STORING DICTIONARIES AND THESAURUSES USING NL-ADDRESSING

Krassimira B. Ivanova, Koen Vanhoof, Krassimir Markov, Vitalii Velychko

***Abstract**: Our main goal in this paper is to propose a new approach for storing dictionaries and thesauruses using only the names but not pointers and this way to simplify and to speed up the corresponded computer programs. It is called "Natural Language Addressing" (NLA). This approach is a possibility to access information using natural language words or phrases as direct addresses of the information in the computer memory. For this purpose the internal encoding of the letters is used to generate corresponded address co-ordinates. This paper outlines the main idea of NLA approach. An extended example based on the WordNet thesaurus is given.*

***Keywords**: addressing; natural language addressing.*

***ACM Classification Keywords**: D.4.3 File Systems Management, Access methods.*

Introduction

An Introduction to Natural Language Addressing (NLA) was given in [Ivanova et al, 2012; Ivanova et al, 2013]. Below we will remember the main idea of NLA.

We will use concept "address" in sense accepted in the Computer Science: *the code that identifies where a piece of information is stored* [WordNet, 2012]; a name or number used in information storage or retrieval that is assigned to a specific memory location; the *memory location* identified by this name or number; a name or a sequence of characters that designates an e-mail account or a specific site on the Internet or other network [AHD, 2009].

We usually make difference between human aspect of the concept "address" and its computer "understanding". The NLA approach is based on using human representation of the address (by natural language words) as computer memory address.

➢ Computer indexes

The textual information of dictionaries and thesauruses may be stored in (internal or external) computer memory. Locating concepts and connected to them definitions may be done by:

- Direct scanning the files;
- Indexing and based on it search of the pointer to address of text element.

Scanning files is convenient only for small volumes of concepts. Some rationalization is possible using some algorithms like binary search.

Indexing is creating tables (indexes) that point to the location of folders, files and records. Depending on the purpose, indexing identifies location of resources based on file names, key data fields in a database record, and text within a file [PC mag, 2013].

The main idea of indexing is to facilitate the search by search in (multi-level) index and after that to ensure the direct access to address given by pointer. In other words, the goal of data indexing is to ease the search of (and

access to) data at all times. This is done by creating a data structure called index and providing faster access to the data. Accessing data is determined by the physical storage device being used. Indexing could potentially provide large increases in performance for large-scale analysis of unstructured data. In addition, the implementation of the chosen index must be suitable in terms of index construction time and storage utilization [Faye et al, 2012].

Indexing needs resources: memory for storing additional information and time for processing, which may be quite a long, especially for updating of the indexes when new elements are added or some old ones are removed.

➢ Names and addresses

Basic element of an index is couple: (name, address). In different sources the "name" is called "key", "concept", etc. The address usually is given by any "number", "pointer", etc.

There are two interpretations of the couple:

1) Address is a connection of the concept with its definition, i.e. practically we have triple:

(concept, address, definition).

2) Concept is a name of the address and may be used for user friendly style of programming and the third part of the triple (definition) may be variable.

NLA approach is based on using the computer encoding of name (concept) letters as address of connected to it information. This way no indexes are needed and high speed direct access to text elements is available. It is similar to natural order addressing in a dictionary book where no explicit index is used but the concept by itself locates definition. For instance, let have the next concept and corresponded definition:

"London: The capital city of England and the United Kingdom, and the largest city, urban zone and metropolitan area in the United Kingdom, and the European Union by most measures."

In computer memory, for example, the definition may be stored at address "FF084920". The index couple is:

("London", "FF084920"),

i.e. at memory address "FF084920" the main text, "*The capital … measures.*" will be stored.

To read/write the main text, firstly we need to find name "London" in the index and after that to access memory address "FF084920" to read/write the definition.

If we assume that name "London" in the computer memory is encoded by six numbers (letter codes), for instance by using ASCII encoding system London is encoded as (76, 111, 110, 100, 111, 110), than we may use these codes as direct address to memory, i.e.

("London", "76, 111, 110, 100, 111, 110")

One may remark that above we have written two times the same name and this is truth. Because of this we may omit this couple and index, and read/write directly to the address (76, 111, 110, 100, 111, 110).

For human this address will be shown as "London", but for the computer it will be "76, 111, 110, 100, 111, 110".

From other point of view, the array (76, 111, 110, 100, 111, 110) may be assumed as co-ordinates of point in multidimensional (in this case – six dimensional) information space and the definition can be stored in this point.

The natural language does not contain words only of six letters long. The length of the words is variable and in addition there exits names as phrases. This means that we really have multidimensional address space defined by set of all natural words and phrases.

What we need is a program function which converts such multidimensional addresses in concrete linear machine locations (on the hard disk, for example) and corresponded access method.

➢ Storing Dictionaries and Thesauruses

Dictionary is a reference resource, in printed or electronic form, which 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].

Dictionary may be a book, optical disc, mobile device, or online lexical resource containing a selection of the words of a language, giving information about their meanings, pronunciations, etymologies, inflected forms, derived forms, etc., expressed in either the same or another language; lexicon; glossary. Print dictionaries of various sizes, ranging from small pocket dictionaries to multivolume books, usually sort entries alphabetically, as typical CD or DVD dictionary applications do, allowing one to browse through the terms in sequence. All electronic dictionaries, whether online or installed on a device, can provide immediate, direct access to a search term, its meanings, and ancillary information [Dict, 2013].

Thesaurus is a book or catalog of words with the same or nearly the same meanings, or synonyms, and their opposites, or antonyms. An online thesaurus provides immediate electronic access to lists of alternate terms for the queried word, covering its various shades of meaning [Collins, 2003; AHD, 2009; Dict, 2013]

What is important is that the main dictionary construction is the couple (concept, definition). In the electronic variant it becomes triple (concept, address, definition).

The computer storing model for dictionaries is simple because the one-one correspondence “word-definition”. For computer dictionary it is enough to have a sorted list of words and a tool for binary search and pointers to the corresponded definitions.

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

WordNet thesaurus

WordNet® is a large lexical database of English. It 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. WordNet 3.0 database contains 155 287 words organized in 117659 synsets for a total of 206941 word-sense pairs; in compressed form it is about 12 megabytes in size [WordNet, 2012].

WordNet superficially resembles a thesaurus, in that it groups words together based on their meanings. However, there are some important distinctions.

- First, WordNet interlinks not just word forms — strings of letters — but specific senses of words. As a result, words that are found in close proximity to one another in the network are semantically disambiguated.
- Second, WordNet labels the semantic relations among words, whereas the groupings of words in a thesaurus do not follow any explicit pattern other than meaning similarity.

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 was to write source files that contain the basic lexical data — the contents of those files are lexical substance of WordNet.
- The second task was to create a set of computer programs that would accept 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 to convert these files into the WordNet lexical database;
- The WordNet lexical database;
- The suite of software tools used to access the database.

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.

WordNet organizes nouns, verbs, adjectives and adverbs into sets of cognitive synonyms (synsets), each expressing a distinct concept. Synsets are interlinked by means of conceptual-semantic and lexical relations. Further they are arranged into a set of lexicographers' source files by syntactic category and other organizational criteria. Adverbs are maintained in one file, while nouns and verbs are grouped according to semantic fields. Adjectives are divided between two files: one for descriptive adjectives and one for relational adjectives.

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 Grinder is a multi-pass compiler that is coded in C++. The first pass uses a parser to verify that the syntax of the input files conforms to the specification of the input grammar and lexical items, and builds an internal representation of the parsed synsets.

In its second pass, the Grinder resolves all of the semantic and lexical pointers. To do this, the pointers that were specified in each synset are examined in turn, and the target of each pointer (either a synset or a word form in a synset) is found. The source pointer is then resolved by adding an entry to the internal data structure which notes the ''location'' of the target. In the case of reflexive pointers, the target pointer's synset is then searched for a corresponding reflexive pointer:

- If a reflexive pointer is found, it is replaced by the pointer to the original source.
- If a reflexive pointer is not found, the Grinder automatically creates one with all the pertinent information.

A subsequent pass through the list of word forms assigns a polysemy index value, or sense count, to each word form found in the on-line dictionary. There is a separate sense count for each syntactic category that the word form is found in.

The Grinder's final pass generates the WordNet database.

The resulting network of meaningfully related words and concepts can be navigated with a specialized browser [Fellbaum et al, 1998; Miller, 1995].

WordNet database

WordNet database is divided into five lexical and grammatical classes: noun, verb, adjective, adverb and functional words. The parts of speech are organized in a hierarchy of synsets (nodes).

The internal representation of the lexicographic data is a network of interrelated linked lists. A *hash table* of word forms is created as the lexicographers' files are parsed. Lower-case strings are used as keys; the original orthographic word form, if not in lower-case, is retained as part of the data structure for inclusion in the database files.

As the parser processes an input file, it calls functions which create data structures for the word forms, pointers, and verb frames in a synset. Once an entire synset had been parsed, a data structure is created for it which includes pointers to the various structures representing the word forms, pointers, and verb frames.

All of the synsets from the input files are maintained as a single linked list. The Grinder's different passes access the structures either through the *linked list* of synsets or the *hash table* of word forms. A list of synsets that specify each word form is maintained for the purposes of resolving pointers and generating the database's index files.

For each syntactic category, two files represent the WordNet database — index.pos and data.pos, where "pos" is either "noun", "verb", "adj" or "adv" (the actual file names may be different on platforms other than Sun-4). The 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. Each index file is an alphabetized list of all of the word forms in WordNet for the corresponding syntactic category. Each 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*.

The index and data files are interrelated. Part of each entry in an index file is a list of one or more byte offsets, each indicating the starting address of a synset in a data file. The first step to the retrieval of synsets or other information is typically a search for a word form in one or more index files to obtain all data file addresses of the synsets containing the word form. Each address is the byte offset (in the data file corresponding to the syntactic category of the index file) at which the synset's information begins [Fellbaum, 1998].

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].

WordNet storing model

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 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 1):

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

WordNet stores information about words in four main data files (for nouns, verbs, adjectives and adverbs). The data structure is the same in each of these 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 2 and Figure 3). This value is the unique identifier of the synset. The synset data elements are separated by spaces. We should note that links to other synsets are given again by the absolute 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 2. 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 3. 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 simple: at the first place the word is given and, after some coded information, absolute addresses of corresponded synsets are given (Figure 4 and Figure 5).

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

Figure 4. Record for the word "accession" in the index of nouns

accession v 1 2 @ ; 1 0 **00989696**

Figure 5. 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 absolute addresses are collected and then system reads synsets directly from 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 и 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 absolute address of a synset (Figure 6).

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 6. Records for the word "accession" in the sense index

In this case, to reach all synsets of a word, firstly a binary search is made in the sense index and the corresponded absolute 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 $O(\log(n))$, where $n= n_n+n_v+n_a+n_r$, i.e. 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.

Disadvantages of WordNet storing model

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 search and, based on it, extracting needed information via direct access based on absolute addresses in data files.

Many disadvantages of the WordNet organization are discussed in [Poprat et al, 2008]. An important shortcoming of the WordNet database's structure is that although all files are in ASCII, and are therefore editable, and in theory extensible, in practice *this is almost impossible*. One of Grinder's primary functions is the calculation of addresses for synsets in data files. Editing any of the database files would (most likely) create incorrect byte offsets, and would thus derail many searching strategies. At present time, building a WordNet database requires using of the Grinder and processing of all lexicographers' source files at the same time [Fellbaum, 1998]. In addition, Grinder utility is available only for Unix/Linux operating systems.

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

1. Absolute addressing is convenient for computer processing, but it is difficult to be used by customer;
2. Manual creating of numerical addresses is impossible, and their use can be done only by corresponded program;
3. 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 WordNet program "Grinder" and 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.

Our goal is to show that WordNet lexical database may be realized without using pointers and this way to avoid pointed above limitations and recompilation of the database after every update.

NL-addressing

Let see a small example shown on Figure 7 - two variants of the synset of the word "accession" from different compilations of 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].

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"

а) Version from the 2012 year

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"

б) Version from the 2011year

Figure 7. Synset of the word "accession" from the data files for nouns

The difference between absolute addresses is visible.

Due to absolute 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.

Our main goal in this work is to propose a NLA as a new approach for storing WordNet database, using only the names but not pointers and this way to simplify and to speed up the corresponded computer programs.

Analyzing the database structure of synset we may see an important peculiarity.

In the computer memory, all characters are represented by numerical codes (one, two or four bytes depending on the encoding system such as ASCII or UNICODE). This way, in WordNet storing model, the unique digital code of word is combined with another unique numeric code of its place in the file (absolute address). This duplicating of the codes could be avoided if NLA model of information storing is used.

Let remember, ASCII internal representation of word “accession” is the following unique sequence of numbers: (97, 99, 99, 101, 115, 115, 105, 111, 110). It can be used as a spatial address in nine-dimensional space. At this address we can record the synset and access it again via this address.

In this case, for customer, the address will be presented by word “accession” and for computer by vector (97, 99, 99, 101, 115, 115, 105, 111, 110).

If we apply this opportunity to WordNet data, we will obtain a result that is understandable for users and, at the same time, fully recognizable for the computer programs.

As example two variants of the synset of word "accession" are shown in Figure 8: (a) WordNet version and (b) NLA-version.

An important feature of NLA-version is that it can be updated dynamically without recompilation of the database.

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"

а) 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"

б) NLA-version

Figure 8. WordNet and NLA-versions of the synset of the word "accession"

Experiments

➢ Experiments for storing dictionaries

Our first experiment was 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 23412 words in English and Bulgarian with their definitions in Bulgarian, stored in a sequential file with size of 2410 KB.

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 experimental program "WordArM" used for the experiments is specially designed for storing dictionaries and thesauruses based on NL-addressing.

For storing dictionaries we use simple model: definitions are stored at the NL-addresses formed from words corresponded to them.

The speed for storing, accessing, and size of the work and permanent files are given in Table 1.

Table 1. Experimental data for NL-storing of a dictionary

operation	number of instances	total time in milliseconds	average time for one instance	work file	permanent file
NL-writing	23 412	22 105	0.94 ms	80 898 KB	2 939 KB
NL-reading	23 412	20 826	0.89 ms		

The analysis of the results in Table 1 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.

The work memory taken during the work was 80898 KB. To analyze the system performance, work memory was chosen to be in a file but not in the main memory. After finishing the work, occupied permanent memory for compressed archive is 2939 KB. This means that the NL-indexing takes 2103 KB additional compressed memory (the sequential file with initial data is 2410 KB and compressed by WinZip it is 836 KB).

In further realizations of WordArM, the work memory 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.

➢ Experiments for storing thesauruses

Storing thesauruses is similar to storing dictionaries. The main difference is that the computer storing models for thesauruses are more complicated due to existing more than one corresponded definitions for a given word (synonyms) as well as other relations between words (hyperonymy, hyponymy, meronymy, etc.) which need to be encoded in proper way and the database has to support such relations. We had no goal to realize our own thesaurus. Because of this for experiments we have chosen the WordNet thesaurus.

Using NL-addressing, the WordNet lexical database may be realized without using pointers and this way to avoid the limitations and recompilation of the database after every update.

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

The total number of instances (file records) is 117 871 from which 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 remark that there are equal synsets in several lexicographer files. This has matter when we integrate the 45 files in one source file for representing a thesaurus.

The program used for the experiments is the experimental program “WordArM”. The results are given in Table 2. A screenshot from the WordArM for the case of WordNet as thesaurus is shown at Figure 9.

Table 2. 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 memory: 15 603 KB; source text: 1 333 KB			

We receive practically the same results as for storing dictionaries. The analysis of results in Table 2 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.

Work memory taken during the work was 385538 KB. To analyze the system performance, work memory was chosen to be in a file but not in main memory. In further realizations, to accelerate the speed and reduce of used disk space, work memory may be realized as part of main memory (as dynamically allocated memory or as a file mapped in memory).

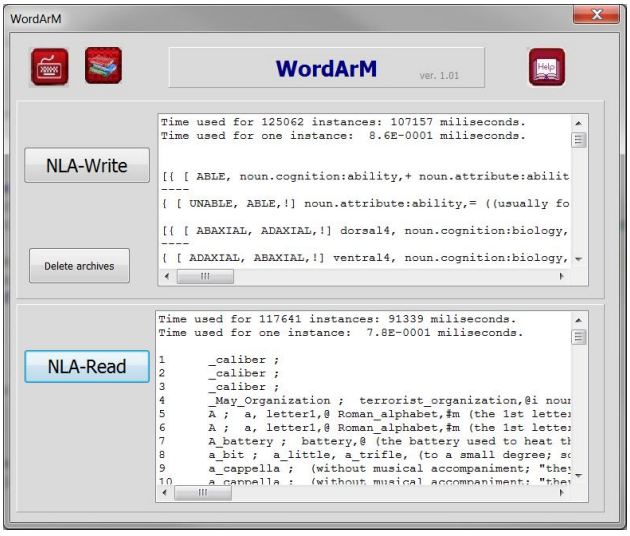

Figure 9. WordArM results for the case of WordNet as thesaurus

Comparing results in Table 1 and Table 2 we may see that more great number of instances in the case of thesauruses is stored and accessed faster than less number of instances in the case of dictionaries. The main reason for this is that the system needs some time for preparing internal indexing structures. Main part of this time is spent during creating archive and initial empty structures.

After finishing the work, occupied permanent memory for compressed archive is 15603 KB, i.e. in this case NL-indexing takes 14270 KB additional compressed memory (the sequential file with initial data is 1333 KB).

To compare our results with “Grinder” (original WordNet program for generating the database) we provided experiments on separate computer under operating system “Linux open source 12.3”. Computer configuration was: Processor: Celeron Dual Core T35.00 2.1 GH; Memory: 2GB; HDD: 230 GB. It is important that Grinder works in single command prompt mode, which permits high speed processing. Multitasking and Graphical User

Interface of MS Windows takes additional resources (processor time and main memory) for supporting its work and this way the work of the user program became slower. Pseudo parallelism of Windows 7 processes is additional factor for delay of applied programs work.

The steps of Grinder program were: Resolving pointers; Getting sense counts; Figuring out byte offsets; Dumping data files; Dumping index files; Dumping sense index. The time used for generating the data base was about 7 seconds. If we assume that access to hard disk blocks is at least 100 times slower than access to main memory blocks, we may accept that WordArM will process the same information for about one second if it is only in the main memory, i.e. working only in main memory, WordArM will store information at least seven times faster.

It was impossible to measure the WordNet access to information in compiled files.

The size of data and index files created by Grinder is 32.3 MB (33 921 109 bytes); compressed by WinZip they have size of 9584 KB, the size of the WordNet compressed archive is 15603 KB, i.e. our archive takes about one and half more disk space.

Analyzing other software systems for maintenance of dictionaries and thesauruses we found that for more of them there exist limitations for maximal length of the concepts which vary from 60 up to 255 bytes [Will & Will, 2004]. As example of the size of thesaurus database we may point the MultiTes system which thesaurus database needs 15 MB of disk space for every 50,000 terms [MultiTes, 2013]. WordArM needs the same disk space for two times more terms.

Conclusion

Natural Language Addressing (NLA) is a possibility to access information using natural language words or phrases as direct addresses of the information in the computer memory. For this purpose the internal encoding of the letters is used to generate corresponded address co-ordinates. In this paper we outlined the main idea of NLA and illustrated its genesis.

NLA is applicable for storing dictionaries and thesauruses.

The conclusions about the gain and loss from using NL-Addressing for storing dictionaries and thesauruses are:

- The loss is additional memory for storing internal address 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:

  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.

Bibliography

[AHD, 2009] The American Heritage® "Dictionary of the English Language" Fourth Edition copyright© 2000 by Houghton Mifflin Company, Updated in 2009; Published by Houghton Mifflin Company. All rights reserved.

[Angelov, 2012] St. Angelov. SA Dictionary http://www.thediction.com/ (accessed: 11.01.2013).

[Collins, 2003] "Collins English Dictionary – Complete and Unabridged", HarperCollins Publishers, 1991, 1994, 1998, 2000, 2003

[Dict, 2013] Dictionary.com LLC, http://dictionary.reference.com/ (accessed: 14.04.2013)

[Faye et al, 2012] David C. Faye, Olivier Cure, Guillaume Blin. A survey of RDF storage approaches. Received, December 12, 2011, Accepted, February 7, 2012, ARIMA Journal, vol. 15 (2012), pp. 11-35.

[Fellbaum et al, 1998] Fellbaum, Christiane, ed. WordNet: An Electronic Lexical Database/MIT Press, Cambridge, MA, 1998. pp. 422

[Fellbaum, 1998] Fellbaum Christiane (ed.) WordNet. “An Electronic Lexical Database”, ISBN: 978026206197, MA: MIT Press; 1998, pp. 422

[Ivanova et al, 2012] Krassimira Ivanova, Vitalii Velychko, Krassimir Markov. About NL-addressing (К вопросу о естествено-языконой адрессации). In: V. Velychko et al (ed.), Problems of Computer in Intellectualization. ITHEA® 2012, Kiev, Ukraine - Sofia, Bulgaria, ISBN: 978-954-16-0061-0 (printed), ISBN: 978-954-16-0062-7 (online), pp. 77-83 (in Russian).

[Ivanova et al, 2013] Krassimira B. Ivanova, Koen Vanhoof, Krassimir Markov, Vitalii Velychko, “Introduction on the Natural Language Addressing”, International Journal of Information Technologies and Knowledge, Vol.7, Number 2, 2013, pp. 139–146. ISSN 1313-0455

[Miller, 1995] Miller G. A. "WordNet: a lexical database for English"/G. A. Miller. – Communications of the ACM 38: 11, 1995. pp. 39–41.

[MultiTes, 2013] MultiTes. http://www.multites.com (accessed: 18.04.2013)

[Palagin et al, 2011] Palagin A.V., Krivii S.L., Petrenko N.G. Ontological methods and instruments for processing domain knowledge. (А. В. Палагин, С. Л. Крывый, Н. Г. Петренко. Онтологические методы и средства обработки предметных знаний: монография/Луганск: изд-во ВНУ им. В. Даля, 2011. – 300 с.), (in Russian).

[PC mag, 2013] PC Magazine Enciclopedia http://www.pcmag.com/encyclopedia_term/0,1237,t=indexing&i=44896,00.asp (accessed: 23.01.2013)

[Poprat et al, 2008] Poprat Michael, Elena Beisswanger, Udo Hahn. Building a BioWordNet by Using WordNet’s Data Formats and WordNet’s Software Infrastructure — A Failure Story. Software Engineering, Testing, and Quality Assurance for Natural Language Processing, Columbus, Ohio, USA, June 2008. Association for Computational Linguistics, 2008, pp. 31–39.

[Will & Will, 2004] L.D. Will, S.E. Will. Willpower Information. http://www.willpowerinfo.co.uk/thestabl.htm (accessed: 18.04.2013)

[WordNet, 2012] Princeton University "About WordNet", WordNet, Princeton University, 2010 http://WordNet.princeton.edu (accessed: 23.07.2012)

Authors' Information

Ivanova Krassimira *– University of National and World Economy, Sofia, Bulgaria. Institute of Mathematica and Informatics, BAS, Sofia, Bulgara.*

e-mail: krasy78@mail.bg

Major Fields of Scientific Research: Software Engineering, Business Informatics, Data Mining, Multidimensional multi-layer data structures in self-structured systems

Vanhoof Koen - *Universiteit Hasselt; Belgium*

e-mail: koen.vanhoof@uhasselt.be

Main research areas: data mining, knowledge retrieval.

Markov Krassimir *– Institute of Mathematics and Informatics, BAS, Bulgaria*

e-mail: markov@foibg.com

Major Fields of Scientific Research: General theoretical information research, Multi-dimensional information systems

Velychko Vitalii – *Institute of Cybernetics, NASU, Kiev, Ukraine*

e-mail: Velychko@rambler.ru

Major Fields of Scientific Research: Data Mining, Natural Language Processing