**Abstract.** In natural languages, a semantically completely independent entity, settled word combination, often arises from combination of several words (e.g. a multi-word name of a municipal community or chemical compounds). Such entity is than called collocation.
In this work, we concentrate on description of a system for saving individual collocations that would be able to detect even the changes of inflection by means of rules. For our system we further use the term database of collocations.
We shall analyse the properties of the kinds of collocations described by us for Czech and suggest a formalism for the database of collocations. We shall describe algorithm by means of which it is possible to search the collocations with low time demands. By means of implementation the database of collocations and the described algorithm, a module shall arise that enables to improve the quality of written text, for example to identify the collocations in corpora. We have devised and implemented the database of collocations by a tool for interactive edition.

## 1 Kinds of collocations

As mentioned above, by a collocation we understand a combination of two or more independent lexical items (words). The entity created in this way, a new lexical unit, also often bares a new meaning, for example: "širý svět" (the entire world), "hodný kus cesty" (rather a long journey), ... In literature, collocations are often defined as multi-word lexical units. Although specialised conferences are being held on collocations[1], collocation has never been entirely rigorously defined.

A very good linguistic enumeration of types of collocations in Czech language could be find in v [Čer01], the author uses "steady state" as a basic criterion for the classification of collocations.

For the construction of the formalism to description of individual types of collocations the mentioned criterion does not have an essential meaning. Our focus is above all the heterogenity from the point of view of composition and variation of collocations. For further information see [Čer00].

As a collocation we will consider:

- idioms — expressions in the usage of a language that is peculiar to itself having a meaning that cannot be derived from the conjoined meanings of its elements: "neklame-li mne zrak" (if my eyes do not deceive).
- multi-word units used for:
  - names, pertinently including academical degrees: "prof. Tomáš Garrique Masaryk", ...
  - toponyms, the place-names of a region: "Karlovy Vary" (Carlsbad), "Bílé Labe" (White Elbe, Czech river), ...
  - terms (technical, linguistics, ... ): "fotoelekrický článek" (electric eye), ...
  - chrematonyms (proper names of human figments and institutions) "Matice česká" (Czech Fellowship)
- synsematics: "to co" (this, what), "až na to" (except that), ...
- (authorial) metaforic word's units: "mrzlo, až praštělo" (it was freezing cold), ...
- grammar-semantics word units: "letní dovolená" (summery holiday), "pionýrský tábor" (youth camp), ...
- analytical forms: "šel by" (he would go), "byl stanoven" (it was set), ...
- practice words units: "nastoupit do tramvaje" (get on tram), "nakrájet nadrobno" (cat up to small pieces), ...
- multi-word time entries: "včera večer" (in the yesterday evening), "17.červen 1979" (17-th july 1979), "První Máj" (First of May), ...

Some formalisms used for collocations description and storage are described in [And95,Eli,AAA+]. A system using a relational database designed for the particular needs of representing lexical collocation can be find [Kre00a,Kre98,Kre00b]. Some info is contained in the works about (extracting collocations from text corpora), see [Hei99,Uni,ISU96,Fra96].

[1] for example Computational Approaches to Collocations, July 22-23. 2002 Vienna

## 2 Properties of collocations

One of our aims is to devise and implement a tool for creating and correcting the *database of collocations*[2].

For additional work with the database of collocations it is essential to realise several basic properties of collocations:

- collocations are lexical units containing more than two components
- some collocations are inflecting "(bez) Karlových Varů" ((without) Carlsbad), "věšela mu bulíky na nos" (she was foll him), k "(k) Mont Blanku" ((to) Mount Blanck), "(počínaje)17.červnem 1979" (starting from the 17-th July 1979), ...
  - there are component in some collocations
  - flection of one element can be dependent on others
- sometimes elements can be abbreviated using one or more initial letters followed by tittle : "T.G.M.", "K. Vary", "ox. siřičitý" (sulphur dioxide), ("17.6.1979" will be considered as abbreviated form too), ...
- the meaning cannot be always derived from the conjoined meanings of its elements
- collocation corresponds to self-reliant semantic entity, so there is a need to to carry analogous attributes as other (one word) lexical units:
  - type: proper name, geographical unit, term, ...
  - domain: chemistry, medicine, linguistics, ...
  - sometimes semantic denotation, word class, grammatical unit, ...
- collocation elements can be variables, for example: *Brát* **něco** *na lehkou váhu* (make light of **sth.**), replacement of variables is restricted by valency constraints (rules)
- order of elements cannot by strictly fixed, included variable elements can be complicated structures

## 3 Database of collocations

Let us try than, for describing the *database of collocations*, to create a system of rules that would be possible to combine with the morphological database and that would cover the kinds of collocations described above (see Chapter 1). Our requirement to the constructed system, which would search the collocations saved in the database, is to provide a simple partial analysis of a text.

We define collocation as:

1. ordered set of nodes, where nodes are defined by following attributes:
   - `morf_id` — then number, unambiguously corresponding to the base form (lemma), extended by identification of it's semantic sense and determining pattern for the inflectional process
   - fixed part of the morphological tag — set of tuples attribute + value

[2] a term *dictionary of collocations* is also used

   - attributes
     - var — extension enabling definition of the variable elements
     - short — says if this element can/cannot be abbreviated
2. set of restricting rules (relations among the elements):
   - juice or some other relation among attributes of the morphological tags
   - relationship among positions (giving their physical ordering) in a analysed text
3. pertinently pointer (reference) to new sense (of whole collocation)
   - designed for connecting with external lexikons, glossaries, encyclopedias, ...
   - the attributes (see chapter 2) are placed at the referenced place
   - such a reference enables usage of this entity (in the next collocation)

We assume that the devised system can by with slight changes used also for the word formation process *compounding of words*. To do so, it is essential to extend the system of a tag indicating the fact whether there is/is not a word hyphen (gap). It is also necessary to introduce the system of nonterminals which would provide, in combination with regular grammar, recursive chaining of individual units of the entity. As an example of usage of the formalism extended of nonterminals, it is possible to cite the formation of compound numerals.

## 4 Algorithm for detection of collocations

If a suitable searching algorithm is added to the database of collocations, we will be able to search collocations in the presented entry (in the documents of corpora, eventually in any written text). The mentioned searching of collocations than facilitates a computer analysis of a written text or computer assisted translation.

We decided to use the following method for effective searching so generally described collocations in machine-readable texts:

1. divide input into tokens (tokenize) by patterns based on regular expressions:
   - the token can be:
     - a continuous chain of alphanumeric characters from the alphabet of given language
     - repetition of one arbitrary non alphanumeric character , for example: ".", "..."
   - any chain composed from so called "white chars"[3] are omitted and division into tokens is enforced in those places
   - between tokens, which was not created by division from the input text in the places of omitted "white chars", we insert special structure tag `<g>` (glue)[4]
2. we try to find all possible base forms (lemmas) and applicative morphological tags for any token using morphological analysis (for example programme `ajka` or library `alib`, see [Sed99]); we determine corresponding values `morf_id` (see Chapter 3)

[3] space, tabulator, new line, carriage return

[4] see example at the end of the chapter 4.1

3. we lookup candidates for collocations in given interval of tokens, we have to search all possible combinations of tokens containing relevant values of `morf_id` desiderative in some collocation stored in the database (it is possible to do that effectively using structures and algorithm described in chapter 4.1)
4. from the set of found candidates we must skip collocations, which do not match *restricting rules* (see chapter 3)
5. we determine the resultant morphological tag (determination of the appropriate result lemma will be good idea in the case of collocation with variable elements)

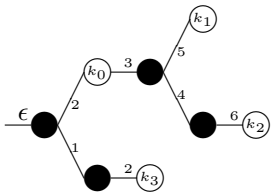
### 4.1 Finding the candidates for collocations

This algorithm requires unique numeration of all basic forms (lemmas) including the paradigm used for inflection and the meaning, which is obvious from the example: *jeřáb popelavý* (grus grus), where *jeřáb* is a bird not a machine (crane) or a tree (mountain ash).

**Creating the searching structures** We shall perform the searching separately for continuous collocations with fixed succession of units and separately for the others. In both cases, we shall create a tree structure *trie* (see [Knu73]) $S_1$ a $S_2$, where separate edges in these structures are composed of `morf_id` numbers which unequivocally correspond to the lemmas (extended by appropriate paradigm of inflection, respectively by identification of the meaning). Into the node corresponding to the end of such chain (*endpoint node* of the path from the tree root in the relevant structure), we add a reference to the corresponding collocation.

**Example:**

$L_{k_0} \equiv [2]$
$L_{k_1} \equiv [2, 3, 5]$
$L_{k_2} \equiv [2, 3, 4, 6]$
$L_{k_3} \equiv [1, 2]$




The chain $L_{k_i}$ of numbers `morf_id` corresponds to list of elements of collocation $k_i$.

For all collocations $k_1$, $k_2$, ...

- for the first type of collocation, we insert chain $L_{k_i}$ = [ `morf_id`$_1$, `morf_id`$_2$, ... ] into the structure $S_1$
- for the second type
  1. sort elements of the collocation by `morf_id`
  2. we insert chain sort($L_{k_i}$) = [ `morf_id`$_{s_1}$ < `morf_id`$s_2$ < ... ] into the structure $S_2$

**Searching** Input for the searching algorithm are:

- pre-built structures $S_1$, $S_2$
- ordered set of tokens (FIFO) created from an input text $P \equiv [p_1, p_2, \ldots]$
- and ordered set of pointers to results of morphological analysis $M \equiv [m_1, m_2, \ldots]$.

Token $p_i$ is represented by structure with following attributes:

- `w` — text of token (word form)
- `ll` — set of values $l_{i,j}$ of the `morf_str` type (see below) acquired by morphological analysis of the word form $p_i$.`w`

We will use following structure `morf_str` for the representation of the morphological analysis results. The attributes of `morf_str` are:

- `morf_id` — identification of the base form (lemma, pattern, sense)
- `father` — pointer to the associated[5] token
- `tt` — list of applicative morphological tags

The set of pointers $M$ is sorted owning to the attribute `morf_id` ($\mathtt{m}_1\mathtt{.morf\_id} < \mathtt{m}_2\mathtt{.morf\_id} < \ldots$)
We can use three operations during the analysing process:

1. adding of the next token $p_{n+1}$ at the end of the list $P$ — this operation among others accomplish morphological analysis of the word form $p_{n+1}.w$ and fill the list $p_{n+1}.ll$ (it was empty till the time), at the end the pointers to the members of $p_{n+1}.ll$ are added into the list $L$
2. relaxation of the first token from $P$ — moreover the pointers to the members of the list $p_{n+1}.ll$ are released from the list $L$
3. analysis
   (a) we trace (gradually) the paths in $S_1$ corresponding to chains $H \equiv [h_i | i \in \{0 \ldots n\} : h_i \in p_{j+i}.ll]$ ; if we reach the endpoint node, we have found a candidate for collocations (applicative to given to endpoint node), the candidate is formed with positions $[p_j, \ldots, p_{j+n}]$
   (b) we trace (gradually) the paths in $S_2$ corresponding to chains $H \equiv [h_i | i \in \{0 \ldots n\} : h_i \equiv m_{j+i} \in M]$ ; if we reach the endpoint node, we have found a candidate for collocations (applicative to given to endpoint node), the candidate is formed with positions $[h_0.father, \ldots, h_n.father]$

When implementing the structures of the *trie*, it is good to realise that the number of successions to the root will probably be sharply higher than the number of successions to the other nodes. In any case, this value is limited by the maximum degree of the `morf_id` values.

[5] this object was created by analysis of referenced ed token

**Abbreviated units of collocations** Searching in accordance to the given algorithm is heavily complicated by the fact that the units of some collocations might have abbreviated forms. One of the possible solutions to this problem is to create a dictionary of all possible forms of abbreviations occurring in collocations, and than to each abbreviation (in the entry) assign a set of accessible forms of abbreviations. With the mentioned process, we would gain, though, too vast number of possible forms, especially for one-letter abbreviations, which would disproportionally increase the time demandingness of the entire searching. That is why we shall create another structure of the trie (lets inscribe it $S_3$) in the similar way to the structure $S_2$ (see Chapter 4.1). In the endpoint nodes $t$ of structure $S_3$ shall not only be references to the particular collocations, but also other trees of the *trie* $S_{3,t}$ (edges are composed of the letters of alphabet, endpoint dots correspond to the possible abbreviated forms).

For adding the chain into $S_3$:

1. we sort the elements of collocations ascendent according to `morf_id`
2. we insert into the structure $S_3$ the chain
   $\text{sort}(\text{L}_{k_i}) = [\ \texttt{morf\_id}_{s_1} < \texttt{morf\_id}s_2 < \ldots \mid \texttt{morf\_id}_{s_1}\texttt{.short} \neq 1\ ]$
3. endpoint node $t$ we add to $S_{3,t}$ gradually all lemmas corresponding to individual member of the list $[\ \texttt{morf\_id}_{s_1}, \texttt{morf\_id}s_2 < \ldots \mid \texttt{morf\_id}_{s_1}\texttt{.short} \equiv 1\ ]$

The searching of candidates including the tags we accomplish this way

1. while reading the input we gradually keep the list of abbreviated forms $X$
2. if $X$ is not empty: we accomplish searching in the list $L$ according to the structure $S_3$; in all found endpoint nodes $t$ we continue searching of members from the list $X$ in the structures $S_{3,t}$; we mark the applicable positions including the abbreviations $X$ by means of lemmas found in $S_{3,t}$ (replenishment of the abbreviations is given by the chains contained in the subtree of $S_{3,t}$, the subtree determined by worded part of the abbreviation)
3. we search the positions (with new marked abbreviated forms) by means of structures $S_1$ a $S_2$
4. we check the rules for found candidates
5. we determine resultant tagging

In other words: in the first passage we can find candidates for collocations that might contain abbreviations; we shall develop these abbreviations and continue in accordance with the original process.

From the formal point of view it is necessary to add that an abbreviation is formed with three[6] positions:

1. with the initial letters of the paced word
2. with the structural `<g>` to mark the coherence

[6] respective with two positions and one structural tag

3. with the character "." (dot) marking that is probably an abbreviation, but it could be an ending of the sentence also

When marking the abbreviations it is essential that the system for dividing the entering text to positions (tokenize) is rather reorganised. We shall add virtual positions; in the place of an abbreviation two variants shall arise and instead of linear chain, it is necessary to browse a directed graph.

**Example:**

| | | | |
|---|---|---|---|
| Vstup: | P. Veliký přijel do Karlových Varů. | | |
| Tokens | \| P \| <g> \| . \| Veliký \| přijel \| do \| Karlových \| Varů \| <g> \| . \| | | |
| Morf. anal. | P | | |
| | Veliký | veliký | k2eAgMnSc[15]d1, k2eAgInSc[145]d1 |
| | přijel | přijet | k5eAp[MI]nStMmPaP |
| | do | do | k7c2 |
| | Karlových | Karlův | k2eAg[MINF]nPc[26] |
| | do | do | k7c2 |
| | Varů | Vary | k1gInPc2 |
| Determination of $X$ | "P.", "Vary." are candidates | | |
| Abbreviation Correction | for "Veliký" found "P." which could be shortening of the word "Petr"<br>a triplet of positions \| P \| <g> \| . \| → \| P. \|<br>could be an abbreviation of: "P." and its lemma could be"Petr" | | |
| Searching of candidates | collocations found "Petr Veliký", "Karlovy Vary" | | |

## 5 Conclusion

In the direct consequence to the morphological analysis, we have created the formalism that enables maintaining the database (dictionary) of collocations. The described formalism has been introduced with reference to the Czech language but can be easily applied even to other languages. We have outlined the algorithm for effective searching of collocations in the corpora or in the analysed texts. The low time complexity is preserved in spite of the possible shortening of individual units by abbreviations, also those collocations that are not contiguous in the text (collocations with nested elements) are permitted. The system selected can be applied also to generating; furthermore, we can inflect particular collocational entities by virtue of constraint rules. For the Czech language, an already functioning testing database has been developed that can be easily extended by means of interactive editor.

## References

[AAA+] I. Aduriz, J. Aldezabal, X. Artola, N. Ezeiza, and R. Urizar. Multiword lexical units in euslem, a lemmatisertagger for basque.

[And95] Fr'ed'erique Segond And. Using a finite-state based formalism to identify and generate multiword expressions, 1995.

[Eli] Fr'ed'erique Segond Elisabeth. Idarex: Formal description of german and french multi-word expressions with finite state technology.

[Čer00] F. Čermák. Combination, collocation and multi-word units. In *Proceedings of the Ninth Euralex International Congress 2000*, pages 489–495, Inst. fuer maschinelle Sprachverarbeitung, Universitaet Stuttgart, 2000.

[Čer01] F. Čermák. Syntagmatika slovníku, typy lexikálních kombinací. In P. Karlík Z. Hladká, editor, *Čeština - univerzália a specifika*, volume 3, pages 223–232, Brno, Czech Republic, 2001. Masarykova univerzita v Brně.

[Fra96] K. Frantzi. Extracting nested collocations, 1996.

[Hei99] U. Heid. Extracting terminologically relevant collocations from german technical texts, 1999.

[ISU96] S. Ikehara, S. Shirai, and H. Uchino. A statistical method for extracting uniterrupted and interrupted collocations from very large corpora, 1996.

[Knu73] Donald E. Knuth. *The Art of Computer Programming: Sorting and Searching*. Addison-Wesley, 1973.

[Kre98] Brigitte Krenn. A Representation Scheme and Database for German Support-Verb Constructions. In *Proceedings of KONVENS '98*, Bonn, Germany, 1998.

[Kre00a] B. Krenn. CDB – a database of lexical collocations. In *Proceedings of the 2nd International Conference on Language Resources & Evaluation*, Athens, Greece, 2000.

[Kre00b] B. Krenn. Collocation mining: Exploiting corpora for collocation identification and representation, 2000.

[Sed99] Radek Sedláček. Morfologický analyzátor češtiny. Master's thesis, Fakulta informatiky Masarykovy university, Brno, 1999.

[Uni] Brigitte Krenn Universitat. Acquisition of phraseological units from linguistically interpreted corpora a case study on german pp-verb collocations.