A Query Language for WordNet-Like Lexical Databases

Marek Kubis

Uniwersytet im. Adama Mickiewicza,
Faculty of Mathematics and Computer Science,
Department of Computer Linguistics and Artificial Intelligence,
ul. Umultowska 87, 61-614 Poznań, Poland
mkubis@amu.edu.pl

Abstract. WordNet-like lexical databases are used in many natural language processing tasks, such as word sense disambiguation, information extraction and sentiment analysis. The paper discusses the problem of querying such databases. The types of queries specific to WordNet-like databases are analyzed and previous approaches that were undertaken to query wordnets are discussed. A query language which incorporates data types and syntactic constructs based on concepts that form the core of a WordNet-like database (synsets, word senses, semantic relations, etc.) is proposed as a new solution to the problem of querying wordnets.

Keywords: Wordnet, lexical database, domain-specific language.

1 Introduction

Databases that employ data organization inspired by that of WordNet [8] are used in many natural language processing tasks ranging from word sense disambiguation to information extraction and sentiment analysis. The data collected in a wordnet[1] form a network of *synsets*, i.e. semantically related sets of synonymous words. A synset represents a common meaning of the words it contains. Synsets are connected through relations that represent the semantic relationships that hold between the word meanings they represent. Although wordnets and the systems to maintain them have been created for more than 30 natural languages, no common tool has emerged that would allow to access the data collected in the different WordNet-like lexical databases in a uniform manner.

The data collected in a database management system are usually provided to the user through a dedicated programming language (*query language*) that allows the user to create expressions (*queries*) which describe the criteria which the data requested by the user have to fulfill. A wide variety of query languages of both theoretical and practical importance has been proposed for relational,

[1] The terms *wordnet* and *WordNet-like lexical database* are used in the paper interchangeably. The term *WordNet* is used to refer to the database described in [8].

J.-S. Pan, S.-M. Chen, N.T. Nguyen (Eds.): ACIIDS 2012, Part III, LNAI 7198, pp. 436–445, 2012.

object-oriented and semi-structured database management systems. These languages possess data types based on the concepts that appear in the data models for which they have been developed, and provide dedicated syntactic constructs that allow to formulate concise and easy-to-understand queries. We propose a similar approach to data management with respect to WordNet-like lexical databases by introducing WQuery – a query language that incorporates data types and syntactic constructs based on concepts that form the core of a wordnet. The WQuery language interpreter may be freely obtained from the project website located at `http://www.wquery.org`. The interpreter has been used in two projects. First, as a part of an access layer to the lexical database in an NLP system [12]. Second, as a supporting tool in the development of PolNet [19].

2 WordNet-Like Databases

As stated in the introduction, a WordNet-like lexical database is a network of synsets. A synset consists of a set of *words* and a *gloss*. All the words in the synset belong to the same part of speech (POS). Every synset represents a single common meaning of the words it contains. For example, an automobile is represented in WordNet [8] by the set *{ car, auto, automobile, machine, motorcar }* accompanied by the gloss *"a motor vehicle with four wheels; usually propelled by an internal combustion engine (...)"*. A word may occur in several synsets. Every occurrence of the word has an assigned *sense number*. The sense number is a positive integer unique among all the occurrences of the word within the synsets that share the POS. For instance, the word *car* has sense number 1 assigned in the synset above and has sense number 3 in the synset *{ car, gondola }* accompanied by the gloss *"the compartment that is suspended from an airship and that carries personnel and the cargo and the power plant"*. A triple that consists of a word, sense number and POS symbol is called a *word sense*.[2] Synsets are connected through *semantic relations*, i.e. binary relations which represent semantic relationships that hold between the meanings of synsets. For instance, the *hypernymy* relation links synsets representing more general concepts to synsets representing more specific ones. Word senses are connected through *lexical relations*, i.e. binary relations which represent lexical relationships that hold among words. For example, the *antonymy* relation links word senses that represent opposite concepts. Some relations in a wordnet may be inferred from others. For instance, *hyponymy* connects synset A to synset B if and only if *hypernymy* connects synset B to synset A. Besides the data types mentioned above, wordnets also encompass additional data that vary largely among them. For instance, EuroWordNet [20] stores relations that hold between synsets and Inter-Lingual Index, and PolNet [19] provides for synsets examples of usage of the words that belong to them.

[2] In the examples ":" is placed to separate the word, the sense number and the POS symbol of a word sense.

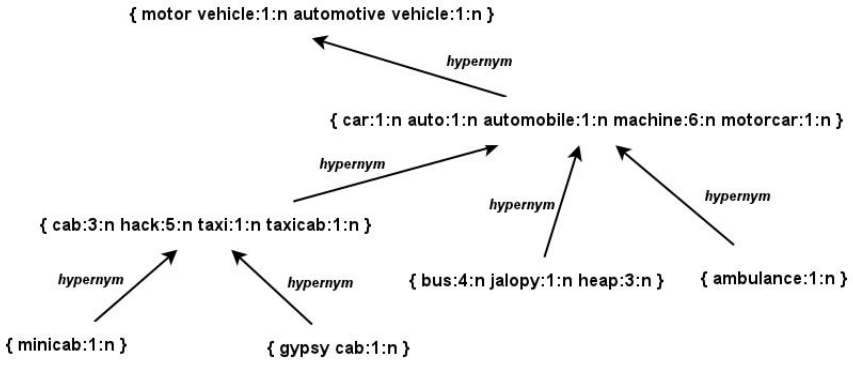

Fig. 1. A fragment of WordNet

3 Wordnet-Specific Queries

Although WordNet-like lexical databases are used in many natural language processing tasks, the class of queries to be answered by a wordnet-specific query language has not been previously investigated. Therefore, we analyzed papers from major wordnet conferences (e.g. [17,16]) and gathered wordnet-specific queries. Since we noticed multiple attempts to use relational databases to store and query wordnets (e.g. [1]), we assumed that the minimal usable wordnet query language should express at least the queries that are representable in relational algebra. Therefore, we present in this section only those queries that require more expressive languages than the relational complete ones.

3.1 Tree Queries

A tree query is a query which for a set of synsets S and a semantic relation r retrieves all paths such that the first synset of a path belongs to S and every other synset of the path is accessible from the previous one through r.

Example 1. (Tree query) Find all paths that link (through hypernymy) synsets containing the word *car* to their hypernyms, hypernyms of their hypernyms, hypernyms of hypernyms of their hypernyms, etc.

Tree queries may be used to build tree views for semantic relations in wordnet browsers and editors. They may also be used to compute structural measures, such as minimum/maximum depth and height of the hypernymy hierarchy[3] and to detect cycles in semantic relations that by definition should be acyclic (e.g. hypernymy). In [12] tree queries are used in the process of building frame-based representations for natural language sentences.

[3] In [6] values of such measures obtained for WordNet are analyzed.

3.2 Reachability Queries

A reachability query is a query which for two given synsets α and β and a set R of semantic relations retrieves all paths that connect the synset α to the synset β through relations from the set R.

Example 2. (Reachability query) Find all paths that connect through hypernymy the synset that contains the fourth noun sense of the word *bus* to the synset that contains the first noun sense of the word *vehicle*.

The results of reachability queries are used to determine values of path-based similarity and relatedness measures, which are further exploited in the word sense disambiguation task [14]. For instance, the Leacock-Chodorow measure [13] of similarity between words a and b is defined by the formula $sim_{ab} = max[-log(Np/2D)]$, where Np means the number of nodes of a path p between a and b, and D means the depth of hyponymy. The results of reachability queries are also determined in WordNet-based anaphora resolution methods [7]. As in the case of tree queries, one may formulate reachability queries that detect cycles in semantic relations.

3.3 Least Common Subsumers

The least common subsumer of two synsets α and β is a common hypernym γ of α and β such that no other hypernym of α and β is placed lower in the hypernymy hierarchy than γ.

Example 3. (Least Common Subsumer) The synset *{ car:1:n auto:1:n automobile:1:n auto:1:n automobile:1:n }* presented in Figure 2 is the least common subsumer of *{ bus:4:n jalopy:1:n heap:3:n }* and *{ minicab:1:n }*.

As in the case of reachability, the notion of the least common subsumer is exploited in similarity and relatedness measures (e.g. [14] the Resnik, Jiang-Conrath and Lin measures).

4 Related Work

A tool called Hydra that provides a modal logic based language for querying wordnets has been described in [15]. The Hydra language may be translated to relational calculus by adjusting the standard translation from the modal language to the first-order language defined in [3]. Therefore, one cannot express in Hydra any of the wordnet-specific queries described in Section 3 because these queries involve computation of the transitive closures of relations, which are not computable in relational calculus. The *WN* language [11] is another modal logic based formalism designed to query WordNet-like databases. *WN* represents the transitive closure of hypernymy as a separate binary relation. However, it does not provide operators that would allow to compute the entire paths that transitively connect synsets through arbitrary chosen semantic relations. Hence, one

cannot formulate in *WN* queries defined in Section 3.2. Neither *WN* nor Hydra encompass arithmetic expressions. Therefore, they are unable to answer aggregate queries such as "How many synsets exist in the wordnet?" or "What is the average polysemy?", and they cannot be used to compute the Leacock-Chodorow measure mentioned in Section 3.2. DEBVisDic [10] – a tool for browsing and editing wordnets – provides an option to search for synsets by specifying the word form, the word sense and several other properties that conform to the underlying XML representation of a wordnet. These search criteria may be combined together by using logic operators. Such queries may be expressed in WQuery using the synset generator followed by a filter with a suitable condition (Section 6.2). The wordnet-specific queries defined in Section 3.2 and 3.3 are not directly expressible using the search form of DEBVisDic.

As for general purpose tools adapted to query wordnets, relational databases are reported to have been used in several projects, e.g. [1]. Since the queries described in Section 3 involve computation of transitive closures, they cannot be formulated in relational complete languages. Besides the use of relational databases, there have been attempts to store wordnets in XML [10,18] and RDF [9]. Thus, one may consider using one of the languages designed for these models, such as XQuery or SPARQL. However, the XML and RDF query languages do not include the wordnet-specific data types described in Section 2. Another option is to access a wordnet from a general purpose programming language through one of the APIs mentioned in [1] or the DEBVisDic server API. However, regardless of the choice, general purpose tools cannot be more expressive than WQuery since it expresses all computable queries over the wordnet data model, as stated in Section 6.4.

Path expressions, which form the core of the WQuery language, are similar to constructs available in other regular path languages, such as Lorel [2] and XPath [5], but the preservation of variable bindings in functions is not, to the best of our knowledge, supported by any of the tools mentioned above.

5 Data Model

According to the description of a WordNet-like lexical database presented in Section 2, a wordnet query language has to operate on such domain-specific data types as synsets, word senses, words, glosses, POS symbols and sense numbers. One may notice that the relationships which define the structure of a wordnet may be modeled as a set of binary relations between the aforementioned data types that consists of:

1. The relation *synset* that links word senses to their synsets.
2. The relation *word* that links word senses to their words.
3. The relation *sensenum* that links word senses to their sense numbers.
4. The relation *pos* that links word senses to their part of speech symbols.
5. The relation *gloss* that links synsets to their glosses.

However, in order to embrace extensions specific to particular existing wordnets, we amplify this model in several ways. First, we include two additional data types

– floating point numbers and booleans. Secondly, we assume that words, glosses and POS symbols are encompassed by the generic character string data type and that sense numbers belong to the integer data type. Thirdly, we represent the inferred relations by relation names bound to regular expressions composed over other previously defined relations.

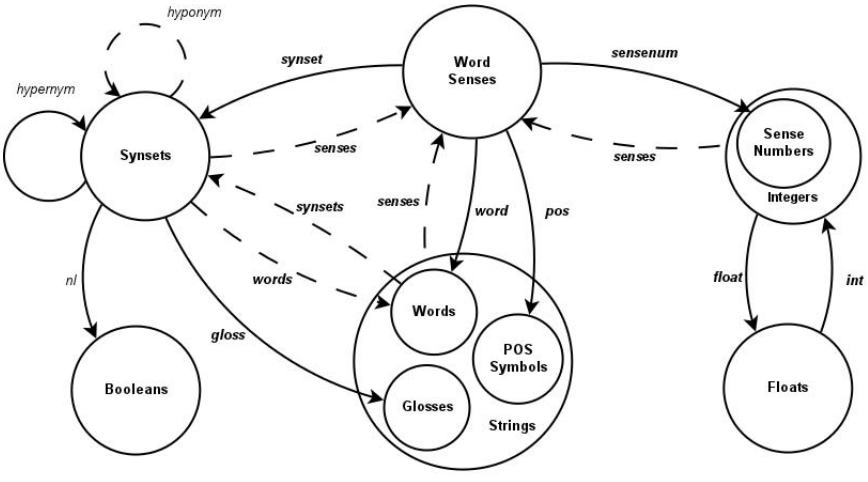

Fig. 2. An instance of the WQuery wordnet model. The mandatory relations are marked by boldfaced labels. The inferred relations are represented by dashed edges.

6 Syntax and Semantics

6.1 Path Expressions

The syntax and semantics of the WQuery language are focused on searching for paths in instances of the data model defined in the previous section. Paths are represented at the syntactic level by *path expressions*, and at the semantic level by tuples that consist of values of data types defined in Section 5, interleaved with the names of the relations that connect the consecutive values. A path expression consists of one or more *steps*, such that the first one (*generator*) describes a set of nodes and the following ones (*transformations*) extend the paths defined by the preceding expressions by attaching edges and nodes to them. For example, the expression

```
{car}.(hypernym|meronym).words
```

consists of a generator `{car}` which represents the set of synsets that contain the word *car*, the transformation `.(hypernym|meronym)` which extends one-element paths generated by `{car}` with edges reachable through either the `hypernym` or `meronym` relation, and the transformation `words` which extends the resulting

paths with words that belong to the synsets reached by the previous transformation. The following tuples belong to the result of the query formulated above[4]

```
{ car:1:n ... } meronym { air bag:1:n } words 'air bag'
{ car:1:n ... } meronym { bumper:2:n } words bumper
{ car:3:n ... } hypernym { compartment:2:n } words compartment
```

Transformations may involve conventional regular operators, such as `*` (zero or more), `+` (one or more), `|` (alternative) and parentheses. One may also use the `^` operator to traverse relations in the opposite direction.

6.2 Variable Bindings

In order to select data from paths, one may augment a path expression with variables placed after a step. For instance, the path expression

```
{car}$a.hypernym+@P$b
```

will bind the variable `$b` to the last transitive hypernym on the path, the path variable `@P` to all elements of the path generated by the transformation `.hypernym+` that precede `$b` and the variable `$a` to a node generated by `{car}`.

Values bound to variables may be referenced in *filters* and *projections* – two constructs that can be placed after a step of a path expression in order to modify the result. A filter consists of a logical condition enclosed in square brackets. If the condition inside the filter holds for a path, then the path is added to the resulting multiset, otherwise it is eliminated. For example, the query

```
{car}.hypernym+$a[$a = {vehicle:1:n}]
```

finds paths that connect synsets that contain the word *car* to the synset that contains the word sense *vehicle:1:n* through the `hypernym` relation. Filters that compare a value bound to the last step of the path with a value of a particular generator may also be abbreviated by placing the generator after the dot instead of the bracketed expression. Thus, the query above may be reformulated as

```
{car}.hypernym+.{vehicle:1:n}
```

A projection consists of a path expression enclosed in `<` and `>`. The result of the evaluation of the enclosed expression replaces the path for which the projection is evaluated. For example, the query

```
{car}$a.hypernym.meronym$b<$a,$b>
```

finds all pairs that consist of a synset that contains the word *car* and a meronym of its hypernym.

[4] The queries in the paper are invoked against WordNet [8], version 3.0.

6.3 Functions

WQuery provides a set of built-in functions which take as arguments and return as values arbitrary multisets of tuples. Among them there are conventional aggregate operators, such as `count`, `max` and `avg`, mathematical functions, such as `log` and `exp`, and special path-oriented operators, such as `shortest` and `longest`, which return the shortest and longest tuple of a multiset, respectively. What differs WQuery functions from their counterparts in other query languages is that the functions that do not modify, remove or replace elements of their arguments preserve variable bindings. Thus, the query

```
longest({car:5:n}.hypernym+$a)
```

not only returns the longest path from `{car:5:n}` to the top of the hypernymy hierarchy, but also binds the synset on the top to the variable `$a`.

6.4 Other Constructs

Although the queries presented in Section 3 may be formulated using the constructs presented in the previous sections, we have introduced additional constructs to make the language robust. First, we extended the language with multipath operators that are counterparts of relational algebra operators. The `union`, `intersect`, `except` and cross product (`,`) operators treat paths as tuples and make it possible to translate queries formulated in relational complete languages to WQuery. Secondly, we added conventional arithmetic operators to make it easier to compute similarity measures mentioned in Section 3 directly in WQuery. Thirdly, we introduced imperative constructs such as `if-else` statements, `while` loops and sequential execution blocks that made it possible to prove that the WQuery language can express any query computable over an instance of the wordnet data model. Due to the limited space, we do not present the formal proof here, but it can be easily derived by constructing the translation from the QL language [4] to WQuery.

7 Wordnet-Specific Queries in WQuery

We will now show how the wordnet-specific queries described in Section 3 may be expressed in WQuery.

7.1 Tree Queries

Let `g` be a generator that represents a set of synsets S and `r` be the name of a semantic relation r. The tree query of S under r is defined by the expression

```
g.r+
```

For instance, to answer the example query presented in Section 3.1 one may formulate the following expression

```
{car}.hypernym+
```

7.2 Reachability Queries

Let `g` and `h` be generators for synsets α and β, respectively. Let `r_1`, ..., `r_k` be the names of the semantic relations in set R. The reachability query from α to β with respect to R is fulfilled by the expression

```
g.(r_1|...|r_k)*.h
```

For example, to find paths that connect the synset that contains the word sense *bus:4:n* with the synset that contains the word sense *vehicle:1:n* through hypernymy one may formulate the expression

```
{bus:4:n}.hypernym*.{vehicle:1:n}
```

7.3 Least Common Subsumers

Let `g` and `h` be generators for synsets α and β, respectively. The least common subsumer of α and β is determined by the query

```
longest((g.hypernym*$a<$a>
    intersect h.hypernym*$b<$b>)$s.hypernym*)<$s>
```

For instance, the least common subsumer of the synsets that contain the senses *minicab:1:n* and *bus:4:n* is computed by the query

```
longest(({minicab:1:n}.hypernym*$a<$a>
    intersect {bus:4:n}.hypernym*$b<$b>)$s.hypernym*)<$s>
```

8 Conclusion

We have presented in the paper several types of wordnet-specific queries that arise in natural language processing tasks, such as word sense disambiguation and relatedness measurement. We found that none of the existing wordnet-specific tools are capable of expressing all of the analyzed queries. Therefore, we proposed a new solution to the problem of querying WordNet-like lexical databases. The solution is based on the data model that incorporates wordnet-specific data types and a query language that utilizes the concept of a path. We have shown that by using the WQuery language one can formulate all wordnet-specific queries described in Section 3. In the future we would like to introduce optimization techniques for query evaluation based on the structure of a WordNet-like database.

Acknowledgments. The author is a scholarship holder within the project "Scholarship support for Ph.D. students specializing in majors strategic for Wielkopolska's development", Sub-measure 8.2.2 Human Capital Operational Programme, co-financed by European Union under the European Social Fund.

References

1. WordNet - Related Projects, http://wordnet.princeton.edu/wordnet/related-projects/ (access date: April 28, 2011)
2. Abiteboul, S., Quass, D., McHugh, J., Widom, J., Wiener, J.L.: The Lorel Query Language for Semistructured Data. International Journal on Digital Libraries 1(1), 68–88 (1997)
3. Blackburn, P., van Benthem, J.: Modal Logic: a Semantic Perspective. In: Blackburn, P., et al. (eds.) Handbook of Modal Logic, pp. 1–84. Elsevier (2007)
4. Chandra, A.K., Harel, D.: Computable Queries for Relational Data Bases. Journal of Computer and System Sciences 21(2), 156–178 (1980)
5. Clark, J., DeRose, S.: XML path language (XPath) version 1.0. W3C recommendation, W3C (1999), http://www.w3.org/TR/1999/REC-xpath-19991116/
6. Devitt, A., Vogel, C.: The Topology of WordNet: Some Metrics. In: Sojka, et al. (eds.) [17], pp. 106–111
7. Fan, J., Barker, K., Porter, B.: Indirect Anaphora Resolution as Semantic Path Search. In: K-CAP 2005: Proceedings of the 3rd International Conference on Knowledge Capture, pp. 153–160. ACM Press (2005)
8. Fellbaum, C. (ed.): WordNet: An Electronic Lexical Database. MIT Press, Cambridge (1998)
9. Graves, A., Gutierrez, C.: Data Representations for WordNet: A Case for RDF. In: Sojka, et al. (eds.) [16], pp. 165–169
10. Horak, A., Pala, K., Rambousek, A., Povolny, M.: DEBVisDic - First Version of New Client-Server Wordnet Browsing and Editing Tool. In: Sojka, et al. (eds.) [16]
11. Koeva, S., Mihov, S., Tinchev, T.: Bulgarian Wordnet – Structure and Validation. Romanian Journal of Information Science and Technology 7(1-2), 61–78 (2004)
12. Kubis, M.: An Access Layer to PolNet – Polish WordNet. In: Vetulani, Z. (ed.) LTC 2009. LNCS, vol. 6562, pp. 444–455. Springer, Heidelberg (2011)
13. Leacock, C., Chodorow, M.: Combining Local Context and WordNet Similarity for Word Sense Identification. In: Fellbaum (ed.) [8], ch. 11, pp. 265–283 (1998)
14. Patwardhan, S., Banerjee, S., Pedersen, T.: Using Measures of Semantic Relatedness for Word Sense Disambiguation. In: Gelbukh, A. (ed.) CICLing 2003. LNCS, vol. 2588, pp. 241–257. Springer, Heidelberg (2003)
15. Rizov, B.: Hydra: a Modal Logic Tool for Wordnet Development, Validation and Exploration. In: Calzolari, N., et al. (eds.) Proceedings of the Sixth International Conference on Language Resources and Evaluation, LREC 2008 (2008)
16. Sojka, P., et al. (eds.): Proceedings of the Third International WordNet Conference – GWC 2006. Masaryk University, Brno (2005)
17. Sojka, P., et al. (eds.): Proceedings of the Second International WordNet Conference-GWC 2004. Masaryk University, Brno (2003)
18. Soria, C., Monachini, M., Vossen, P.: Wordnet-LMF: Fleshing out a Standardized Format for Wordnet Interoperability. In: Proceeding of the 2009 International Workshop on Intercultural Collaboration, pp. 139–146. ACM, New York (2009)
19. Vetulani, Z., Kubis, M., Obrębski, T.: PolNet - Polish WordNet: Data and Tools. In: Calzolari, N., et al. (eds.) Proceedings of the Seventh International Conference on Language Resources and Evaluation. ELRA, Valletta (2010)
20. Vossen, P.: EuroWordNet: general document. Vrije Universiteit, Amsterdam (2002), http://dare.ubvu.vu.nl/handle/1871/11116