Pieter C. Masereeuw and Iskandar Serail,
University of Amsterdam

DictEdit: a computer program for dictionary data entry and editing

ABSTRACT: The program DictEdit has been developed at the Computer Department of the Faculty of Arts, University of Amsterdam. In this paper we argue the need for such a program in view of the increasing demand for and use of electronic dictionaries. We summarize the requirements of such dictionaries and of a program to create them. In the last part of this paper, we describe DictEdit's functionality in more detail.

1. Computers: dictionary producers and dictionary users

It is impossible to give an accurate account of all relations between computer technology and lexicographical practice within the limited space available. Therefore, we shall only emphasize two aspects that we consider most important for understanding the principles that underlie DictEdit.

First, computers are used by lexicographers during the dictionary compilation process. Text processing facilities are used to enter dictionary data, database programs to organize them. Furthermore, large text corpora are rapidly becoming available for exploration by the lexicographer, providing a rich source of evidence on the use of words, phrases and idioms. Thus they facilitate more comprehensive and detailed collection of relevant examples. These aspects of computer use during the dictionary compilation process have been described by, for example, Clear (1987).

Second, computers also become 'dictionary users'. State-of-the-art application software depends to a large extent on reliable computerized lexicons. Spelling checkers, grammar and style checkers, natural language interfaces to databases and thesauruses for information retrieval systems require computer readable dictionaries or lexicons. This is also reflected in the scientific fields of computer science, computational linguistics and artificial intelligence. Natural language processing is a major topic in these disciplines. Contemporary theories on natural language processing tend to put a heavy burden on the lexicon, requiring detailed information on morphological, grammatical, syntactic and semantic properties of lexical items. Computational lexicography is considered to provide an answer to these demands (see e.g. Boguraev and Briscoe, 1989).

Maybe it is important at this stage to mention that we do not feel that this second aspect will decrease the importance of printed dictionaries, as there are many situations in which the human user will prefer to have a book at hand. Nor do we think that at any time dictionaries will be automatically produced by machines. The observation we make is that there is a growing need for large scale reliable lexicons (or dictionaries, if you like) on the one hand, and that the possibilities to create them are rapidly improving on the other. We feel that computer technology is of crucial importance for these developments.

2. The need and possibility to re-use resources

The compilation of dictionaries is an expensive process, as is clear from the amounts of time, money and human resources involved. With an increasing demand for dictionaries both in electronic and in printed form, lexicographers (and others involved, such as dictionary publishers) have to consider the reusability of their investments. It is very attractive to enter source data once and derive several dictionaries from them. Or, as an alternative, to transform one dictionary into another.

Many research projects have focused on the theme of reusability of dictionary data. As examples, we refer to some projects carried out (at least partly) at the University of Amsterdam:

- the ASCOT-project (Akkerman e.a., 1988), which resulted (among other things) in a computerized lexicon for morphological tagging purposes;
- the LINKS-project (Vossen, 1990), in which the dictionary definitions were syntactically analyzed and semantically interpreted;
- the ACQUILEX-project,[1] which links up these analyzed definitions with analyses of Dutch, Italian and Spanish monolingual dictionaries in order to build a multilingual lexical knowledge base for natural language processing;
- the project "Conversion of a Dutch-Russian dictionary" (Honselaar and Elstrodt, fc.) in which an existing Russian-Dutch bilingual dictionary was "inverted" to provide the basis for a new Dutch-Russian counterpart.

The first three projects had the Longman Dictionary of Contemporary English (Procter, 1978) as their source dictionary, the last one the Dutch-Russian dictionary by Van Den Baar (1989).

2.1. Reusability of dictionary data

The projects mentioned above made clear that an important change of perspective was required: if dictionary data are to be exploited for several purposes, the electronic form of the dictionary should no longer be considered a spin-off of the printed form, but rather the other way around. The new electronic dictionary must meet the following requirements:

- a formal and unambiguous specification of the structure of dictionary entries. This means that there must be a formal description of all possible entry fields, the values allowed within each field and the possible relations and dependencies between (the values of) the fields. Examples of these properties are given in section 3 below;
- consistency with this specification of all entries throughout the dictionary;

- an explicit relation between the structure of each particular entry and the general specification;
- a clear distinction between the content of the dictionary and its printed output.

DictEdit is a computer program specially designed to ensure that the dictionary data satisfy these requirements. It operates on the basis of a formal definition of the structure of an entry, checks the data entered for consistency with the structure and stores data in such a way that the structure of each entry is always unambiguously recoverable.

We want to point out very clearly that there is no way in which DictEdit (or any other program) could decide what should be in a dictionary and how it should be represented. These decisions can only be made by the lexicographer, using his or her insights in the languages involved and the intended use of the dictionary (either by humans or by computers). The structure of the dictionary entries must be defined in such a way that it allows all relevant material to be included, yet prohibits inconsistencies. It should not keep the lexicographer from expressing his/her intuitions and findings, but from expressing them in an arbitrary way.

Appropriate use of DictEdit will force the lexicographer to plan his/her material even more carefully than he/she would otherwise have done. We agree with Landau (1989, 277) that

> "(...) a computerized file demands considerably more planning than conventional files since one must plan not only for the current dictionary but for its subsequent revisions, abridgments and derivative works, and at a time when these offshoots seem remote and problematical."

On the other hand, decisions regarding the exact format and arrangement of the final (printed) result may be postponed till a later stage, since the separate elements of the entries are clearly distinguished and may be rearranged in any desirable way. In the end, the computer-assisted approach will undoubtedly be profitable.

2.2. Reusability of the program itself

DictEdit also provides another kind of reusability: it is basically usable in any dictionary compilation project without it needing substantial redesign. This claim implies the following major features:

- dictionary independency: once a dictionary definition has been specified, the program's behaviour is completely driven by it. In order to tailor it for another dictionary, the lexicographer only has to specify another definition and the program will behave accordingly;
- language independency: the program supports different alphabets (with their own sorting orders), character sets and fonts, even within a single dictionary definition;
- theory independency: the program does not assume any particular linguistic theory;
- platform independency: the main part of the program is developed in standard PASCAL, and is hence portable between virtually any computer system and operating system. The user interface, however, is necessarily platform-specific; it is initially developed for the Apple Macintosh and we are currently trying to implement versions for VAX/VMS and DOS.

3. Functionality of DictEdit

Essentially, editing a dictionary entry with DictEdit is nothing more than filling in a form. This form has to be defined by the editor in chief. When data is entered into the form, DictEdit performs continuous checks to make sure that nothing invalid is entered.

DictEdit behaves like a mixture of a database program and a simple word processor:

- like a database, a DictEdit entry consists of fields, but unlike most databases, DictEdit's fields are hierarchical, i.e. the existence of some fields depends on the existence of other, higher-level fields;
- as with a word processor, the editor has complete freedom in entering data into the form; there is no predefined order in which data should be entered, and the editor can make corrections at any stage, just by bringing the cursor back to the position he or she wants to correct.

The differences with ordinary word processors and databases are:

- the editor may not move the cursor to another field if the current field contains invalid data;
- sometimes, fields are inserted or deleted as a result of values entered;
- text has some well-defined properties depending on the prior specifications, such as the kind of the text, its alphabet and corresponding keyboard, subsets of allowable characters and so on.

The chief editor can also define a print format for entries; this makes it possible to edit the dictionary irrespective of its final printed form; defining a print format is not necessary: it is equally possible that DictEdit in a given context is simply used as a tool for searching and retrieving dictionary data according to various criteria.

The DictEdit program is not finished. In the rest of this text, we will use the following marks to indicate the development status of features that have not yet been fully implemented.

[RSN]	Real Soon Now: finished between now and a few months.
[NF]	Near future: finished between a few months and a year.
[FF]	Far future: finished after one year or more.

3.1. Properties of a DictEdit dictionary definition

A DictEdit dictionary definition describes a dictionary entry in terms of fields:

- the relation between fields is defined in terms of groups, menus, repeatability and optionality;
- the content of fields is defined in terms of field type, alphabets, functions, label sets and abbreviations.

In the rest of this section we will give a short discussion of these features.

3.2. The content of fields

3.2.1. *Alphabets*

A dictionary can be written in one or more alphabets. Internally, these alphabets are translated into their corresponding fonts, keyboard layouts and sorting order.[2] Each field in an entry defines which alphabets it allows. It is also possible to define a subset of an alphabet, e.g.: only digits for a numeric field.

3.2.2. *[RSN] Functions*

With "functions" we refer to special abbreviatory marks used in dictionaries. Every field in the dictionary definition declares which function(s) it allows (if any). By means of functions, DictEdit formalizes and controls the use of tildes (entire main word string), hyphens (definable part of the main word), cross-references, enumeration (comma, slash), several types of parenthesis and abbreviations.

3.2.3. *[RSN] Labels*

With "labels" we refer to text elements that constitute a special class of abbreviations (often printed in a special format) which supply extra information to the dictionary user, such as, for instance, the language "register" of a word or its meaning (e.g., "[Vulg]"). Every field in the dictionary definition defines which set of labels it allows and [NF] at what position (front, end, anywhere) in the field. DictEdit does not define a scope for labels. Perhaps it will do so later.

3.3. Relations between fields

3.3.1. *Hierarchy: fields, groups and menus*

All fields in a dictionary definition have a unique name. The editor need not be aware of these names: he or she only sees the descriptive "prompt" of the field, which need not be unique.

Fields can be combined into groups. Like fields, groups have unique names. Groups of fields are used to define menus. The same group may occur in more than one menu.

We now come to the most important property of a DictEdit definition: hierarchical structure. Depending on its content, a field can have a different submenu. For instance, a category field will have different submenus for nouns and verbs.

Some additional remarks:

- one menu is assigned a special status: the so-called main menu. It contains the headword, identifying the entry;
- recursion, whether direct or indirect, is forbidden;
- submenus are positioned after their parent field;
- those who see a resemblance here with context free grammars are quite right of course.

3.3.2. Field type

DictEdit distinguishes several field types:

- fixed value fields: the content of a field can only be chosen from a user-defined list. Every value can lead to a different submenu. It is possible for fields to have a default value (hence, a field can have a default submenu). A default value is automatically filled in by the program;
- free text fields: the editor is free to enter whatever text is allowed by the definition of content of the field. Free text fields cannot have a default value; they can have a submenu, though, which is present if the field is filled. [FF] Later, more elaborate checking of free text fields may take place, e.g. on-line checking of controlled vocabulary or even the syntax of word definitions;[3]
- [NF] cross-reference and label fields: fields that only allow a cross-reference or a label;
- [RSN] several kinds of comment fields: 1) ordinary comments; 2) alert comments that "pop up" when an entry is opened; 3) comments generated by the program if an entry is retrieved that does not fully comply with the definition.[4]

3.3.3. Repeatability

The definition may state that a field is repeatable; if a field with a submenu is repeated, the repeated field follows the last field of the submenu. Some fields are repeated so frequently that they can be made to repeat automatically.

For repeated fixed value fields, it is possible to define the order in which values should be entered in the repeated fields. For instance, one could state that a definition of an item as "adjective" should precede definition as "adverb".

Groups may also be repeatable. This feature may be used to collectively repeat fields that belong closely together, such as an example field and the example's translation.

3.3.4. Presence of fields

A field can be (the two options are NOT mutually exclusive):

- obligatory or not. If it is, it must have some legal content;
- always present or present on demand. "Always present fields" can never be deleted from the entry. "Present on demand fields" can be freely removed or added (at the right spot). They can be used to define infrequent properties. It is possible to switch between full menus (all fields present) and short menus (empty fields that are present on demand are removed or not generated).

3.3.5. [NF] Breaking the hierarchy: parameters

Not every kind of relation can be expressed easily within a hierarchical setup, especially if there is more than one relation at the same time. DictEdit will make it possible to break the hierarchy by means of parameters. Unfortunately, space limitations prevent us from going into more detail about these parameters, but those who guess that there is a resemblance with context free grammars extended with parameters are right.

3.4. [FF] Searching

DictEdit will provide extensive search and retrieval facilities. This part of the program is underspecified as yet, but it is our intention that the editor should formulate his query in terms of fields (optionally taking into account the field's hierarchical position), values and words. Use of Boolean operators and wildcards will be possible. In contrast with many existing search programs, DictEdit will take information about alphabets into account when searching.

3.5. [FF] Printing

Dictionary entries will be printed according to a print format specification made by the editor in chief. When an entry is printed, DictEdit will rearrange and even edit its contents in such a way that it "looks good on paper". When creating the print format specification, the editor in chief has the following facilities:

- rearranging fields;
- mapping characters to another font for printing;
- defining the style for each element (italics, bold, etc.);
- editing the text of dictionary data, for instance: printing "irregular" when a field named "regular y/n" has value "no";
- adding numbers to specific sections of an entry, for instance, repeated fields/field groups.

3.6. Data format/use of SGML

The internal data format has a SGML compatible syntax. Although DictEdit does not operate on DTDs (document type definitions), it is possible to translate a DictEdit definition into a DTD (which can, in principle, be done automatically).

This implies that it is relatively easy to exchange dictionary data with other systems, especially if the receiving party has SGML tools. On the other hand, DictEdit is not able to read just any SGML document: DictEdit can only process documents that comply with its own internal definition.

4. Projects currently using DictEdit

At the moment, DictEdit is used in two major projects at the University of Amsterdam. The first is the creation of a new Modern Greek-Dutch bilingual dictionary, carried out by the Modern Greek Department. This project is in the stage of defining and refining the dictionary structure and testing the definition by entering a representative sample of entries.

The second is the completion of the Russian-Dutch dictionary on the basis of the result of the "inversion" of the Dutch-Russian as described by Honselaar and Elstrodt (fc.). This project is in the stage of converting this result to DictEdit's format and refining the structure definition.

As already stated above, DictEdit is not finished yet. We expect that experience from these projects will provide us with useful feedback for the further development of the program.

Endnotes

1 The AQUILEX project (ESPRIT BRA 3030) is a joint project of the Universities of Amsterdam, Barcelona, Cambridge (UK), Dublin and Pisa and Cambridge University Press, funded by the EEC.

2 This includes special sorting rules, such as spanish "ll" sorting between "l" and "m" and the ligature character "æ" which sorts like "ae".

3 One could think of defining the syntax of descriptions in terms of relations like "part of", "is a", etc. See Vossen, Meijs and den Broeder (1989).

4 It is possible to modify the dictionary definition. For existing entries, DictEdit tries to map them as well as possible, but if this fails, it creates an error field.

Bibliography

AKKERMAN, E., VOOGT-VAN ZUTPHEN, H. and MEIJS, W. (1988): A Computerized Lexicon for Word-Level Tagging: ASCOT Report No 2. Rodopi, Amsterdam.

BOGURAEV, B. AND BRISCOE, T. (1990): Computational Lexicography for Natural Language Processing. Longman Group UK Ltd., Harlow.

CLEAR, J. (1987): "Computing". In: Looking Up. An account of the COBUILD project in lexical computing. Ed. by J. M. Sinclair. Collins ELT, London.

HONSELAAR, W. and ELSTRODT, M. (fc.): "The electronic conversion of a dictionary: from Dutch-Russian to Russian-Dutch". To appear in the Proceedings of the Fifth Euralex International Congress.

LANDAU, S. I. (1989): Dictionaries: The Art and Craft of Lexicography. Cambridge University Press, Cambridge.

PROCTER, P. (1978): Longman Dictionary of Contemporary English. Longman Group Ltd., Harlow.

VAN DEN BAAR, A.H. (1989): Nederlands-Russisch Woordenboek. Kluwer, Deventer-Antwerpen.

VOSSEN, P. (1990): A Parser Generator for the Meaning Descriptions of the Longman Dictionary of Contemporary English. Technical Report NWO, project nr. 300-169-007. University of Amsterdam, Amsterdam.

VOSSEN, P., MEIJS, W. and DEN BROEDER, M. (1989): "Meaning and structure in dictionary definitions". In: Boguraev, B. and Briscoe, T.: Computational Lexicography for Natural Language Processing. Longman Group UK Ltd., Harlow, pp. 171-192.

KEYWORDS: software, computer-aided lexicography