

Inflectional Morphology, Reverse Similarity and Data Mining – Finding and Applying Compact and Transparent Descriptions of Verb Systems of Natural Languages

Alfred Holl
Fakultät Informatik
Georg-Simon-Ohm-Hochschule Nürnberg
Postfach 210320, 90121 Nürnberg
Germany
Alfred.Holl@ohm-hochschule.de



ABSTRACT: Under the term “data mining”, the field of computer science includes many different techniques for data analysis, among them methods of cluster analysis. In the approach presented, a special method is designed for the analysis of inflectional systems. The algorithm is independent of individual natural languages and parts of speech. It finds two types of clusters: morphologically homogeneous ones, which contain reversely similar (which possess the same trailing letters), morphologically analogous lexemes of the examined language-part-of-speech combination, and morphologically inhomogeneous ones, in which the largest part of the lexemes is morphologically homogeneous. The resulting registers are compact and transparent as well as easily extensible and correctible. In condensed form, they provide linguistically and didactically usable, structural results on inflectional systems. For instance, it is possible to assign arbitrary lexemes to clusters with a detailed explanation based upon the structure of an inflectional system. The approach is most often applied to verb systems in inflecting and agglutinating languages.

Key words: Analogy, Aphasia Rehabilitation, Cluster, Conjugation, Data Mining, Inflection, Inflectional Class, Inflection Form, Inflection Type, Inflectional Feature, Inflectional System, Inflectional Morphology, Language Learning, Language Teaching, Morphological Analogy, Morphology, Pattern Verb, Principal Part, Reverse Order, Verb

Received: 9 January 2011, Revised 18 February 2011, Accepted 2 March 2011

© 2011 DLINE. All rights reserved

1. Introduction and Overview

This research approach combines data mining and structural linguistic analysis of inflectional systems.

Linguistic data analysis is nothing new. When one establishes morphological, syntactical or phonological rules, data analysis is necessary to analyze linguistic material (texts, grammar books, dictionaries and especially inflectional systems). Every existing list of irregular verbs, for example, is the result of a manual data analysis.

My approach, however, attempts to put the analysis of inflectional systems on a consistent formal and automated platform using methods of computer science. For that purpose, I recur to my analytic algorithm, independent of individual languages and parts of speech, first published in Holl / Behrschmidt / Kühn 2004, II.55-II.73, improved in Holl / Suljiæ2010, 54-57, a previous version in Holl 1988, 183-184.

The algorithm was successfully used for the examination of a couple of language-part-of-speech combinations in Indo-European languages: the noun system of Swedish (2007) and the verb systems of Latin (1988), Catalanian (1988), Portuguese (1988), Rumanian (1988), Italian (1988, 2002), Spanish (1988, 2002), French (1988, 2002, 2003), German (2002, 2004), Russian (2004), Greek (2006), Ancient Greek (2006), English (2002, 2007), Swedish (2002, 2007) and Croatian (2010). I focus mostly on verb systems, as they are a lot easier to describe than noun systems.

The basic assumptions and strategies of my approach remained the same since the 1980s although I had to adjust some parameters during the course of time and learnt some new aspects with each language-part-of-speech combination I examined. So when looking in one of my earlier publications in this field, you will find slight differences compared with the more mature and more elaborate version of my approach presented here. This is especially due to the recent exact investigation of the requirements to automatically assign arbitrary lexemes to clusters in Holl / Zimnik 2009.

This paper is intended to be easily readable for all kinds of researchers in the area of computer linguistics. Therefore, I start with some introductory definitions regarding inflection (Section 2). My approach is motivated by the discussion of analogical reasoning strategies based upon the reverse similarity of lexemes (Section 3). The goal is to find linguistically interesting clusters which contain reversely similar lexemes. Possible types of clusters are presented in Section 4. In order to apply my approach, I have to take some principal decisions independent of the language-part-of-speech combination examined (Section 5). In the analytic part, the divisive cluster algorithm, which automatically structures pre-processed linguistic material, is presented. The result of the algorithm is manually post-processed and improved to generate a register which contains different types of clusters (Section 6). In the synthetic part, a search algorithm is designed which uses the register as a collection of rules in order to automatically detect which cluster some arbitrary lexeme belongs to (Section 7). The paper terminates with some perspectives on future research work (Section 8).

In this paper, I will omit some details of my approach and concentrate on its main ideas and procedures. Examples from the English verb system will be presented in order to ensure broader understanding, although modern English has only a reduced inflectional system.

2. Basic Terminology

Linguists use the term lexeme as label for a “word” in a lexicographic sense. A **lexeme** is an abstract basic unit of lexicography which can occur in different inflection forms. In a dictionary entry, a lexeme is represented by the one of its inflection forms, which is considered as most important, namely its **lexical base** (lemma). The lexical base of a lexeme is used as the “name” with which a lexeme is referred to. In the case of a verb, its lexical base is – depending on the language – its present infinitive of active voice, briefly infinitive, or its 1st person singular of present tense. In addition, I will use the term **lexeme ending** as an abbreviation of the term **ending of the lexical base of the lexeme** by identifying a lexeme by its lexical base.

Applying the above definition, **inflection** means the rule-based modifications of lexemes in order to mark certain inflectional categories. Inflection is an umbrella term comprising declension and conjugation. **Declension** comprises the modifications of nouns, pronouns and adjectives to indicate case, number and gender; **conjugation** those of verbs to indicate tense, mode, voice, person and number. **Inflectional morphology** (in this paper briefly morphology) is the linguistic discipline which deals with inflection.

The phenomenon of inflection can be found in inflecting and agglutinating languages. In inflecting languages, one can split a word form into a stem which carries the semantic meaning (and sometimes also a part of the inflectional meaning) and an ending which carries the largest part of the inflectional meaning. In agglutinating languages, the splitting of meanings is a lot clearer, as uniting several components of the meaning on one morpheme rarely occurs. The border between inflecting and agglutinating languages can often not be drawn exactly.

Morphological analogy of two or more lexemes means that they possess the same morphological properties, that is, the same inflectional features. In other words, their inflection forms possess the same structure and the same components. When an inflection form of a lexeme is known, the corresponding one of an analogous lexeme can be derived using analogical reasoning, e.g. the inflection forms of the verb *cling* can be derived from those of the verb *fling*.

The set of the inflectional features of a lexeme is called its **inflection type**. The ordered set of the inflection forms of a lexeme is its **averbo**.

Infinitive forms		Present infinitive		Imperative	
		Gerund Present participle			2 nd sg
		Past participle			2 nd pl
Present tense	1 st sg				1 st sg
	2 nd sg				2 nd sg
	3 rd sg				3 rd sg
	1 st pl				1 st pl
	2 nd pl				2 nd pl
	3 rd pl				3 rd pl
Past tense	1 st sg				1 st sg
	2 nd sg				2 nd sg
	3 rd sg				3 rd sg
	1 st pl				1 st pl
	2 nd pl				2 nd pl
	3 rd pl				3 rd pl




-  inflection forms derived from the infinitive
-  inflection forms derived from past tense 1st sg
-  inflection forms derived from past participle

Figure 1. Stem distribution of the English verb (Holl / Maroldo / Urban 2007, 94)

Given a lexeme, one can choose about half a dozen “main inflection forms”, traditionally called **principal parts**, the first of which is the lexical base. The principal parts serve as a basis to derive all the other inflection forms of the lexeme’s averbo using derivation rules. There are only very few exceptions where derivation does not work: “strange” inflection forms, such as *I am*, *you are*, *he is* etc., cannot be derived from principal parts with derivation rules. They have to be listed separately and learnt as exceptions with every verb.

The concept of principal parts can be generalized to entire language-part-of-speech combinations. Within a given one, the choice of inflectional categories of the principal parts can be made in a way so that it meets two conditions:

1. The choice is invariant for the entire language-part-of-speech combination, that is, the inflectional categories of the principal

parts are equal for every lexeme, e.g. present infinitive, past tense, past participle are the inflectional categories for all of the English verbs.

2. Independently of a given lexeme, a certain inflection form is always derived from the same principal part usually with a fairly simple, at least a not too complex decatenation and concatenation rule.

The value of this fact is even more evident in languages with more complex inflectional systems, such as in Romance languages (cf. Holl 1988).

Every principal part represents a stem in a formal sense, that is, the remaining rest of the principal part when the ending is cut off. As a principal part is used as the basis to derive a certain invariant set of inflection forms within a language-part-of-speech combination, one can also speak of an invariant stem distribution, which can be represented with a differently shadowed empty *averbo* table as shown in Figure 1 for the English verb.

3. Motivation

Up until now, detailed presentations of inflectional systems (of specific language-part-of-speech combinations) are products of manual cluster analysis. Roughly spoken, clusters in terms of data mining mean sets with two qualities: elements within a set have similar properties, elements of different sets have different properties (cf. Ester / Sander 2000, 45). Clusters of morphologically analogous lexemes are established if all of the lexemes with the same inflectional features are listed together with every pattern lexeme or if, in a complete register of all of the lexemes of a language-part-of-speech combination, the identifier of its pattern lexeme is indicated together with every lexeme.

But what is the advantage of such lists of morphologically analogous lexemes from a didactic point of view? They are so large, unstructured and unmotivated that it is very hard to learn them by heart and almost impossible to completely memorize them, as additional structural information is not taken into consideration.

Let me say it in other words: Undoubtedly, it is possible to construct registers to collect the information about the inflection types of a language-part-of-speech combination and about the assignment of lexemes to inflection types. These registers are efficient if you just want to look up the inflectional features of some arbitrary lexeme without desiring any further explanation. This technique is used by spelling verifiers and verb dictionaries, some of which are mentioned in the references.

These registers, however, do not contain any linguistic structural information. They describe an inflectional system without explaining it. Therefore, they do not help linguists to understand it. As a consequence, they do not facilitate language learning and teaching.

In contrast to this traditional technique, my approach examines inflectional systems using additional structural information in the form of the reverse similarity of lexemes (details later in this section) in order to establish clusters of morphologically analogous ones. It is obvious that I have to describe an inflectional system as well, but I arrange the description in a way that it becomes shorter, better motivated, more transparent and easier to learn. Given a lexeme, you see which inflectional properties similar lexemes possess and how and why it is assigned to a certain pattern (in the form of a cluster) and not to another one. That way, my approach has an explanatory value. Furthermore, where verb registers need 200 pages, I often arrive at only 20 pages. How did I find the basic ideas of my approach? I was inspired by the description of conjugation in Latin and Romance languages and by strategies, success and errors of language learners, by mistakes of children and by hypercorrectisms of native speakers. So, let me return to language learners. How do they try to solve the problem of dealing with lexeme lists? In order to minimize their learning effort, they use strategies based upon analogical thinking. They would prefer to get the inflectional features right from the lexical base, just as one actually recognizes the inflection type of regular verbs in Romance languages by looking at their infinitive endings. Once an *averbo* (often in the reduced form of principal parts) has been learned with much effort, the learner attempts to get the most of his knowledge and tries to extend it to other “similar” lexemes which he “condemns” to having the same inflectional features. As *tertium comparationis* (basis of comparison) to “detect” similarities, the reverse similarity is chosen.

Reverse similarity can be seen in a common ending of lexical bases. In this sense, the term “ending” is not used as a morphological category, but simply means a sequence of letters at the end of a word. The length of an ending is defined pragmatically depending on the particular case. An *n*-digit ending is defined as an ending of the length *n*, that is, the trailing *n* letters of a lexical base.

An example for the learning strategy mentioned (analogical transfer based upon reverse similarity) can be shown with the English verbs *link*, *blink* and *slink*. They have similarities in the last 4 letters, in their ending *link*. If a learner of a language would learn the word *link* with its other principal parts (*linked*, *linked*) as the first one of this group, he would want to analogically transfer its regularities to the two other ones. This transfer is correct for *blink* (*blinked*, *blinked*), but not for *slink* (*slunk*, *slunk*). The latter has to be memorized as an explicit exception of the verbs with the ending *link*.

This strategy is correct for every basic lexeme and its prefixed lexemes (with very few exceptions), for regular lexemes and a part of the irregular ones, but it is not generally correct, which results in a considerable source of errors. This strategy can be helpful as well as misleading. Only when it fails, will the learner intensively memorize further lexemes with their inflectional features in a second step.

There are other facts which support the phenomenon of reverse similarity as basis of a learning strategy (and of my approach):

1. Rule-based particularities of otherwise regular verbs can depend on the ending of a lexeme: *e*-insertion in the 3rd person singular of present tense for English verbs ending in *-ch*, *-sh*, *-Co*, *-ss*, *-x*, *-Cz* (where *C* means a consonant grapheme).
2. There is a general tendency to simplify inflectional systems. Irregular forms are only stable if they are used very often. Otherwise parallel regular forms arise. As a consequence, regular inflection features are spread.
3. Only regular inflection types are productive, that is, new lexemes (neologisms) will never be irregular.
4. Basic conjugation types in Latin and Romance languages are defined by the lexeme ending.
5. End rhymes in poems support the reverse perspective although they are not directly connected to inflection.

Let me summarize the situation outlined above. In the context of inflection, two competing forms of similarity occur: the reverse similarity of the lexical base (an alphabetic property) and the morphological analogy. They are not at all identical and, within a part of speech, one cannot draw conclusions from the reverse similarity of two lexical bases onto the morphological analogy of the corresponding lexemes. The analogical reasoning (conclusion from partial to complete similarity) which underlies this strategy is a common, essential – often unconscious – principle of human learning and thinking. Therefore, it cannot simply be switched off. The learner as well as the teacher of a language has to deal with it consciously. It is best to show the learner of a language from the very beginning in which cases reverse similarity implies morphological analogy and in which ones it does not. This is shown in Holl 2002, 151-158 in detail. Furthermore, there are recent studies on how children use analogical thinking to learn languages (Friederici 2010, 68-69).

4. Goals

Regarding the background of the preceding motivation, it is a promising approach to include the aspect of reverse similarity of lexical bases (as a seduction for analogical thinking) in the description of inflectional systems. That way, one will find structures which make it clearer and more transparent. The structures carry important linguistic information which helps linguists, language teachers, language learners and aphasia patients in their work of understanding, explaining and learning these systems. The structures are clusters (in the sense of data mining) with equal or mostly equal morphological properties.

Before I can describe the clusters in detail, I have to take the first decision of my approach, the one for the representation of linguistic objects. I focus on a graphemic and synchronic view due to its high practical relevance and to its independence of theoretic assumptions necessary in the case of a phonemic representation (cf. the detailed argumentation in Holl 1988, 171, and Kempgen 1989, 4, summarized in Holl / Maroldo / Urban 2007, 26-27). Furthermore, a language learner can use the results without dealing with problems of phonology and language history.

I distinguish the following types of clusters:

A morphologically **homogeneous cluster** (abbreviated homC) is a mathematically connected set (does not have any gaps or interruptions) in a reversely ordered sequence of morphologically analogous lexemes.

A morphologically **inhomogeneous cluster** is a connected set in a reversely ordered sequence of lexemes whose morphological properties are different (at least one lexeme has different properties as all the other ones).

A **basic cluster** (abbreviated basC) is an inhomogeneous cluster, in which most of the lexemes have the same morphological

properties. The threshold percentage of the majority, i.e. the probability that an arbitrary lexeme of this cluster belongs to the majority, can be chosen depending on linguistic reasons. In general, one can say that an inhomogeneous cluster with more than 70% morphologically analogous lexemes can already be interesting from a linguistic point of view and be worth defining a basic cluster. Basic clusters often correspond to traditional inflectional classes.

In my approach, every cluster is formally represented as an **abstract lexeme**, quoting its abstract lexical base (that is, the first principal part) and its other abstract principal parts together with its inflection type (cf. the following examples). In analogy to lexemes (cf. Section 2), the lexical base of a cluster is used as its “name”, and the term **ending of the lexical base of the cluster** is abbreviated to **cluster ending**, identifying a cluster with its lexical base.

Regarding reverse similarity, the lexical base of a cluster is always identical to an alphabetic property which the endings of the lexemes belonging to the cluster have to possess. In other words, comparing a lexeme with a cluster always means comparing the lexeme’s lexical base with the cluster’s lexical base.

Except for clusters with only one element, called one-lexeme clusters, the graphemic description of a cluster (that is, of its lexical base) always contains joker characters in the sequence ~ or #, grapheme type character(s), normal character(s):

~ represents an arbitrary sequence of characters only at the beginning of a cluster name.

marks the left word limit and is only used together with at least one immediately following grapheme type character.

C is used as a grapheme type character and represents a consonant grapheme.

V is used as a grapheme type character and represents a vowel grapheme.

Clusters with grapheme type characters (C, V) are abbreviations for sets of clusters. They are mathematically non-connected sets, as the alphabetical sequence of consonant graphemes is always interrupted by vowel graphemes (and the other way round), e.g. *a* is a vowel grapheme, *b, c, d* are consonant graphemes, *e* is a vowel grapheme, *f, g, h* are consonant graphemes, and so on.

The following clusters from the English verb system are described as abstract verbs mentioning their principal parts: infinitive (past tense, past participle).

Basic clusters are always many-lexeme clusters. They appear in three types:

1. Basic clusters without grapheme type characters and without word delimitation:

Example: ~*ch* (~*ched*, ~*ched*); all verbs whose infinitives end in *-ch*. Most of the verbs are regular with *e*-insertion in the 3rd person singular of present tense whose ending is extended to *-es*.

Elements: *reach* (*reached*, *reached*) etc.

Exceptions: *teach* (*taught*, *taught*); *catch* (*caught*, *caught*).

2. Basic clusters (precisely: sets of basic clusters) with grapheme type characters and without word delimitation:

Example: ~C (~*Ced*, ~*Ced*); all verbs whose infinitives end in a consonant grapheme. Most of the verbs are regular.

Elements: *look* (*looked*, *looked*) etc.

Exceptions: *bring* (*brought*, *brought*) etc. There are many exceptions, but statistically only a few, as the cluster comprises all of the innumerable regular verbs ending in a consonant grapheme.

3. Basic clusters (precisely: sets of basic clusters) with grapheme type characters and with explicit word delimitation:

Example: #CVg (#CVgged, #CVgged); all basic verbs whose infinitives have the form consonant grapheme plus vowel grapheme plus *g* and the prefixed verbs of these basic verbs. Most of the verbs are regular with consonant grapheme doubling in past tense, past participle and gerund.

Elements: *jig* (*jigged*, *jigged*) etc.

Exception: *dig* (*dug*, *dug*).

There are four types of homogeneous clusters:

1. Many-lexeme homogeneous clusters, without grapheme type characters and without word delimitation:

Example: *~ling* (*~lung*, *~lung*); all verbs whose infinitives end in *-ling*. All of them have the same irregularities.

Elements: *cling* (*clung*, *clung*); *fling* (*flung*, *flung*); *sling* (*slung*, *slung*).

2. Many-lexeme homogeneous clusters (precisely: sets of homogeneous clusters) with grapheme type characters and without word delimitation:

Example: *~Cz* (*~Czed*, *~Czed*); all verbs whose infinitives end in a consonant grapheme plus *z*. All of the verbs are regular with *e*-insertion in the 3rd person singular of present tense whose ending is extended to *-es*.

Elements: *jazz* (*jazzed*, *jazzed*) etc.

3. Many-lexeme homogeneous clusters (precisely: sets of homogeneous clusters) with grapheme type characters and with explicit word delimitation:

Example: *#CCVg* (*#CCVgged*, *#CCVgged*); all verbs whose infinitives have the form two consonant graphemes plus vowel grapheme plus *g*. All of the verbs are regular with consonant grapheme doubling in past tense, past participle and gerund.

Elements: *drag* (*dragged*, *dragged*) etc.

4. One-lexeme homogeneous clusters with implicit word delimitation and without any joker character; they consist of one single basic verb and its prefixed verbs. They have to be dealt with from a formal perspective although they are not interesting from a linguistic one.

Examples: *dig* (*dug*, *dug*); *bring* (*brought*, *brought*) etc.

Clusters can be nested hierarchically and thus represented in the form of trees. Basic clusters can contain basic and / or homogeneous sub-clusters. This situation is illustrated for a part of the English verbs in Fig. 2 using examples mentioned above (cf. Holl / Maroldo / Urban 2007 for the English examples and Holl / Zimmnik 2008, 12).

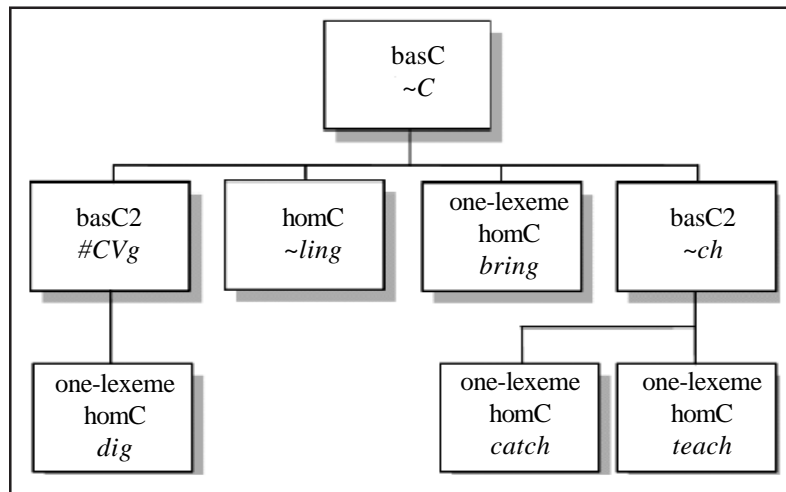


Figure 2. Part of a cluster tree for English verbs

The mostly regular basic cluster *~C* (*~Ced*, *~Ced*) contains (among others):

1. The mostly consonant grapheme doubling basic cluster *#CVg* (*#CVgged*, *#CVgged*) which contains the irregular one-lexeme cluster *dig* (*dug*, *dug*).

2. The consonant grapheme doubling homogeneous cluster *#CCVg* (*#CCVgged*, *#CCVgged*).

3. The irregular homogeneous cluster *~ling* (*~lung*, *~lung*).

4. The irregular one-lexeme cluster *bring* (*brought*, *brought*).

5. The regular *e*-inserting basic cluster *~ch* (*~ched*, *~ched*) which contains the irregular one-lexeme clusters *catch* (*caught*, *caught*) and *teach* (*taught*, *taught*).

In order to design a well-defined cluster tree, a linguist has to pay attention to the following requirement: it must be possible to unambiguously assign a search lexeme to a cluster. This requirement is met if the two sets of lexemes representing two clusters are either disjoint or in a superset-subset relation. The lexical base of a parent cluster always has to contain fewer or as many characters than the lexical bases of its child clusters. In the case of an equal number of characters, the lexical base of a parent cluster has to contain more grapheme type characters. That is, regarding lexical bases (alphabetic properties) as conditions: the condition of a parent cluster is always weaker than the conditions of its child clusters. One-lexeme clusters can only appear as leaves of a cluster tree. This argumentation is needed for the synthetic part of my approach and will be continued in Section 7. Furthermore, it is important to remark that nested clusters can have different morphological properties. Nested basic clusters are abbreviated basC2 (level-two basic clusters), basC3 etc. according to their level in a cluster tree.

5. Principal Decisions

In order to efficiently apply my approach to specific language-part-of-speech combinations, I have to take several decisions (cf. Holl / Maroldo / Urban 2007, 27-34, 36-37 and 42-44).

1. Graphemic representation of lexemes (explained above in Section 4).

2. Prefix treatment: prefixes are detached using a prefix marker (underscore *_*) as prefixed verbs have to be put together with their basic lexemes in a reversely sorted lexeme list, because they in most cases possess the same morphological properties. E.g. *in_lay*, *under_lay*, *mis_lay*, *out_lay* should appear immediately after *lay* in a reversely sorted register and not somewhere between *flay*, *play*, *parlay*, *slay* etc., which would happen without a prefix marker. Prefixes are pragmatically marked according to a semantic-synchronic prefix definition: A lexeme is called prefixed if an educated, linguistically untrained native speaker spontaneously understands it as a synchronically transparent composition. That is, if it is interpreted as a prefixed lexeme due to the similar meaning and alphabetic shape of the corresponding basic lexeme.

3. Lexemes with two inflection types (**inflectional variants**) are treated as different lexemes. If a variant is combined with a semantic difference, numbers are attached at the beginning of the lexical bases and the meanings are mentioned, e.g. *llie* (*lied*, *lied*) ‘make an untrue statement’ vs. *llie* (*lay*, *lain*) ‘be in a horizontal position’. If there is no semantic difference between the variants, small Greek letters are attached, e.g. *á+knit* (*knitted*, *knitted*) vs. *â+knit* (*knit*, *knit*). This pattern of marking is adopted from dictionaries.

4. Delimitation to the syntax: lexemes can have synthetic (one-word) and analytic (many-word) inflectional forms. The latter are omitted in my approach due to the following reasons (cf. Kempgen 1989, 3, and Holl 1988, 175). Regarding nouns, analytic case forms are constructed with a preposition and a synthetic inflection form. Regarding verbs, analytic forms consist of a synthetic infinite verb form (past participle, *ing*-form) and a finite verb form of an “auxiliary verb”, which in turn is recorded as a verb, e.g. *I have found*. If the auxiliary verb form is analytic in turn, it must be split up in an analogical way, e.g. *I have been found*.

5. Delimitation to the lexicon: the final registers do not offer complete listings of lexemes of a certain language-part-of-speech combination. They are reduced to morphologically interesting material. Defective lexemes, i.e. the ones that cannot form all of the inflectional forms, are not treated in a specific way. Unusual or non-existing principal parts are marked by () or *, but in individual cases, a dictionary has to be considered with regard to further details.

6. Analytic part

In the analytic part of my approach, the inflectional system of a language-part-of-speech combination is examined. Like other data mining projects, this part is split into the three phases pre-processing, processing (running the data mining algorithm) and post-processing. As usual, pre-processing with data selection, data clearing and transformation takes up most of the time (cf. Alpar / Niedereichholz 2000, 6-7, and Kruse / Borgelt 2002, 81). Post-processing uses up a medium portion and processing itself the smallest portion of time.

6.1 Pre-processing

In the pre-processing phase, the linguistic data is gathered, delimited and categorized according to the principles in Section 5. The primary sources are grammar books, dictionaries, reverse dictionaries or even part of speech dictionaries (such as verb dictionaries) to establish a database consisting of one table with linguistically important lexemes. Depending on the quality of the previous examination of a language-part-of-speech combination in the literature, this task can require more or less effort (cf. Holl / Maroldo / Urban 2007, 34-35).

This database is extended by adding the other principal parts and the inflection type to each lexeme registered. For this purpose, one determines adequate invariant inflectional categories for the principal parts and a collection of suitable inflection types analyzing and adjusting the ones used in literature. The desired result is an invariant stem distribution, so that quite simple decatenation and concatenation rules to derive the other inflection forms from the principal parts can be designed. In the few cases where a derivation is not possible, exceptional inflection forms are listed (cf. Section 2).

Prefixed lexemes and lexemes with more than one inflection type are examined and, if linguistically interesting, added to the database.

The final database has to contain at least the following columns (cf. Holl / Zimnik 2009, 13):

1. Inflection type.
2. Analytic lexical base: the standard orthographic form extended by prefix markers and markers for lexemes with inflectional variants; used by the cluster algorithm of the analytic part.
3. Synthetic lexical base: the pure standard orthographic form; used by the search algorithm of the synthetic part.
4. The other principal parts.
5. Comment: the meaning of lexemes with more than one inflection type; exceptional inflection forms not derivable with rules etc.

6.2 Processing

During the processing phase, a data mining algorithm is run. The following one was first published in Holl / Suljiæ 2010, 54-57. It is independent of the language-part-of-speech combination examined and detects clusters which are as big as possible and contain morphologically analogous lexemes with the same ending. For this purpose, the algorithm uses a top-down cluster analysis strategy (divisive method). It analyzes clusters of lexemes with the same ending with respect to morphological analogy. If a cluster is morphologically analogous, the objective is reached. If not, the ending is increased by one letter to the left, the cluster is divided and one continues as above.

The database established in the pre-processing phase has to be sorted reversely using the column “analytic lexical base”, like a reverse or rhyme dictionary. The column “lexeme cluster” initialized with NULLs is added. This attribute is meant to be filled with the name of the cluster which a lexeme is assigned to by the algorithm.

The core of the algorithm (Figure 4) consists of two nested loops:

step by step, the outer loop processes all of the lexemes;

step by step, the inner loop processes all ending lengths.

In detail, a general step of the algorithm runs as follows.

n-th step of the outer loop:

The next lexeme is picked out which is not assigned to a cluster, that is, which has the initial value in the column “lexeme cluster” (*lexeme_cluster* == *NULL*). This lexeme is called **reference lexeme** (variable *ref_lexeme*).

m-th step of the inner loop:

1. The variable *Counter_lexeme_ending_length* is increased by 1 and thus set to *m*. The algorithm gets the reference lexeme’s ending (*ref_lexeme_ending*, depending on the current value of *counter_lexeme_ending_length*) and its inflection type (**reference inflection type**). The algorithm is now going to compare the next lexemes with the reference lexeme. Therefore, the variables of the following comparison loop are initialized with 0.

2. In a third loop (comparison loop), all of the lexemes which have the same n -digit ending (the same n trailing letters in the lexical base), are examined (**compare lexemes**). All of the compare lexemes with the reference inflection type are counted in the variable *equiv_lexemes*. Those with a different inflection type are counted in the variable *non_equiv_lexemes*.

3. Depending on the values of the variables *equiv_lexemes*, *non_equiv_lexemes* and *counter_lexeme_ending_length*, four cases are distinguished (Figure 3).

	<i>equivalent_ lexemes</i>	<i>non_equivalent_ lexemes</i>	<i>counter_lexeme_ ending_length</i>
Case 1	0	0	—
Case 2	—	≥ 1	$= \text{ref_lexeme_length}$
Case 3	≥ 1	0	—
Case 4	—	≥ 1	$< \text{ref_lexeme_length}$

Figure 3. Cases after counting equivalent and non-equivalent compare lexemes

Case 1 (there are no compare lexemes):

one-lexeme cluster consisting of the reference lexeme only:

```
if non_equivalent_lexemes == 0 and equivalent_lexemes == 0
    then lexeme_cluster := ref_lexeme_ending
```

Case 2 (there are 1 or more compare lexemes with a different inflection type than the reference inflection type, and the current ending length is already equal to the number of letters of the reference lexeme):

one-lexeme cluster consisting of the reference lexeme only:

```
if non_equivalent_lexemes  $\geq 1$ 
and counter_lexeme_ending_length  $\geq$  ref_lexeme_length
    then lexeme_cluster := ref_lexeme
```

Case 3 (all of the compare lexemes have the reference inflection type):

many-lexeme cluster consisting of the reference lexeme and at least 1 more lexeme:

```
if non_equivalent_lexemes == 0 and equivalent_lexemes  $> 0$ 
    then
    for the reference lexeme
        lexeme_cluster := ref_lexeme_ending
    evaluation loop
    for each processed compare lexeme
        lexeme_cluster := ref_lexeme_ending
```

Remark: After cases 1 to 3, the inner loop terminates.

Case 4 (there are 1 or more compare lexemes with a different inflection type than the reference inflection type, and the current ending length is still less than the number of letters of the reference lexeme):

no cluster:

```
if non_equivalent_lexemes  $\geq 1$ 
and counter_lexeme_ending_length  $<$  ref_lexeme_length
    then no action
```

Only in case 4 (no cluster was found), the inner loop continues. Its $(m+1)$ -st step starts with an increase of the current ending

length (*counter_lexeme_ending_length*) by 1 and deals with a further examination of the current reference lexeme. Otherwise (a homogeneous cluster was found), the algorithm continues with the (*n+1*)-st step of the outer loop. The next reference lexeme is examined.

This decomposition is continued until all of the homogeneous clusters are detected, until there is no lexeme left which has the value *NULL* in the column “lexeme cluster”.

The algorithm presented can also support the search for basic clusters. This search is executed manually during the post-processing phase and uses the relations of the values of the variables *equivalent_lexemes* and *non_equivalent_lexemes* – both confined to basic lexemes – for each ending examined by the algorithm. If the first one is high compared to the second one, there is a candidate for a basic cluster (cf. Sections 3 and 6.3).

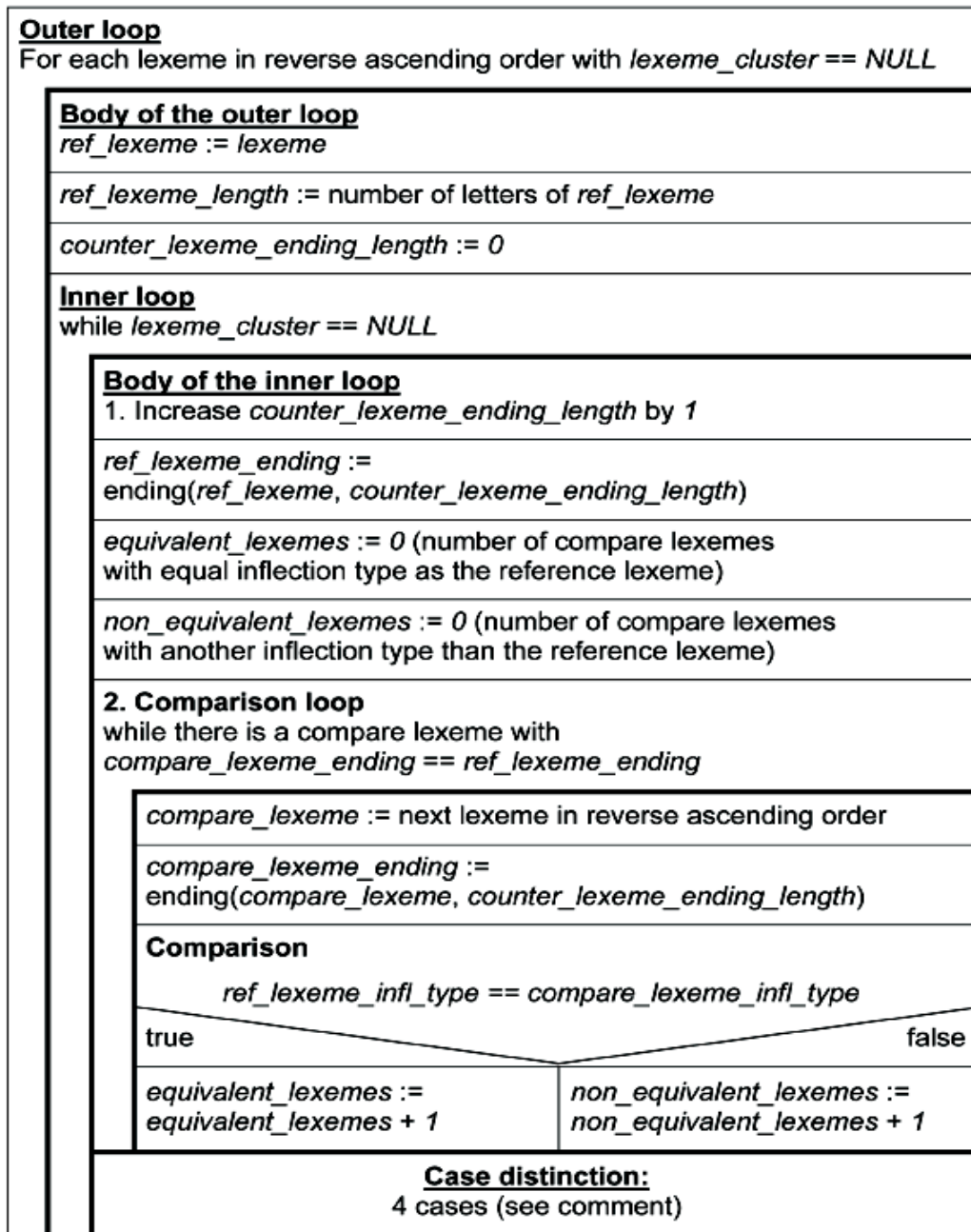


Figure 4. Nassi-Shneiderman diagram of the cluster algorithm

6.3 Post-processing

“After using data mining techniques, the results have to be interpreted, checked and evaluated” (Kruse / Borgelt 2002, 81, translated from German). The final lexeme register is a manual modification of the input database of the analytic cluster algorithm. In the final lexeme register, some redundancies, important for understanding and learning the system of a language, have to be maintained. They support the explanatory value of a register for linguist and language learner. To avoid misunderstandings, it is useful for the learner to explicitly find all lexemes of non-productive classes (classes which new lexemes are not assigned to). Therefore, every basic lexeme of this type should be mentioned explicitly, not only implicitly with a representative of a homogeneous cluster.

In contrast, the search algorithm of the synthetic part does not need these redundancies to detect the adequate cluster to some arbitrary lexeme. Therefore, those entries (table lines) which the search algorithm really needs are marked in a new column “Flag” of the lexeme register. *s* and *v* mean “to be treated by the search algorithm” where *v* marks an inflectional variant.

Homogeneous clusters found by the analytic algorithm have to be judged by a linguist as not all of them are linguistically interesting, e.g. small clusters containing only completely regular lexemes. Only the important homogeneous clusters are added as abstract lexemes (cf. Section 4) to the final lexeme register.

As already mentioned in Section 6.2, some numeric results can help the linguist to design profitable basic clusters without grapheme type characters (such as *C*, *V*). Basic and homogeneous clusters with grapheme type characters cannot be found with my analytic algorithm. Regarding this task, however, an experienced linguist is always a lot more reliable than any elaborate algorithm. It is smart manual work to find these types of clusters.

Basic clusters are added as abstract lexemes (cf. Section 4) to the final lexeme register as is done with homogeneous clusters.

Then the final lexeme register is complete. An extract of the English verb register corresponding to a segment of Figure 2 is displayed in Figure 5.

Even if a lexeme register is not used by the search algorithm in Section 7, its clusters provide linguists and language learners with a lot of structural information about a language-part-of-speech combination.

7. Synthetic part

The synthetic part consists of a search algorithm which uses the register of a given language-part-of-speech combination (the manually optimized result of the cluster algorithm, cf. Section 6.3) as a collection of rules in order to automatically decide which cluster some arbitrary lexeme belongs to. More than one cluster is found in the case of an inflectionally ambiguous lexeme with variants which can be inflected in two different ways.

The arbitrary lexeme, which I now call **search lexeme**, is spelled according to current orthography with the only exception that prefixes are detached with the marker underscore. It is possible to mark several prefixes in a row with several underscores which is necessary for languages which allow multiple prefixes.

I will now continue the argumentation in the comment to cluster trees in Section 4. Every cluster corresponds to a rule which says that a lexeme that matches the condition (lexical base, alphabetic property) of a cluster has to be assigned to the same inflection type as the cluster. An entire well-defined cluster tree represents a rule hierarchy. As the condition of a parent cluster is always weaker than the ones of its child clusters, more lexemes will match it, and the rule corresponding to it is less specific than the rules corresponding to its child clusters.

The search algorithm, however, has to look for the best, that is, the longest matches of the search lexeme with conditions of clusters. It has to look for the strongest conditions, for the most specific rules which can be applied to the search lexeme. Therefore, the search algorithm starts from the leaves of the cluster tree (cf. Figure 2). Looking for matches of a lexeme with a cluster means comparing the lexeme’s lexical base with the lexical base of the cluster.

The search algorithm runs as follows (cf. Holl / Zimnik 2009, 22-23).

The search lexeme is passed to a loop which forwards it to the “total length compare” routine: in the first run of the loop, the original search lexeme is forwarded, in a later run, the modified search lexeme, a prefix of which was cut off.

Cluster	Inflection type	Fl ag	Analytic lexical base	Synthetic lexical base	Past tense	Past participle	Comment
<i>basC</i>	<i>reg</i>	<i>s</i>	<i>~C</i>	<i>~C</i>	<i>~Ced</i>	<i>~Ced</i>	<i>e.g. look</i>
<i>basC</i>	<i>reg</i>	<i>s</i>	<i>~V</i>	<i>~V</i>	<i>~Ved</i>	<i>~Ved</i>	<i>e.g. visa, boo</i>
<i>basC2</i>	<i>reg / CC</i>	<i>s</i>	#CVg	#CVg	<i>CVgged</i>	<i>CVgged</i>	<i>e.g. jig</i>
<i>homC</i>	<i>reg / CC</i>	<i>s</i>	#CCVg	#CCVg	<i>CCVgged</i>	<i>CCVgged</i>	<i>e.g. drag</i>
	<i>reg / CC</i>		drag	drag	dragged	dragged	
	i-u-u / 0 / CC	s	dig	dig	dug	dug	
	<i>reg / CC</i>		jig	jig	jigged	jigged	
	<i>reg / CC</i>		pig	pig	pigged	pigged	
	<i>reg</i>	<i>s</i>	1hang	hang	hanged	hanged	kill with a rope
	a-u-u / 0	v	2hang	hang	hung	hung	suspend
	<i>reg</i>		whang	whang	whanged	whanged	
	<i>reg</i>		king	king	kinged	kinged	
<i>homC</i>	<i>i-u-u / 0</i>	<i>s</i>	<i>~ling</i>	<i>~ling</i>	<i>~lung</i>	<i>~lung</i>	<i>cling, fling, sling</i>
	i-u-u / 0		cling	cling	clung	clung	
	i-u-u / 0		fling	fling	flung	flung	
	i-u-u / 0		sling	sling	slung	slung	
	<i>reg</i>	<i>s</i>	1ring	ring	ringed	ringed	provide with a ring
	i-a-u / 0	v	2ring	ring	rang	rung	sound
	i-ou-ou / D	s	bring	bring	brought	brought	
	i-a-u / 0	s	α+spring	spring	sprang	sprung	BE
	i-a-u / 0	v	β+spring	spring	sprung	sprung	AE
	i-u-u / 0	s	string	string	strung	strung	
	i-u-u / 0	s	wring	wring	wrung	wrung	
	i-a-u / 0	s	sing	sing	sang	sung	
	<i>reg</i>		ting	ting	tinged	tinged	
	i-u-u / 0	s	sting	sting	stung	stung	
	<i>reg</i>		wing	wing	winged	winged	
	i-u-u / 0	s	swing	swing	swung	swung	
	<i>reg</i>		catalog	catalog	cataloged	cataloged	
	<i>reg / CC</i>	<i>s</i>	hum_bug	humbug	humbugged	humbugged	
<i>basC2</i>	<i>reg / +e</i>	<i>s</i>	<i>~ch</i>	<i>~ch</i>	<i>~ched</i>	<i>~ched</i>	<i>e.g. reach</i>
	<i>reg / +e</i>		reach	reach	reached	reached	
	ea-au-au / D / +e	s	teach	teach	taught	taught	
	<i>reg / +e</i>		screech	screech	screeched	screeched	
	<i>reg / +e</i>	<i>s</i>	α+be_seech	beseech	beseched	beseched	
	ee-ou-ou / D	v	β+be_seech	beseech	besought	besought	
	<i>reg / +e</i>		batch	batch	batched	batched	
	a-au-au / D / +e	s	catch	catch	caught	caught	
	<i>reg / +e</i>		scratch	scratch	scratched	scratched	
	<i>reg / +e</i>		watch	watch	watched	watched	

Figure 5. Part of the English verb register (cf. Holl / Maroldo / Urban 2007, 114-115)

The “total length compare” routine checks whether there is a one-lexeme homogeneous cluster in the register column “synthetic lexical base” which is completely identical with the search lexeme.

If this is the case (an identical one-lexeme cluster was found), the loop terminates successfully.

Otherwise (there is no identical one-lexeme cluster), the algorithm checks whether the (original or modified) search lexeme contains a prefix marker.

If the test is negative, the loop terminates as well, but without success.

Otherwise (the search lexeme contains a (further) prefix marker), it is passed to the “prefix cut” routine which cuts the prefix off the search lexeme, and the loop starts from the beginning.

If the loop terminates, that means either

1. The original or modified search lexeme matches a one-lexeme cluster (one cluster type) in the register or
2. The original (or modified) search lexeme does not yield any match in the register and does not contain any prefix marker.

The algorithm continues after the loop.

If suitable one-lexeme clusters (plural in the case of inflectional variants) were found in the register, they are displayed.

Otherwise, the original (or modified) search lexeme is forwarded to the “alternative total length compare” routine which generates search alternatives in addition to the search lexeme itself. This is done by replacing leading letters of the search lexeme by grapheme type characters (C, V) and conserving trailing letters. The n -th search alternative conserves n trailing characters and replaces the leading characters by grapheme type characters. The “alternative total length compare” routine checks search alternatives in the sequence from 4 to 0 (sufficient according to my experience) for matches with homogeneous or basic clusters with grapheme type characters and explicit word delimitation (two cluster types).

If a suitable many-lexeme cluster was found in the register, it is displayed.

Otherwise, the search lexeme and its search alternatives are forwarded to the “longest match” routine which reduces them from the left by one character after the other until the reduced search lexeme or one of its reduced search alternatives (from 4 to 0) matches a cluster in the register. A cluster found with this routine is a basic or homogeneous cluster without word delimitation, no matter whether with or without grapheme type characters (four cluster types). The details of this procedure are omitted as they would go beyond the scope of this paper.

The algorithm is even able to treat neologisms, as they are always assigned to regular inflection types which are either completely regular or contain at the most regular graphemic adjustments (such as e -insertion, e -deletion, consonant grapheme doubling in the case of English verbs). Such inflection types are always represented by basic clusters. It is possible that neologisms change a homogeneous cluster to a basic cluster.

Once the cluster the search lexeme belongs to is found, it is easy to construct the principal parts of the search lexeme in analogy to those of the cluster.

8. Outlook

Although the main decisions for my research approach have been taken, there remains a lot of research work to be done in the future:

1. The analysis of other language-part-of-speech combinations, especially of non-Indo-European languages. The description of the Turkish verb is forthcoming. Due to vowel harmony, it requires the distinction between four vowel grapheme type characters.
2. The comparative investigation of lexemes with more than one inflection type in several languages.
3. The development of an explanation component: the steps how the search algorithm arrives at the assignment of a lexeme to a cluster are already transparent. This process, however, should be presented in a user-friendly way.
4. The production of the complete averbos of search lexemes, that is, the derivation of inflection forms starting from the

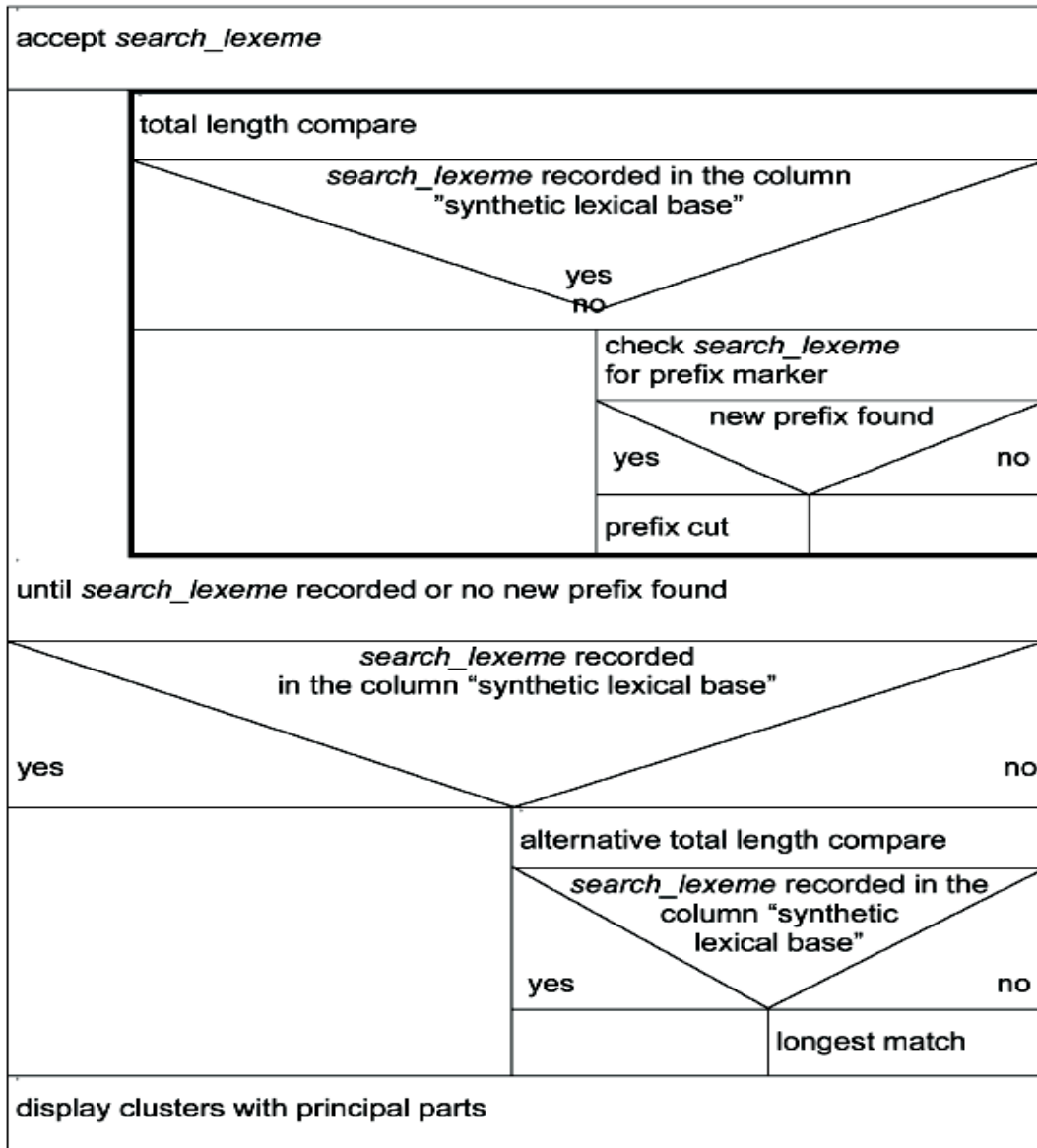


Figure 6. Nassi-Shneiderman diagram of the search algorithm

corresponding principal part. This task requires an effective rule parser for the interpretation and execution of decatenation and concatenation rules, which are partly dependent on certain alphabetic properties.

5. A promising application in medicine is a support tool for the rehabilitation of aphasia patients who have lost parts of their command of a language as a consequence of an accident or a stroke. Logopedics (speech therapy) report that those patients often remember the lexical bases, but have difficulties with the derivation of special inflection forms. The tool should allow them to enter some arbitrary lexeme, plus information about inflectional categories, such as person, number, tense, mode and voice for verbs, into a cell phone to be provided via internet with the inflection forms looked for.

I am sure that there will be other challenging research issues beyond the ones mentioned.

References

[1] Alpar, Paul, Niedereichholz, Joachim, (2000). *Data Mining im praktischen Einsatz*. Braunschweig / Wiesbaden.

- [2] Ester, Martin, Sander, Jörg, (2000). *Knowledge discovery in databases. Techniken und Anwendungen*. Berlin.
- [3] Friederici, Angela (2010). Wie das Gehirn zur Sprache kommt. *Spektrum der Wissenschaft*, 1, 66-71.
- [4] Holl, Alfred, Suljiæ, Ivan, (2010). *Rückläufiges Wörterbuch zur kroatischen Verbmorphologie. Aufbereitung mit Datenanalyseverfahren der Informatik (Data Mining)*. Regensburg: Roderer [= *Studia et exempla linguistica et philologica*, Series V: Lexica, Tom. 6].
- [5] Holl, Alfred; Zimnik, Gordon, (2009). Data Mining und natürlichsprachliche Verbmorphologien. [= *Schriftenreihe der Georg-Simon-Ohm-Hochschule Nürnberg* 43(2008) 1-54]. Nürnberg: Ohm-Hochschule.
- [6] Holl, Alfred, Maroldo, Sara, Urban Reinhard, (2007). *The inflectional morphologies of the Swedish noun, the Swedish verb and the English verb. Reverse dictionaries based upon data mining methods*. Växjö [= *Mathematical modelling in physics, engineering and cognitive sciences*, vol. 12].
- [7] Holl, Alfred, Pavlidis, Stilianos, Urban, Reinhard, (2006). *Rückläufiges Wörterbuch zur alt- und neugriechischen Verbmorphologie. Aufbereitung mit Datenanalyse-Verfahren der Informatik (Data Mining)*. Regensburg [= *Studia et exempla linguistica et philologica*, Series V: Lexica, Tom. 5].
- [8] Holl, Alfred; Behrschmidt, André, Kühn, Alexander, (2004). *Rückläufige Register zur russischen und deutschen Verbmorphologie. Aufbereitung mit Datenanalyse-Verfahren der Informatik (Data Mining)*. Regensburg 2004 [= *Studia et exempla linguistica et philologica*, Series V: Lexica, Tom. 4].
- [9] Holl, Alfred, (2003). Datenanalyseverfahren der Informatik (Data Mining) als Grundlage einer didaktischen Darstellung der französischen Verbmorphologie. In: Bernhard, Gerald; Kattenbusch, Dieter; Stein, Peter (ed.): *Namen und Wörter. Festschrift Josef Felixberger zum 65. Geburtstag*. Regensburg, 107-119.
- [10] Holl, Alfred, (2002). Licht und Schatten von Analogieschlüssen auf der Basis rückläufiger Ähnlichkeit in der Verbmorphologie romanischer und germanischer Sprachen. In: Heinemann, Sabine; Bernhard, Gerald; Kattenbusch, Dieter (ed.): *Roma et Romania. Festschrift Gerhard Ernst zum 65. Geburtstag*. Tübingen, 152-167.
- [11] Holl, Alfred, (2001). The inflectional morphology of the Swedish verb with respect to reverse order: analogy, pattern verbs and their key forms. *Arkiv för nordisk filologi* 116, 193-220.
- [12] Holl, Alfred, (1988). *Romanische Verbmorphologie und relationentheoretische mathematische Linguistik. Axiomatisierung und algorithmische Anwendung des klassischen Wort und Paradigma-Modells*. Tübingen [= *Linguistische Arbeiten* 216].
- [13] Kruse, Rudolf, Borgelt, Christian, (2002). Suche im Datenschungel. *Spektrum der Wissenschaft*, 11, 80-81.

Selected examples of verb registers and reverse dictionaries

- [1] Alinei, Mario L.: *Dizionario inverso italiano*. Den Haag 1962.
- [2] Edmonds, David: *The Oxford reverse dictionary*. Oxford 2002.
- [3] Einberger, Angela: *Langenscheidt Verb-Tabellen Englisch*. Berlin 2005, 2000.
- [4] Elia, Pietro: *I verbi italiani per gli stranieri*. Mailand [1955] 1983.
- [5] Goulding, Sylvia: *Englisch. Verben. Basiswissen. Eine leicht verständliche Beschreibung des englischen Verbsystems*. Princeton NJ 1998.
- [6] Juilland, Alphonse: *Dictionnaire inverse de la langue française*. Den Haag 1965.
- [7] Kempgen, Sebastian: *Grammatik der russischen Verben*. Wiesbaden 1989.
- [8] Langendorf, Dieter: *L'art de conjuguer. Le Nouveau Bescherelle: Dictionnaire de douze mille verbes*. Frankfurt 1986.
- [9] Lehnert, Martin: *Rückläufiges Wörterbuch der englischen Gegenwartssprache*. Leipzig 1973.
- [10] Mateo, Francis: *El arte de conjugar en espanol. Diccionario de 12 000 verbos. Collection Bescherelle*. Paris 1995.
- [11] Mater, Erich: *Rückläufiges Wörterbuch der deutschen Gegenwartssprache*. Leipzig 1965.
- [12] Muthmann, Gustav: *Reverse English dictionary based on phonological and morphological principles*. Berlin 2002, 1999.
- [13] Muthmann, Gustav: *Rückläufiges deutsches Wörterbuch*. Tübingen 2001.

- [14] Scott, Samantha: *Pons Verbtabelle Englisch*. Stuttgart 2001.
- [15] Sleumer, Albert: *Die unregelmäßigen Zeitwörter der goldenen und silbernen Latinität in ihren einfachen und zusammengesetzten Formen*. Bonn ²1962.
- [16] Stahl, Fred A.; Scavnick, Gary E. A.: *A reverse dictionary of the Spanish language*. Urbana 1973.
- [17] Weermann, Eva Maria: *Pons Verbtabelle Deutsch*. Stuttgart 2001.
- [18] Zaliznjak, Andrej A.: *Grammatičeskij slovar' russkogo jazyka: slovoizmenenie*. Moskau ⁴2003 = ³1987 = ¹1977
[= Зализняк, Андрей А.: *Грамматический словарь русского языка: словоизменения*]