Recent Advances in Intelligent Information Systems
ISBN 978-83-60434-59-8, pages 51–64

Automatic Construction of the Polish Nominal Lexicon for the OpenCyc Ontology

Aleksander Pohl[1,2]

[1] Jagiellonian University, Cracow, Poland
[2] University of Science and Technology, Cracow, Poland

Abstract

This paper describes the automatic construction of the Polish nominal lexicon for the OpenCyc ontology. The ontology is shipped with the English lexicon, thus the goal is achieved by translating the English words corresponding to the OpenCyc symbols into Polish. As a result the mapping between the ontology symbols and the Polish words is created. The developed algorithm utilizes semantic properties of the ontology, as well as a machine readable English-Polish dictionary („Wielki Multimedialny Słownik angielsko-polski polsko-angielski Oxford/PWN"). The algorithm is heuristics-based and purely symbolic.

Keywords: Cyc, dictionary, lexicon, ontology

1 Introduction and Motivation

The OpenCyc[1] ontology (Lenat and Guha, 1990; Lenat, 1995) is well known for its breadth and depth and was reported to be useful in the field of NLP (cf. Curtis *et al.*, 2006) as well as general AI (cf. Schneider *et al.*, 2005). The recent interest in the Semantic Web is not passed over by the creators of Cyc – a part of its contents[2] is available in the form of OWL file; moreover each Cyc concept has an URI pointing to its definition presented in the form of RDF data[3]. The fact that Cyc was chosen one of the key technologies in the €10 Million EU-funded LarKC project (Fensel *et al.*, 2008) is even more compelling.

Still at present this large ontology, empowered with a sophisticated inferencing engine, is only useful for the English-speaking community. Even though it was designed in a language-agnostic manner[4], it contains only the English lexicon, that is a mapping between the Cyc symbols and the English words. By contrast, the other popular „ontology" – the English WordNet (Fellbaum, 1998) – is translated

[1]The OpenCyc ontology is publicly available version of the Cyc ontology. Since its content is the same as the full version, we will use the names *Cyc* and *OpenCyc* conversely.

[2]Covering the `#$isa` and `#$genls` relations.

[3]See: `http://sw.opencyc.org`

[4]This might be obscured by the fact, that Cyc symbols have English names – still they need to be mapped to English words. On the other hand – the language-agnosticism is hard to defend against the famous Sapir-Whorf hypothesis.

into many languages, despite the fact that its logical expressiveness[5] is far much lower.

The gap between Cyc and natural languages other than English, might be bridged in a number of ways. First of all Cyc is mapped to the English WordNet (EnWN), so provided that there is a mapping between given language and EnWN, it could be exploited to create Cyc lexicon for this language. However this approach has a number of limitations. In OpenCyc only 11297 concepts[6] were mapped to EnWN synsets. The other problem is that, the mapping between given language and EnWN as well as EnWN and Cyc is never 100% accurate, and this would significantly impact the accuracy of the obtained mapping. But the most important problem, in the context of Polish, is that even though there are attempts to build the Polish WordNet (Vetulani *et al.*, 2007; Derwojedowa *et al.*, 2008), they are not yet finished[7].

The other way of building discussed lexicon, could be based on the fact that recently the ontology was automatically mapped to the English Wikipedia (Medelyan and Legg, 2008) which is massively linked to the Polish Wikipedia. The results of the mapping are quite good in terms of accuracy (93%) and recall (52,690 Cyc concepts were mapped to the Wikipedia articles). This way of building the lexicon is promising, but since the latest downloadable OpenCyc doesn't contain the mapping, it is still to be investigated.

The way of building the lexicon discussed in this article is similar to the way EnWN is automatically translated into other languages – i.e. exploits a machine readable bilingual dictionary (Farreres and Rigau, 1998). But since WordNet and Cyc have different structure (e.g. the basic unit of meaning in WordNet is a synset, while in Cyc a constant, which might be a collection, an individual, a relation or even a function), the techniques developed for WordNet has to be adopted or even replaced.

The motivation for the construction of the Polish lexicon is manifold. The first obvious result of the mapping would be the ability to search the contents of the ontology using Polish words, to display the symbols' definitions with Polish titles, etc. thus making it easier accessible for the Polish research community. The second result (achievable only with additional effort), would be the ability to identify and describe senses of Polish words in terms of Cyc symbols, thus to create a Cyc-based Polish semantic dictionary. The last, rather long-term result, would be the ability to use the knowledge stored in it and its inferencing engine in sophisticated NLP algorithms for Polish. Still, since the described algorithm processes only nominals, it is just the first step in construction of the full Polish lexicon. Its main purpose is to give an answer whether semi-automatic construction of it is feasible in a reasonable amount of time.

Sections 2 and 3 of this article contain the description of the resources used in the construction of the lexicon – namely the OpenCyc ontology and the English-Polish dictionary. Section 4 discuss the details of the algorithm used for the lexicon construction. Section 5 contains the results of the algorithm, while section

[5]Measured by the number of employed relations, means for knowledge organization, etc.

[6]All numbers are checked against the OpenCyc v. 1.0, KB: 5006, System: 1.11058.2.20.2.3

[7]After the experiment was finished, it was reported that the Polish WordNet containing approx. 20 thousands of synsets was constructed.

6 presents some conclusions of the work.

2 The OpenCyc ontology

The OpenCyc[8] ontology is a publicly available version of the Cyc[9] ontology, which is developed since 1984 by Lenat and Guha (1990) and the ©Cycorp company. According to Mascardi *et al.* (2006) it is one of the few upper ontologies available nowadays and doubtlessly it is the biggest of them: it contains approx. 300 thousands of concepts, nearly 3 millions of assertions and 15 thousands of relations. It is distributed with a compiled inferencing engine, which contains many submodules designed to speed up the inferencing in particular domains (taxonomy, space and time relations, etc.).

The Cyc ontology provides basic notions, such as concepts, individuals, predicates, functions, etc., as well as more complex ones, e.g.: time, space and modalities, which are represented by Cyc symbols. They take the form of `#$NAME`, where NAME, is the name of the symbol. It is usually some English word, but since words are ambiguous and symbols cannot be, it may be some concatenation of English words or its abbreviations, e.g. `#$BankTopographical`, `#$BankOrganization`, `#$MakingABankDeposite`, `#$MotherFn` and so on.

2.1 Lexicon

The fact that given Cyc symbol is mapped to some words may be expressed in a number of ways:

1. by the `#$synonymousExternalConcept` predicate
2. by the `#$denotation` predicate
3. by the `#$prettyString` and `#$prettyString-Canonical` predicates

The multiplicity of choices seems to be odd at the first glance, but we shall explain it briefly. In fact the first option is only useful when we map a concept to the other concept[10] in a different lexical resource, such as WordNet. Even though this is not a „direct" mapping (we need to look into that resource, and check the actual word), it might be the only option in case we wish to rapidly extend the ontology. As it was stated – Cyc was designed in a language-agnostic manner, but when it comes to the lexicon, the language-agnosticism has to be dropped, and it is obvious that the means for expressing the language-specific information are English-biased (e.g. there is no way to represent noun cases). Thus instead of extending the way Cyc „understands" words, in particular circumstances it's easier to give it pointers to external resource, which is better suited for storing that information.

Employing the `#$denotation` predicate is the opposite of the first option – it binds Cyc symbols with other Cyc symbols, namely the symbols which represent words. By convention they have the form of `#$X-TheWord`, where X is the represented word. On the one hand, this allows assertions concerning the word-symbols

[8]See: `http://www.opencyc.org`
[9]See: `http://www.cyc.com`
[10]Which has some words attached to it.

to be stated[11], on the other – the actual string corresponding to given symbol[12] is not bound directly.

The last option, employing the `#$prettyString` and `#$prettyString-Canonical` predicates is an intermediate solution. Unlike the first option – the string corresponding to given Cyc symbol is stored directly in the ontology, unlike the second – it doesn't allow to express any additional information about the word. Both predicates require two arguments of which the first is a Cyc symbol and the second is a string, e.g. `(#$prettyString #$Dog „hound'')`. The difference between them is that the first predicate may be employed many times for given symbol, while the second – only once (within given microtheory). As the name suggests – the second predicate is used to indicate the „canonical" string representation of the symbol, e.g. `(#$prettyString-Canonical #$Dog „dog'')`.

It is not surprising, that in the OpenCyc the lexicon is described in terms of the `#$prettyString` and `#$prettyString-Canonical` predicates. Even though there is a mapping between Cyc symbols and EnWN synsets, it is far from being complete (covers only 11 thousands of the Cyc symbols), while the `#$prettyString--Canonical` is applied more than 230 thousands times. On the other hand – employment of the `#$denotation` predicate would require much more work (the mapping was introduced in the last version of the OpenCyc ontology).

When it comes to the creation of Polish lexicon for the Cyc ontology, the most important question is: „how to obtain symbols mapped to given English word?". The answer is short: there is a function in the SubL language, namely `denotation-mapper`, which returns all the symbols mapped to the string being the argument of the function, bound by the `#$denotation` predicate, as well as `#$prettyString` and `#$prettyString-Canonical` predicates. Obviously the results of the function are ambiguous, e.g. `(denotation-mapper „dog'') : ((„dog'' .#$Dog)(„dog'' . #$HotDog))`, (which means that the English word „dog" is mapped in Cyc to the `#$Dog` and `#$HotDog` symbols) and have to be disambiguated against the Polish translations present in the bilingual dictionary.

2.2 Taxonomy

The knowledge in Cyc is organised around the taxonomy of concepts. All symbols (i.e. concepts, relations, functions, etc.) belong to the root collection: `#$Thing`. Collections belong to the `#$Collection` collection, individuals to the `#$Individual` collection, relations to the `#$Relation` collection, and so on.

The `#$Collection` collection contains all collections, of which first-order collections may be perceived as concepts. Collections can contain (via `#$isa` predicate) other collections or individuals, and may be generalized (via `#$genls` predicate) to other collections, but do not have any space-time qualities nor parts. On the other hand – individuals, such as the `#$EiffelTower`, which belong to the `#$Individual` collection, may have some space-time qualities (such as height) and may have parts, but they cannot contain any other individuals nor collections and cannot be generalized.

[11] Such as the part of speech of the word.
[12] Mapped by the `#$denotation` predicate.

3 English-Polish Dictionary (GMD)

The second resource which was used in the construction of the lexicon was the „Wielki Multimedialny Słownik Angielsko-Polski Polsko-Angielski Oxford/PWN”[13] which will be abbreviated as GMD (Great Multimedia Dictionary) (GMD, 2004). It contains 77684 entries[14], of which 47321 are nominals. The earlier version of the dictionary was used by Jassem (2004) in machine translation system *Translatica*. That paper describes the transformation of the human readable dictionary to the format applicable for machine translation, but the result is not publicly available.

In this section we will describe GMD from the point of view of the automatic translation of the Cyc lexicon. This task is in fact simpler than the one described in the mentioned paper, since e.g. the syntactic information, the compound expressions, as well as usage examples didn't have to be treated so carefully. We concentrate on simple translations (one or two words) and qualifications which are very useful in the disambiguation process. On the other hand the parsing of the dictionary definitions was much more harder, since the described version of the dictionary, didn't have special XML tags to distinguish between say the translation and the qualification[15].

First of all, we shall summarize the issues described by Jassem (2004) and briefly present our treatment of them. Then we shall discuss the features of the dictionary entries, which are particularly important in creating the Polish lexicon for the Cyc ontology. Finally we shall describe the method used to parse the dictionary definitions.

3.1 Usage obstacles

The first problem described by Jassem (2004, pages 100–101) concerns the separation of entries, that is the problem of creating a consistent index of definitions. The author lists the following detailed issues:

- References – some entries were merely references to other entries. In our treatment of the problem we simply attached the referenced definition to the original entry.
- Graphical variants of entries:
  - *bias(s)ed* – the middle „s” in parenthesis is optional. That issue was quite important since in many cases only one variant was recognizable by the Cyc `denotation-mapper` function. It was solved during the entries index construction by creating two entries pointing to the same definition.
  - *Balkanization*, *balkanization* – the first letter capitalization. This problem was ignored, since the used function is case-insensitive.
  - *baldachin, baldaquin* – the variants are separated by comma. Like in the first case – the problem was important and solved during the index construction by duplication of the entries.

[13]The Great English-Polish Polish-English Multimedia Dictionary Oxford/PWN

[14]Two words which have the same base form, but are different parts of speech, are considered different entries.

[15]In the version described by Jassem (2004, page 101) there was special tag: `COLL` to separate the qualification from the translation.

- *billet*[1]*, billet*[2] – separation of the entries which are the same parts of speech. The definitions were merged.
- *behalf* – entries lacking direct equivalents. These entries were ignored.
- Phrasal verbs – ignored, since the constructed lexicon contains only nominals.

The second problem was the acquisition of attribute values (Jassem, 2004, page 101):

- The flexional descriptions were simply ignored, as they were not needed.
- The syntactical information on the nominal complements was important, since it was used in the disambiguation. Unfortunately the XML tag `INDIC`, was not present in the used version of the dictionary. This issues is discussed later.
- Semantic attributes were particularly important during the automatic disambiguation. Like in the previous case, the special tag `COLL` was not present.
- Only the „domain" of the *context* attribute was taken into account, since the other (i.e. „style" and „dialect") were not useful in the construction of the Cyc lexicon.

We skip the last problem discussed in Jassem (2004, pages 101–102), namely the merging of senses, since it was integrated with the translation algorithm and thus treated differently.

3.2 Entry features

There are two types of features of the entries particularly important from the point of view of the automatic construction of the Polish Cyc lexicon. The first is the number of different translations connected with one English „entry"[16] and the second – the means for differentiating these translations.

One might think, that in the first respect the only important distinction is between entries having one and entries having many translation. The reality is quite different. First of all – the definition connected with given entry might contain sections concerning different parts of speech. These sections were marked with Latin digits and abbreviations for names of parts of speech. Since the constructed lexicon contains only nominals, three types of abbreviations were taken into account:

1. *n* – common nouns
2. *npl* – *plurale tantum* nouns
3. *prn* – proper names

The treatment of the nominals belonging to the first and the second group should be the same, as the syntactical information could not be used, due to the limitations of the `#$prettyString` predicated used in the OpenCyc ontology. But the presence of the last abbreviation was quite important – it indicated, that the concept corresponding to the translation is an individual, so it should belong (via `#$isa` predicate) to the `#$Individual` collection. As a consequence – even if there

[16]We intentionally don't use the more precise „lexeme" term, since the „entry" term retains the vagueness of its definition.

arch^1 /a:tʃ/
I *n*
1. `Archit` (dome) łuk *m*, łęk *m*; (archway) sklepione przejście *n*; (of bridge) przęsło *n*; (triumphal) łuk *m*
2. `Anat` (of foot) podbicie *n*; (of eyebrows) łuk *m*; **to have fallen arches** mieć płaskostopie poprzeczne
II *vt* wygi|ąć, -nać w łuk *or* pałąk; ...

Figure 1: Sample entry of the GMD (2004).

was more than one translation, the situation might be further differentiated for cases where there were nominals only from the first and the second *or* from the third group and cases where the nominals belonged to all of them.

Still the part of speech separation was not the lowest level of granulation – many of such entries were further split into something we called semantic groups or sub-entries. Each semantic group was marked with Arabic digit, had some semantic or pragmatic qualification and could be further divided into smaller groups of translations separated by semicolons (each containing their own qualifications, etc.) and commas. The sample entry is given on figure 1.

Such organization of the entry definition imposed some restrictions on the translation algorithm. It was clear that treating every single translation in separation won't be the best option. In the basic situation, the Cyc concepts shouldn't be mapped against them, but rather against semantic sub-entries. This problem is discussed in section 4.1.

The second type of feature of the dictionary entries – i.e. the means for differentiating the translations within one definition – was further divided into three categories:

1. *paradigmatic* qualifications
2. *syntagmatic* qualifications
3. *domain* qualifications

All of them are present in the sample entry. The first category covered situations in which the qualification (given in parenthesis) was (in most cases) a generalization or a synonym of the concept described by the entry (cf. *arch* and *dome* which are connected by the paradigmatic relation of hyperonymy).

The second category covered situations in which the qualification (also given in parenthesis) was connected with the described concept by some syntagmatic relation (cf. *arch of bridge*). In most cases the first category of features was easily distinguishable from the second, by the presence of a preposition in the latter[17].

The last category covered situations in which the qualification was an abbreviation of the domain of the concept (cf. *Archit* – architecture, *Anat* – anatomy). Since the set of the domain abbreviations was fixed it was quite easy to extract them.

[17]But in some rare cases (cf. *triumphal bridge*) even though the qualification was syntagmatic, there was no preposition.

Each of these groups of features was treated differently during the process of lexicon translation, so it was important to clearly distinguish between them. Even though in the general situation it was quite easy, the very flexible, not to say inaccurate, structure of the entry definition imposed huge problems on the appropriate ascription of the qualifications to the translations. This problem is discussed briefly in the next section.

3.3 The structure and parsing of entries

The structure of the entry definition was quite flexible and complicated, as it was mentioned in the previous paragraph. In fact, there was no one structure – there were many structures used on different occasions. In most of the cases given entry had only one part of speech section attached, thus there were no Latin digit at the beginning of it. In many cases the definition for given POS was not split into semantic groups. Within given semantic group, there were qualifications applied to the whole group, as well as those applied only to some translations. Often the syntagmatic qualifications were juxtaposed within the parentheses, but the preposition was attached only to the first word. In some cases the qualifications were in front, while in the other at the end of the translations. There are countless examples of such inconsistencies in the structure of the entry. This is not a surprise, since the dictionary was designed to be used by humans, but it entailed measurable effort for the designer of the parsing algorithm, to cope with all of them.

What is more – even though the definitions were written in SGML, there was no separation between the visual and the semantic tags, e.g. the sections for different POSes were separated only by `<P>` (paragraph) and `<IMG>` (image of the Latin digit) tags. The only strictly semantic tags used were `<GB>` and `<PL>` indicating that given segment contains English or Polish words.

While creating the parsing algorithm, it quickly appeared that there is no other way of achieving the accurate result, than preparing many hand-crafted rules, since the general-enough assumptions lead to very poor effects. The parsing was based on a finite-state automata, whose transition table was created manually. A special tool was written to visualise and modify it on-the-fly. Suffice it to say, that after few man-weeks of the construction of the transition table, the automata was good enough to correctly[18] recognize 99,5% out of more than 70 thousands entries[19].

4 Cyc lexicon translation algorithm

The goal of the algorithm designed for the lexicon translation can be stated as follows: „for given Cyc symbol find as many as possible Polish words, which correspond to it". Since the correspondence of Polish and English words is described in the dictionary, the algorithm treated it is as the primary resource. Thus in general it looks as follows:

[18]The correctness was checked loosely – there might be up to 4 consequent symbols, which didn't cause any transition.

[19]The detailed description of the structure of entries, the algorithm and the visual tool will be described in separate paper.

1. Take the first entry from the English-Polish dictionary
2. For given entry find all Cyc symbols corresponding to it (`Cyc-set`)
3. For given entry extract from its definition all Polish sub-entries within the nominal sections (`Polish-set`)
4. If the entry is not a proper name, from the `Cyc-set` remove symbol which are individuals `(#$isa symbol #$Individual)`[20]
5. Try to find the best matching between the elements of the `Cyc-set` and the `Polish-set`
6. If there are not-processed entries in the dictionary, take the next one and go to the 2nd step.
7. Post-process the matchings between Cyc symbols and Polish words.

The most problematic step of the algorithm is the 5th, since it covers the disambiguation of Polish words against the Cyc concepts. The last step is also important, because it is crucial for the precision of the algorithm (it allows to remove invalid matchings from the final result).

4.1 Mapping heuristics

A number of heuristics were applied in the 5th step – they were based on the features of the entry definitions described in section 3.2. We shall call them *semantic* if they were based on the qualifications of the translations and *grouping* if they took into account the cardinality of the `Cyc-set` and `Polish-set`. Furthermore a system of scoring was devised to reflect the confidence of the mapping:

- *strong* confidence – for mappings created according to the semantic heuristics and trivial one-to-one mappings
- *medium* confidence – for mappings created according to some grouping heuristics
- *weak* confidence – for mappings which were preserved only if for given word there were no other mappings with higher confidence

The application of the heuristics was interwoven and started from the trivial grouping heuristic, but we shall first describe the heuristics from the semantic group.

The **paradigmatic** heuristic took into account the paradigmatic qualifications of given Polish word. The heuristic highlighted given Cyc symbol s as corresponding to given word w iff:

$$\exists w_1 \in W_1 \colon genl?(s, w_1) \vee genl?(w_1, s) \tag{1}$$

where `genl?` corresponds to the Cyc `#$genl` predicate[21] and W_1 is a union of sets of symbols returned by the `denotation-mapper` function for each paradigmatic

[20]In fact this step was more sophisticated, since some of the symbols, which are individuals according to Cyc, are not proper names, e.g. `#$Meter`. It turned out, that the lack of clear separation between proper names and common nouns, significantly influenced the accuracy of the algorithm.

[21]Which is described in section 2.2.

qualification of the word w. What might be surprising is the second term of the alternative, since, as it was stated in section 3.2, the paradigmatic qualification should be more general than the described word, so the corresponding Cyc symbols, also should be more general than the symbol in question. However it appeared that in some cases the relation was turned over.

The **domain** heuristic was based on the domain qualifications. For each domain there were taken the most general concepts from Cyc which in our opinion were closely related to it. E.g. for the *Botany* domain there were chosen three symbols `#$Plant`, `#$NaturalTangibleStuff` and `#$OrganismPart`. Thus each domain d_i had a set of symbols $D_i = \{s_1, s_2, s_3, ...\}$ attached to it. The heuristic highlighted given Cyc symbol s as corresponding to given word w iff:

$$\exists s_j \in D\colon genl?(s, s_j) \tag{2}$$

where D is the union of sets attached to all the domain qualifications of the word w.

The **syntagmatic** heuristic was the most problematic. Even though in the Cyc there are means for expressing syntagmatic relations (like the `#$conceptually-Related` predicate) or at least we can specify a query, that will scan over the relations trying to find something that corresponds to it, they are not easily accessible. We tried the query:

```
(cyc-query '(#$thereExists ?col1 (#$thereExists ?pred
  (#$thereExists ?index1 (#$thereExists ?index2
    (#$and (#$argIsa ?pred ?index1 ?col1)
      (#$argIsa ?pred ?index2 ?col2) (#$genls term1 ?col1)
      (#$genls term2 ?col2)))))) #$UniversalVocabularyMt)
```

Where `term1` and `term2` were substituted with actual values, but the inference was very slow. So we had to devise more robust solution. Analyzing a number of examples we arrived at solution based on the symbols used in the pragmatic heuristic, corresponding to some general semantic categories (such as Abstract-Object, Animal, BodyPart, Plant, etc.). For each of these categories c_i we defined a set of semantically close Cyc symbols $C_i = \{s_1, s_2, s_3, ...\}$, such that the concepts covered by them (via the `#$genl` predicate) might be connected with symbols belonging to the category (via the `#$genl` predicate) by means of syntagmatic relations.

For instance the English word „foot" which has a number of meanings, of which one is an animal body part (`#$Foot-AnimalBodyPart` in Cyc and *stopa, łapa, noga* in Polish) fallen into BodyPart category. This category was connected with the `#$BiologicalLivingObject` symbol, as the syntagmatic qualifications present in the entry definition for „foot" were names of living objects (*of cat, dog* $\rightarrow$ *łapa*, *of bird, insect* $\rightarrow$ *noga*).

The heuristic highlighted given Cyc symbol s as corresponding to given word w iff:

$$\exists c_i \exists s_j \exists s_k\colon c_i \in C \wedge s_j \in syn(c_i) \wedge s_k \in S \wedge genl?(s, c_i) \wedge genl?(s_k, s_j) \tag{3}$$

where C is the set of general semantic categories, $syn(c_i)$ is a set of Cyc symbols semantically close to the category c_i and S is the union of sets of symbols returned

TABLE 1: Overall accuracy and recall of the algorithm

Cyc symbol type	mapped/all	strong con.	medium con.	weak con.	**overall**
`#$Collection`	8247/40137	69.21%	53.87%	21.59%	**58.14%**
`#$Individual`	1871/55250	37.71%	35.96%	26.47%	**35.50%**
`#$Relation`	604/13742	68.31%	63.46%	50.0%	**65.26%**
NART	1297/133740	56.89%	47.10%	22.5%	**47.65%**
all	**12019/242869**	**64.67%**	**49.84%**	**23.09%**	**54.34%**

by the `denotation-mapper` function for each syntagmatic qualification of the word *w*.

If any of the semantic heuristics highlighted given word as corresponding to given Cyc symbol, they were mapped with strong confidence.

As it was stated, the *grouping* heuristics took into account the cardinality of the `Cyc-set` and the `Polish-set` that is the number of Cyc symbols returned for given English word by the `denotation-mapper` function and number of semantic groups present in the entry definition. There were four general cases (`#Polish-set` to `#Cyc-set`):

1. 1 to 1 – trivial heuristic: map each word in the semantic group to the Cyc concept with strong confidence
2. 1 to n – apply the semantic heuristics first, if without success, map each word in the semantic group to each Cyc concept with medium confidence
3. n to 1 – apply the semantic heuristics first, if without success, map each word in the first group with medium confidence and words in other groups with weak confidence (this heuristic reflects the fact that the most popular meaning of a word is listed first in its definition).
4. n to n – apply the semantic heuristics first. If with success, remove the matched words/word groups and concepts from each set. If the case is reduced to one of the previous cases, apply the appropriate heuristic with lower confidence (strong $\rightarrow$ medium, medium $\rightarrow$ weak). If not – create the Cartesian product of the sets and mark each mapping with weak confidence.

4.2 Post-processing of the result

At the end of the algorithm there were some post-processing procedures applied. First of all, the mappings were sorted and grouped by Polish words, that is all mappings for given Polish word were put into one group. Then, according to the semantics of the weak mapping confidence, if in given group there were mappings with weak and higher confidence, the former were removed. At the end the mappings were checked pairwise, and if one of the mapped symbols was generalization of the other, the mapping corresponding to the latter was removed.

5 Results

The overall accuracy of the algorithm is presented in table 1. In general more than 12 thousands out of 240 thousand symbols were mapped with overall accuracy

Table 2: Detailed accuracy of the algorithm for general semantic categories

	AbstractObj	**Animal**	**Artifact**	**BodyPart**	**Event**	**Food**
size	4652	878	4807	758	6957	489
strong	48.29%	87.5%	44.86%	70.42%	54.69%	84.31%
medium	38.97%	61.64%	32.69%	84.21%	32.30%	60.0%
weak	22.22%	18.75%	31.11%	15.38%	16.26%	21.42%
overall	**42.39%**	**76.42%**	**40.22%**	**66.01%**	**35.29%**	**67.77%**
	Human	**Instrument**	**Location**	**Meter**	**NaturalObj**	**Plant**
size	2551	3486	2373	110	1432	208
strong	80.23%	57.26%	62.42%	91.89%	76.92%	97.5%
medium	79.10%	54.90%	63.43%	80.95%	60.52%	83.33%
weak	29.62%	12.0%	14.28%	100.0%	69.23%	25.0%
overall	**74.71%**	**54.37%**	**59.61%**	**88.33%**	**72.61%**	**89.28%**
	Proper	**Self**	**Set**	**State**	**Structure**	—
size	168	659	592	1590	358	—
strong	79.31%	53.84%	51.61%	82.95%	60.60%	—
medium	73.03%	62.5%	30.0%	69.23%	31.81%	—
weak	54.54%	46.42%	10.0%	37.14%	0.0%	—
overall	**72.86%**	**54.65%**	**39.34%**	**69.71%**	**41.53%**	—

reaching 54%, tested against 3467 out of approx. 27 thousands of mappings ($\sim$ 12%). The results for the mappings with strong confidence are significantly better, reaching 64%.

Following results should be highlighted:

- the symbols belonging to the `#$Collection` collection, which may be perceived as concepts, and the symbols belonging to the `#$Relation` collection, have the highest accuracy (reaching 70% for the strong confidence)
- the symbols belonging to the `#$Individual` collection, which may be perceived as proper names, have significantly worse accuracy reaching only 38% for the strongest confidence
- the NARTs (Non-atomic reified terms, such as `(#$JuvenileFn #$HomoSapiens)`) which may be perceived as compound expressions, were mapped with moderate accuracy

In table 2 there are presented details of the accuracy of the algorithm for each of the general semantic categories mentioned in the description of the *domain* heuristic (cf. section 4.1). For each of the category, at least 10% of the mappings were checked. It should be stressed, that given word could belong to more than one category, so the sum of the categories' sizes is not equal to the number of mappings.

Following results should be highlighted concerning the semantic categories:

- for some of the categories (cf. Animal, Food, Human, Meter, Plant and State) the accuracy for the strong confidence is greater than 80% which is quite good result
- only for two categories, namely BodyPart and Self[22], the accuracy of mappings

[22]Category containing symbols which did not fit to any other category.

with medium confidence is better than those with strong confidence

- the accuracy of the mapping for the largest categories (AbstractObj, Artifact, Event and Instrument) is worse than average
- the largest category, namely Event, has the worst overall accuracy

6 Conclusions

It is hard to compare the results of the described algorithm with results obtained in the automatic translation of the English WordNet to other languages, since the structure and contents of Cyc and EnWN are significantly different. What is more – most of the algorithms devised for the translation of WN, was based on the semantic unit of it, that is a synset, which doesn't have its counterpart in the Cyc ontology. That's why we shall draw some conclusions based only on the results of the described algorithm.

First of all the idea of the confidence level has proved its usefulness – in most of the cases the mappings with strong confidence were more accurate than those with medium confidence, while these being more accurate than those with weak confidence.

Second important observation concerned the proper names mapped to symbols from the `#$Individual` collection. While the proper names (belonging to the Proper general semantic category) were mapped with high accuracy (54% – 80%), other mappings of the Cyc symbols from the `#$Individual` collection significantly worsened the overall result. This is caused by the fact, that not all individuals are proper names (e.g. meter, gram, minute, etc.), but the OpenCyc doesn't distinguish between them and common nouns. As a consequence, some more sophisticated heuristics should be devised concerning this special case.

On the other hand, the fact that mappings of symbols belonging to the `#$Collection` collection and to the `#$Relation` collection have high accuracy ($\sim$ 70% for the strong confidence) means that these two types of symbols whose semantics is easily recognizable are mapped with ease.

The conclusions which may be drawn from the results for the semantic categories are as follows. It appeared that the smaller the category was, the better the results. It is probable, that by developing a larger number of categories (by splitting AbstractObj, Artifact, Event and Instrument categories, to finer-grained ones) would lead to significantly better performance of the algorithm.

The lack of lexical categorization (e.g. part of speech tags) in the English lexicon of OpenCyc caused major problem for the algorithm, which may be observed in the case of Event category. The algorithm has no means to distinguish between nouns (e.g. „arrest" `#$arrests` → pol. *aresztowanie*) and verbs („to arrest" `#$ArrestingSomeone` → pol. *aresztować*), what caused many incorrect mappings.

All these issues will be taken into account in the further development of the algorithm, for the purpose of planed mapping of the Cyc concepts to Polish verbs and adjectives.

References

Jon Curtis, John Cabral, and David Baxter (2006), On the application of the Cyc ontology to word sense disambiguation, in *Proceedings of the 19th International Florida Artificial Intelligence Research Society Conference*, pp. 652–657.

Magdalena Derwojedowa, Maciej Piasecki, Stanisław Szpakowicz, Magdalena Zawisławska, and Bartosz Broda (2008), Words, Concepts and Relations in the Construction of Polish WordNet, in *Proceedings of the Global WordNet Conference, Seged, Hungary*, pp. 162–177.

Xavier Farreres and German Rigau (1998), Using WordNet for Building WordNets, in *Proceedings of COLING-ACL Workshop on Usage of WordNet in Natural Language Processing Systems*, pp. 65–72.

Christiane Fellbaum (1998), *WordNet: An Electronic Lexical Database*, MIT Press.

Dieter Fensel, Frank van Harmelen, Bo Andersson, Paul Brennan, Hamish Cunningham, Emanuele Della Valle, Florian Fischer, Zhisheng Huang, Atanas Kiryakov, Tony Kyung il Lee, Lael School, Volker Tresp, Stefan Wesner, Michael Witbrock, and Ning Zhong (2008), Towards LarKC: a Platform for Web-scale Reasoning, URL `http://www.larkc.eu/wp-content/uploads/2008/05/larkc--icsc08.pdf`, to appear.

Oxford/PWN GMD (2004), Wielki Multimedialny Słownik Angielsko-Polski Polsko-Angielski Oxford/PWN.

Krzysztof Jassem (2004), Applying Oxford-PWN English-Polish Dictionary to Machine Translation, in *Proceedings of the 9th EAMT Workshop, „Broadening horizons of machine translation and its applications”*, pp. 98–105.

Douglas B. Lenat (1995), CYC: A Large-Scale Investment in Knowledge Infrastructure, *Communications of the ACM*, 38(11):33–38.

Douglas B. Lenat and Ramanathan V. Guha (1990), *Building Large Knowledge-Based Systems*, Addison Wesley.

Viviana Mascardi, Valentina Cordì, and Paolo Rosso (2006), A comparison of upper ontologies, Technical report.

Olena Medelyan and Catherine Legg (2008), Integrating Cyc and Wikipedia: Folksonomy meets rigorously defined common-sense, in *Proceedings of the WikiAI Workshop at AAAI-2008*, to appear.

David Schneider, Cynthia Matuszek, Purvesh Shah, and Robert Kahlert (2005), Gathering and Managing Facts for Intelligence Analysis, in *Proceedings of the International Conference on Intelligence Analysis. (2005).*

Zygmunt Vetulani, Justyna Walkowska, Tomasz Obrębski, Paweł Konieczka, Przemysław Rzepecki, and Jacek Marciniak (2007), PolNet – Polish WordNet project algorithm, in *Proceedings of 3rd Language & Technology Conference*, pp. 172–176.