MATES — an experimental linguistic database system

Martin Holub and **Pavel Míka**

Center for Computational Linguistics
Faculty of Mathematics and Physics
Charles University, Prague
Malostranské nám. 25, 118 00 Praha, Czech republic
e-mail: `holub@ksi.ms.mff.cuni.cz`, `mika@uhelnytrh.cz`

Abstract

In this paper we give brief information on an originally developed linguistic database system. It concerns its architecture, provided means for managing data and the possible utilization of this system.

1 Introduction

The linguistic database system MATES (Master of Text Sources) has been intended, designed and implemented and is being used as an experimental system; it works with Czech textual documents annotated either manually (in the framework of the Prague Dependency Treebank project) or automatically (using automatic procedures of morphological and syntactic tagging). The MATES system serves primarily for academic and research purposes, particularly for investigating NLP tasks and doing experiments with empirical data. MATES enables storing, processing and searching large amounts of linguistic data needed both for development of NLP methods and for testing their possible applications.

When developing the MATES system, we preferred robustness and universality of the system as criteria of design suitability rather than an efficient implementation of selected parts of the system. The consistency of the database is ensured by transaction processing.

MATES was being developed for 12 months by a group of six people. Since the end of the development it has been in a one-year trial operation (it has been tested by about six people different from the developers); during this time, we have got much valuable experience with its performance. Now we are developing a new version with respect to the requirements that came from the users who have practically worked with this system.

2 System architecture

MATES is a multiuser system which widely makes use of relational database technology for storing data. It consists of three basic parts (see Fig. 1-A): first, a MATES server, second, a database machine (namely the Oracle v. 8 database server), and third, a client part of the system, the MATES text client. Each of the three main modules can be run on a different computer (they can communicate via network).

2.1 Data entities in MATES

There are three basic data entities in MATES: documents, collections and lemmas. Documents are unstructured (or arbitrarily-structured) texts in natural language. MATES is set for Czech; it would be possible to adapt it for another language, but not uncomplicated. Collections are sets of documents. A document can be a member of more collections. Lemmas are (morphologically) basic forms of words. In fact, documents are sequences of word forms, but we can consider them as sequences of lemmas. Each lemma and each document is assigned a unique identifier, a lemma ID (LID) and a document ID (DID), respectively. When working with lemmas or documents, MATES almost always deals with these integers.

Furthermore, there is another (very important) data structure in MATES that makes space for storing values that characterize documents or lemmas or their relations. These data are called features and are of two types: system features that are read-only for users (their values are computed and stored by the system) and user features that are completely managed by users. The meaning of the features and their structure will be described in subsection 3.2.

2.2 Users of MATES

How can users manage data in MATES? As for putting data into the database, they can create collections of documents, put documents into collections, select a method of their linguistic processing and index the inserted documents. The process

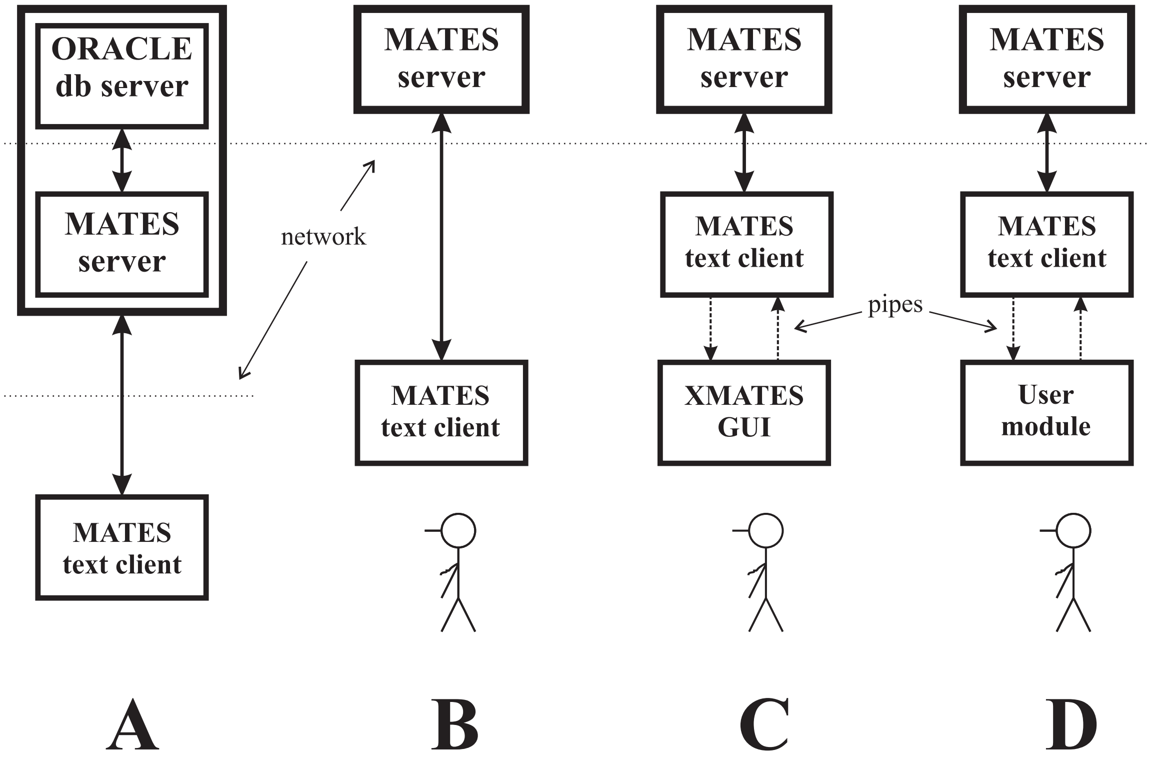

Figure 1: A rough scheme of the MATES architecture.

of the indexing will be explained in more details in subsection 4.1; among others, it causes two important effects: lemmas found in the inserted documents are stored in the MATES dictionary, and the system features are computed. Then users can access MATES data and retrieve them under conditions specified in appropriate MATES queries. Users can define both structure and content of the user features and in these data structures store presumably useful data. Then stored values of features can be either retrieved or used as conditions for retrieving another data.

The user who puts a document into the MATES database is the owner of the document. Analogously, the user who creates a collection is its owner. Each document and each collection has a private or public status (set by their owners). Private documents and collections can be accessed only by their owner; the public ones can be read or searched by any other user. Only the owner of a collection has the right to insert/delete a document into/from the collection and create or update user features.

2.3 The Mates Query Language

The MATES text client provides a low-level communication interface in the form of the Mates Query Language (MatesQL or MQL). MQL is a non-procedural query language; it consists of several dozens statements that cover all operations that a user (or a program that makes use of the MATES system) can run including data definition, manipulation, and retrieval. SQL-like statements make MQL powerful and easy to learn. Since MQL has been designed for low-level communication, it is suitable for developers or for machine communication rather than for an end-user (it is too uncomfortable for humans).

2.4 The XMATES and user modules

There are three basic ways how users can work with MATES; they differ in the way of communication with the system. The first, and the simplest, but also the most uncomfortable, possibility is to work in text mode using only the MATES text client as a communication interface (see Fig. 1-B). In this case, the communication language is the MQL.

Second, users can use the graphic user interface

XMATES, which is an X-windows application (see Fig. 1-C). Many of the basic functions that every user often needs can be performed using mouse and a menu system. XMATES also provides a window for editing MQL statements. Users can browse the history of entered queries and returned answers and edit them. The way of presentation of MATES answers is also much better than in the case of working with the mere MATES text client.

The third alternative how to work with MATES is to create and use a specialized user module which can define a specialized user language for communication with users according to their needs (see Fig. 1-D). User's requirements or queries are then translated into a set of MQL commands and results of the (possibly very complex) interaction with MATES are presented to user in a suitable form.

3 Data in MATES

3.1 Documents and lemmas

MATES stores both original and linguistically annotated versions of documents and both the versions can be accessed by users. Since the annotated texts have a SGML format and take up large space, they are stored in a compressed form. When they are read by users, they are first transferred from the MATES server to the client and then the decompression runs on the client computer.

We have a limited collection of texts manually annotated in the framework of the Prague Dependency Treebank project that contains about 1.4 mil. words. However, most of the data stored in MATES are annotated by a sequence of automatic procedures. They work stochastically, i.e. the results of annotation are not always correct. The process of automatic annotation will be described in more details in subsection 4.2.

Lemmas are stored in the MATES dictionary. MATES knows only lemmas contained in documents that have been inserted into the database. During inserting, word forms from inserted documents are processed by the built-in lemmatizer and corresponding lemmas are either found in the MATES dictionary or they are inserted to it. The lemmatizer provides also some lexical information about lemmas (for instance their part of speech). This information is also stored in the dictionary and can be retrieved or used as a condition in a MATES query.

3.2 System and user features

Both system and user features are information about documents and lemmas in a given collection; we can consider them as values of functions defined on five domains: $\mathcal{L}$, $\mathcal{D}$, $\mathcal{L} \times \mathcal{D}$, $\mathcal{L} \times \mathcal{L}$ and $\mathcal{D} \times \mathcal{D}$, where $\mathcal{L}$ is the set of lemmas and $\mathcal{D}$ is the set of documents in a given collection. Values of all features are stored in five relational database tables (due to those five domains). The primary keys of these tables are LID, DID, the couple (LID, DID), the couple (LID, LID) and the couple (DID, DID), respectively. Each feature is represented by one column in a table. The values can be of three types: integers, real numbers or strings. We also speak about five types of features: L-features, D-features, LD-features, LL-features and DD-features which characterize lemmas as such, documents as such, binary relations between lemmas and documents, between lemmas and between documents, respectively.

MATES computes and stores three system features: document frequency (df), i.e. the number of documents that contain a given lemma (an L-feature), term frequency (tf), i.e. the number of occurrences of a given lemma in a given document (an LD-feature), and term frequency in collection (tfc), i.e. the number of occurrences of a given lemma in all documents in a collection (an L-feature). The structure of user features is defined for each collection of documents individually by owners of collections.

3.3 The index of word occurrences

The index of word occurrences is the only data which are not stored in relational tables of the database machine. The index is structured as a set of so called inverted files. Each inverted file uniquely corresponds to a lemma and consists of a list of documents in which the given lemma occurs. Retrieving documents that contain a given lemma is a very frequent operation and therefore it should be implemented effectively. This requirement was the only reason for developing a special data structure in which the index is stored. The primary criterion for the design was the effectiveness of retrieving a given inverted file. This criterion has been fulfilled by the property of implementation that each inverted file is stored in a continuous part of file on disk. In order to keep this property even if new documents are indexed, we developed an original algorithm for inserting new records into the index which takes advantage of the fact that the statistical distribution

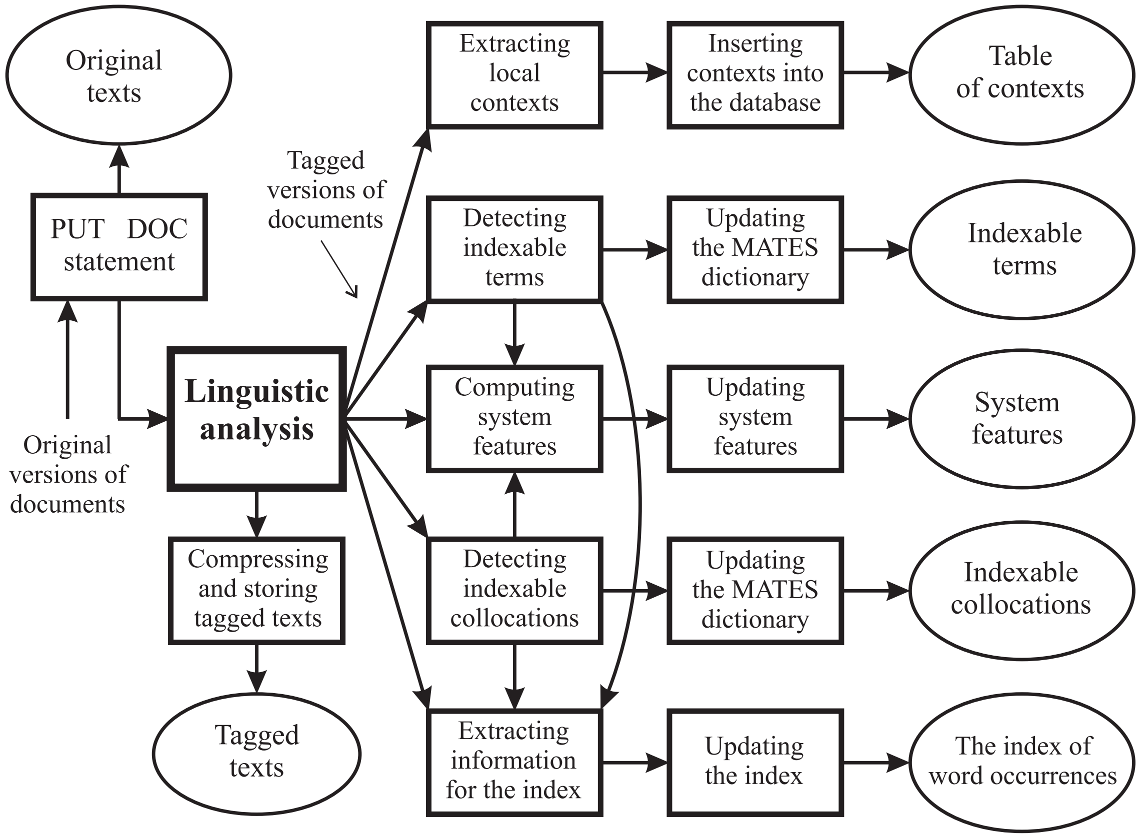

Figure 2: The processing of the tagged texts in the indexing process.

of words in natural language respects the so called Zipf's law.

Now we very briefly describe basic properties of the index and the method of its update. The index is divided into a number of so called areas. Each area contains a set of segments of the same capacity and each segment contains one inverted file. The length of an inverted file is the number of documents in it. All inverted files in an area are of approximately equal length, i.e. they correspond to lemmas of approximately equal frequency in collection. The maximal length of inverted files in the i-th area is equal to the capacity of segments in this area and is denoted by sc_i. The minimal length of inverted files in the i-th area is equal to $sc_{i-1} + 1$. The proportion between the segment capacities in the areas is given by the following equation (which respects the Zipf's law): $sc_{i+1} = \lceil \lambda \cdot sc_i \rceil$ where λ is a constant greater than 1. Note that λ determines the maximal amount of unused free space inside the segments.

If a document is to be added into an inverted file which is stored in the i-th area and whose length is equal to the segment capacity sc_i then the corresponding segment expands and is moved into the (i+1)-th area. This move can recursively cause a limited number of further moves in areas of a greater index. The number of such transfers should be minimal. Details of this algorithm will not be discussed here. Due to Zipf's law all the areas take up approximately equal space, numbers of segment expansions from the areas are balanced, and so the amount of transfered data is optimized. A suitable λ allows to choose a compromise between space taken by the index and effectiveness of update. A greater λ implies more unused space in the index, a smaller λ implies more segment transfers caused by an update operation.

4 Documents preprocessing

4.1 Inserting documents into the database

Inserting documents into the MATES system consists of three processes: inserting the original ver-

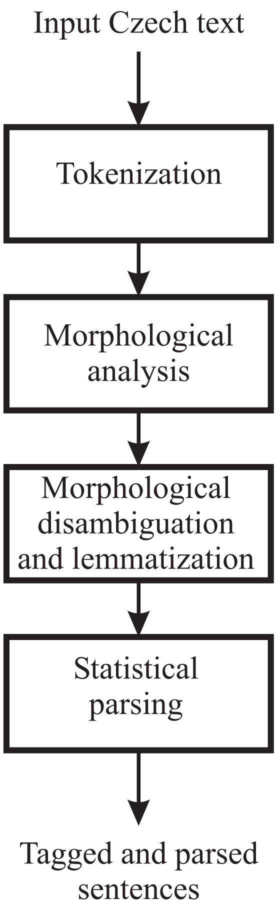

Figure 3: The sequence of the automatic annotation procedures.

sions of documents into the database, then their linguistic preprocessing, i.e. making up the tagged versions of documents using a linguistic analysis, and finally the indexing of the inserted documents. The procedure of the linguistic preprocessing is configurable; users can select it from a given variety that is set by the MATES administrator. The complete scheme of these procedures is depicted in Fig. 2. As it is shown in the figure, the indexing is a very complex process consisting of 1) identification of indexable terms in the text (if they are not found in the MATES dictionary, they are assigned a unique LID and added to it), 2) storing sentence structures in the table of local contexts, 3) computing and updating the system features, 4) detection of indexable collocations (they are assigned unique LIDs in the same way as single indexable terms), and 5) updating the index of word occurrences.

4.2 The automatic linguistic analysis

The automatic linguistic analysis consists of the automatic annotation procedures (see Fig. 3). The MATES project makes use of the Prague Dependency Treebank (for details see [1]) and follows its course of linguistic annotation, both at the morphological and syntactic levels. The input of the linguistic analysis is a Czech text. First, the text is tokenized, i.e. divided into lexical atoms, tokens, and the boundaries of sentences and paragraphs are determined. Then the morphological analysis assigns each word a set of possible lemmas and morphological tags. They are disambiguated in the next stage. Finally, syntactic dependency relations inside each sentence are recognized by a statistical parser; these dependencies make a tree structure. The final result is a sequence of morphologically tagged and parsed sentences.

4.3 Local contexts storing

The support for storing and searching contexts belongs to most significant functions of MATES. During the linguistic preprocessing, each sentence from an analysed document is assigned a sentence identifier (SID) which is unique within the document. Information about the syntactic structure of sentences, which is needed for a local context extraction, is extracted from the tagged texts and stored in a database table. The primary key of this table is the couple (DID, SID) and each row stores information about one sentence. The described data structure allows to retrieve a local context of a word occurrence within a given sentence or the neighbouring sentences within a given distance.

5 Searching for information in MATES

Users can retrieve a set of word occurrences, documents and their attributes, lemmas and information about them, values of both system or user features, and contexts of a given word occurrence or of a given sentence. The search conditions can concern occurrences of lemmas in documents, various document attributes (title, author, etc.), values of any feature, and information about lemmas. The search conditions can be logical combinations of the mentioned types. We cannot go into details here, and so we give only four examples:

Queries for retrieving documents have the form

```
SELECT DOC WITH (<conditions>)
AND/OR WHERE (<conditions>);
```

	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.
1. Electronic dictionaries		•					•	•	•	•				
2. EuroWordNet (for Czech)									•	•	•	•	•	•
3. Annotated corpora				•	•					•				
4. Morphology and lemmatization			•		•	•	•	•		•	•		•	•
5. Syntactic dependency relations			•			•	•	•						
6. Statistic processing of texts	•						•	•		•	•	•		
7. Contexts extraction	•					•		•		•	•	•		
8. Significant collocations	•					•	•		•	•	•		•	•
9. Hierarchy of concepts		•					•			•	•	•	•	•
10. Word sense disambiguation						•	•				•	•	•	•
11. Semantic classes						•	•		•	•			•	•
12. Significance of concepts													•	•
13. Document clustering											•			•
14. Modeling semantic content													•	

Table 1: Mutual support of the NLP tasks which can help to model semantic contents.

for retrieving lemmas have the form

```
SELECT LEMMA WHERE (<conditions>);
```

for retrieving features

```
SELECT FEATURES COLUMNS
(<features>) WHERE (<conditions>);
```

and for retrieving contexts

```
GET CONTEXT LOCAL/SENTENCE/
PARAGRAPH/DOCUMENT LID="..."
[DID="..." [SID="..."]].
```

6 Intended applications

MATES is to support the investigation of NLP tasks, and their applications, as well. Currently, due to our main interests, the system is employed as a tool allowing to effectively combine several NLP methods that can be more or less helpful in the area of intelligent information retrieval. Our aim is to construct suitable data representations of semantic content of text documents. This goal is supported especially by the means listed in Table 1. These means are not mutually independent; many of them can more or less help to master another, relations among them are very complex. We can say that the first five means are basic tools for lexical, morphological and syntactic analysis of texts. A preparatory statistical processing is represented by the next two methods. And the last seven tasks directly support an effective information retrieval. Table 1 shows in each row which procedures can be supported by those in the first column. The most important relations are depicted in a reduced scheme in Fig. 4.

How can MATES be utilized for solving such tasks? For example, when we want to find significant collocations, probably we will work with several stochastic variables. Their values can be computed from large amounts of empirical data stored in MATES as a large collection of documents and then can be stored as user features. Then the experimenter can effectively retrieve sets of values of the variables and can try and combine them to compute a measure of collocating of selected pairs of lemmas. Again, this measure can be stored in MATES as a user feature for testing its usefulness.

7 Conclusions

The MATES system can be viewed as a universal environment for experimental work. It is capable to safely manage large amounts of linguistic data. It supports storing statistical quantities as functions that are defined by user and that can characterize properties of statistical distribution of various linguistic phenomena and their (cor)relations.

Above all we point out that without the statistical processing of empirical data we can know neither the statistical distribution of language phenomena nor the character of the contexts in which they appear; both can be obtained by processing of a large collection of texts. Especially the knowledge of contexts of language structures is necessary if we want to solve the ambiguity of their meaning. MATES is able not only to store results of such statistical investigations, but also to effectively retrieve

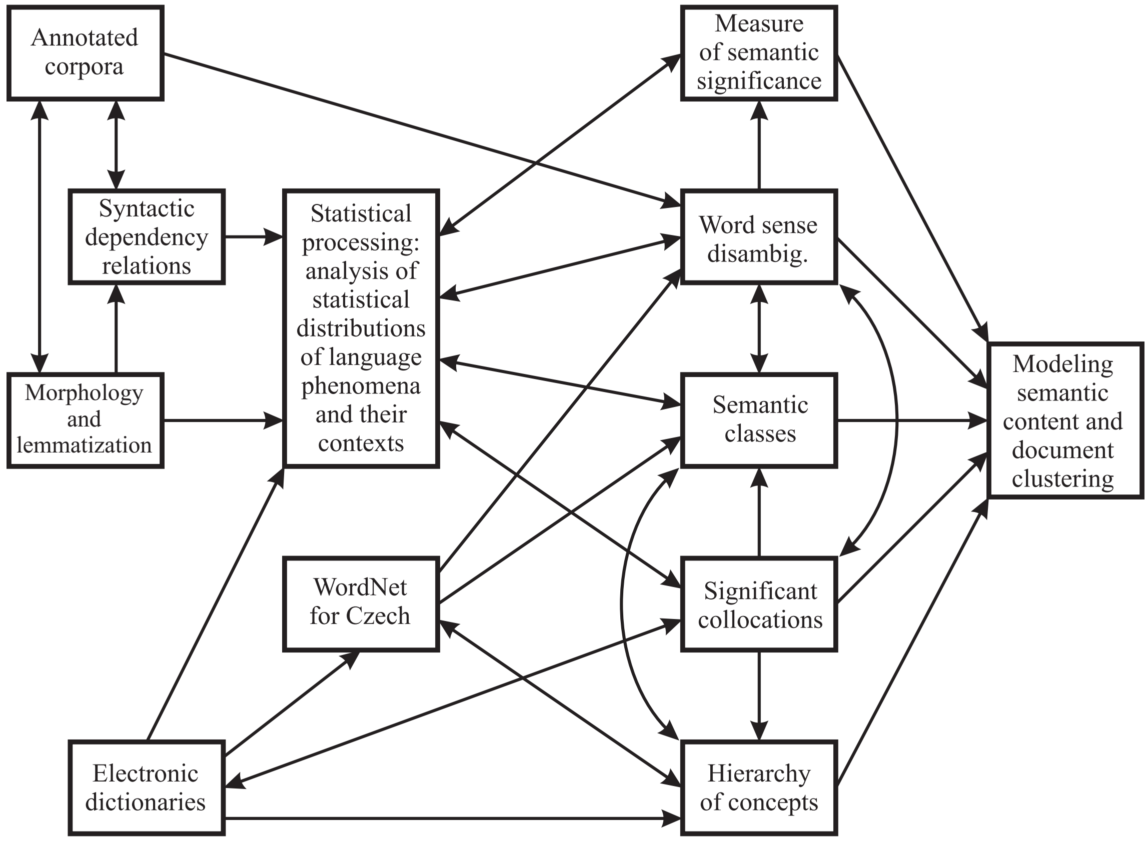

Figure 4: A scheme of some important relations among NLP tasks.

specified sets of values of selected variables. This functionality is very helpful when we look for suitable statistical methods and test their applicability to solving practical NLP problems.

8 Acknowledgements

This work has been supported by the Ministry of Education, project Center of Computational Linguistics (project LN00A063).

Reference

[1] Hajičová, E.; Hajič, J.; Hladká, B.; Holub, M.; Pajas, P.; Řezníčková, V.; Sgall, P.: The Current Status of the Prague Dependency Treebank. Proceedings of the 4th International Conference TSD 2001, ed. by V. Matoušek et al., Berlin: Springer 2001, pp. 11–20.