DOI: 10.15439/2015F280
ACSIS, Vol. 5

Concepts extraction from unstructured Polish texts: a rule based approach

Piotr Szwed
AGH University of Science and Technology
E-mail: pszwed@agh.edu.pl

***Abstract*—We present recently developed solution allowing extraction of concepts from unstructured Polish texts with special focus on correct morphological forms of obtained concept names. As Polish is a highly inflected language, detected names need to be transformed following Polish grammar rules. We propose a user-friendly method for specification of transformation patterns, which is based on a simple annotations language. Annotations prepared by a user are compiled into transformation rules. During the concept extraction process the input document is split into sentences and the rules are applied to sequences of words comprised in sentences. Recognized strings forming concept names are aggregated at various levels and assigned with scores. We report also results of initial experiments performed on a medical text.**

***Index Terms*—NLP, Text Mining, concept extraction, unstructured text, inflection, rules**

I. Introduction

IN THIS PAPER we investigate the problem of concepts extraction from unstructured Polish texts. The term concept is defined here as a sequence of words (an n-gram) occurring frequently in analyzed documents. However, as primary function of concepts is to represent objects, ideas, events or activities, extracted names should comprise words belonging to specific parts of speech classes (usually nouns or verbal nouns with complements) and also satisfy certain syntactical restrictions stemming from language conventions.

Concepts retrieved from unstructured textual data have several applications. Their usage improves relevance of documents returned by search queries due to more accurate indexing and clustering. They may help to assure data privacy by identifying sensitive information and removing it from published texts (documents anonymization). Models based on concepts can be used to express and structure knowledge existing within an organization, which is often distributed between large number of documents of various types: e-mails, reports or internal regulations.

The goal of our work was to develop a tool allowing to extract referenced concepts from documents in Polish language. The key assumption made was that concept names should have correct morphological forms according to Polish grammar rules. Such feature was selected due to aesthetic reasons (in various applications concept names are presented to end users), as well as the expectation that in many situations using the correct form may improve sense disambiguation. Hence, the designed solution, apart from identifying concepts, should provide automatic translation of n-grams representing concepts to their nominative form. This task is particularly challenging for the Polish language, which is characterized by the high degree of inflection.

We decided to apply a rule based approach to perform transformation of concept names. The rule language, inspired by Petri nets, allows to define patterns of input tokens and required transformations. For this purpose part of speech (POS) information is used. Considering the complexity of POS tags, specification of rules may be a complicated and tedious task. We build the translation rules fully automatically by defining samples of translation patterns (annotations) and compiling them to rules.

The paper is organized as follows: next Section II discusses the problem of concept extraction, as well as several tools dedicated to Polish language processing. It is followed by Section III, which presents general features of our approach. Next Section IV describes shortly Morfologik, a dictionary and POS tagging/lemmatizing library. In Section V we discuss models, algorithms and other details of the solution. Results of initial experiments are given in Section VI. Last Section VII provides concluding remarks.

II. Related works

Ontology learning is a process of building ontologies from various data representations. It includes such tasks as identification of concepts, their relations and attributes, arranging into hierarchies. Such task is apparently easier in the case of structured or semi-structured data. A challenging problem, however, is building ontologies (usually taxonomies) comprising concepts extracted from unstructured text documents [14]. For this purpose various text mining techniques can be used: syntactical analysis, Formal Concept Analysis [4] and clustering [7].

Concepts (sometimes also referred as compound terms or phrases) are important features used in Text Mining [23]. Compound terms processing is a technique aiming at improving accuracy of search engines by indexing documents according to compound terms, i.e. combinations of 2 or more single words. During query execution searched compound terms are also extracted form queries (which can be phrases in natural language) and then matched with compound terms attributed to documents. A good example can be a compound term “table wine”. A document referencing it is more likely to discuss wines, than pieces of furniture. Such approach outperforms solutions that use in queries keywords combined with boolean operators. The idea of statistical compound terms processing was proposed and commercialized by Concept Search company [5]. On the other hand, a syntactical technique based on rules defining patterns for various European languages was developed within CLAMOUR project (see reports at http://webarchive.nationalarchives.gov.uk/20040117000117/ http://statistics.gov.uk/methods_quality/clamour/default.asp).

In [9] Dalvi, Kumar et al. from Yahoo Labs. coined an idea of a *web of concepts*. They claimed that the current model of the web in form hyperlinked pages represented by bags of words should be augmented by extracting concepts and creating a new rich view of all available resources for each concept instance. The paper discussed several use cases enabled by the new approach, including: more accurate web search, browsing optimization, session optimization and advertising. The paper also indicated several challenges, related mainly to information extraction, potential uncertainty and changing data. An algorithm for concept extraction following this idea was proposed in [20].

Blake and Pratt [2] used automatically retrieved concepts to build searchable representations of medical texts. In [3] authors also extracted ontologies from medical texts and showed that using concepts improves search results.

Osinski and Weiss described Lingo, a concept driven document clustering algorithm [19]. Concepts (frequent sequences of terms), were used to label clusters in a document-term matrix.

An important task in NLP language classification is *tagging*, i.e. assigning part of speech (POS) information to inflected forms of words. This is a challenging task for a highly inflected languages like Polish. According to [1] words in English language can be described with about 200 tags whereas for Polish their number ranges at 1000.

In our work we used Morfologik stemming library [17], which is discussed in detail in Section IV. The software and the dictionary was used in several projects including Language tool [18], carrot search [19], PSI toolkit [10], PELCRA [21] and Smyrna [11]. Morfologik dictionary was integrated as a part of PoliMorf [27].

Language tool [18], [16] is a proofreading program supporting a number of languages including Polish. It allows to define text correction rules referring to explicit token values, lemmatized words and part of speech tags. Rules can be specified either in XML format or written directly as Java classes. The tool provides also a visual rule editor, which allows to construct a rule in an interactive manner. As the rules yield suggestions, their core functionality is somehow similar to the concepts extraction rules described in this paper.

III. TOWARDS POLISH CONCEPTS EXTRACTION

Concepts are terms denoting sets of objects sharing common properties. The most general concepts are represented in texts as single nouns (referring to tangible or abstract objects) and verbal nouns describing actions. Examples of both types can be a *car* (a tangible object) or *driving* (an action). Subclasses of general concepts are usually represented by various complements, e.g. a *green car* (a car having the green color) or *car driving* (an action performed on a specific type of vehicle).

Hence, concepts in texts are represented by n-grams (sequences of tokens), whose elements satisfy grammar rules related to correct words ordering and their morphological forms. The rules are language specific; this in particular applies to the Polish language, in which the complements can be declined depending on the noun case, e.g. for the green car: *zielony samochód* (Nominative), *zielonego samochodu* (Accusative), *zielonemu samochodowi* (Dative), etc.

Table I gives examples of several sentences and concepts identified in a supervised manner. Underlined words indicate parts of concept names denoting their superclasses, e.g. *szybki* <u>*samochód*</u> $\sqsubset$ *samochód*. (The English equivalent is: *FastCar* $\sqsubset$ *Car*).

The concepts listed in Table I constitute typical parts of speech combinations: adjective+noun (*szybki samochód*, *słone jezioro*, *mały ludzik*), nount+noun (*architektura systemu*, *hurtownia danych*) or verbal noun+noun (*leczenie pacjenta*, *zbieranie informacji*). They appear in source sentences as sequences of tokens that after identifying them should be transformed into appropriate nominative forms following the language grammar rules, especially, as regards noun complements. In most cases concepts occur in distinct parts of sentences, however sometimes they may overlap, e.g three concepts can be selected for the entry 6 in the table. Similarly, for the entry 10 showing two adjectives separated by a coma and a noun, three concepts forming a small taxonomy are possible:

- *mały ludzik* $\sqsubset$ *ludzik*
- *zielony ludzik* $\sqsubset$ *ludzik*
- *mały zielony ludzik* $\sqsubset$ *mały ludzik* $\sqcap$ *zielony ludzik*

A. Rule based approach

The proposed approach to concept extraction consists in defining rules that search for selected patterns in an input text, then apply appropriate morphological transformations to matched words in order to obtain correct forms of concepts.

This is shown in Fig. 1. A window having the length equal to size of a rule input pattern slides through the input text. Sentences are the natural boundaries for the analysis, i.e. the window stops at the sentence end.

If the rule is applicable for a current n-gram appearing in the window, a set of output sequences of tokens is produced. They constitute candidates for concepts, that can be further analyzed and aggregated according to various attributes: sentences, where they occurred, frequency in the whole document, weights describing confidence, etc.

B. Specification of rules - learn by example

Selection of the language used to specify extraction rules is an important decision, as the language capabilities have strong impact on both on the process on rules definition and obtained results. In our approach we decided to define rules in a semi-formal way, by giving samples of expected translations rather

TABLE I
IDENTIFICATION OF CONCEPTS IN SENTENECES AND EXAMPLES OF ANNOTATIONS.

#	Sentence	Concept	Annotation	Function
1.	Jeżdżę *szybkim samochodem.*	szybki samochód	@(szybki samochód = szybki @samochód)	extract noun + verb
2.	Morze Kaspijskie jest *jeziorem słonym.*	słone jezioro	@(jeziorem słonym = słone @jezioro)	inversion + Instrumental $\rightarrow$ Nominative
3.	*Leczenie pacjenta* przebiegało bardzo wolno.	leczenie pacjenta	@(leczenie pacjenta = @leczenie pacjenta)	extract verbal noun + complement
4.	Opracowałem *architekturę systemu.*	architektura systemu	@(architekturę systemu = @architektura systemu)	Accusative $\rightarrow$ Nominative
5.	*Rozwojowi rolnictwa* towarzyszyło *zanikanie lasów.*	rozwój rolnictwa	@(rozwojowi rolnictwa = @rozwój rolnictwa)	Dative $\rightarrow$ Nominative
		zanikanie lasów	@(zanikanie lasów = @zanikanie lasów)	extract verbal noun + complement in plural form
6.	Następnie poświęcił się *zbieraniu informacji* o *losach misjonarza.*	zbieranie informacji	@(zbieraniu informacji = @zbieranie informacji)	Dative $\rightarrow$ Nominative
		los misjonarza	@(losach misjonarza = @los misjonarza)	Locative plural $\rightarrow$ Nominative sing.
6.	Obecnie coraz częściej stosowane są *zwinne metodyki zarządzania projektami informatycznymi.*	zwinna metodyka	@(zwinne metodyki = zwinna @metodyka)	Nominative plural $\rightarrow$ Nominative sing.
		metodyka zarządzania	@(metodyki zarządzania = @metodyka zarządzania)	Nominative plural $\rightarrow$ Nominative sing.
		zarządzanie projektami informatycznymi	@(zarządzania projektami informatycznymi = @zarządzanie projektami informatycznymi)	Accusative $\rightarrow$ Nominative
8.	*Hurtownie danych* stanowią osobną klasę *systemów informatycznych.*	hurtownia danych	@(hurtownie danych = @hurtownia danych)	Nominative plural $\rightarrow$ Nominative sing.
		system informatyczny	@(systemów informatycznych = @system informatyczny)	Genitive plural $\rightarrow$ Nominative sing.
9.	Małe, *zielone ludziki* rozłożyły się u *podnóża dębu.*	zielony ludzik	@(zielone ludziki = zielony @ludzik)	Nominative plural $\rightarrow$ Nominative sing.
		podnóże dębu	@(podnóża dębu = @podnóże dębu)	Locative $\rightarrow$ Nominative
10.	*Małe, zielone ludziki* rozłożyły się u podnóża dębu.	mały ludzik, zielony ludzik, mały zielony ludzik	@(małe $, zielone ludziki = mały @ludzik | zielony @ludzik | mały zielony @ludzik)	Nominative plural $\rightarrow$ Nominative sing.

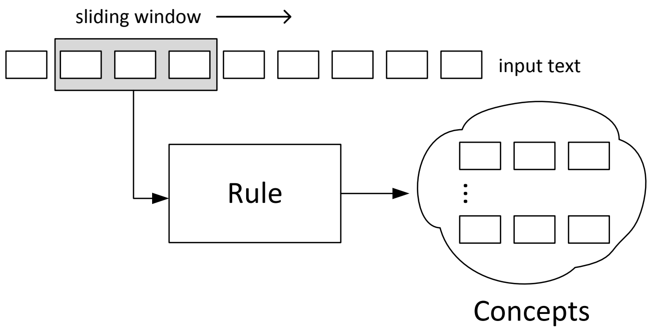

Fig. 1. General rule design

than using formally defined rules. The translation patterns are defined using special, relatively simple textual annotations that can be embedded in a source text or placed in a separate file.

The third column of Table I gives examples of annotation matching the identified concepts. Each annotation starts with '@' (at sign) followed by the annotation body put between two parentheses. The '=' sign separates the input pattern and a list of output sequences, whereas '|' is the output sequence separator. Optional '@' sign within an output sequence identifies a key, i.e. a possible concept superclass.

The last column of Table I specifies the expected function for each annotation example. Usually, annotations define transformations of declination cases and plural forms, which according to Polish grammar rules should be applied both to nouns and their complements. However, they may be used to specify extraction patterns only, not accompanied by morphological transformation. Examples are given in rows 1, 3 and 5.

It should be remarked that, basically, annotations do not specify exact matches of words (for an exact match the dollar sign is used as in row 10). The intent of annotations is to specify indirectly rules applicable to classes of tokens. For example a rule derived from the annotation `@(zwinne metodyki = zwinna @metodyka )` (agile methodologies $\rightarrow$ agile methodology) in row 6 of Table I applies to feminine nouns in plural form with an adjective. Hence, it should match also the pair of words, which are tagged with the same POS information, e.g. *długie wstążki* (long ribbons), and properly convert it to the singular form *długa wstążka*.

An advantage of the proposed indirect method of rules specification is its simplicity: translation patterns can be defined very quickly, moreover, specifications are not bounded to a particular rule language or execution engine. Depending on a compiler used, rulsets in various rule languages can be obtained.

In many cases defined translation patterns may result in conflicting rules. Let us consider the following annotations:

1) `@(hurtowniom danych=@hurtownia danych)`
2) `@(grubościom pokryw=@grubość pokrywy)`

The first translation results in a noun having a complement in plural form, whereas int the second case the complement is singular. However, they both represent valid transformations. To tackle with such problems the language used to represent rules should allow to attribute results with weights representing likelihood of concept occurrence.

C. Process of concept extraction

The outline of the process of concept extraction is presented in Fig. 2. A text file containing manually prepared annotations is compiled yielding a set of rules stored in an XML file. The input text is split into sentences and then the rule set is executed for each sentence giving candidate concepts. Finally, the concept occurrences are aggregated and sorted. As different rules may return identical results, the aggregation is at first performed at the sentence level, then for the whole document.

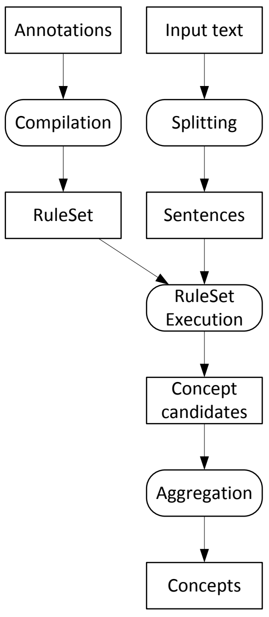

Fig. 2. Process of concept extraction

IV. Morfologik

During annotations processing and rules execution input words are checked for specific part of speech properties and undergo morphological transformations. This function is provided by the Morfologik.

Morfologik is both a a comprehensive dictionary of Polish inflected forms and a software library written in Java accompanied by a number of utility tools. The main function offered by the Morfologik is *stemming* (lemmatization) of Polish words, i.e. finding a stem (lemma) accompanied by grammar information for an inflected form. Examples of inflected forms and corresponding stems can be: *psu* – *pies* (noun: dog), *czystego* – *czysty* (adjective: clean) or *pisaniu* – *pisać* (verbal noun writing and verb: to write).

Morfologik dictionary can be seen as a relation

$$D \subset IF \times S \times \mathcal{P}, \tag{1}$$

where IF is a set of inflected forms, S is a set of stems, $S \subset IF$, and $\mathcal{P}$ is a set of POS tags defining properties of inflected forms (part of speech, gender, singular vs. plural, declination case, etc.)

A few entries from the Morfologik dictionary are shown in Table II. It can be observed that several stems may be found for an inflected form, e.g. *czarnym* – *czarna*, *czarny*. Moreover, pairs of inflected forms and stems can be attributed with multiple tags (separated with '+' sign).

A tag is a string of symbols (usually abbreviations) separated by colon signs. Examples in the table show typical tag elements: *subst* – noun, *adj*-adjective, *ger*–verbal noun, *sg*–singular, *pl* –plural, *nom*, *gen*, *dat*, *acc*, *inst*, *loc* – declination cases. For a given part of speech class, tag components appear in the same order. In some cases multiple symbols separated by dot sign may occur, e.g. "m1.m2" - various classes of masculine forms.

Based on the Morfologik dictionary scheme two functions can be defined: $stem$ (2) and $synth$ (3). The first takes as input an inflected form and returns a set of stems with accompanying tags, the second synthesizes inflected forms from an input stem and tag.

$$stem\colon IF \to 2^{S \times \mathcal{P}} \tag{2}$$

$$synth\colon S \times \mathcal{P} \to 2^{IF} \tag{3}$$

Morfologik fully supports stemming. The library uses internally an efficient dictionary representation based on Finite State Machine (FSM) model, which is characterized by compact data size and short access times [8]. The current Morfologik dictionary for the Polish language (version 2.0) constitutes a large 317 MB text file, whereas the same data compiled to FSM are stored in 2.7 MB binary file. The precompiled dictionary is a part of Morfologik distribution, however, the library offers also a number of awk scripts to preprocess the dictionary data, as well as tools allowing to compile it into FSM.

Unfortunately, the synthesizing function is not provided directly by the library, although the documentation suggests that it is possible to rebuild (revert) the dictionary and recompile to FSM form. As the guidelines for implementing an FSM

TABLE II
SAMPLE ENTRIES FROM THE MORFOLOGIK DICTIONARY.

IF - inflected form	S – stem (lemma)	$\mathcal{P}$ - part of speech tags
czarnym	czarna	subst:pl:dat:f
	czarny	adj:pl:dat:m1.m2.m3.f.n1.n2.p1.p2.p3:pos +adj:sg:inst:m1.m2.m3.n1.n2:pos +adj:sg:loc:m1.m2.m3.n1.n2:pos +subst:pl:dat:m1+subst:sg:inst:m1+subst:sg:loc:m1
czystego	czysty	adj:sg:acc:m1.m2:pos+adj:sg:gen:m1.m2.m3.n1.n2:pos
psu	pies	subst:sg:dat:m1+subst:sg:dat:m2
pisaniu	pisać	ger:sg:dat.loc:n2:imperf:aff:refl.nonrefl
	pisanie	subst:sg:dat:n2+subst:sg:loc:n2

based synthesizer seemed quite difficult to follow, we left this possibility for further improvements. Instead, we developed a synthesizing function, which uses the dictionary data stored in a local PostgreSQL database. It was populated with 288657 stems, 1173 tags broken on plus (+) signs and 7410145 triples comprising inflected forms, stems and tags.

The implemented $synth(s, p)$ function comprises two steps:

1) A query to the database is made to select a set of triples matching the stem s. As the query result set $IFT = \{(if, s', p') \in D \colon s' = s\}$ is returned. The set is usually small, in most cases it contains up to 30 entries.
2) From the triples in IFT a set of inflected forms $IF = \{if \colon (if, s', p') \in IFT \wedge match(p, p')\}$ is selected, whose tags p' match p. The $match(t, t')$ function takes into account order of symbols appearing in tags, as well as alternate properties indicated by the dot separator.

The time efficiency of the implemented $synth$ function is obviously inferior to the $stem$ function offered by Morfologik. Single call to $stem$ (based on FSM representation) takes about 0.085 ms, whereas $synth$ (based on database queries) ranges to 0.75 ms. It should be remarked that we have also implemented an ORM version of $synth$. In this case the execution time is about 1.25 ms.

V. METHODS

The language used to define rules is based on Petri nets. Each rule comprises ordered sets of input and output places, which are linked by transitions.

Fig. 3 gives an example of the rule comprising three input places, three output places and four transitions. Multiple transitions, here *t1* and *t2* may link a pair of input and output places. The net layout shows also, how inversion of tokens can be achieved.

Each rule transition is assigned with two sets of tags: input $itag$ and output $otag$. Input tags are used as guards, they allow check if the transition applies to a word tagged with part of speech information. Output tags are used to synthesize target inflected form from a word stem (lemma). Additionally, each transition has assigned weight, that can be used to differentiate less and more likely translation schema.

Below we give a formal definition of the language used to define rules.

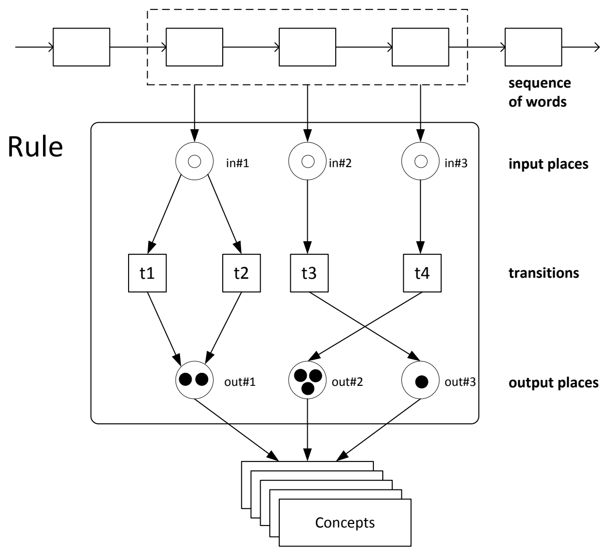

Fig. 3. Detailed rule design

Definition 1. Concept extraction rule is a tuple $R = (\mathcal{P}, I, O, T, itag, otag, itoken, \mu)$, where:

- $\mathcal{P}$ is a set of tags
- $I = \{i_1, i_2, \ldots, i_n\}$ is an ordered set of input places,
- $O = \{o_1, o_2, \ldots, o_m\}$ is an ordered set of output places,
- $T \subset I \times (O \cup \{nil\})$ is a set of transitions,
- $itag \colon T \to 2^{\mathcal{P}}$ is a function assigning to a transition a set of *input tags*,
- $otag \colon T \to 2^{\mathcal{P}}$ is a function assigning to a transition a set of *output tags*,
- $itoken \colon I \to \mathcal{A}$ is a function assigning to an input place an exact (possible empty) string form $\mathcal{A}$
- $\mu \colon T \to [0, 1]$ is a transition weight function

The symbol nil used in transition definition denotes a fake output place. It is used when a transition serves as a guard allowing to check if an input place contains a word appertaining to a particular class without translating it. The $itoken$ function is used to assign particular token values to input places. They result from compilation of '\$'-token specifications in input annotations (see Table I row 10).

A. Compiling annotations to rules

An annotation defining multiple outputs $@(\sigma = \pi_1|\pi_2|\dots|\pi_k)$ is split into k rules corresponding to annotations $@(\sigma = \pi_1)$, $@(\sigma = \pi_2)$,..., $@(\sigma = \pi_k)$. (Symbols σ and π stand here for input and output sequences of words.

For an annotation $@(\sigma = \pi)$, n input places and m output places are created, where n and m denote the lengths of σ and π respectively.

Then each word in sequences σ and π is stemmed and transitions between input and output places are derived based on stem matching.

Fig. 4 illustrates this process on an example of annotation `@(informacji przetwarzaniu = @przetwarzanie informacji)`. The equivalent English term is *information processing*. The annotation defines a translation pattern that includes inversion and changing the case of the whole expression from Dative to Nominative form.

Rounded rectangles depict words appearing in the input sequence σ (above) and output sequence π (below). For each word in the sequence the stemming information is determined with Morfologik $stem$ function. It consists of a stem (lemma) and a set of tags. The word *informacji* is classified as the noun *informacja* tagged as singular or plural form with various declination cases (see the first transition in Fig. 5 for details). The stemming information for *przetwarzanie(u)* gives two options: it is either a noun *przetwarzanie* (Eng. *processing*) or a verbal noun (tagged by Morfologik as *ger*) derived from the verb *przetwarzać* (Eng. *to process*).

Hence, by performing stem/tags matching three transitions are created (indicated by arrows in Fig. 4): one for places corresponding to the mapping *informacji* $\rightarrow$ *informacji* and two for *przetwarzaniu* $\rightarrow$ *przetwarzanie*. XML code for the resulting rule is presented in Fig. 5.

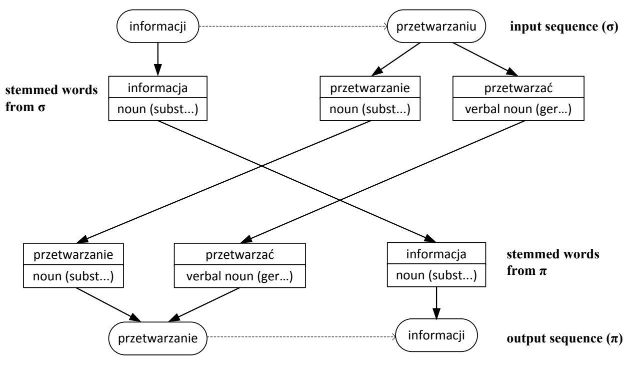

Fig. 4. Translation of annotation `@(informacji przetwarzaniu = @przetwarzanie informacji)`

It is expected that rules obtained as results of the annotations compilation satisfy the following conditions:

- Each output place is a target of a transition: $\forall o \in I.\exists t = (i_s, o_e) \in T\colon o = e_e$
- All transitions leaving an input place must target the same output place: $\forall t_1 = (i_1, o_1), t_2 = (i_2, o_2) \in T.i_1 = i_2 \rightarrow o_1 = o_2$

If the compilation returns a rule, which does not satisfy the above conditions, the source annotation is reported as ambiguous. Obviously, with the presented translation algorithm it is not possible to compile such annotation as `@(danym danym = dane @dane)` although it can be quite easily interpreted as *given data*.

To our consolation, statistical translation functions of Google Translate (as of 2015) also have problems in this case: *danym danym* is translated to *given the*, wheras *dane dane* to *data data*. However, after specifying the input n-gram explicitly, i.e entering *"dane dane"* the correct form *given data* is returned.

```
<rule id="r:13" weight="1.0">
  <source>@( informacji przetwarzaniu  = @przetwarzanie informacji ) @line:34</source>
  <inputPlaces>
    <inputplace isExact="false" ord="0">
      <transitions>
        <transition target="op:30" isGuard="false" weight="1.0">
          <inputTags>
            <intag>subst:pl:gen:f</intag>
            <intag>subst:sg:gen:f</intag>
            <intag>subst:sg:dat:f</intag>
            <intag>subst:sg:loc:f</intag>
          </inputTags>
          <outputTags>
            <outtag>subst:pl:gen:f</outtag>
            <outtag>subst:sg:gen:f</outtag>
            <outtag>subst:sg:dat:f</outtag>
            <outtag>subst:sg:loc:f</outtag>
          </outputTags>
        </transition>
      </transitions>
    </inputplace>
    <inputplace isExact="false" ord="1">
      <transitions>
        <transition target="op:29" isGuard="false" weight="1.0">
          <inputTags>
            <intag>subst:sg:loc:n2</intag>
            <intag>subst:sg:dat:n2</intag>
          </inputTags>
          <outputTags>
            <outtag>subst:sg:nom:n2</outtag>
            <outtag>subst:sg:acc:n2</outtag>
            <outtag>subst:sg:voc:n2</outtag>
          </outputTags>
        </transition>
        <transition target="op:29" isGuard="false" weight="1.0">
          <inputTags>
            <intag>ger:sg:dat.loc:n2:imperf:aff:refl.nonrefl</intag>
          </inputTags>
          <outputTags>
            <outtag>ger:sg:nom.acc:n2:imperf:aff:refl.nonrefl</outtag>
          </outputTags>
        </transition>
      </transitions>
    </inputplace>
  </inputPlaces>
  <outputPlaces>
    <outputplace id="op:29" ord="0" isKey="true"/>
    <outputplace id="op:30" ord="1" isKey="false"/>
  </outputPlaces>
</rule>
```

Fig. 5. A rule in XML format

B. Execution of rules

As the described rules (see Definition 1) follow the Petri net approach, to define their execution such notions as tokens and marking are necessary. The set of tokens TK is defined as $TK = IF \times S \times \mathcal{P} \times \mathbb{R}^{0+}$. (See also formula 1.) Components of a token tuple $(if, s, p, w) \in TK$ are the following: if denotes an inflected form, s is a stem, p is a part of speech tag and w is a non-negative weight.

Definition 2 (Marking)**.** *Marking* for a rule $R = (\mathcal{P}, I, O, T, itag, otag, itoken, \mu)$ is defined as a function that assigns sets of tokens to places $M : I \cup O \to 2^{TO}$

Before executing rules, each word in an input sequence $\pi = (\pi_i)$ is submitted to Morfologik $stem$ function (see formula 2) and corresponding sequence of stemming information is obtained $\rho = (\rho_i)$, where $\rho_i \in 2^{S \times \mathcal{P}}$. Hence, π_i is an input inflected form of a word and ρ_i a set of possible stem–tag combinations for π_i.

Execution of a rule comprises the following steps:

1) Input places are filled with tokens starting from position b in sequences π and ρ (b defines the beginning of the sliding window). Each k-th input place receives tokens from π_{b+k} and ρ_{b+k}. Its marking is gets: $M(I_k) \leftarrow \{\pi_{b+k}\} \times \rho_{b+k} \times \{0\}$.
2) For each input place $I_k \in I$ it is checked, if there exists at least one enabled transition (or exact word matching, if specified). Hence, for each transition $t \in T(I_k)$ and each token $tk = (if, s, p, w) \in M(I_k)$ it is checked, if the sets of transition input tags $itag(t)$ and token tags p match. The matching function compares tags in both sets, splitting them into smaller parts where needed. A pair (t, tk), such that the transition t is enabled is called an enabled binding.
3) Finally, transitions are executed for all enabled bindings (t, tk). If O_i is the output place for the transition t, and $tk = (if, s, p, w)$, then the marking for O_i is modified according to (4):

$$M(O_i) \leftarrow M(O_i) \cup synth(s, otag(t)) \times \{s\} \times \\ \times (p \cap otag(t)) \times weight(t, tk) \quad (4)$$

Interpretation of the formula (4) is the following: a stem (lemma) together with output tags are submitted to the $synth$ function, which returns a set of inflected forms. In consequence, for an input word if_{in} the chain of translations is accomplished:

$$if_{in} \xrightarrow{p_i n \cap itag(t)} \{s\} \xrightarrow{p \cap otag(t)} \{if_{out}\}$$

In the first step if_{in} is converted into a set of stems $\{s\}$. In the second stems are converted back to altered inflected forms $\{if_{out}\}$ according to the set of output tags assigned to the transition.

The $weight(t, tk)$ function is used to assign weight value to tokens. It calculates the Jaccard index of two sets: token tags p and transition input tags $itag(t)$ and then multiplies it by the transition weight $\mu(t)$.

$$weight(t, tk) = \frac{|p \cap itag(t)|}{|p \cup itag(t)|} \cdot \mu(t) \quad (5)$$

Weight calculation according to formula (5) is based on a certain intuition: the Jaccard index allows assessing the similarity between a prototype word appearing in the annotation, from which the rule originates and the input term.

After executing a rule, its output places may contain several tokens. Such situation is shown in Fig 3, where the place *out#1* contains two tokens, the place *out#2* three and the place *out#3* one. For the presented example six separate strings representing concept candidates, each of them having different content and weights are obtained.

C. Aggregation

Aggregation of results constitutes the final stage of text processing and concepts extraction. Actually, it is rather a chain of aggregations performed at subsequent levels (see Fig. 6). All data processed within the chain are attributed with numerical weights (scores). At each step, apart from data conversions (e.g. combining tokens into strings) the assigned weights are summarized using various norms: sum,min, max, etc.

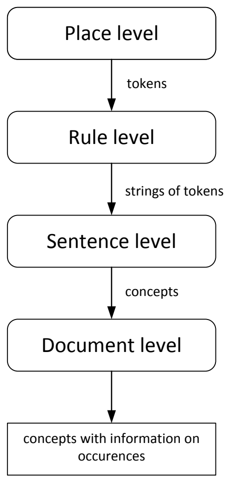

Fig. 6. The chain of aggregations

- During the aggregation occurring at the *place level* identical tokens assigned to an output places combined into one. The default norm is sum.
- Aggregation at the *rule level* aims at creating strings from sets of tokens residing in output places. In the same time weights tokens are aggregated and obtained values are assigned to strings. The default norm is min.
- Various rules may return identical concepts. At present there are no measures implemented that would allow to manage the ruleset, and in particular remove redundant rules [13]. Aggregation at the *sentence level* combines multiple strings using the selected norm (default is max).
- Finally, aggregation at the *document level* keeps track of occurrences of concepts in sentences, counts them and aggregates the weights. (The default norm is sum).

D. Implementation

The software implementing the discussed approach was implemented in Java language, what enables integration with Morfologik stemming libraries and Jena for OWL output handling. It includes the following modules:

- A tool allowing to load the Morfologik dictionary into PostgreSQL database (see Section IV)

- Synthesizer shortly described in Section IV
- Annotations compilation tool, which create rules and serializes them with JAXB libraries
- Sentence scanner allowing to split the input text into sentences and perform additional preprocessing
- Rule execution modules
- Configurable aggregation modules, e.g. it is possible to select various aggregation norms
- Output modules that renders results in various formats including CSV and OWL

VI. Experiments and results

In this section we present initial results of concept extraction from a large text file specifying medical guidelines for asthma treatment. The file is a Polish translation of a document issued by Global Initiative for Asthma (GINA) in 2011. The file size is 308KB it contains about 40000 words and 2000 sentences.

We selected this document, as it was used in previous works aiming at building ontologies of medical guidelines and performing fuzzy reasoning based on ontological models [24]. Moreover, the presented here solution was intended to be an improvement of a prototype tool (unpublished) that used internally FSM to extract concepts from texts. Both tools were tested on the same file and we wanted to compare the results of the described method and the previous one (available at http://home.agh.edu.pl/~pszwed/en/doku.php?id=ontologies#in_polishgina_guidelines_glossary). The main difference with respect to the previous solution is the focus on correct morphological form of concept names and different approach to evaluation of the accuracy of extracted concepts .

The ruleset used during the experiment resulted from compilation of an annotations file comprising translation patterns for single words and pairs of words only. Single word translations were limited to nouns and verbal nouns, 2-grams included variations comprising nouns, verbal nouns and adjectives in various cases. The resulting ruleset contained 223 rules.

The average processing time (concepts extraction without rules compilation) was about 36.785 seconds (executed on Intel Core i7-2675QM laptop at 2.20 GHz, 8GB memory under Windows 7). The extraction process returned 5151 concepts.

Table III shows selected results ordered by the weight values. It comprises top 20 results, 10 concepts from the middle and 10 concepts with the lowest weights. The word in capital letter indicates the key, i.e. the part of the concept name referring to a prospective superclass. Improperly identified concepts are underlined.

In the whole table the entry 2495 is apparently incorrect according to syntactical rules. The correct form should be: *EKONOMIKA_wdrażania*. Other underlined entries are formed correctly according to grammar rules, however their semantics does not match the document content. Position 18 WIEKO (Eng. lid) should be replaced by WIEK (Eng. age). The word NIEODPOWIEDNI (Eng. inaccurate) is rather an adjective. However, according to Morfologik it can be classified both as adjective and noun. The word SŁUŻĄCY can be interpreted as "a servant" and "serving to". In this context rather the second meaning was used over the document.

TABLE III
EXTRACTED CONCEPTS FROM THE POLISH TRANSLATION OF ASTHMA TREATMENT MEDICAL GUIDELINE

Pos	Concept	Count	weight
1	DROGI_oddechowe	161	644.00
2	LECZENIE	487	496.35
3	LECZENIE_astmy	87	348.00
4	ASTMA	965	289.50
5	ZAOSTRZENIE_astmy	42	126.00
6	LECZENIE_zaostrzenia	23	92.00
7	GLIKOKORTYKOSTEROID_wziewny	46	92.00
8	ZAOSTRZENIE	103	90.90
9	BADANIE	87	90.75
10	ROZPOZNANIE_astmy	30	90.00
11	RYZYKO	98	87.30
12	POSTĘPOWANIE	73	70.20
13	ZATOKI_przynosowe	17	68.00
14	ciężkie_ZAOSTRZENIE	22	66.00
15	PRZEPŁYW_powietrza	31	62.00
16	WYSTĘPOWANIE	65	61.50
17	ROZPOZNANIE	68	60.30
18	WIEKO	85	59.85
19	STOSOWANIE	192	57.60
20	GRUPY_wiekowe	14	56.00
2488	szkodliwa_CZĄSTKA	1	1.00
2489	DZIAŁANIE_glikokortykosteroidu	1	1.00
2490	ZAKRESY_zużycia	1	1.00
2491	JAMA_nosowa	1	1.00
2492	źródłowy_DOKUMENT	1	1.00
2493	nowoczesny_LEK	1	1.00
2494	charakterystyczna_CECHA	1	1.00
2495	EKONOMIKI_wdrażania	1	1.00
2496	OCENA_lekarska	1	1.00
2497	PRZETWÓRNIA_ryb	1	1.00
5141	MOC	1	0.04
5142	ODPOWIEDZIALNOŚĆ	1	0.04
5143	OKOLICZNOŚĆ	1	0.04
5144	NIEODPOWIEDNI	1	0.04
5145	POŚCIEL	1	0.04
5146	MNIEJSZOŚĆ	1	0.04
5147	PRACUJĄCY	1	0.04
5148	CAŁOŚĆ	1	0.04
5149	SŁUŻĄCY	1	0.04
5150	WPŁYW_CFC	1	0.04
5151	BETA	1	0.01

Fig. 7 shows extracted concepts related to the term *ryzyko* (Eng. risk) arranged into a small hierarchy. The term repeated quite frequently over the document. It can be seen that in two cases 3-grams would be more adequate, because some complements are missing, e.g. *RYZYKO_utraty* (Eng. risk of loosing).

The presented software is still in development phase, hence only preliminary results of concept extraction can be given. During future experiments several parameters controlling the execution should be tuned to provide high accuracy ratio.

The configurable parameters include norms used at various stages of aggregation (see Section V-C), as well as weights assigned to rules and individual transitions. Taking into account the complexity of rule definitions (see Fig. 5) and the number of rules, weights tuning must be performed in an automated way.

We already started to implement such mechanisms. During the experiment described in Section VI rules with single word transformation patterns were attributed with smaller weights (0.3). This value was selected quite arbitrarily.

Another example is related to handling a situation described as *tag conflict*. Quite often Morfologik identifies a word a member of two various part of speech classes, e.g. an adjective vs. a noun or a verbal noun vs. a noun. In this case, while compiling the annotations, we attribute higher weights to transitions derived from adjectives and verbal nouns (0.9), than nouns (0.1).

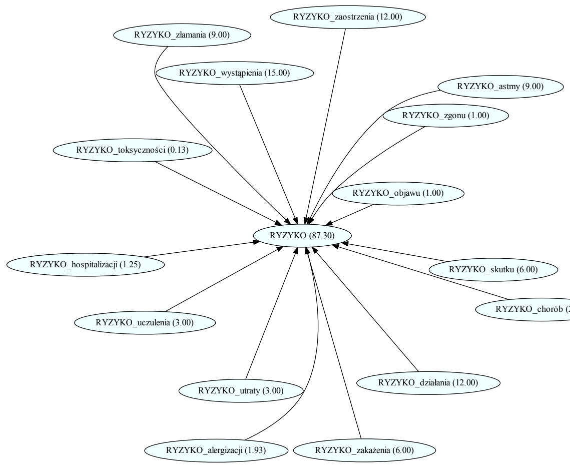

Fig. 7. Concepts related to the term *ryzyko* (risk)

VII. Conclusions

In this paper we present a solution for concept extraction from Polish texts with special focus on correct morphological forms of obtained concept names. As Polish is a highly inflected language, detected names need to be transformed following Polish grammar rules. We propose a user-friendly method for specification of transformation patterns, which is based on a simple annotations language. Annotations prepared by a user are compiled into transformation rules. During the concept extraction process the input document is split into sentences and the rules are applied to sequences of words comprised in sentences. Recognized strings forming concept names are aggregated as various levels. Although the annotation language is simple, it is flexible enough to specify various translation patterns, for example it is possible to apply inversion or to omit such tokens appearing in the input sequence as commas or prepositions.

The design of rules follow the language of Petri nets, combining elements of colored [12] and fuzzy [6] Petri nets. Rules comprise input places (filled with elements of analyzed n-grams), output places, where results are collected, and transitions linking them. Similarly to colored Petri nets, tokens are tuples – in this case they represent words and part of speech information. Tokens are also equipped with fuzzy weights. The internal behavior exhibited during rules execution borrows from fuzzy Petri nets (i.e. there is no conflict between transitions that may fire simultaneously and produce multiple tokens in output places). A similar approach was used in our previous works related to semantic event recognition and it turned out to be very efficient [26], [25].

Although we did not provide the formalization in the flavor of fuzzy rules and Fuzzy Inference Systems and fuzzy rules [22], [15], the discussed approach follows this direction. A token put in an input place receives a weight based on a kind of membership function (see Section V-B, step 3). This step resembles the fuzzification stage in fuzzy inference systems. Further, after a rule is executed, weights of output token are combined using configurable aggregation norms. This in turn corresponds to defuzzification step. Nevertheless, the transformation rules are not semantic, they are not prepared by experts and do not reference linguistic terms in the sense of fuzzy logic.

The rules are obtained fully automatically. This point also differs the presented approach from the Language tool [18] discussed in Section. II.

The developed software is tightly coupled with the Morfologik dictionary for the Polish language and accompanying library API. However, the proposed approach is quite general and can be applied to texts in other languages, provided that a language specific dictionary, as well as a software supporting lemmatization and synthesizing are available.

There are several avenues for future work. We plan to define 3-gram translation patterns and patterns comprising verbs, .e.g. *he* $\underline{\textit{reads a book}} \rightarrow \underline{\textit{book reading}}$. Further improvement to the extraction accuracy can be achieved by tuning several parameters: rule and transition weights and norms. Another direction may be related to rule management, removing redundant rules, refactoring (e.g. splitting transitions) and purging tag sets assigned to transitions.

References

[1] S. Acedański, “A morphosyntactic brill tagger for inflectional languages,” in *Advances in Natural Language Processing*. Springer, 2010, pp. 3–14.

[2] C. Blake and W. Pratt, “Better rules, fewer features: a semantic approach to selecting features from text,” in *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*. IEEE, 2001, pp. 59–66.

[3] S. Bloehdorn, P. Cimiano, and A. Hotho, “Learning ontologies to improve text clustering and classification,” in *From Data and Information Analysis to Knowledge Engineering*, ser. Studies in Classification, Data Analysis, and Knowledge Organization, M. Spiliopoulou, R. Kruse, C. Borgelt, A. Nürnberger, and W. Gaul, Eds. Springer Berlin Heidelberg, 2006, pp. 334–341. [Online]. Available: http://dx.doi.org/10.1007/3-540-31314-1_40

[4] C. Carpineto and G. Romano, *Concept data analysis: Theory and applications*. John Wiley & Sons, 2004.

[5] J. Challis, “Lateral thinking in information retrieval white paper,” Concept Searching, Tech. Rep., 2003.

[6] S.-M. Chen, J.-s. Ke, and J.-F. Chang, “Knowledge representation using fuzzy petri nets,” *Knowledge and Data Engineering, IEEE Transactions on*, vol. 2, no. 3, pp. 311–319, Sep 1990.

[7] P. Cimiano, A. Hotho, and S. Staab, “Learning concept hierarchies from text corpora using formal concept analysis.” *J. Artif. Intell. Res.(JAIR)*, vol. 24, pp. 305–339, 2005.

[8] J. Daciuk, "Incremental construction of finite-state automata and transducers, and their use in the natural language processing," Ph.D. dissertation, Gdansk University of Technology, ETI faculty, Gabriela Narutowicza 11/12, 80-233 Gdansk Poland, 1998.

[9] N. Dalvi, R. Kumar, B. Pang, R. Ramakrishnan, A. Tomkins, P. Bohannon, S. Keerthi, and S. Merugu, "A web of concepts," in *Proceedings of the twenty-eighth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. ACM, 2009, pp. 1–12.

[10] F. Graliński, K. Jassem, and M. Junczys-Dowmunt, "Psi-toolkit: A natural language processing pipeline," in *Computational Linguistics*, ser. Studies in Computational Intelligence, A. Przepiórkowski, M. Piasecki, K. Jassem, and P. Fuglewicz, Eds. Springer Berlin Heidelberg, 2013, vol. 458, pp. 27–39. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-34399-5_2

[11] D. Janus, "Smyrna prosty konkordancer obsługujący język polski," 2015, accessed: May 2015. [Online]. Available: http://smyrna.danieljanus.pl/

[12] K. Jensen, *Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use*. Springer, 1996, vol. 1, no. Basic Concepts.

[13] A. Ligeza, *Principles of Verification of Rule-Based Systems*. Springer, 2006.

[14] A. Maedche and S. Staab, "Ontology learning for the semantic web," *Intelligent Systems, IEEE*, vol. 16, no. 2, pp. 72–79, Mar 2001.

[15] E. H. Mamdani and S. Assilian, "An experiment in linguistic synthesis with a fuzzy logic controller," *International Journal of ManMachine Studies*, vol. 7, no. 1, pp. 1–13, 1975. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0020737375800022

[16] M. Miłkowski, "Developing an open-source, rule-based proofreading tool," *Software: Practice and Experience*, vol. 40, no. 7, pp. 543–566, 2010.

[17] ——, "Morfologik," 2015, accessed: May 2015. [Online]. Available: http://morfologik.blogspot.com/

[18] D. Naber, "Language tool style and grammar check," 2015, accessed: May 2015. [Online]. Available: https://www.languagetool.org/

[19] S. Osinski and D. Weiss, "A concept-driven algorithm for clustering search results," *Intelligent Systems, IEEE*, vol. 20, no. 3, pp. 48–54, 2005.

[20] A. Parameswaran, H. Garcia-Molina, and A. Rajaraman, "Towards the web of concepts: Extracting concepts from large datasets," *Proceedings of the VLDB Endowment*, vol. 3, no. 1-2, pp. 566–577, 2010.

[21] P. Pęzik, "Wyszukiwarka PELCRA dla danych NKJP," 2012.

[22] T. Ross, *Fuzzy Logic with Engineering Applications*. Wiley, 2009.

[23] A. Stavrianou, P. Andritsos, and N. Nicoloyannis, "Overview and semantic issues of text mining," *ACM Sigmod Record*, vol. 36, no. 3, pp. 23–34, 2007.

[24] P. Szwed, "Application of fuzzy ontological reasoning in an implementation of medical guidelines," in *Human System Interaction (HSI), 2013 The 6th International Conference on*, June 2013, pp. 342–349.

[25] ——, "Video event recognition with Fuzzy Semantic Petri Nets," in *Man-Machine Interactions 3*, ser. Advances in Intelligent Systems and Computing, A. Gruca, T. Czachórski, and S. Kozielski, Eds. Springer International Publishing, 2014, vol. 242, pp. 431–439. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-02309-0_47

[26] P. Szwed and M. Komorkiewicz, "Object tracking and video event recognition with fuzzy semantic petri nets," in *Proceedings of the 2013 Federated Conference on Computer Science and Information Systems, Kraków, Poland, September 8-11, 2013.*, M. Ganzha, L. A. Maciaszek, and M. Paprzycki, Eds., 2013, pp. 167–174. [Online]. Available: http://fedcsis.org/2013/

[27] M. Wolinski, M. Milkowski, M. Ogrodniczuk, and A. Przepiórkowski, "Polimorf: a (not so) new open morphological dictionary for polish." in *LREC*, 2012, pp. 860–864.