implementation was derived from a CLISP package [2] that provides mappings from SQL functions in a library written in C to the form usable in Common LISP using FFI[3] package.

[2] http://www.atp-media.de/odbc-cl/
[3] Foreign Function Call

Chapter 3

Approach

3.1 Preliminaries

Since the legal application is to be implemented as a part of the STEP system, in order to understand how the application was about to be developed, one has to know how the STEP system actually works.

3.1.1 Interaction between the STEP system and the user

STEP [1] is a system for natural language access to relational databases. Online demo for the system can be accessed on the web page. It takes requests to the database in form of natural language sentences and also generates answers in the natural language, There are many systems that process natural language; unique on this system is that LISP processes natural language in bidirectional way - input is natural language and output is in form of natural language sentence. Currently attempts are going on establishing voice interface to the STEP system, using SPHINX package. Sample session looks as follows: The user enters the query as a sentence in the natural language. Sample queries provide cues how the query can be formulated. System parses query and returns the answer to the user. The user can get 4 kinds of results:

- An exact match - system has found an exact match for the query and returned results for the query
- A noisy parse - system has guessed more or less, what the user was asking for. In this case, user is provided with a paraphrase of the guessed query.
- An ambiguous match - multiple interpretations of the query are possible
- No match - The system did not find a parse for the given query.

3.1.2 Internal architecture of the STEP system

STEP runs as a server application and is written in LISP. It can be called through a CGI interface from web browser. In order to process requests in the natural language, the system looks for the query that would correspond to the sentence that the user has

[1] Schema Tuple Expression Processor, http://www.cs.umu.se/ mjm/step/

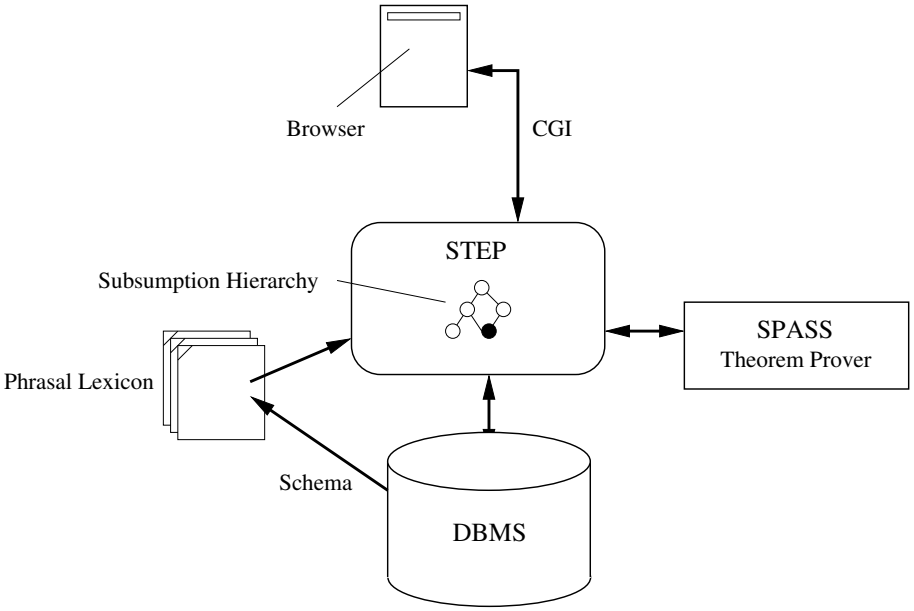

Figure 3.1: Architecture of the STEP system.

asked to the system. In order to generate results in the natural language, the query is sorted into subsumption hierarchy based on the phrasal lexicon and the natural language sentence is constructed.

STEP checks satisfiability of the queries using SPASS theorem prover. For better understanding of the question user has asked, STEP looks for synonyms of the words in the phrasal lexicon using WordNet.

3.1.3 Components of a application in STEP

A legal application works as a module within the STEP system. Therefore it makes sense to discuss what kind of components each STEP application should contain. STEP enables new applications to be included easily, by setting up the following parameters:

- Database schema
- Phrasal lexicon
- Database connection parameters

The database schema includes a set of relations and its attributes and represents the data that is available. Primary keys and foreign keys are also to be specified. A proper specification of the database schema will ensure consistency of the data and will speed up answering SQL queries. For the database schema of the legal application, see Appendix.

Each application has its own phrasal lexicon; set of phrases that system will "understand" when parsing the query and that will be used to generate the answers. An important point is to supply the phrasal lexicon with synonyms to enable as much

queries to be understood as possible. The phrasal lexicon is compiled into a subsumption hierarchy that will be used to parse phrases into the right position in the hierarchy.

The last component of each STEP application is the connection string the enables interfacing with the database specified above. One can specify rights such as protecting the application with an appropriate password.

3.2 Task steps

This project involved the following steps:

3.2.1 Analyzing data model

The first stage of the project was analyzing the data model. It includes what kind of information will be stored in the database. Legal acts are complex documents there is a lot of features that are related to it. It was not feasible to store each and every information about legal regulations. There was also no need for that. Goal of this project was to prove usability of the natural language interface and not to store a complete data about each legal document.

3.2.2 Preeliminary database schema

The next step involved developing preeliminary database schema. Entity Relationship Diagrams notations has been used.

3.2.3 Collecting data

After developing of the preeliminary database schema, the next aim was to collect data about legal acts pertaining to the Intellectual Property legal domain. Many sources was available, most complete one was CLEA (Collection of Law for Electronic Access) online library. The Collection of Laws for Electronic Access is a unique electronic database providing easy access to intellectual property legislation from a wide range of countries and regions as well as to treaties on intellectual property. It is an invaluable information resource made available free of charge by WIPO to all interested parties, including researchers, legal professionals, policy-makers, students and administrators. It stores information about legal acts that are in force in particular countries, as well as international conventions. Different language versions of each text are available in HTML and PDF format.

3.2.4 Final database schema

When the collecting of data was done, not all kind of information appeared necessary. Some attributes about legal acts have been dropped. After deciding on relevant part of legal information, final database schema has been developed.

3.2.5 ODBC interface for PostgreSQL

At the initial stage of the project, STEP system was using a proprietary interface to the PostgreSQL DBMS. One of the goals of the project was to establish the database connectivity via the ODBC interface. This standard enables using most useful SQL

features of popular SQL database systems. Thanks to implementing this feature, STEP can interface with any DBMS system that provides ODBC driver. Appendix C contains the source code of the ODBC interface.

3.2.6 Populating the database

In the end of the project the database schema has been transformed into a physical representation inside the database. Primary, the DBMS system was Microsoft Access. Actually PostgreSQL DMBS is used to store the data, since the STEP system works as a server and it is meant to handle many requests at the same time. PostgreSQL can handle many concurrent connections properly and therefore it is a proper system to be used.

3.2.7 Development of the phrasal lexicon and the sample queries

One of the most crucial steps in developing the system was providing a proper legal lexicon. This lexicon is meant to contain words and phrases that will be understood by the system. On the other hand, the time to load the STEP systems loads logarithmically with the size of the phrasal lexicon. Therefore it was necessary to balance up between enough number of words and synonyms for each phrase, and picking the words carefully not to overload the system with unimportant information. One source of help was WordNet lexicon which provides synonyms to phrases. WordNet lexicon contains words that are in common use and does not contain words and phrases which are parts of legal language. Therefore it was necessary to develop a separate legal lexicon.

3.2.8 Initial tests of the application

In the next stage initial tests of the application were performed. It showed that most users are not aware of the strict meaning of some words that are part of the legal knowledge. Users do not distinguish well between different kinds of legal acts. For instance, they often use words ‘statue‘, ‘code‘, ‘decree‘ as synonyms, despite the fact that do not always mean the same.

3.2.9 Refinements of the phrasal lexicon, and updating the sample queries

Adding more synonyms was necessary and more example queries were needed to guide the user on the possible usage of the system.

3.2.10 Final evaluation

Final evaluation was performed by users with and without knowledge of legal language. Results of this are in the Results chapter.

Chapter 4

Results

4.1 Database schema

The first task was to create an appropriate database schema. In order to depict the information that was subject of interest, an EER diagram has been created. (Figure 1).

4.1.1 Entities and relationships

The purpose of the database was to store information about legal acts that pertain to the intellectual property legal domain, as well as information related to these legal acts. Following entities and relationships were used (Figure 2)

LegalAct

The most important entity in the schema is the LegalAct entity. It represents a legal act object and has following attributes:

- id - an unique identifier of a single legal act and the primary key of the entity
- name - a full name of the legal act
- pubDate - date of the publication of the given legal act
- url - full URL link to the text of the regulation, unless not provided
- country - country code of the country which released the legal act.

Category

As described above, the database represents legal acts that somehow pertain to the intellectual property legal domain. Within this domain one can distinguish some subcategories, which are help to categorize legal regulations more precisely. The Entity Category has the following attributes:

- id - unique identifier of a single category and the primary key of the entity
- name - name of the category

Figure 3 depicts all categories available in the database.

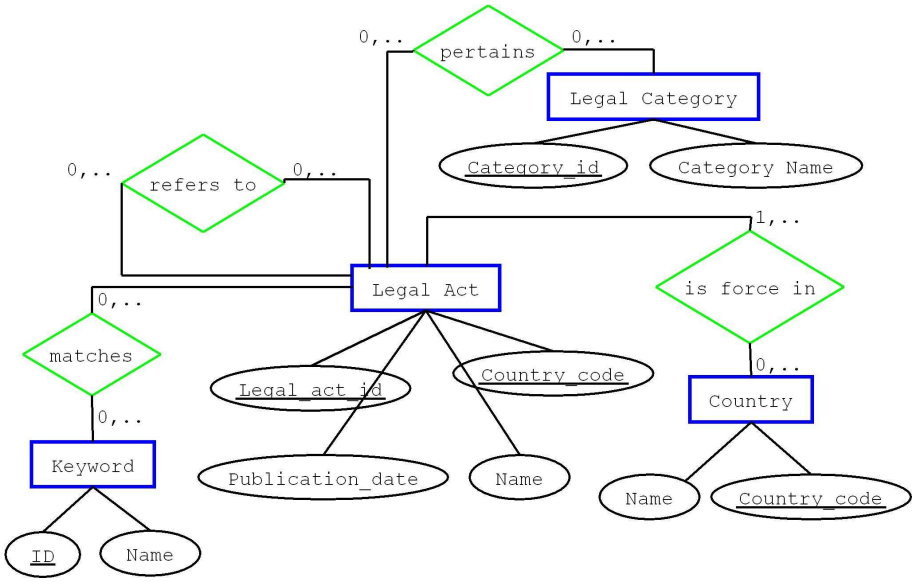

Figure 4.1: EER diagram.

	category_id	category_name
	GEN	Any administrative procedures not covered above
	APO	Appellations of origin / geographical indications
	CIV	Civil judicial procedures and remedies
	CDA	Copyright and Related Rights
	CRI	Criminal procedures
	ENF	Enforcement
	IND	Industrial designs
	TOP	Layout designs (topographies) of integrated circuits
	MRK	Marks
	OTH	Other
	PAT	Patents
	PVV	Plant variety protection
	ABU	Prevention of the abuse of intellectual property rights
	UND	Protection of undisclosed information
	JUD	Provisional judicial measures
	BRD	Special border measure requirements
	UCD	Trade names and unfair competition
	TRT	Transfer of technology
	UMU	Utility models

Figure 4.2: Categories of the Intellectual Property legal domain.

Keyword

Next entity is named Keyword. It represents keywords, that a legal act might match. It has 2 attributes:

- id - a unique identifier of a keyword and the primary key of the entity
- name - the keyword

Country

This entity is meant to depict the countries in the database.

- code - a unique identifier of a country and the primary key of the entity
- name - the name of the country

Pertains

This relationship represesents a dependency between a legal act and a legal category. It has following attributes:

- actId - an identifier of the legal act(entity LegalAct)
- catId - an identifier of the category to which the legal act pertains(entity Category).

This relationship enables connecting of a single legal act with many categories to which the legal act may pertain.

Matches

This relationship represesents a dependency between a legal act and a keyword in the database that the legal act matches. It has the following attributes:

- actId - an identifier of the legal act
- keyId - an identifier of the category to which the legal act pertains(entity Keyword).

Enforces

This relationship represesents a dependency between a legal act and a country that the legal act is in force in. It has the following attributes:

- actId - identifier of the legal act
- code - code of the country

This relation enables binding legal regulation to multiple countries, which is useful by international conventions.

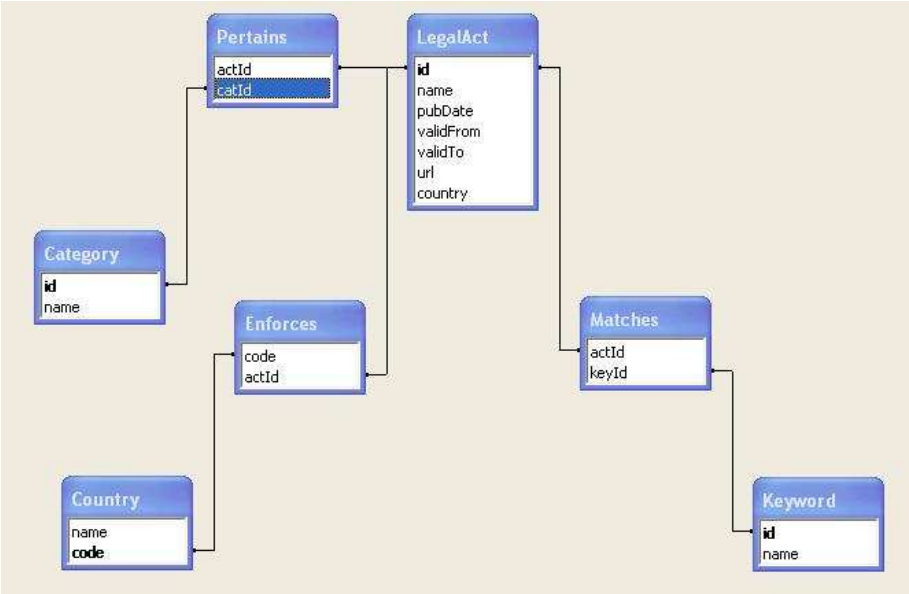

Figure 4.3: Tables.

4.2 Physical representation

The conceptual model was transformed into a schema in the SQL database. The PostgreSQL database DBMS was used. All entities and relations described above became tables in the database, and their attributes became fields of the records in the appropriate tables. The database contains currently round 12000 records, 4500 are records the represent relations between regulations and categories. Round 3500 records represent legal acts.

4.2.1 Representation of phrases in STEP

An important part of every application in STEP are phrasal lexicons, that determine what vocabulary is available for the user in order to ask queries. These phrasal lexicons are developed by specifying a set of entries to be compatible with the database schema. [9] Each entry is of the form hq : p+i where q is a query expression and p+ is one or more phrasal patterns that express the meaning of q. Patterns can specify following parts of the query:

- head - usually a noun
- modifier - adjective
- complement
- answer material, that indicates the way in which answers will be sent back to the user.

```
;acts that pertain to the specified category
 '((x LegalAct
```

```
     (+ (y1 y2) (Category y1) (Pertains y2)
 (*y1)
 (= x id y2 actId) (= y1 id y2 catId)
 ))
   (:comp ("pertaining to " (:ref *y1)))
   )
```

In the phrase "Polish statues that pertain to the patent category", "Polish" is a modifier, "statues" is a head and " with less than 10,000,000 people" is a complement.

In the entry below the elemental expression x—LegalAct(x) is coupled with eight head patterns where features sing and pl indicate singular and plural forms.

```
'((x LegalAct)
  (:head ("legal acts") (:pl))
  (:head ("legal act") (:sing))
  (:head ("acts") (:pl))
  (:head ("act") (:sing))
  (:head ("statues") (:pl))
  (:head ("statue") (:sing))
  (:head ("decretes") (:pl))
  (:head ("decrete") (:sing))
  )
```

For complete phrasal lexicon of the legal database application, see Appendix C.

4.3 Evaluation of the application

There are many metods to evaluate usability of interactive systems. One of them that seems applicable for the legal applicaion, is Cognitive Walkthrough.

4.3.1 Cognitive Walkthrough

Cognitive Walkthrough involves the following steps:

- The user sets a goal to be accomplished with the system
- The user searches the interface for currently available actions
- The user selects the action that seems likely to make progress toward the goal
- The user performs the selected action and evaluates the system's feedback for evidence whether progress is being made toward the current goal. [10]

Using Cognitive Walkthrough when evaluating the legal application, one can test whether the application meet the expectations of the user.

4.3.2 Testing of the legal application using Cognitive Walkthrough

The user was supplied with a web page that contains a text box. In the text box the user can type in queries to the system. In this manner the sample queries are helpful, with them the user might try to check how system works.

The goal to be achieved by the user was to find information about:

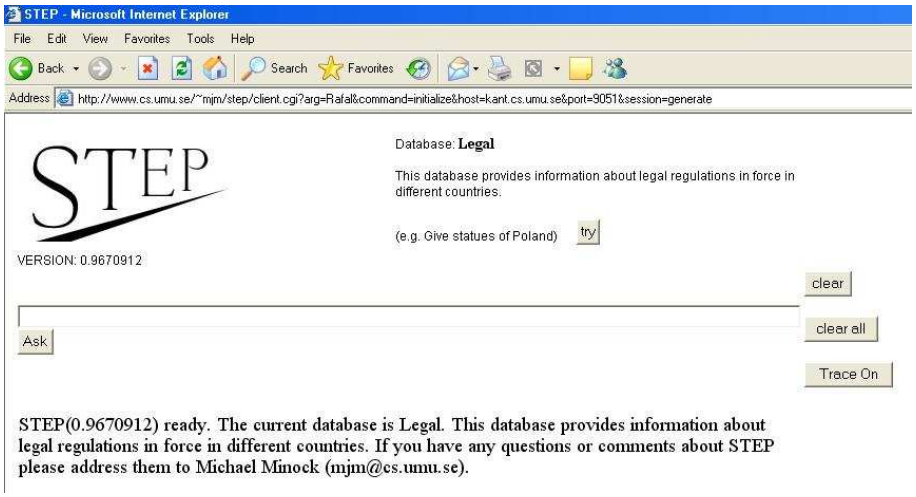

Figure 4.4: Welcome page of the legal application.

- available categories (see Figure 4.9)
- legal acts in particular countries (see Figure 4.5)
- legal acts in particular countries and pertain to the particular category (see Figure 4.8)
- legal acts that match some specified keyword(see Figure 4.6)

If the user asks questions about legal acts, the system presents answers in form of links, that the user might follow to retrieve text of the desired legal regulation. Otherwise, results (if any) are simply listed below.

After submitting the query and evaluating the feedback, user can learn whether the query was parsed succesful and whether it has given result as expected.

4.3.3 Results of the evaluation

It can be stated that evaluation of the application has given satisfiable results. Even users with no knowledge about legal terminology could retrieve the information that they were looking for. It is also crucial for preventing the user from getting frustrated. To point the user in the right way of asking questions to the systems, sample queries are of big importance. After the natural language interface has proven its usefulness for accessing legal databases, the system can be improved by answering more complex legal queries. For some proposals, see section Future Work.

Figure 4.5: Legal regulations in a specified country.

Figure 4.6: statues that match the specified keyword.

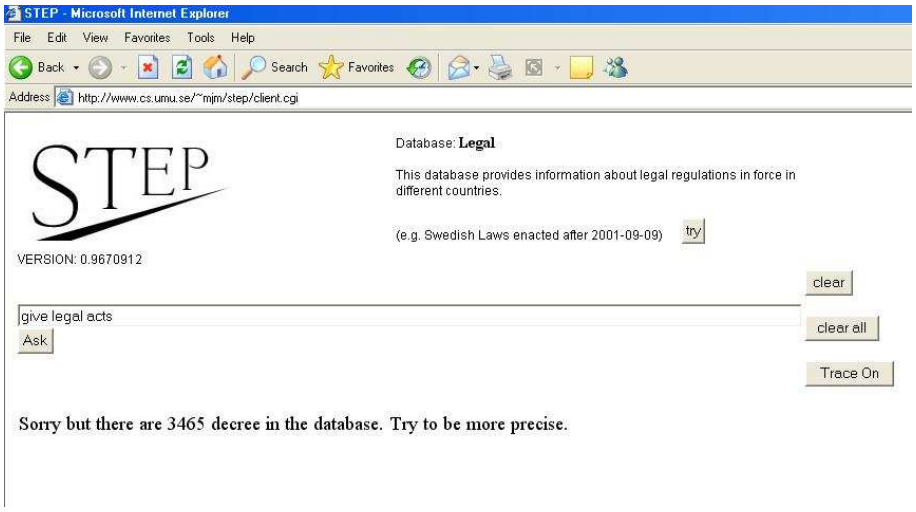

Figure 4.7: Answer to the query about available statues.

Figure 4.8: statues that pertain to the specified category in a specified country.

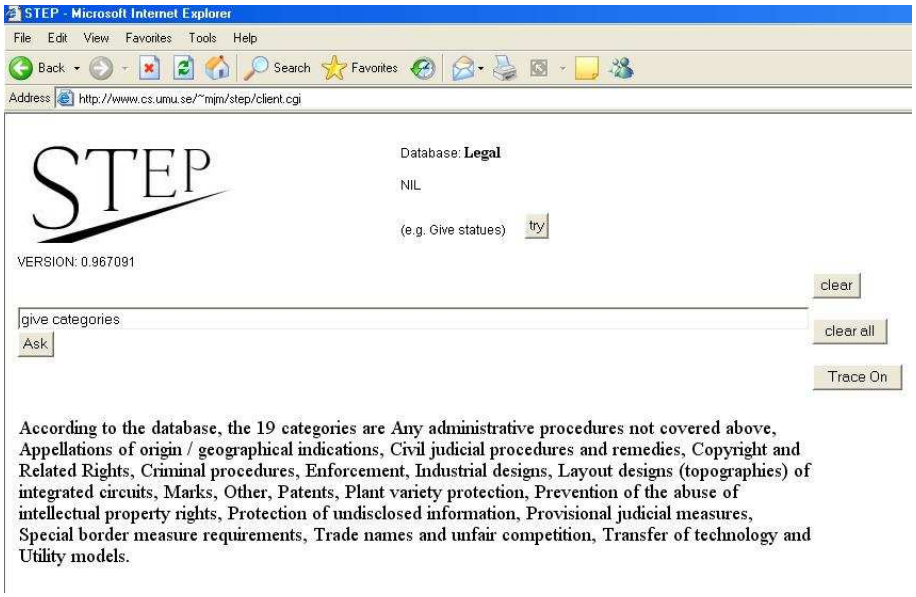

Figure 4.9: Available categories in the database.

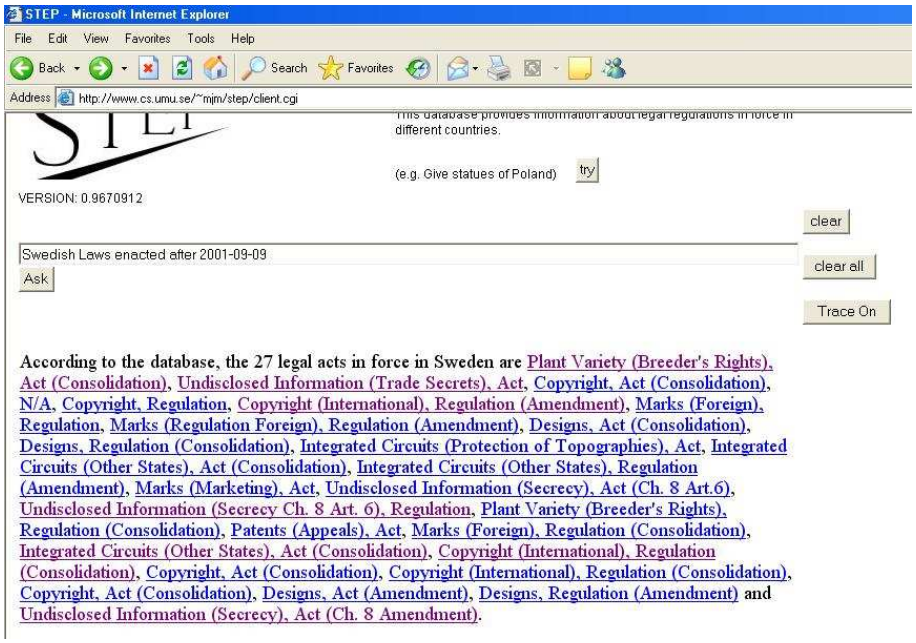

Figure 4.10: Statues in a country enacted after a specified date.

Chapter 5

Conclusions

Legal regulations are a suitable subject for natural language processing systems. First, legal acts are written in very formal language, which makes them easy to be processed. Next, natural language is more intuitive for users that look for legal information, especially lawyers, because they usually work with documents written in natural language, such as notes etc. Furthermore, legal domains fit well into legal hierarchies, what facilitates providing more general information, that user did not specify, but might have interest in. With some extensions and refinements the legal application could become very useful source of a legal information.

Quite large amount of the time involved collecting data. Development of the database schema was relatively straightforward. Creating of phrasal lexicon was not so time consuming, some refinements appeared to be necessary after primary tests of the system. The primary goal was intended to supply the user with more comprehensive information about legal acts and relationships between them, but these issues does not affect value of the project in a very big extent. This goal has been achieved.

5.1 Limitations

The current application runs only over the Intellectual Property legal domain. Furthermore, the number of keywords covered in the database, is small. The intention was to evaluate a natural language interface to legal database, rather than to provide a complete source of a legal information, which would definitely be more time consuming and not helpful for the goal that was set.

5.2 Future work

System described above can provide a link with the content of a given legal act. A useful extension of the system would be extracting of these sections of the legal act that are really relevant for the specified keyword, instead of providing the entire content. For providing a complete legal knowledge base system in the Intellectual Property domain, the database would have to be updated with the international conventions, that are important part of the IP legislation.

Chapter 6

Acknowledgements

I would like to thank my supervisor, Michael Minock for his help at each stage of the project. I would also like to thank Per Lindstrom for helping me during the entire time I spent in Umea. Thanks to Michal Antkowiak for helping me to deal with all LaTeXissues.

References

[1] Joseph D. Becker. The phrasal lexicon. In *Proceedings of the 1975 workshop on Theoretical issues in natural language processing*, pages 60–63, Morristown, NJ, USA, 1975. Association for Computational Linguistics.

[2] Graham Creenleaf, Andrew Mowbray, and Alan Tyree. The datalex legal workstation: integrating tools for lawyers. In *ICAIL '91: Proceedings of the 3rd international conference on Artificial intelligence and law*, pages 215–224, New York, NY, USA, 1991. ACM Press.

[3] Deborah L. Edwards and Dirk E. Mahling. Toward knowledge management systems in the legal domain. In *GROUP '97: Proceedings of the international ACM SIGGROUP conference on Supporting group work : the integration challenge*, pages 158–166, New York, NY, USA, 1997. ACM Press.

[4] F. Haft, R. P. Jones, and Th. Wetter. A natural language based legal expert system for consultation and tutoring: the lex project. In *ICAIL '87: Proceedings of the 1st international conference on Artificial intelligence and law*, pages 75–83, New York, NY, USA, 1987. ACM Press.

[5] Karen Sparck Jones. Natural language and databases, again. In *Proceedings of the 22nd conference on Association for Computational Linguistics*, pages 182–183, Morristown, NJ, USA, 1984. Association for Computational Linguistics.

[6] Klaus-Dieter Krageloh and Peter C. Lockemann. Access to data base systems via natural language. In *Natural Language Communication with Computers*, pages 49–86, London, UK, 1978. Springer-Verlag.

[7] Minock M., Flank A., and Olofsson H. The STEP system. 2004.

[8] L. J. Matthijssen. An intelligent interface for legal databases. In *ICAIL '95: Proceedings of the 5th international conference on Artificial intelligence and law*, pages 71–80, New York, NY, USA, 1995. ACM Press.

[9] Michael Minock. A phrasal approach to natural language interfaces over databases. Technical report, Dept. of Comp. Sc., University of Umeå, Umeå, Sweden, 2005.

[10] William M. Newman, Michael G. Lamming, and Mik Lamming. *Interactive System Design.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1995.

[11] William C. Ogden and Susan R. Brooks. Query languages for the casual user: Exploring the middle ground between formal and natural languages. In *CHI '83: Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pages 161–165, New York, NY, USA, 1983. ACM Press.

[12] Anja Oskamp, Maaike W. Tragter, and Arno R. Lodder. Mutual benefits for ai & law and knowledge management. In *ICAIL '99: Proceedings of the 7th international conference on Artificial intelligence and law*, pages 126–127, New York, NY, USA, 1999. ACM Press.

[13] Vijay Sethi. Natural language interfaces to databases: Mis impact, and a survey of their use and importance. In *CPR '86: Proceedings of the twenty-second annual computer personnel research conference on Computer personnel research conference*, pages 12–26, New York, NY, USA, 1986. ACM Press.

[14] Andre Valente and Joost Breuker. On-line: an architecture for modelling legal information. In *ICAIL '95: Proceedings of the 5th international conference on Artificial intelligence and law*, pages 307–315, New York, NY, USA, 1995. ACM Press.

[15] Pepijn R. S. Visser, Robert W. van Kralingen, and Trevor J. M. Bench-Capon. A method for the development of legal knowledge systems. In *ICAIL '97: Proceedings of the 6th international conference on Artificial intelligence and law*, pages 151–160, New York, NY, USA, 1997. ACM Press.

[16] Robert Wilensky and Yigal Arens. Phran: a knowledge-based natural language understander. In *Proceedings of the 18th conference on Association for Computational Linguistics*, pages 117–121, Morristown, NJ, USA, 1980. Association for Computational Linguistics.

[17] Marianne Winslett, Kenneth Smith, and Xiaolei Qian. Formal query languages for secure relational databases. *ACM Trans. Database Syst.*, 19(4):626–662, 1994.

[18] Uri Zernik and Michael G. Dyer. Disambiguation and language acquisition through the phrasal lexicon. In *Proceedings of the 11th coference on Computational linguistics*, pages 247–252, Morristown, NJ, USA, 1986. Association for Computational Linguistics.

Appendix A

Source Code of the ODBC interface

```
(in-package :db)


(defvar *henv* nil)
(defvar *hdbc* nil)
(defvar *hstmt* nil)
(defvar *errors* nil)

(defvar row nil)

(defun odbc_success (ret-val)
  (or (eql ret-val SQL_SUCCESS) (eql ret-val SQL_SUCCESS_WITH_INFO)))

(defun append_error (text)
  (setf *errors* (append *errors* (list text))))

(defun clear_errors ()
  (setf *errors* nil))

(defun no_errors ()
  (zerop (length *errors*)))

(defun ce ()
  (clear_errors))

(defun commit ()
  (let (ret)
    (when *hdbc*
      (setf ret (SQLEndTran SQL_HANDLE_DBC (%VAL *hdbc*) SQL_COMMIT)))
    (when (odbc_success ret) "OK")))

(defun disconnect ()
```

```
  (let ((ret nil))
    (unless *hdbc*
      (append_error "not connected")
      (return-from disconnect))
    (setq ret (SQLDisconnect (%VAL *hdbc*)))
    (unless (odbc_success ret)
      (append_error "disconnect from database failed"))
    (setq ret (SQLFreeHandle SQL_HANDLE_DBC (%VAL *hdbc*)))
    (setf *hdbc* nil)
    (unless (odbc_success ret)
      (append_error "release of db handle failed"))
    (setq ret (SQLFreeHandle SQL_HANDLE_ENV (%VAL *henv*)))
    (unless (odbc_success ret)
      (format t "release of environment handle failed!~%")))
  (when (no_errors) "OK"))

(defun sql-exec-i (statement)
  (let (ret)
    (setq ret (SQLAllocHandle SQL_HANDLE_STMT (%VAL *hdbc*) (%ADR *hstmt*)))
    (unless (odbc_success ret)
      (append_error "allocation of statement handle failed")
      (return-from sql-exec-i))
    (setq ret (SQLExecDirect (%VAL *hstmt*) statement))
    (unless (odbc_success ret)
      (append_error "execution of statement failed"))
    (sql-diag))
  (when (no_errors) "OK"))

(defun sql-exec (statement)
  (let (ret)
    (sql-exec-i statement)
    (sql-diag)
    (setq ret (SQLFreeHandle SQL_HANDLE_STMT (%VAL *hstmt*)))
    (unless (odbc_success ret)
      (append_error "release of statement handle failed")
      (return-from sql-exec)))
  (when (no_errors) "OK"))

(defun sql-diag ()
  (let ((sqlstate (make-c-string 8))
        (sqlnative (make-c-long))
        (sqlmsg  (make-c-string 256))
        (txtlen (make-c-long))
        ret)
    (setf ret (SQLGetDiagRec SQL_HANDLE_STMT (%val *hstmt*) 1
                             (%adr sqlstate) (%adr sqlnative)
     (%adr sqlmsg) 256 (%adr txtlen)))
    (when (odbc_success ret)
      (format t "~%~a" (%val sqlmsg)))
```

```
    ))

(defun describe-col (hstmt index)
  (let ((ret nil)
        (ColumnNamePtr (make-c-string 80))
        (NameLengthPtr      (make-c-long))
        (DataTypePtr        (make-c-int))
        (ColumnSizePtr      (make-c-long))
        (DecimalDigitsPtr   (make-c-long))
        (NullablePtr        (make-c-long)))
    (setf ret (SQLDescribeCol (%VAL hstmt)
                              index
                              (%ADR ColumnNamePtr)
                              80
                              (%ADR NameLengthPtr)
                              (%ADR DataTypePtr)
                              (%ADR ColumnSizePtr)
                              (%ADR DecimalDigitsPtr)
                              (%ADR NullablePtr)))
    (if (not (odbc_success ret))
(progn
  (append_error (format nil "SQLDescribeCol failed with: ~a" ret))
  (setf ret nil))
      (setf ret
    (list :name           (%VAL ColumnNamePtr)
  :name-length    (%VAL NameLengthPtr)
  :data-type      (%VAL DataTypePtr)
  :column-size    (%VAL ColumnSizePtr)
  :decimal-digits (%VAL DecimalDigitsPtr)
  :nullable       (%VAL NullablePtr))))))

(defun dsc-col (index)
  (describe-col *hstmt* index))


(defun odbc-exec (statement)
  (setq reslist())
  (setq result())
  (let ((col-count-ptr (make-c-int))
        (ret nil)
        (cols nil)
        (size nil)
        (row nil))
    (clear_errors)

    (sql-exec-i statement)
    (when (no_errors)
      (setf ret (SQLNumResultCols (%VAL *hstmt*) (%ADR col-count-ptr)))
      (unless (odbc_success ret)
```

```
        (append_error (format t "NumResultCols returned ~a" ret))
        (return-from query))
      (setf cols (%VAL col-count-ptr))
;
      (setf row nil)
      (let ((i 0)
            (tmp1)
            (line ""))
        (setf row nil)
        (loop
 (when (> (incf i) cols) (return))
 (setf line
       (string-append line
      (format nil "~12@a" (getf (dsc-col i) :name))))
 (setf tmp1 (make-c-long))
 (setf (%VAL tmp1) 0)
 (setf size (getf (dsc-col i) :column-size))
 (when (> size 4096) (setf size 4096))
 (setf row (append row (list (make-c-string size) tmp1)))
 (setf ret (SQLBindCol (%VAL *hstmt*)
       i
       SQL_C_CHAR
       (%ADR (nth (* (1- i) 2) row))
       255
       (%ADR (nth (1+ (* (1- i) 2)) row))))
 (unless (odbc_success ret)
   (append_error (format nil "SQLBindCol returned: ~a" ret))
   (return-from query))
 )
        )
;
      (unless (zerop cols)
        (loop ;;////////petla iteracji po tuplach, czyli rzedach
 (setf ret (SQLFetch (%VAL *hstmt*)))
 ;; (sql-diag)
 (unless (odbc_success ret)
   ;;(print (format nil "SQLFetch returned: ~a" ret))
   (return))
 (let ((i 0)
       (tmp nil)
       (line ""))
   (loop ;;petla iteracji po columnach -- ???
    (when (> (incf i) cols) (return))
    (setf tmp (nth (* (1- i) 2) row))

    (push (%VAL tmp) result)

    (when tmp (setf tmp (%VAL tmp)))
    (setf tmp (string-trim '(#\Space) tmp))
```

```
    (setf tmp1 (%VAL (nth (1+ (* (1- i) 2)) row)))
    (when (= tmp1 -1) (setf tmp "NULL"))
    (setf line (string-append line (format nil "~12@a" tmp)))
    ;;(setf line (string-append line (format nil "~12@a" tmp1)))
    )

   ;;//////////my tests
   (push (reverse result) reslist)
   (setq result nil)
   ;;//////////my tests

   )))
      (setq ret (SQLFreeHandle SQL_HANDLE_STMT (%VAL *hstmt*)))
      (unless (odbc_success ret)
        (append_error "release of statement handle failed"))
      ;;  (return)))
      ))
  (if (no_errors)
      (reverse reslist)
    *errors*))

(defun odbc-connect (&key (database "") (user "") (password "")
  (library "/usr/lib/libiodbc.so.2"))
  (setf *odbc-on* t)
  (init-odbc-callouts library)
  (setf kernel::*map-to-lower-case* nil)
  (let ((ret nil))
    (setf *henv*  (make-handle))
    (setf *hdbc*  (make-handle))
    (setf *hstmt* (make-handle))
    (setq ret (SQLAllocHandle SQL_HANDLE_ENV SQL_NULL_HANDLE (%ADR *henv*)))
    (unless (odbc_success ret)
      (append_error "failed to allocate environment handle")
      (return-from odbc-connect))
    (setf ret (SQLSetEnvAttr (%VAL *henv*)
     SQL_ATTR_ODBC_VERSION SQL_OV_ODBC3 SQL_NTS))
    (unless (odbc_success ret)
      (SQLFreeHandle SQL_HANDLE_ENV (%VAL *henv*))
      (append_error "setting environment to ODBC 3 failed")
      (return-from odbc-connect))
    (setq ret (SQLAllocHandle SQL_HANDLE_DBC (%VAL *henv*) (%ADR *hdbc*)))
    (unless (odbc_success ret)
      (SQLFreeHandle SQL_HANDLE_ENV (%VAL *henv*))
      (append_error "allocation of connection handle failed")
      (return-from odbc-connect))
    (setq ret (SQLConnect (%VAL *hdbc*) database user password))
    (unless (odbc_success ret)
      (SQLFreeHandle SQL_HANDLE_ENV (%VAL *henv*))
      (SQLFreeHandle SQL_HANDLE_DBC (%VAL *hdbc*))
```

```
    (append_error "connection could not be established")
    (return-from odbc-connect)))
"OK")
```

Appendix B

Database specification

```
;;(log-kernel)
(setf *log-odbc* nil) ;;

(set-db
 :name "Legal"
 :description "This database provides information about legal
regulations in force in different countries."
 :schema
 '(
   (LegalAct (strid id name country date url)
     :date (date)
     :key (strid) :source (name country)
     :unique (name))
   (Country (name code)
    :key (code) :unique (name) :source (name))

   (Category (id name) :key (id) :source (name))
   (Pertains (id legalact category)
             :key (legalact category))


   (Enforces (id country legalAct))

   (Refers (act1 act2) :source(act1 act2))
   (Keyword (id name) :key (id) :source (name))
   (Matches (id legalAct keyword) :source(keyword))
   )

 :connection
 (odbc-connect :database "legal" :user "legal" :password "legal"
               :library "/usr/lib/libiodbc.so.2"
       ;;:library "odbc32.dll"
       )
```

```
:example-queries
'("Give statues of Poland"
  "Laws regarding Patents of Sweden"
  "German Laws"
  "Swedish Laws enacted after 2001-09-09"
  )
)
```

Appendix C

Phrasal lexicon

```
(set-pl
 :english

 '((x LegalAct)
   (:head ("legal acts") (:pl))
   (:head ("legal act") (:sing))
   (:head ("laws") (:pl))
   (:head ("law") (:sing))
   (:head ("acts") (:pl))
   (:head ("act") (:sing))
   (:head ("statues") (:pl))
   (:head ("statue") (:sing))
   (:head ("decrees") (:pl))
   (:head ("decree") (:sing))
   )

 '((x LegalAct (= x name $c1))
   (:comp ("named " $c1))
   (:head ($c1) () (:proper-name))
   )

 '((x LegalAct (= x url $c1) (= x name $c2))
   (:answer ("<a href=\"" $c1 "\">" $c2 "</a>"))
   )

 '((x LegalAct (> x date $c1))
   (:comp ("enacted after " $c1)))

 '((x LegalAct (< x date $c1))
   (:comp ("enacted before " $c1)))

 '((x LegalAct (>= x date $c1) (< x date $c2))
```

```
  (:comp ("enacted between " $c1 " and " $c2)))

'((x LegalAct
     (+ (y1) (Country y1)
(= x country y1 code)
(*y1)
))
  (:comp ("in force in " (:ref *y1)))
  (:comp ("of " (:ref *y1)) (:nlu))
  )

'((x LegalAct
     (+ (y1) (Country y1)
        (= x country y1 code)
        (= y1 name $c1)
        ))
  (:comp ("in force in " $c1))
  (:comp ("of " $c1) (:nlu))
  (:mod ($c1) (:nlu)))


'((x LegalAct
     (+ (y1 y2) (Category y1) (Pertains y2)
(*y1)
(= x strid y2 legalAct) (= y1 id y2 category)
))
  (:comp ("pertaining to " (:ref *y1)))
  (:comp ("regarding " (:ref *y1)) (:nlu))
  (:comp ("concerning " (:ref *y1)) (:nlu))
  (:comp ("about " (:ref *y1)) (:nlu))
  )


'((x Country)
  (:head ("countries") (:pl))
  (:head ("country") (:sing))
  (:head ("nations") (:pl))
  (:head ("nation") (:sing))
  )

'((x Country (= x name $c1))
  (:comp ("named " $c1))
  (:head ($c1) () (:proper-name))
  (:answer ($c1))
  )



'((x Category)
```

```
  (:head ("categories") (:pl))
  (:head ("category") (:sing))
  )

'((x Category (= x name $c1))
  (:comp ("named " $c1))
  (:head ($c1) () (:proper-name))
  (:answer ($c1))
  )

)
```