

5 Experiments for NL-storing of small datasets

Abstract

In this chapter we will present several experiments aimed to show the possibilities of NL-addressing to be used for NL-storing of small size datasets which contain up to one hundred thousands of instances.

The goal of the experiments with different small datasets, like dictionary, thesaurus or ontology, is to discover regularities in the NL-addressing realization. More concretely, two regularities of time for storing by using NL-addressing will be examined:

- *It depends on number of elements in the instances;*
- *It not depends on number of instances in datasets.*

This chapter starts with introduction of the idea of knowledge representation. Further in the chapter three experiments with small size datasets are outlined: for NL-storing of dictionaries, thesauruses, and ontologies. Presentation of every experiment starts with introductory part aimed to give working definition and to outline state of the art in storing concrete structures.

We start with analyzing the easiest one: NL-storing dictionaries. After that, NL-storing of thesauruses will be analyzed. An experiment with WordNet thesaurus and program WordArM based on NL-addressing will be discussed.

At the end, a special attention will be given to NL-storing ontologies. This part of the chapter begins with introducing the basic ontological structures as well as the corresponded operations and tools for operating with ontologies. Further, NL-storing models for ontologies will be discussed and experiments with OntoArM program for storing ontologies based on NL-addressing will be outlined.

5.1 Knowledge representation

In a letter written to Philip Jourdain in 1914, Gottlob Frege had written:

“Let us suppose an explorer travelling in an unexplored country sees a high snow-capped mountain on the northern horizon.

By making inquiries among the natives he learns that its name is 'Aphla'. By sighting it from different points he determines its position as exactly as possible, enters it in a map, and writes in his diary: 'Aphla is at least 5000 meters high'.

Another explorer sees a snow-capped mountain on the southern horizon and learns that it is called Ateb. He enters it in his map under this name.

Later comparison shows that both explorers saw the same mountain. Now the content of the proposition 'Ateb is Aphla' is far from being a mere consequence of the principle of identity, but contains a valuable piece of geographical knowledge. What is stated in the proposition 'Ateb is Aphla' is certainly not the same thing as the content of the proposition 'Ateb is Ateb'.

Now if what corresponded to the name 'Aphla' as part of the thought was the reference of the name and hence the mountain itself, then this would be the same in both thoughts. The thought expressed in the proposition 'Ateb is Aphla' would have to coincide with the one in 'Ateb is Ateb', which is far from being the case. What corresponds to the name 'Ateb' as part of the thought must therefore be different from what corresponds to the name 'Aphla' as part of the thought. This cannot therefore be the reference which is the same for both names, but must be something which is different in the two cases, and I say accordingly that the sense of the name 'Ateb' is different from the sense of the name 'Aphla'.

Accordingly, the sense of the proposition 'Ateb is at least 5000 meters high' is also different from the sense of the proposition 'Aphla is at least 5000 meters high'. Someone who takes the latter to be true need not therefore take the former to be true. An object can be determined in different ways, and every one of these ways of determining it can give rise to a special name, and these different names then have different senses; for it is not self-evident that it is the same object which is being determined in different ways.

We find this in astronomy in the case of planetoids and comets. Now if the sense of a name was something subjective, then the sense of the proposition in which the name occurs, and hence the thought, would also be something subjective, and the thought one man connects with this proposition would be different from the thought another man connects with it; a common store of thoughts, a common science would be impossible.

It would be impossible for something one man said to contradict what another man said, because the two would not express the same thought at all, but each his own.

For these reasons I believe that the sense of a name is not something subjective (crossed out: in one's mental life), that it does not therefore belong to psychology, and that it is indispensable" [Frege, 1980].

What is important in this example is [Ivanova et al, 2013c]:

- The names Ateb and Aphla refer different parts of the same natural object (mountain, let call it *Pirrin*);
- The position of the referred object (mountain) is fixed by any artificial system (geographical co-ordinates, address) which is another name of the same object;
- The names and the address correspond one to another and both to the real object but without the explorer's map, respectively – the explorer's diary, it is impossible to restore the correspondence;

- At the end, the names Ateb and Aphla are connected hierarchically to the name Pirrin and the relations are:
 - Aphla *is_a_South_Side_of* Pirrin;
 - Ateb *is_a_North_Side_of* Pirrin.

The last case forms a simple vocabulary (Table 23):

Table 23. *A simple vocabulary*

<i>name</i>	<i>definition</i>
Aphla	The South Side of Pirrin mountain
Ateb	The North Side of Pirrin mountain
Pirrin	A mountain in the unexplored country with co-ordinates (x,y)

In addition, all cases given above form a simple ontology with four concepts which may be represented by a graph (Figure 35):

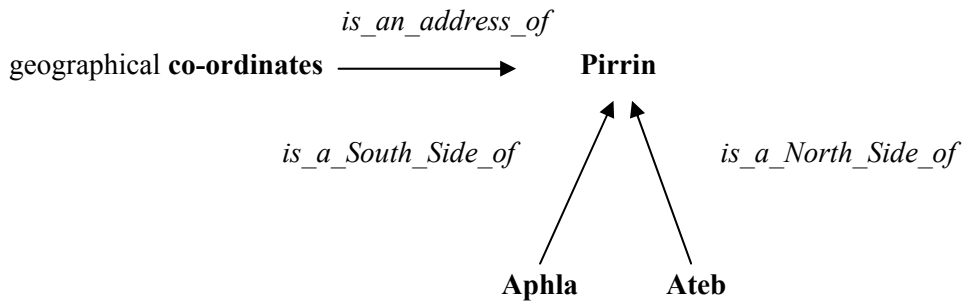


Figure 35. *A simple ontology*

The same information may be represented by a table (Table 24):

Table 24. *A simple ontology*

object	<i>is_a_South_Side_of</i>	<i>is_a_North_Side_of</i>	<i>is_an_address_of</i>
Pirrin	Aphla	Ateb	co-ordinates

What vocabularies, taxonomies, thesauruses, and ontologies, **all have in common** are [Pidcock & Uschold, 2012]:

- They are approaches to help structure, classify, model, and or represent the concepts and relationships pertaining to some subject matter of interest to some community;
- They are intended to enable a community to come to agreement and to commit to use the same terms in the same way;

- There is a set of terms that some community agrees to use to refer to these concepts and relationships;
- The meaning of the terms is specified in some way and to some degree;
- They are fuzzy, ill-defined notions used in many different ways by different individuals and communities.

The **major differences** that distinguish these approaches [Pidcock & Uschold, 2012]:

- How much meaning is specified for each term?
- What notation or language is used to specify the meaning?
- What is the thing for? Taxonomies, thesauruses, and ontologies have different but overlapping uses.

At the end, some additional information may be connected to the names. For instance, it may be the type of mountain, minerals found, some photos, textual descriptions, etc. All such information is connected to names and has to be accessed by names as keywords or paths to it, i.e. its computer representation has to be organized using corresponded pointers, indexes of keyword, etc.

In this case the concept “knowledge representation” is used. As we have seen above, the ontologies are useful approach for knowledge representation, which is understandable for humans as well as for the specialized software.

Knowledge representation is closely connected to data models, i.e. the information structures used for organizing the information in the internal or external computer memory. In other words, knowledge representation is depended on the storing patterns and program tools for accessing data.

Below in this chapter and in the next chapter, storing of knowledge, represented by structured and semi-structured data sets, will be discussed from point of view of using the NL-addressing and NL-ArM for this purpose. Results from provided experiments will be analyzed.

In this chapter we will present several experiments aimed to show the possibilities of NL-addressing to be used for NL-storing of small size datasets which contain up to one hundred thousands of instances.

The experiments were provided at PC SONY Vaio, with Intel® Core™2 Duo CPU T9550 @ 2.66GHz 2.67GHZ, RAM 4.00 GB, 64-bit operating system Windows 7 Ultimate SP1.

The goal of the experiments with different small datasets, like dictionary, thesaurus or ontology, is to discover regularities in the NL-addressing realization. More concretely, two regularities of time for storing by using NL-addressing will be examined:

- It depends on number of elements in the instances;
- It not depends on number of instances in datasets.

5.2 Experiment for NL-storing dictionaries

Our first experiment is to realize a small multi-language dictionary based on NL-addressing. For this purpose, we have taken data from the popular in Bulgaria “SA Dictionary” [Angelov, 2012]. SA Dictionary is a computer dictionary, which translates words from Bulgarian language to English and vice versa.

For experiments we take a list of **23 412** words in English and Bulgarian with their definitions in Bulgarian, stored in a sequential file with size of 2 410 KB.

The experimental program “WordArM” used for the experiments is specially designed for storing dictionaries and thesauruses based on NL-addressing. It is outlined in the Appendix A.

➤ ***Definition of dictionary***

For the purposes of this research, next definition of dictionary is appropriate:

Dictionary: *a reference resource, in printed or electronic form, that consists of an alphabetical list of words with their meanings and parts of speech, and often a guide to accepted pronunciation and syllabification, irregular inflections of words, derived words of different parts of speech, and etymologies* [Collins, 2003]:

This definition is modeled by the construction:

<name> —→ <definition>.

➤ ***Multi-language dictionary based on NL-addressing***

For storing dictionaries we use simple model: the words (concepts) are used as paths to their definitions stored in corresponded terminal containers.

The speed for storing, accessing, and size of the work memory and permanent archives are given in Table 25.

Work memory is the memory taken for storing hash tables and service information during the work of NL-ArM. Usually it has to be part of main computer memory. To analyze its real size in our experiments, work memory is allocated as file.

Permanent archives are static copies of work memory (zipped files), aimed for long storing the information. They have to be of small size and converting to and from expanded work memory structures has to be quick (usually several seconds or minutes). For compressing of work memory we use a separate archiving program.

Table 25. Experimental data for NL-storing of a dictionary

operation	number of instances	total time in milliseconds	average time for one instance	work memory	permanent archive
NL-writing	23 412	22 105	0.94 ms	80 898 KB	5 938 KB
NL-reading	23 412	20 826	0.89 ms		

The work memory taken during the work was **80 898** KB.

After finishing the work, occupied permanent compressed archive is **5 938** KB. This means that the NL-indexing takes 5 102 KB additional compressed disk memory (the sequential file with initial data is 2 410 KB and compressed by WinZip it is 836 KB).

To analyze work of the system, work memory was chosen to be in a file but not in the main memory. In further realizations of WordArM, it may be realized as a part of main memory of computer as:

- Dynamically allocated memory;
- File mapped in memory.

In this case, the speed of storing and accessing will be accelerated and used hard disk space will be reduced.

The analysis of the results in Table 25 shows that the NL-addressing in this realization permits access practically equal for writing and reading for all data.

The speed is more than a thousand instances per second.

Reading is possible immediately after writing and no search indexes are created.

5.3 Experiment for NL-storing thesauruses

The NL-storing model for vocabularies was simple because the one-one correspondence “word-definition”.

The storing models for thesauruses are more complicated due to existing more than one corresponded definitions for a given word (synonyms). Because of this, below we will outline and analyze one such model – the storing model of WordNet thesaurus [WordNet, 2012].

The idea is to use NL-addressing to realize the WordNet lexical database and this way to avoid recompilation of its database after every update.

The program used for the experiments is “WordArM” (see Appendix A).

➤ *Definition of thesaurus*

For the purposes of this research, the next definition of *thesaurus* is appropriate:

Thesaurus: a book or catalog of words and their synonyms and antonyms [YourDictionary, 2013].

A **thesaurus** is a networked collection of controlled vocabulary terms. This means that a thesaurus uses associative relationships in addition to parent-child relationships. The expressiveness of the associative relationships in a thesaurus varies and can be as simple as “correlated to terms” as in term **A** is related to term **B**. The thesaurus has two kinds of links: broader/narrower term, which is much like the generalization/specialization link, but may include a variety of others [Pidcock & Uschold, 2012].

➤ *WordNet thesaurus*



WordNet® is a large lexical database of English (<http://WordNet.princeton.edu>).

Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. Synsets are interlinked by means of conceptual-semantic and

lexical relations. The resulting network of meaningfully related words and concepts can be navigated with the browser.

WordNet is freely and publicly available for download. WordNet's structure makes it a useful tool for computational linguistics and natural language processing [Fellbaum et al, 1998; Miller, 1995].

WordNet was created and is being maintained at the Cognitive Science Laboratory of Princeton University under the direction of psychology professor George A. Miller. Development began in 1985.

As of November 2012 WordNet's latest Online-version is 3.1 (announced on June 2011), but latest released version is 3.0 (released on December 2006).

The 3.0 database contains *155 287 words* organized in *117 659 synsets* for a total of *206 941 word-sense pairs*; in compressed form, it is about 12 megabytes in size [WordNet, 2012].

➤ **WordNet system**

The mathematical model of the WordNet is a graph $V = (X, R)$, where X is the set of graph nodes, and R is the set of edges between them.

The set X is divided into two disjoint subsets: $X = X_1 \cup X_2$, $X_1 \cap X_2 = \emptyset$. The nodes from X_1 correspond to the words and phrases, and nodes from X_2 - to their meanings (interpretations). Each of meanings correlates to one of the parts of speech: noun, verb, adjective or adverb.

The set of edges is also divided into two subsets which are not intersecting: $R = R_1 \cup R_2$, $R_1 \cap R_2 = \emptyset$. Edges of R_1 connect the words with theirs meanings i.e. the elements X_1 with elements X_2 . These edges represent relationships which belong to $X_1 \times X_2$. Edges of R_2 connect "words with words" and "meanings with meanings", i.e. represent relationships that belong to $X_1 \times X_1$ and $X_2 \times X_2$ respectively [Bashmakov, 2005]. Other types of relationships are defined by typification of edges from R_2 .

Technically, the WordNet is an electronic thesaurus which defines a wide range of meanings of words bounded together by semantic pointers. WordNet logical structure is shown in Figure 36.

In developing WordNet lexical database, it has been convenient to divide the work into two interdependent tasks which bear a vague similarity to the traditional tasks of writing and printing a dictionary [Fellbaum, 1998]:

- One task is to write the source files that contain the basic lexical data — the contents of those files are the lexical substance of WordNet;
- The second task is to create a set of computer programs that would accept the source files and do all the work leading ultimately to the generation of a display for the user.

The WordNet system falls naturally into four parts:

- The WordNet lexicographers' source files;
- The software utility called "Grinder" aimed to *convert* lexicographers' source files into the WordNet lexical database;
- The WordNet lexical database;
- And the suite of software tools used to access the database.

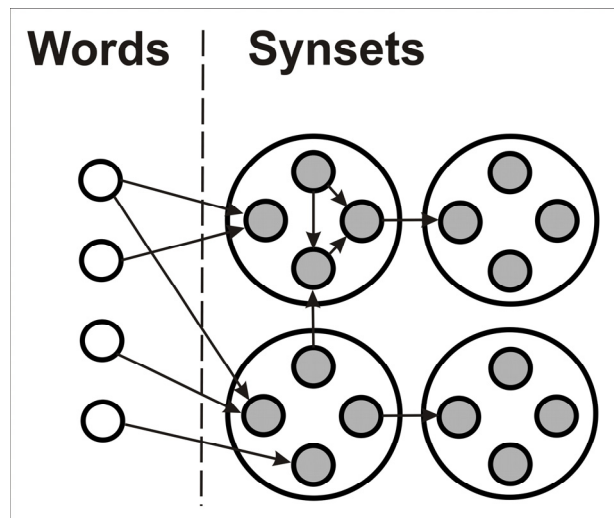


Figure 36. Logical structure of the WordNet

WordNet's source files are written by lexicographers. They are the product of a detailed relational analysis of lexical semantics: a variety of lexical and semantic relations are used to represent the organization of lexical knowledge.

The “Grinder” utility compiles the lexicographers' files. It verifies the syntax of the files, resolves the relational pointers, then generates the WordNet database that is used with the retrieval software and other research tools. *To build a complete WordNet database, all of the lexicographers' files must be processed at the same time.*

The main relation among words in WordNet is synonymy, as between the words “shut” and “close” or “car” and “automobile”. Synonyms (words that denote the same concept and are interchangeable in many contexts) are grouped into unordered sets (synsets). Each of WordNet's 117 000 synsets is linked to other synsets by means of a small number of “conceptual relations”. Additionally, a synset contains a brief definition (“gloss”) and, in most cases, one or more short sentences illustrating the use of the synset members. Word forms with several distinct meanings are represented in as many distinct synsets. Thus, each form-meaning pair in WordNet is unique [WordNet, 2012].

Consider a representation of a synset of the word “accession” in the WordNet lexical database:

```
00047131 04 n 02 accession 0 addition 0 001 @ 09536731 n 0000 |
  something added to what you have already;
  "the librarian shelved the new accessions";
  "he was a new addition to the staff"
```

The number 00047131 is a unique identifier of the synset of the noun {accession, addition}. The part of the record between the symbols "@" and "|" indicates that this synset is subordinated to the synset with ID 09536731 which correspond to meaning "acquisition". The last part of the record (after

the symbol "|") is interpretation of synset and some examples of using the words included in the synset.

From a software standpoint, this record requires a number of additional indexes for service the access, which of course needs additional resources.

As an example, consider the information about the word "accession". As an answer to the request for "accession", the WordNet system returns the following information (Figure 37):

The noun accession has 6 senses (no senses from tagged texts)

1. **{13251723}** <noun.process> accession#1 -- (a process of increasing by addition (as to a collection or group); "the art collection grew through accession")
2. **{13170404}** <noun.possession> accession1#2 -- ((civil law) the right to all of that which your property produces whether by growth or improvement)
3. **{13082910}** <noun.possession> accession#3, addition#4 -- (something added to what you already have; "the librarian shelved the new accessions"; "he was a new addition to the staff")
4. **{07078650}** <noun.communication> accession2#4, assenting#1 -- (agreeing with or consenting to (often unwillingly); "accession to such demands would set a dangerous precedent"; "assenting to the Congressional determination")
5. **{05115154}** <noun.attribute> entree#2, access#1, accession#5, admittance#1 -- (the right to enter)
6. **{00232781}** <noun.act> accession3#6, rise to power#1 -- (the act of attaining or gaining access to a new office or right or position (especially the throne); "Elizabeth's accession in 1558")

The verb accession has 1 sense (no senses from tagged texts)

1. **{00989696}** <verb.communication> accession#1 -- (make a record of additions to a collection, such as a library)

Figure 37. Answer by WordNet system to a query for the word "accession"

➤ *WordNet storing model*

The WordNet database is in an ASCII format that is human- and machine-readable, and is easily accessible to those who wish to use it with their own applications.

There are two main types of database files:

- Data file - contains all of the lexicographic data gathered from the lexicographers' files for the corresponding syntactic category, with relational pointers resolved to addresses in data files;
- Index file - an alphabetized list of all of the word forms in WordNet for the corresponding syntactic category.

WordNet stores information about words in four main data files and four main index files (for nouns, verbs, adjectives and adverbs).

The index and data files are interrelated.

The data structure is the same in each of data files – one or more synsets are stored for every word and the access is performed by the address of the first bytes of the synsets, which is apparently given by an eight digit number beginning namely in this byte (Figure 38 and Figure 39). This value is the unique identifier of the synset. It is its “*relative address*” from the beginning (first byte) of the file. The synset data elements are separated by spaces. We should note that links to other synsets are given again by the relative addresses.

13251723 22 n 01 **accession** 0 001 @ **13323403** n 0000 | a process of increasing by addition (as to a collection or group); "the art collection grew through accession"

13170404 21 n 01 **accession** 1 002 @ **13070995** n 0000 ;c **08338303** n 0000 | (civil law) the right to all of that which your property produces whether by growth or improvement

13082910 21 n 02 **accession** 0 addition 0 001 @ **13082742** n 0000 | something added to what you already have; "the librarian shelved the new accessions"; "he was a new addition to the staff"

07078650 10 n 02 **accession** 2 assenting 0 002 @ **07076600** n 0000 + **00795631** v 0102 | agreeing with or consenting to (often unwillingly); "accession to such demands would set a dangerous precedent"; "assenting to the Congressional determination"

05115154 07 n 04 entree 0 access 0 **accession** 0 admittance 0 003 @ **05113619** n 0000 + **02426186** v 0401 ~ **05119817** n 0000 | the right to enter

00232781 04 n 02 **accession** 3 rise_to_power 0 003 @ **00060914** n 0000 + **01989112** v 0101 + **02358456** v 0101 | the act of attaining or gaining access to a new office or right or position (especially the throne); "Elizabeth's accession in 1558"

Figure 38. Synsets of the word “accession” in WordNet data file for nouns

00989696 32 v 01 **accession** 0 002 @ **00990286** v 0000; c **00897092** n 0000 01 + 08 00 | make a record of additions to a collection, such as a library

Figure 39. Synsets of the word “accession” in WordNet data file for verbs

What is important for us now is the *algorithm of reaching the synsets*.

There are four index files of WordNet (for nouns, verbs, adjectives and adverbs). They are sorted in alphabetical order of words and for each word a special record is stored at separated line. Its structure is clear: at the first place the word is given and, after some coded information, the relative addresses of the corresponded synsets in corresponded data files are given (Figure 40 and Figure 41).

accession n 6 4 @ ~ + ; 6 0 13251723 13170404 13082910 07078650 05115154 00232781

Figure 40. Record for the word "accession" in the index of nouns

accession v 1 2 @ ; 1 0 00989696
--

Figure 41. Record for the word "accession" in the index of verbs

To reach all synsets of a word, firstly a binary search is made in all index files, the corresponded relative addresses are collected and then system reads the synsets directly from the data files.

Algorithmic complexity in this case is $O(\log(n_n)+\log(n_v)+\log(n_a)+\log(n_r))$, where n_n , n_v , n_a and n_r are the quantities of nouns, verbs, adjectives and adverbs, respectively.

There is a second way to reach synsets. It is served by so called "*sense index*". This index is also sorted, but for every word there exist as much records as number of synsets exists for given word in all data files. For example, the word *accession* has seven records: six for its meanings as a noun and one for its meaning as a verb. Each record contains only one relative address of a synset (Figure 42).

In this case, to reach all synsets of a word, firstly a binary search is made in the sense index and the corresponded relative addresses are collected from all records for the word. Then, the system reads the synsets directly from the data files.

Algorithmic complexity in this case is greater than $O(\log(n))$, $n = n_n + n_v + n_a + n_r$ is the total number of words in the database (nouns + verbs + adjectives + adverbs), because the words may be repeated many times, and further work is needed to retrieve all occurrences of the word.

accession%1:04:03:: 00232781 6 0 accession%1:07:00:: 05115154 5 0 accession%1:10:02:: 07078650 4 0 accession%1:21:00:: 13082910 3 0 accession%1:21:01:: 13170404 2 0 accession%1:22:00:: 13251723 1 0 accession%2:32:00:: 00989696 1 0

Figure 42. Records for the word "accession" in the sense index

➤ *Disadvantages of WordNet storing model*

The WordNet storing model permits quick response of the system during its everyday using. The (binary) search in four types sorted index files and one general sense index, using corresponded hash tables, allows high speed of the search and, based on it, extracting the needed information via direct access based on the relative addresses in the data files.

Many disadvantages of the WordNet organization are discussed in [Poprat et al, 2008]. Due to importance of them, below we will include larger citations.

When the WordNet project started more than two decades ago, markup languages such as SGML or XML were unknown. Because of this reason, a rather idiosyncratic, fully text-based data structure for these lexicographic files was defined in a way to be readable and editable by humans — and survived until today. This can really be considered as *an outdated legacy* given the fact that the WordNet community has been so active in the last years in terms of data collection, but has refrained from adapting its data formats in a comparable way to today's specification standards.

There are two types of problems founded for the data format underlying the WordNet lexicon and the software that helps building a WordNet file and creating an index for this file:

- First, WordNet's data structure puts several restrictions on what can be expressed in a WordNet lexicon. For example, it constrains lexical information to a fixed number of homonyms and a fixed set of relations;
- Second, the data structure imposes a number of restrictions on the string format level.

If these restrictions are violated the WordNet processing software throws error messages which differ considerably in terms of informativeness for error tracing and detection or even do not surface at all at the lexicon builder's administration level.

In addition, it seems that the length of a word is restricted to 425 characters and synsets are only allowed to group up to 988 direct hyponymous synsets.

According to our experiences the existing WordNet software *is hardly (re)usable* due to insufficient error messages that the software throws and limited documentation [Poprat et al, 2008].

In terms of the actual representation format of WordNet we found that using the current format is not only cumbersome and error-prone, but also limits what can be expressed in a WordNet resource [Poprat et al, 2008].

From our perspective this indicates the need for a *major redesign* of WordNet's data structure foundations to keep up with the standards of today's meta data specification languages (e.g., based on RDF [Graves & Gutierrez, 2006], XML or OWL [Lungen et al, 2007]). We encourage the reimplementations of WordNet resources based on such a state-of-the-art markup language (for OWL in particular a representation of WordNet is already available [van Assem et al, 2006]).

Of course, if a new representation format is used for a WordNet resource also the software accessing the resource has to be adapted to the new format. This may require substantial implementation efforts that we think are worth to be spent, if the new format overcomes the major problems that are due to the original WordNet format [Poprat et al, 2008].

Finally, one more shortcoming of the WordNet database's structure is that although all the files are in ASCII, and are therefore editable, and in theory extensible, in practice *this is almost impossible*.

The end user has access only to the static ("compiled") version of the database, which couldn't be extended and further developed. In addition, due to relative addresses that are used as pointers, any change which cause alteration of the number of bytes in any data file makes it unusable and it must be recompiled as well as the corresponded index files.

One of the Grinder's primary functions is the calculation of addresses for the synsets in the data files. Editing any of the database files would (most likely) create incorrect byte offsets, and would thus derail many searching strategies. At the present time, building a WordNet database requires the

use of the Grinder and the *processing of all lexicographers' source files at the same time* [Fellbaum, 1998].

Let see a small example shown on Figure 43 - two variants of the synset of the word "accession" from different compilations of the data file for nouns:

(a) Version of WordNet from August, 2012;

(b) An older version of WordNet from 2011 year, published in [Palagin et al, 2011].

The difference between relative addresses is visible on Figure 43.

<p>13082910 21 n 02 accession 0 addition 0 001 @ 13082742 n 0000 something added to what you already have; "the librarian shelved the new accessions"; "he was a new addition to the staff" a) version from the 2012 year</p> <p>00047131 04 n 02 accession 0 addition 0 001 @ 09536731 n 0000 something added to what you have already; "the librarian shelved the new accessions"; "he was a new addition to the staff" b) version from the 2011year</p>
--

Figure 43. Synset the word "accession" from the data file for nouns

In the main, the WordNet database organization has the following important disadvantages:

1. Relative addressing is convenient for the computer processing, but it is difficult to be used by the customer;
2. Manual creating of numerical addresses is impossible, and their use can be done only by the special program;
3. The end user has access only to the static ("compiled") version of the database, which couldn't be extended and further developed;
4. Building the WordNet database requires the use of the Grinder and the processing of all lexicographers' source files at the same time;
5. Using the current format is not only cumbersome and error-prone, but also limits what can be expressed in a WordNet resource.

We are going to experiment to realize WordNet lexical database without using relative addresses as pointers and this way to avoid the pointed above limitations and recompilation of the database after every update.

➤ *Experiment to store WordNet by NL-addressing*

The main source information of WordNet is published as lexicographer files.

The names of the WordNet lexicographer files and their corresponding file numbers are listed in Table 26, along with a brief description of each file's content and corresponded number of included instances (synsets).

The total number of instances (file records) is 117 871.

206 instances contain service information but not concepts' definitions, so we have 117 665 instances for experiments, distributed in 45 thematically organized lexicographer files.

It is important to note that there is equal synsets in several lexicographer files. This has matter when we integrate the 45 files in one source file for representing a thesaurus.

Table 26. *WordNet lexicographer files*

No.:	Name	Content	number of instances
01	adj.all	all adjective clusters	14435
02	adj.pert	relational adjectives (pertainyms)	3661
03	adj.ppl	participial adjectives	60
04	adv.all	all adverbs	3621
05	noun.Tops	unique beginner for nouns	51
06	noun.act	nouns denoting acts or actions	6650
07	noun.animal	nouns denoting animals	7514
08	noun.artifact	nouns denoting man-made objects	11587
09	noun.attribute	nouns denoting attributes of people and objects	3039
10	noun.body	nouns denoting body parts	2016
11	noun.cognition	nouns denoting cognitive processes and contents	2964
12	noun.communication	nouns denoting communicative processes and contents	5607
13	noun.event	nouns denoting natural events	1074
14	noun.feeling	nouns denoting feelings and emotions	428
15	noun.food	nouns denoting foods and drinks	2574
16	noun.group	nouns denoting groupings of people or objects	2624
17	noun.location	nouns denoting spatial position	3209
18	noun.motive	nouns denoting goals	42
19	noun.object	nouns denoting natural objects (not man-made)	1545
20	noun.person	nouns denoting people	11088
21	noun.phenomenon	nouns denoting natural phenomena	641

No.:	Name	Content	number of instances
22	noun.plant	nouns denoting plants	8159
23	noun.possession	nouns denoting possession and transfer of possession	1061
24	noun.process	nouns denoting natural processes	770
25	noun.quantity	nouns denoting quantities and units of measure	1350
26	noun.relation	nouns denoting relations between people or things or ideas	437
27	noun.shape	nouns denoting two and three dimensional shapes	342
28	noun.state	nouns denoting stable states of affairs	3544
29	noun.substance	nouns denoting substances	2983
30	noun.time	nouns denoting time and temporal relations	1028
31	verb.body	verbs of grooming, dressing and bodily care	547
32	verb.change	verbs of size, temperature change, intensifying, etc.	2383
33	verb.cognition	verbs of thinking, judging, analyzing, doubting	695
34	verb.communication	verbs of telling, asking, ordering, singing	1548
35	verb.competition	verbs of fighting, athletic activities	459
36	verb.consumption	verbs of eating and drinking	243
37	verb.contact	verbs of touching, hitting, tying, digging	2196
38	verb.creation	verbs of sewing, baking, painting, performing	694
39	verb.emotion	verbs of feeling	343
40	verb.motion	verbs of walking, flying, swimming	1408
41	verb.perception	verbs of seeing, hearing, feeling	461
42	verb.possession	verbs of buying, selling, owning	847
43	verb.social	verbs of political and social activities and events	1106
44	verb.stative	verbs of being, having, spatial relations	756
45	verb.weather	verbs of raining, snowing, thawing, thundering	81
		TOTAL:	117871

Let see an example of two variants of the synset of the word "accession" (Figure 44): (a) WordNet version and (b) NL-version.

```

13082910 21 n 02 accession 0 addition 0 001 @ 13082742 n 0000 |
something added to what you already have; "the librarian shelved the new
accessions"; "he was a new addition to the staff"
a) WordNet version

accession 21 n 02 ; 0 addition 0 001 @ acquisition n 0000 | something added
to what you already have; "the librarian shelved the new accessions"; "he
was a new addition to the staff"
b) NL-version

```

Figure 44. WordNet and NL-versions of the synset of the word "accession"

This example gives idea to experiment using NL-addressing to realize the WordNet lexical database without using relative addresses and this way to avoid the limitations and recompilation of the database after every update.

For experiments we have used files from Table 26 as source in two variants:

1. All 45 files concatenated in one big file as thesaurus with more than one hundred thousands of concepts.
2. Every file was assumed as different layer of WordNet ontology.

The first case will be discussed below, and the second case will be outlined in the next section. The program used for experiments in the first case is "WordArM" (see Appendix A). A screenshot from the WordArM for the case of WordNet as thesaurus is shown at Figure 45. The results are given in Table 27.

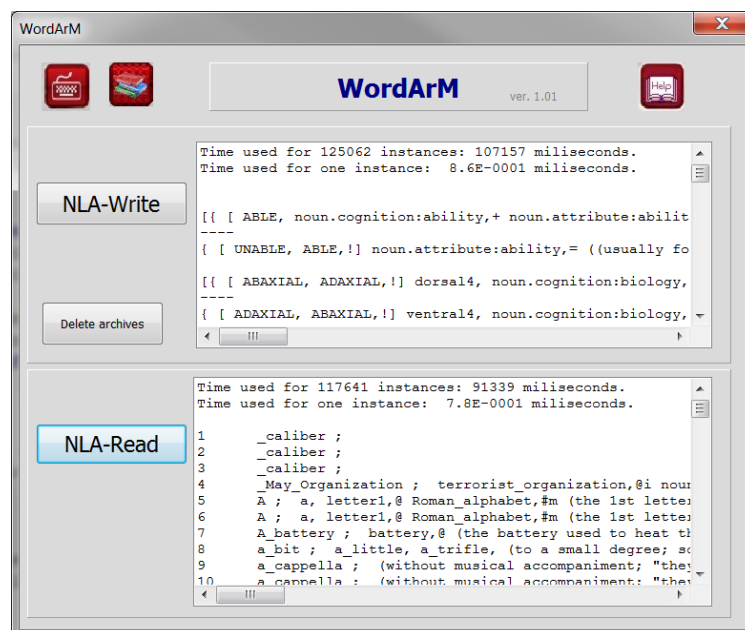


Figure 45. WordArM results for the case of WordNet as thesaurus

Table 27. *Experimental data for storing WordNet as thesaurus*

operation	number of instances	total time in milliseconds	average time for one instance
writing	125 062	107 157	0.86 ms
reading	117 641	91 339	0.78 ms
work memory: 385 538 KB; permanent archive: 15 603 KB; source text: 1 333 KB			

We receive practically the same results as for storing dictionaries.

The analysis of the results in Table 27 shows that the NL-addressing permits access practically equal for writing and reading for all data. The speed is more than a thousand instances per second. Reading is possible immediately after writing and no search indexes are created.

The work memory for hash tables and their containers taken during the work of WordArM was 385 538 KB. To analyze work of the system, work memory was chosen to be in a file in the external memory. In further realizations, to accelerate the speed and reduce of used disk space, the work memory may be realized as part of main memory (as dynamically allocated memory or as file mapped in memory).

After finishing the work, occupied permanent archive for compressed archive is 15 603 KB, i.e. in this case the NL-indexing takes 14 270 KB additional compressed memory (the sequential file with initial data is 1 333 KB).

➤ ***What we gain and loss using NL-Addressing for storing thesauruses?***

The loss is additional memory for storing structures which serve NL-addressing. But the same if no great losses we will have if we will build balanced search trees or other kind in external indexing. It is difficult to compare with other systems because such information practically is not published.

The benefit is in two main achievements:

1. High speed for storing and accessing the information.
2. The possibility to access the information immediately after storing without recompilation the database and rebuilding the indexes.

5.4 Experiment for NL-storing ontologies

Storing graphs and ontologies has one important aspect – the layers which correspond to types of relations between nodes of graph or ontology [Ivanova et al, 2013e]. The example with sample graph in previous chapter indicates that it is important to ensure possibility for multi-layer representation. To make experiment with real data, we will use the WordNet as ontology and its 45 types of relations (given by its files of different types) we store as 45 layers. To provide experiments in this case, we have realized program “OntoArM”. It is outlined in the Appendix A.

➤ **Definition of ontology**

For the purposes of this research, the next definition of **ontology** is appropriate:

Ontology: *a rigorous and exhaustive organization of some knowledge domain that is usually hierarchical and contains all the relevant entities and their relations* [WordNet, 2012].

People use the word **ontology** to mean different things, e.g. glossaries & data dictionaries, thesauruses & taxonomies, schemas & data models, and formal ontologies & inference. A **formal ontology** is a controlled vocabulary expressed in an ontology representation language. This language has a grammar for using vocabulary terms to express something meaningful within a specified domain of interest. The grammar contains formal constraints (e.g. specifies what it means to be a well-formed statement, assertion, query, etc.) on how a term in the ontology's controlled vocabulary can be used together.

People make commitments to use a specific controlled vocabulary or ontology for a domain of interest. Enforcement of ontology's grammar may be rigorous or lax. Frequently, the grammar for "light-weight" ontology is not completely specified, i.e. it has implicit rules that are not explicitly documented [Pidcock & Uschold, 2012].

The word "**ontology**" has been used to refer to all of the above things. When used in the AI/Knowledge_Representation community, it tends to refer to things that have a rich and formal logic-based language for specifying meaning of the terms. Both a thesaurus and taxonomy can be seen as having a simple language that could be given a grammar, although this is not normally done. Usually they are not formal, in the sense that there is no formal semantics given for the language. However, one can create a model in UML and a model in some formal ontology language and they can have identical meaning. It is thus not useful to say one is ontology and the other is not because one lacks a formal semantics. The truth is: there is a fuzzy line connecting these things [Pidcock & Uschold, 2012].

In 1992, Tom Gruber offers a formal description of concepts and relationships between them, called "*ontology*", which is a basis for communication between agents.

Ontology is an explicit specification of a conceptualization. The term is borrowed from philosophy, where *Ontology* is a systematic account of *Existence* [Gruber, 1993a].

People, organizations and software systems must communicate between and among themselves. However, due to different needs and background contexts, there can be widely varying viewpoints and assumptions regarding what is essentially the same subject matter. Each uses different jargon; each may have differing, overlapping and/or mismatched concepts, structures and methods

[Uschold & Gruninger, 1996] stressed that "ontology is a *unifying framework* for the different viewpoints and serves as the basis for:

- *Communication* between people with different needs and viewpoints arising from their differing contexts;
- *Inter-Operability* among systems achieved by translating between different modeling methods, paradigms, languages and software tools;

- *System Engineering Benefits*: In particular:
 - *Re-Usability*: the shared understanding is the basis for a formal encoding of the important entities, attributes, processes and their inter-relationships in the domain of interest. This formal representation may be (or become so by automatic translation) a re-usable and/or shared component in a software system;
 - *Reliability*: A formal representation also makes possible the automation of consistency checking resulting in more reliable software;
 - *Specification*: the shared understanding can assist the process of identifying requirements and defining a specification for an IT system. This is especially true when the requirements involve different groups using different terminology in the same domain, or multiple domains”.

John Sowa notes that *"the art of ranking things in general and species is of no small importance and very much assists our judgment as well as our memory. (...) This helps one not merely to retain things, but also to find them. And those who have laid out all sorts of notions under certain headings or categories have done something very useful"* [Sowa, 2000].

The word “ontology” comes from the Greek “ontos” for being and “logos” for word. It is a relatively new term in the long history of philosophy, introduced by the 19th century German philosophers to distinguish the study of being as such from the study of various kinds of beings in the natural sciences. The more traditional term is Aristotle’s word “category” (kathgoria), which he used for classifying anything that can be said or predicated about anything [Sowa, 2000a].

In the literature on artificial intelligence, the "ontology" is a term used to describe formally represented knowledge based on a conceptualization. It requires a description of a set of objects by corresponded concepts and relationships between these concepts (knowledge).

The word ontology can be used and has been used with very different meanings attached to it. Ironically, the ontology field suffered a lot from ambiguity. The Knowledge Engineering Community borrowed the term “Ontology” from the name of a branch of philosophy some 15 years ago and converted into an object: “ontology”. In the mid-90s philosophers “took it back” and began to clean the definitions that had been adopted [Gandon, 2002].

Formally, the ontology consists of:

- Terms organized in taxonomy;
- Definitions of terms and attributes;
- Axioms and rules for inference.

The ontology formally can be described by the ordered triple [Palagin & Yakovlev, 2005; Gavrilova, 2001; Palagin, 2006; Guarino, 1998]:

$$O = \langle X, R, F \rangle,$$

where X, R, F are a finite sets accordingly:

- X is a set of concepts (terms) from the subject area;
- R is a set of relationships between the elements of X;
- F is a set of functions to interpret the X and/or R.

Classifications of ontologies from different points of view and based on different principles are given in many publications [Bashmakov, 2005; Dobrov et al, 2009].



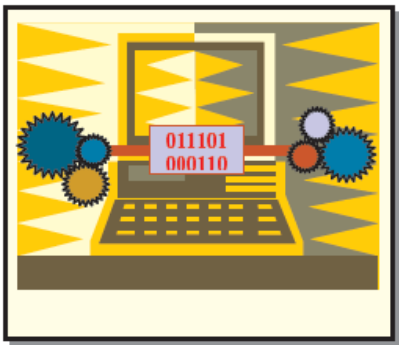
A formal classification scheme is given in [Guarino, 1998]. From its viewpoint: "*ontology is a logical theory accounting for the intended meaning of a formal vocabulary, i.e. its ontological commitment to a particular conceptualization of the world. The intended models of a logical language using such a vocabulary are constrained by its ontological commitment. Ontology indirectly reflects this commitment (and the underlying conceptualization) by approximating these intended models*" [Guarino, 1998].

To illustrate the **usefulness of ontologies** let consider an example given by Prof. Assunción Gómez-Pérez from the Universidad Politecnica de Madrid, during a lecture on ECAI 98, presented in [Gandon, 2002].

The general problem is to formulate a query over a mass of information and get an answer as precise and relevant as possible. In her tutorial at ECAI 98, Prof. Pérez asked: "What is a pipe?"

Extending her example we can imagine three answers to this very same question (Table 28).

Table 28. *Three notions behind the word "pipe"*

		
<p>A short narrow tube with a small container at one end, used for smoking e.g. tobacco.</p>	<p>A long tube made of metal or plastic that is used to carry water or oil or gas.</p>	<p>A temporary section of computer memory that can link two different computer processes.</p>

We have one term and three concepts; *it is a case of ambiguity*. The contrary is one concept behind several terms, and *it is a case of synonyms* e.g. car, auto, automobile, motorcar, etc. These trivial cases poses a serious problem to computerize systems that are not able the see these difference or equivalence unless they have been made explicit to it.

Communication between people is based on an implicit consensus about concepts that are used. For example, when discussing a document, people involved in the discussion implicitly implied that they have a common conceptual consensus about the nature of the document. On a question about "text", the answer "knows" that "text" means the "document".

This basic knowledge is lacking in information systems based only on usual terms and text search (by coincidence).

One possible approach is knowledge to be clearly formulated and presented in a logical structure that can be used by automated systems. This is exactly the purpose of ontology: to capture semantics and relationships of concepts we use, making it clear (explicit) and possibly encoded in

symbolic systems so as to retrieve and exchange between different agents, which in particular can be computer systems.

In conclusion, we recall some of the generally accepted terms and their definitions that are used with ontologies [Gruber, 1993; Guarino & Giaretta, 1995; Bachimont, 2000; Gandon, 2002] (Table 29).

Table 29. Some commonly accepted concepts and definitions

Concepts	Definitions
Notion	- Something formed in the mind, a constituent of thought; - It is used to structure knowledge and perceptions of the world; - Principle, idea semantically evaluable and redeploy able.
A Concept	- Notion usually expressed by a term (or more generally by a sign); - The concept represents a group of objects or beings having shared characteristics “t” that enable us to recognize them a forming and belonging to this group.
A Relation	- Notion of an association or a link between concepts usually expressed by a term or a graphical convention (or more generally by a sign).
Ontology	- “That branch of philosophy which deals with the nature and the organization of reality”; - A branch of metaphysics which investigates the nature and essential properties and relations of all beings as such.
Formal Ontology	- The systematic, formal, axiomatic development of the logic of all forms and modes of being.
Conceptualization	- An intentional semantic structure which encodes the implicit rules constraining the structure of a piece of reality; - It also denotes the action of building such a structure.
An Ontology	- A logical theory which gives an explicit, partial account of a conceptualization; - The aim of ontologies is to define which primitives, provided with their associated semantics, are necessary for knowledge representation in a given context.
Ontological theory	- A set of formulas intended to be always true according to a certain conceptualization.
A Taxonomy	- A classification based on similarities.
A Partonomy	- A classification based on “part-of” relation.

➤ ***Representing and operations with ontologies***

Traditionally, ontologies are built by highly trained knowledge engineers with the assistance of domain specialists. It’s time-consuming and laborious task. Ontology tools also require users to be trained knowledge representation and predicate logic.

There are several approaches for representing ontologies. An example of such approach is using of XML. It is a popular markup language of metadata. With the development of the XML, different definitions of metadata have been proposed such as Dublin Core [Weibel et al, 1998] and ebXML [ebxml, 2012].

However, from the viewpoint of ontology, XML is not suited to describe the interrelationships of resources [Gunther, 1998]. Therefore, W3C has suggested the “Resource Description Framework” (RDF). There are several ontology languages like XML, RDF schema RDF(S), DAML+OIL and OWL. Many ontology tools have been developed for implementing metadata of ontology using these languages [Hertel et al, 2009].

Operations with ontologies are functions of the so called “middleware”. What is called middleware is the layer implementing the access to the physical ontology data store.

Besides an inference mechanism, the access layer should provide functions for creating, querying and deleting data in the store.

While adding data requires parsing and ideally a validation of the incoming ontology sentences, querying the ontology store needs the implementation of some kind of query language as well as an interpretation and a translation of this query language into calls to the physical storage.

Another important feature of this layer is the possibility to export ontology data to a file for exchange with other systems [Hertel et al, 2009].

The *operations with several ontologies* are needed when one application uses multiple ontologies, especially when using modular design of ontologies or when we need to integrate with systems that use other ontologies.

We will summarize some of these operations. The terminology in this area is still not stable and different authors may use these terms in a bit shifted meaning, and so the terms may overlap, however, all of these operations are important for maintenance and integration of ontologies [Obitko, 2007].

- *Merging* of ontologies means creation of a new ontology by linking up the existing ones. Conventional requirement is that the new ontology contains all the knowledge from the original ontologies, however, this requirement does not have to be fully satisfied, since the original ontologies may not be together totally consistent. In that case the new ontology imports selected knowledge from the original ontologies so that the result is consistent. The merged ontology may introduce new concepts and relations that serve as a bridge between terms from the original ontologies;
- *Mapping* from ontology to another one is expressing of the way how to translate statements from ontology to the other one. Often it means translation between concepts and relations. In the simplest case it is mapping from one concept of the first ontology to one concept of the second ontology. It is not always possible to do such “one to one” mapping. Some information can be lost in the mapping. This is permissible; however mapping may not introduce any inconsistencies;
- *Alignment* is a process of mapping between ontologies in both directions whereas it is possible to modify original ontologies so that suitable translation exists (i.e. without losing information during mapping). Thus it is possible to add new concepts and

relations to ontologies that would form suitable equivalents for mapping. The specification of alignment is called articulation. Alignment, as well as mapping, may be partial only;

- *Refinement* is mapping from ontology A to another ontology B so that every concept of ontology A has equivalent in ontology B, however primitive concepts from ontology A may correspond to non-primitive (defined) concepts of ontology B. Refinement defines partial ordering of ontologies;
- *Unification* is aligning all of the concepts and relations in ontologies so that inference in ontology can be mapped to inference in other ontology and vice versa. Unification is usually made as refinement of ontologies in both directions;
- *Integration* is a process of looking for the same parts of two different ontologies A and B while developing new ontology C that allows to translate between ontologies A and B and so allows interoperability between two systems where one uses ontology A and the other uses ontology B. The new ontology C can replace ontologies A and B or can be used as an inter-lingua for translation between these two ontologies. Depending on the differences between A and B, new ontology C may not be needed and only translation between A and B is the result of integration. In other words, depending on the number of changes between ontologies A and B during development of ontology C the level of integration can range from alignment to unification;
- *Inheritance* means that ontology A inherits everything from ontology B. It inherits all concepts, relations and restrictions or axioms and there is no inconsistency introduced by additional knowledge contained in ontology A. This term is important for modular design of ontologies where an upper ontology describes general knowledge and lower application ontology adds knowledge needed only for the particular application. Inheritance defines partial ordering between ontologies.

Not all of these operations can be made for all ontologies [Obitko, 2007]. In general, these are very difficult tasks that are in general not solvable automatically, for example, because of:

- Undecidability when using very expressive logical languages;
- Insufficient specification of an ontology that is not enough to find similarities with another one.

Because of these reasons these tasks are usually made manually or semi-automatically, where a machine helps to find possible relations between elements from different ontologies, but the final confirmation of the relation is left on human. Human then decides based on:

- The natural language description of the ontology elements;
- The natural language names of the ontology elements and common sense.

➤ *Tools for developing ontologies*

The tools for developing ontologies allow users to define new concepts, relationships and instances, i.e. to create and/or expand existing ontologies. The ontology tools may contain some additional features such as graphical representation, information search and additional tuning [Noy & Musen, 2002]. Such tools are, for instance, SWOOP [Kalyanpur et al, 2005], Top Braid composer

[TBC, 2012], Internet Business Logic [IBL, 2012], OntoTrack [Liebig & Noppens, 2003] and IHMC Cmap Ontology Editor [Hayes et al, 2005].

“Chimaera” helps with merging ontologies. It provides suggestions for subsumption, disjointness or instance relationship. These suggestions are generated heuristically and are provided for an operator, so that he may choose which one will be actually used [Chimaera, 2012]. “PROMT” (or “SMART”) system is a similar system that provides suggestions based on linguistic similarity, ontology structure and user actions. It points the user to possible effects of these changes [Prompt, 2012].

In [OntoTools, 2012] more than 150 tools (ontology editors) are outlined. At the first glance, these tools may be classified on two groups – non commercial and commercial.

For instance, the first group include tools like Protégé [protégé, 2012], OilEd [Bechhofer et al, 2001], Apollo [Apollo, 2012], RDFedit [rdfedit, 2012] OntoLingua [Farquhar et al, 1996], OntoEdit [ontoprise, 2012; Sure et al, 2002; Sure et al, 2003], WebODE [Arpírez et al, 2001], KAON [Kaon, 2012], ICOM [ICOM, 2012], DOE [Bachimont, 2000; Bachimon et al, 2002; Troncy & Isaac, 2002; DOE, 2012], WebOnto [Webonto, 2012], and OntoIntegrator [Nevzorova et al, 2007; Nevzorova & Nevzorov, 2009; Nevzorova & Nevzorov, 2011].

Example of the commercial tools are Medius Visual Ontology Modeller [Polikoff, 2003; sandsoft, 2012], LinKFactory Workbench [Deray & Verheyden, 2003] and K-Infinity [Macris, 2004; OntoLex, 2012].

Many of the tools are closed systems. Therefore, it is not possible to evaluate the full functional capabilities. Thus, the choice of editor of ontologies for practical purposes depends of:

- Free distribution;
- Local use of the web interface;
- Extensibility of functional possibilities of the applications;
- Ability to include modules designed by the user.

The basic features, capabilities, advantages, disadvantages, and comparative analysis of available onto-editors are given in a number of meaningful overviews [Ovdei & Proskudina, 2004; Calvanese et al, 2007; Filatov et al, 2007]. Analysis of literary sources about ontoeditors shows that ontoeditor Protégé is closest to the listed requirements.

The instrumental systems for ontological engineering can be divided into three main groups [Ovdei & Proskudina, 2004]:

The first group includes tools for creating ontologies that provide:

- Maintenance of collaborative development and review;
- Creation of ontologies according to any methodology;
- Maintenance of reasoning.

The second group includes tools for [Noy & Musen, 1999]:

- Unification of ontologies;
- Discovering semantic relations between different ontologies;
- Alignment the ontologies by establishing links between them and allowing the aligned ontologies to reuse information from one another.

The third group includes tools for annotation of Web-based ontology resources.

Adding some new systems to survey of [Youn & McLeod, 2006], in Table 30 and Table 31 the basic and advanced features of several well-known ontological systems are outlined.

Table 30. Basic functions of the well-known ontological systems [Youn & McLeod, 2006]

	Import format	Export format	Graph view	Consistency check	Multi-user	Web support	Merging
Protégé [protégé, 2012]	XML, RDF(S), XML Schema	XML, RDF(S), XML Schema, FLogic, CLIPS, Java html	Via plug- ins like GraphViz and Jambalaya	Via pluggins like PAL and FaCT	Limited (Multi-user capability added to it in 2.0 version)	Via Protégé- OWL plug- in	Via Anchor- PROMPT plug-in
OilEd [Bechhofer et al, 2001]	RDF(S), OIL, DAML+O IL	RDF(S), OIL, DAML+OIL, SHIQ, doty, html	No	Via FaCT	No	Very limited namespaces	No
Apollo [Apollo, 2012]	OCML, CLOS	OCML, CLOS	No	Yes	No	No	No
RDFedt [rdfedit, 2012]	RDF(S), OIL, DAML, SHOE	RDF(S), OIL, DAML, SHOE	No	Only checks writing mistakes	No	Via RSS (RDF Site Summary)	?
OntoLingua [Farquhar et al, 1996]	IDL, KIF	KIF, CLIPS, IDL, OKBC syntax, Prolog syntax	No	Via Chimaera	Via write-only locking, user access levels	Yes	?
OntoEdit (Free version) [ontoprise, 2012; Sure et al, 2002; Sure et al, 2003]	XML, RDF(S), FLogic and DAML+O IL	XML, RDF(S), FLogic and DAML+OIL	Yes	Yes	No	Yes	?
WebODE [Arpírez et al, 2001]	RDF(S), UML, DAML+O IL and OWL	RDF(S), UML, DAML+OIL, OWL, PROLOG, X- CARIN, Java/Jess	Form based graphical user interface	Yes	By synchronization, authentication and access restriction	Yes	Via ODEmerge

	Import format	Export format	Graph view	Consistency check	Multi-user	Web support	Merging
KAON [Kaon, 2012]	RDF(S)	RDF(S)	No	Yes	By concurrent access control	Via KAON portal	No
ICOM [ICOM, 2012]	XML , UML	XML, UML	Yes	Via FaCT	No	No	With inter-ontology mapping
DOE [Bachimont, 2000; Bachimon et al, 2002; Troncy & Isaac, 2002; DOE, 2012]	XSLT, RDF(S), OIL, DAML+OIL, OWL and CGXML	XSLT, RDF(S), OIL, DAML+OIL, OWL and CGXML	No	Via type inheritance and detection of cycles in hierarchies	No	Load ontology via URL	No
WebOnto [Webonto, 2012]	OCML	OCML, GXL, RDF(S) and OIL	Yes	Yes	With global write-only locking	Web based	?
OntoIntegrator [Nevzorova & Nevzorov, 2011]	own format	own format	Yes	?	No	No	?
Medius VOM [Polikoff, 2003; sandsoft, 2012]	XML Schema, RDF and DAML+OIL	XML Schema, RDF and DAML+OIL	UML diagrams via Rose	With a set of ontology authoring wizards	Network based	Via read-only browser from Rose	Limited (only native Rose model)
LinKFactory [Deray & Verheyden, 2003]	XML, RDF(S), DAML+OIL and OWL	XML, RDF(S), DAML+OIL, OWL and html	No	Yes	Yes	Yes	Yes
K-Infinity [Macris, 2004; OntoLex, 2012]	RDF	RDF	With Graph editor	Yes	Network based	No	?

Table 31. Additional functions of the well-known ontological systems [Youn & McLeod, 2006]

	Collaborative working	Ontology library	Inference engine	Exception handling	Ontology storage	Extensibility	Availability
Protégé [protégé, 2012]	No	Yes	With PAL	No	File & DBMS (JDBC)	Via plug-ins	Free
OilEd [Bechhofer et al, 2001]	No	Yes	With FaCT	No	File	No	Free
Apollo [Apollo, 2012]	No	Yes	No	No	Files	Via plug-ins	Free
RDFedit [rdfedit, 2012]	No	No	No	Yes	Files	No	Free
OntoLingua [Farquhar et al, 1996]	Yes	Yes	No	No	Files	No	Free
OntoEdit (Free version) [ontoprise, 2012; Sure et al, 2002; Sure et al, 2003]	No	No	No	No	File	Via plug-ins	Free
WebODE [Arpírez et al, 2001]	Yes	No	Prolog	No	DBMS (JDBC)	Via plug-ins	Free
KAON [Kaon, 2012]	?	Yes	Yes	No	?	No	Free
ICOM [ICOM, 2012]	No	?	Yes	No	DBMS	Yes	Free
DOE [Bachimont, 2000; Bachimon et al, 2002; Troncy & Isaac, 2002; DOE, 2012]	No	No	Yes	No	File	No	Free
WebOnto [Webonto, 2012]	Yes	Yes	Yes	No	File	No	Free web access
OntoIntegrator [Nevzorova & Nevzorov, 2011]	No	Yes	No	?	relational database	No	Free

	Collaborative working	Ontology library	Inference engine	Exception handling	Ontology storage	Extensibility	Availability
Medius VOM [Polikoff, 2003; sandsoft, 2012]	Yes	Yes (IEEE SUO)	Yes	?	?	Yes	Commercial
LinKFactory [Deray & Verheyden, 2003]	Yes	Yes	Yes	No	DBMS	Yes	Commercial
K-Infinity [Macris, 2004; OntoLex, 2012]	Yes	Yes	Yes	?	DBMS	No	Commercial

General disadvantages of the outlined instrumental systems are:

- Lack of automatic (or automated) procedures for forming components of ontologies;
- User interface based only on English, which does not permit using of other languages, such as Bulgarian, Russian, Greek, etc.;
- The structure of concepts may be built by only one type of relationships;
- For most commonly available ontological systems it is impossible to work with ontologies of large volume (e.g. OntoEdit free – up to 50 concepts);
- Many tools store the ontologies in text files, which limits the speed of access to ontologies;
- Some functions are not available in the free versions of the tools;
- User documentation is not good enough.

The above shortcomings of popular English language ontological tools exist in similar instruments from Russian segment, in particular, "Multi-layer ontology editor" [Artemieva & Reshtanenko, 2008], "OntoEditor+" [Nevzorova et al, 2004] and others.

➤ *Experiment to store ontology by NL-addressing*

Comparative analysis of the tools shows that all systems use finished products for data storing, which are limited to text files or relational databases. Both approaches for storing do not meet specific structures of the ontologies. This necessitates the development of new tools for storing ontologies.

Storing graphs and ontologies has one important aspect – the layers which correspond to types of relations between nodes of graph or ontology. The example with sample graph in previous chapter indicates that it is important to ensure possibility for multi-layer representation.

To make experiment with real data, we use the WordNet as ontology and its 45 types of relations (given by its files of different types, see Table 26) we store as 45 layers. For experiments in this case, we have realized a program called "OntoArM". It is outlined in the Appendix A. A screenshot from the OntoArM with results for the case with 45 layers is given in Figure 46.

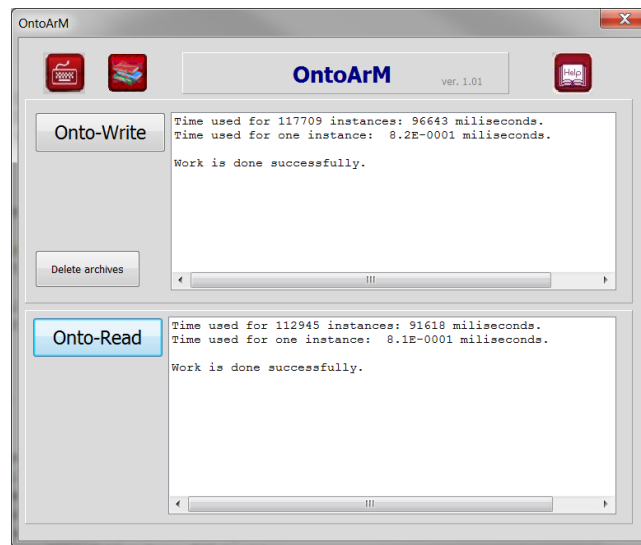


Figure 46. OntoArM results for the case of WordNet with 45 layers

The NL-addressing is case sensitive. The words “cut” and “CUT” are absolutely different as NL-addresses. Because of this, for words “cut” and “CUT” we have to use separate queries (Figure 47). Of course, it is easy to program the system automatically to use both capital and small letters. This is a problem to be solved at the middleware level.

In Figure 48 the OntoArM report for the query “cut; *” is shown.

The information on Figure 48 is shown with “no word wrap” option of WordPad program. In Table 32, the same report is given in whole for both queries (cut; *) and (CUT; *). The definitions are shown “as_is” in the lexicographer files, i.e. the access method does not convert the information to any other style and stores and extracts the information “as_is”.

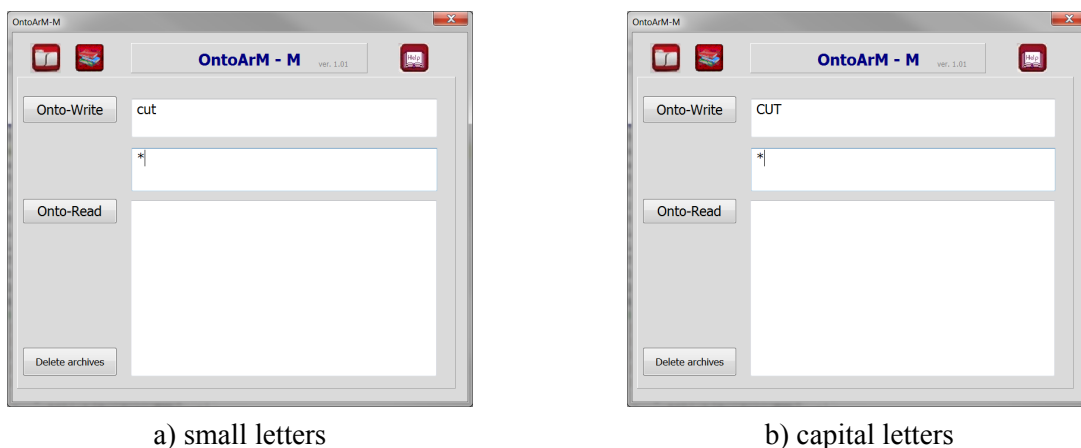


Figure 47. OntoArM panel for manual querying words cut and CUT

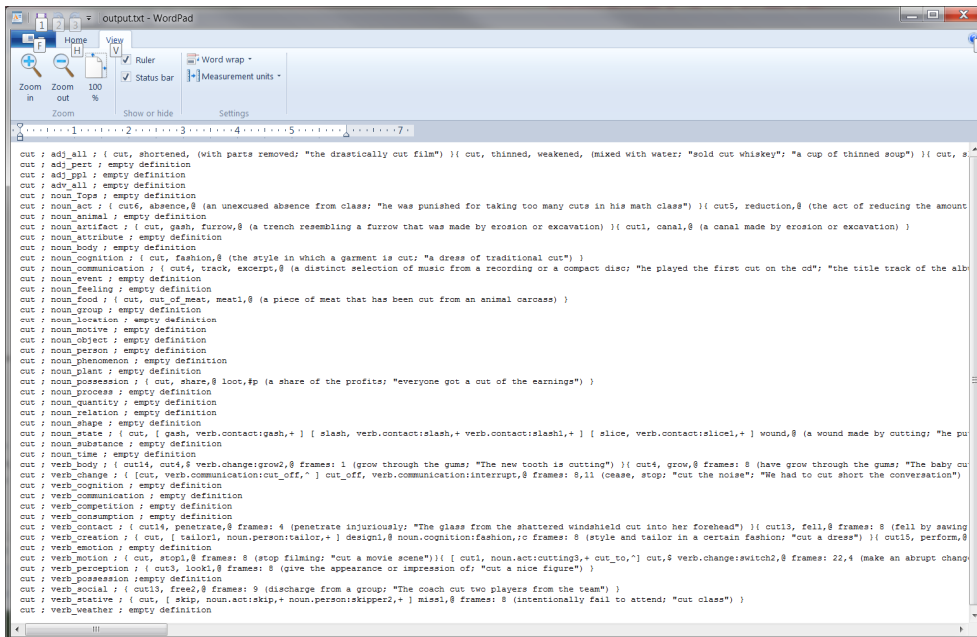


Figure 48. OntoArM report to query from Figure 47 a)

Table 32. Report of the queries from Figure 47 a) and b) for all 45 layers of WordNet and for both queries (cut;*) and (CUT;*)

No.	layer	definition
1	cut ; adj_all ;	{ cut, shortened, (with parts removed; "the drastically cut film") } { cut, thinned, weakened, (mixed with water; "sold cut whiskey"; "a cup of thinned soup") } { cut, slashed, ((used of rates or prices) reduced usually sharply; "the slashed prices attracted buyers") } { cut, emasculated, gelded, ((of a male animal) having the testicles removed; "a cut horse") }
	CUT ; adj_all ;	{ [CUT1, UNCUT1,!] (separated into parts or laid open or penetrated with a sharp edge or instrument; "the cut surface was mottled"; "cut tobacco"; "blood from his cut forehead"; "bandages on her cut wrists") } { [CUT2, UNCUT2,!] ((of pages of a book) having the folds of the leaves trimmed or slit; "the cut pages of the book") } { [CUT3, UNCUT3,!] (fashioned or shaped by cutting; "a well-cut suit"; "cut diamonds"; "cut velvet") }
2	cut ; adj_pert ;	empty definition
3	cut ; adj_ppl ;	empty definition
4	cut ; adv_all ;	empty definition
5	cut ; noun_Tops ;	empty definition

No.	layer	definition
6	cut ; noun_act ;	<p>{ cut6, absence,@ (an unexcused absence from class; "he was punished for taking too many cuts in his math class") }</p> <p>{ cut5, reduction,@ (the act of reducing the amount or number; "the mayor proposed extensive cuts in the city budget") }</p> <p>{ cut, [cutting, verb.creation:cut11,+] cutting_off1, shortening,@ (the act of shortening something by chopping off the ends; "the barber gave him a good cut") }</p> <p>{ cut1, [cutting1, verb.contact:cut10,+ verb.contact:cut,+] division,@ (the act of cutting something into parts; "his cuts were skillful"; "his cutting of the cake made a terrible mess") }</p> <p>{ cut2, [cutting2, verb.contact:cut10,+] opening2,@ (the act of penetrating or opening open with a sharp edge; "his cut in the lining revealed the hidden jewels") }</p> <p>{ cut9, [cutting9, verb.contact:cut5,+] division,@ card_game,#p (the division of a deck of cards before dealing; "he insisted that we give him the last cut before every deal"; "the cutting of the cards soon became a ritual") }</p> <p>{ cut8, [undercut, verb.contact:undercut,+] stroke,@ tennis;;c badminton;;c squash;;c ((sports) a stroke that puts reverse spin on the ball; "cuts do not bother a good tennis player") }</p>
7	cut ; noun_animal ;	empty definition
8	cut ; noun_artifact ;	<p>{ cut, gash, furrow,@ (a trench resembling a furrow that was made by erosion or excavation) }</p> <p>{ cut1, canal,@ (a canal made by erosion or excavation) }</p>
9	cut ; noun_attribute ;	empty definition
10	cut ; noun_body ;	empty definition
11	cut ; noun_cognition ;	{ cut, fashion,@ (the style in which a garment is cut; "a dress of traditional cut") }
12	cut ; noun_communication ;	<p>{ cut4, track, excerpt,@ (a distinct selection of music from a recording or a compact disc; "he played the first cut on the cd"; "the title track of the album") }</p> <p>{ cut, transition,@ ((film) an immediate transition from one shot to the next; "the cut from the accident scene to the hospital seemed too abrupt") }</p>
13	cut ; noun_event ;	empty definition
14	cut ; noun_feeling ;	empty definition
15	cut ; noun_food ;	{ cut, cut_of_meat, meat1,@ (a piece of meat that has been cut from an animal carcass) }
16	cut ; noun_group ;	empty definition

No.	layer	definition
17	cut ; noun_location ;	empty definition
18	cut ; noun_motive ;	empty definition
19	cut ; noun_object ;	empty definition
20	cut ; noun_person ;	empty definition
21	cut ; noun_phenomenon ;	empty definition
22	cut ; noun_plant ;	empty definition
23	cut ; noun_possession ;	{ cut, share,@ loot,#p (a share of the profits; "everyone got a cut of the earnings") }
24	cut ; noun_process ;	empty definition
25	cut ; noun_quantity ;	empty definition
26	cut ; noun_relation ;	empty definition
27	cut ; noun_shape ;	empty definition
28	cut ; noun_state ;	{ cut, [gash, verb.contact:gash,+] [slash, verb.contact:slash,+ verb.contact:slash1,+] [slice, verb.contact:slice1,+] wound,@ (a wound made by cutting; "he put a bandage over the cut") } { cut1, gradation,@ (a step on some scale; "he is a cut above the rest") }
29	cut ; noun_substance ;	empty definition
30	cut ; noun_time ;	empty definition
31	cut ; verb_body ;	{ cut14, cut4,\$ verb.change:grow2,@ frames: 1 (grow through the gums; "The new tooth is cutting") } { cut4, grow,@ frames: 8 (have grow through the gums; "The baby cut a tooth") }
32	cut ; verb_change ;	{ [cut, verb.communication:cut_off,^] cut_off, verb.communication:interrupt,@ frames: 8,11 (cease, stop; "cut the noise"; "We had to cut short the conversation") } { cut12, cut6,\$ decrease1,@ frames: 4 (have a reducing effect; "This cuts into my earnings") } { cut15, dissolve1,@ frames: 11 (dissolve by breaking down the fat of; "soap cuts grease") } { [cut2, cut_back,^ cut_back1,^ cut_out,^] prune, rationalize, rationalise, eliminate1,@ frames: 8 (weed out unwanted or unnecessary things; "We had to lose weight, so we cut the sugar from our diet") } { cut14, shorten9,@ frames: 8 (shorten as if by severing the edges or ends of; "cut my hair") }
33	cut ; verb_cognition ;	empty definition
34	cut ; verb_communication ;	empty definition
35	cut ; verb_competition ;	empty definition
36	cut ; verb_consumption ;	empty definition

No.	layer	definition
37	cut ; verb_contact ;	<p>{ cut14, penetrate,@ frames: 4 (penetrate injuriously; "The glass from the shattered windshield cut into her forehead") }</p> <p>{ cut13, fell,@ frames: 8 (fell by sawing; hew; "The Vietnamese cut a lot of timber while they occupied Cambodia") }</p> <p>{ cut15, reap,@ frames: 8 (reap or harvest; "cut grain") }</p> <p>{ cut7, hit,@ noun.act:sport,;c frames: 8 (hit (a ball) with a spin so that it turns in the opposite direction; "cut a Ping-Pong ball") }</p> <p>{ [cut, noun.person:cutter,+ noun.artifact:cutter,+ noun.act:cutting1,+ cut_away,^ cut_out2,^ cut_up,^ cut_into1,^ cut_off2,^ cut_out1,^] separate1,@ frames: 8,11 (separate with or as if with an instrument; "Cut the rope") }</p> <p>{ [cut5, noun.act:cutting9,+] shuffle,@ frames: 2,8 (divide a deck of cards at random into two parts to make selection difficult; "Wayne cut"; "She cut the deck for a long time") }</p> <p>{ [cut10, noun.act:cutting2,+ noun.act:cutting1,+] frames: 22 (make an incision or separation; "cut along the dotted line") }</p> <p>{ cut11, cut10,\$ verb.stative:be3,@ frames: 1 (allow incision or separation; "This bread cuts easily") }</p> <p>{ cut12, function,@ frames: 1 (function as a cutting instrument; "This knife cuts well") }</p>
38	cut ; verb_creation ;	<p>{ cut, [tailor1, noun.person:tailor,+] design1,@ noun.cognition:fashion,;c frames: 8 (style and tailor in a certain fashion; "cut a dress") }</p> <p>{ cut15, perform,@ frames: 8 (perform or carry out; "cut a caper") }</p> <p>{ [cut11, noun.act:cutting5,+ noun.act:cutting,+] create,@ frames: 8,11 (form or shape by cutting or incising; "cut paper dolls") }</p> <p>{ cut1, cut11,\$ create,@ frames: 8,11 (form by probing, penetrating, or digging; "cut a hole"; "cut trenches"; "The sweat cut little rivulets into her face") }</p> <p>{ cut6, burn5, create3,@ frames: 8 (create by duplicating data; "cut a disk"; "burn a CD") }</p> <p>{ cut4, cut6,\$ verb.communication:record1,@ frames: 8 (record a performance on (a medium); "cut a record") }</p> <p>{ cut5, cut4,\$ verb.communication:record1,@ frames: 8 (make a recording of; "cut the songs"; "She cut all of her major titles again") }</p>
39	cut ; verb_emotion ;	empty definition
40	cut ; verb_motion ;	<p>{ cut, stop1,@ frames: 8 (stop filming; "cut a movie scene") }</p> <p>{ [cut1, noun.act:cutting3,+ cut_to,^] cut,\$ verb.change:switch2,@ frames: 22,4 (make an abrupt change of image or sound; "cut from</p>

No.	layer	definition
		one scene to another") } { cut12, pass_through,@ frames: 8,9 (pass through or across; "The boat cut the water") } { cut13, cut12,\$ pass,@ frames: 22 (pass directly and often in haste; "We cut through the neighbor's yard to get home sooner") } { cut15, move,@ noun.act:boxing,;c frames: 22 (move (one's fist); "his opponent cut upward toward his chin") }
41	cut ; verb_perception ;	{ cut3, look1,@ frames: 8 (give the appearance or impression of; "cut a nice figure") }
42	cut ; verb_possession ;	empty definition
43	cut ; verb_social ;	{ cut13, free2,@ frames: 9 (discharge from a group; "The coach cut two players from the team") }
44	cut ; verb_stative ;	{ cut, [skip, noun.act:skip,+ noun.person:skipper2,+] miss1,@ frames: 8 (intentionally fail to attend; "cut class") }
45	cut ; verb_weather ;	empty definition

To update the content of a definition one may use form for manual work (Figure 49) and to follow the next steps:

- To enter the concept and layer;
- To activate reading current definition pressing the RDF-Read button;
- To update content of definition on screen;
- To press RDF-Write button to store new variant of definition in the correspond archive.

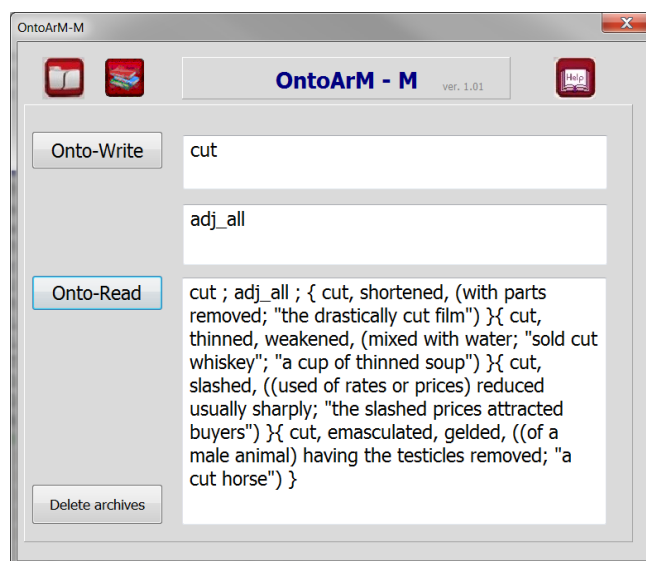


Figure 49. OntoArM panel for manual updating definitions

The results from experiments for storing WordNet as ontology with 45 layers are given in Table 33.

Table 33. Experimental data for storing WordNet as ontology

number of layers	operation	number of instances	total time in milliseconds	average time (ms) for one instance
45	writing	117 709	96 643	0.82
45	reading	112 945	91 618	0.81
45	work memory: 538 408 KB; permanent archive: 17 013 KB source text in 45 files - not compressed: 16 338 KB; compressed by WinZip: 4937 KB			

The work memory for storing hash tables and theirs containers was 538 408 KB. To analyze work of system, the work memory was chosen to be in a file in the external memory. In further realizations of OntoArM, to accelerate the speed and to reduce used disk space, work memory may be realized as part of main memory (as dynamically allocated memory or as file mapped in memory).

After finishing the work, occupied disk memory for compressed permanent archives is 17 013 KB, i.e. in this case the NL-indexing takes 12 076 KB additional compressed memory (the 45 sequential files with initial data occupy 16 338 KB, and compressed by WinZip they take 4 937 KB).

The difference in the numbers of instances in Table 27 and Table 33 is due to removing the equal instances and service information from input files when we use WordNet as thesaurus, stored as one layer archive. For instance, the word “cut” has many instances and when we work with one layer it must be written at least two NL-addresses – using small and capital letters: “cut” and “CUT”.

After updating, no recompiling of the data base is needed. For less than one millisecond after entering new data, the information is ready for using.

➤ *Comparing OntoArM and WordArM programs*

To compare WordArM and OntoArM programs, we made experiment with both programs to store WordNet data as one layer ontology. The results are given in the Table 34.

Table 34. Results for speed of WordArM and OntoArM programs

operation	WordNet as one layer ontology		WordNet as 45 layers ontology
	WordArM average time (ms) for one instance	OntoArM average time (ms) for one instance	OntoArM average time (ms) for one instance
writing	0.86	1.00	0.82
reading	0.78	0.95	0.81

From results in Table 34 we may conclude that OntoArM, working in parallel with 45 layers ensures more high speed than working with one layer. This is due to available separate buffering for each layer and small size of each of 45 archives (in the case of one layer all information is written in one big file).

In addition, OntoArM is slower than WordArM in the case of one layer because of existing operations for control of layers in OntoArM, which is not needed and not realized in WordArM.

➤ ***What gain and loss using NL-Addressing for storing ontologies?***

The conclusions are the same as for the dictionaries and thesauruses.

The loss is additional memory for storing internal hash structures. But the same if no great losses we will have if we will build balanced search trees or other kind in external indexing. It is difficult to compare with other systems because such information practically is not published.

The benefit is in two main achievements:

- High speed for storing and accessing the information;
- The possibility to access information immediately after storing without recompilation the database and rebuilding indexes.

➤ ***Conclusion of chapter 5***

In this chapter we have presented several experiments aimed to show the possibilities of NL-addressing to be used for NL-storing of structured datasets.

Firstly we introduced the idea of knowledge representation. Further in the chapter we discussed three main experiments - for NL-storing of dictionaries, thesauruses, and ontologies.

Presentations of every experiment started with introductory part aimed to give working definition and to outline state of the art in storing concrete structures.

The explanation of the experiments begins with the easiest case – storing dictionaries. Analyzing results from the experiment with a real dictionary data we may conclude that it is possible to use NL-addressing for storing such information.

Next experiment was aimed to answer to question: “What we gain and loss using NL-Addressing for storing thesauruses?”

Analyzing results from the experiment we point that the loss is additional memory for storing hash structures which serve NL-addressing. But the same if no great losses we will have if we will build balanced search trees or other kind in external indexing. It is difficult to compare with other systems because such information practically is not published. The benefit is in two main achievements:

(1) High speed for storing and accessing the information.

(2) The possibility to access the information immediately after storing without recompilation the database and rebuilding the indexes.

The third experiment considered the complex graph structures such as ontologies. The presented survey of the state of the art in this area has shown that main models for storing ontologies are files and relational databases.

Our experiment confirmed the conclusion about losses and benefits from using NL-addressing given above for thesauruses. The same is valid for more complex structures.

Here we have to note that for static structured datasets it is more convenient to use standard utilities and complicated indexes. NL-addressing is suitable for dynamic processes of creating and further development of structured datasets due to avoiding recompilation of the database index structures and high speed access to every data element.

The goal of the experiments with different small datasets, like dictionary, thesaurus or ontology, was to discover regularities in the NL-addressing realization. Analyzing Table 25, Table 27, and Table 33 we may see the main two regularities of storing time using NL-addressing:

- It depends on number of elements in the instances;*
- It not depends on number of instances in datasets.*