An Algorithm for Inconsistency Management in Spatial Knowledge Integration

Marcin Walas and Krzysztof Jassem

Information Systems Laboratory,
Faculty of Mathematics and Computer Science,
Adam Mickiewicz University,
87 Umultowska Street, Poznań, Poland
{mwalas,jassem}@amu.edu.pl

Abstract. In the process of knowledge acquisition it is often necessary to solve the problem of ambiguous entities. The problem occurs when the same names refer to different entities in the knowledge database. We propose a method for resolving some of the ambiguities by means of the constraint calculus, namely RCC-5. We discuss the properties of our algorithm. Then we show that the algorithm can be extended to handle other types of relations. By applying a disambiguation algorithm in the process of knowledge acquisition we are able to increase the quality of the knowledge database acquired from the variety of data resources. The database is used in the prototype of our QA system, Hipisek [11].

Keywords: knowledge acquisition, spatial reasoning, constraint calculus, question answering

1 Introduction

Question Answering (QA) is a task of finding an answer to a question posed in the natural language. Answering a question related to a spatial entity requires a spatial knowledge database. Spatial knowledge may be represented either quantitatively or qualitatively. The former way (e.g. by using absolute geographical coordinates) is certainly useful in most engineering applications. However, for the sake of QA, the qualitative representation — storing relationships between spatial entities — is by far more useful, as this is how spatial information is represented in natural language [7].

There exists several comprehensive knowledge databases with spatial knowledge (both quantitative and qualitative), the best known of which are: Geonames[1] — a comprehensive geographical database and DBPedia[2], which is a structured information database extracted from Wikipedia articles [2]. Although available resources are extensive, they lack information required by a Polish language QA system such as: Polish geographical names, the Polish administrative division or names of specific Polish places (e.g. buildings, touristic attractions).

[1] www.geonames.org
[2] www.dbpedia.org

Our goal is to acquire a comprehensive spatial knowledge database for the purposes of a Polish language QA system. We aim to integrate various spatial knowledge data sources, such as Geographical Information Systems (GIS), structured data sources (e.g. the administrative division database) and semi-structured sources, from which we can extract qualitative spatial information (Wikipedia articles, touristic information web services).

Integration of various data resources is a common method used in the knowledge acquisition process. The Geonames project has incorporated several data resources and applied a wiki mechanism, which allows the users' community to correct mistakes and add new features. The DBPedia project is focused on data extraction from Wikipedia articles.

One of the main problems in the process of data integration is disambiguation of different entities sharing the same name. For example, the name *Drawa* may refer to a Polish river (an inflow of the river *Noteć*) or it may refer to a river in the Central Europe (an inflow of the river *Danube*). During the process of knowledge integration it has to be determined which is the actual entity that the name refers to.

As a solution to this problem we propose the methodology of spatial reasoning. We use Region Connection Calculus (RCC) as a general method for disambiguation. RCC is a family of calculi with the topological approach to spatial representation and reasoning. For our purposes we choose RCC-5, which allows for five relations between regions. The idea is as follows: if a new fact appears in the knowledge database (e.g. *Drawa* is located in *Poland*), then a spatial entity existing in the database (e.g. entity corresponding to the river *Drawa*, an inflow of the river *Noteć*) may be assigned to the subject or the object of the fact only if that does not lead to an inconsistency in RCC-5.

The paper is organized as follows: first we briefly describe the family of calculi (called RCC) and its recent applications in similar tasks. Next, we describe the representation of our knowledge database and the process of converting spatial facts into RCC-5 relations. We describe the algorithm for incorporating facts from various data sources into our knowledge database in accordance to the facts already stored in the database. Then we propose a way to represent imprecise information. Finally, we present the results of the evaluation and formulate conclusions.

2 Region Connection Calculus

2.1 RCC-5 relations

RCC is a topological approach to qualitative spatial representation based on a primitive relation of connection between regions [5]. Relationships are defined by means of the $C(a, b)$ relation (*connection relation*) which holds iff regions a and b share the common point. On the basis of the C relation five basic relations for RCC-5 are defined, namely: DR (*discrete*), EQ (*equal*), PP (*proper part*), PPI (*proper part inversed*) and PO (*partial overlap*) [1]. Figure 1 illustrates the RCC-5 basic relations.

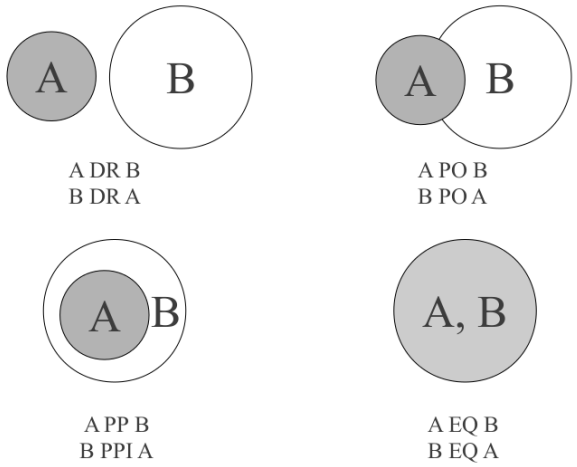

Fig. 1. The set of five RCC-5 basic relations

Any subset of the basic relation set forms a relation in RCC-5. Thus RCC-5 contains 2^5 relations. *Universal relation* (denoted $\top$) is the set of all basic relations, *empty relation* (denoted $\bot$) is the empty set of basic relations. The following operations can be defined for the RCC-5 relations set: union ($R \cup S$), intersection ($R \cap S$), inversion (R') and composition ($R \circ S$) [8].

RCC-5 relations are closed under composition and form a relation algebra [8]. In order to effectively calculate composition, a special table (called a composition table) should be designed.

2.2 Consistency in RCC-5

Using RCC-5 we can represent knowledge on entity relations in the form of constraints. A constraint network is formed from a set of variables V over the domain D and a set of constraints Θ on the variables of V. A network is *consistent* if it has a solution which is an assignment of values of D to the variables of V in a way that all constraints are satisfied [8]. It is proven that the consistency problem is NP-complete for RCC-5 [6].

A *path-consistency* serves as a common approximation of the constraint satisfaction problem. A given constraint network is path-consistent iff for any consistent instantiation of any two variables there exists an instantiation of any third variable s.t. all three variables taken altogether are consistent. To enforce path-consistency, the following operation is used for two variables i and j [8]:

$$\forall_k R_{ij} \Leftarrow R_{ij} \cap (R_{ik} \circ R_{kj})$$

The operation is used until a fixed point is reached. The resulting network is equivalent to the initial network (both share the same set of solutions).

If an empty set is present in the resulting network, then the initial network is not consistent. Otherwise the resulting network is path-consistent [8].

In the general case path-consistency does not imply consistency. Renz and Nebel in [6] define a subclass of RCC-5 relation set (called $\hat{H}_5$) in which path-consistency decides consistency. The $\hat{H}_5$ class contains 28 out of 32 RCC-5 relations, and includes all basic relations and the universal relation.

2.3 RCC-8

RCC-8 is an extension of the RCC-5, which consists of eight basic relations: DC (*disconnected*), EC (*externally connected*), PO (*partially overlap*), TPP (*tangential proper part*), $NTPP$ (*non tangential proper part*), EQ (*equal*), $TPPI$ and $NTPPI$ (inverse of TPP and $NTPP$ respectively).

The consistency problem is proven to be NP-complete in the RCC-8 [6]. Therefore path-consistency method is used to solve RCC-8 consistency problem.

2.4 Applications

Constraints calculi are used in the human-computer interaction systems and systems that use semantic knowledge. Vetulani et al. in [10] describe the XCDC calculus (which is based on spatio-temporal relations) applied in the module of dialogue analysis. Stocker and Sirin in [9] apply the RCC-8 calculus to support qualitative spatial reasoning in a query answering engine. Hogenboom et al. in [4] use RCC-8 to represent spatial knowledge in the Semantic Web (in OWL).

In our approach the reasoning itself is not a main issue. We use spatial reasoning only as one of the criteria to decide which entities should be assigned to processed facts. This helps us disambiguate entities bearing the same name and increase quality of the knowledge database without costly human work.

In our knowledge acquisition system we focus mainly on two relations between spatial entities: *A is located in B* and *A overlaps* with *B*. Therefore we decided to choose RCC-5, whose relations set corresponds to those maintained in our system. However, our algorithms can be easily adapted to use within other constraint calculi. As an example we describe application of the RCC-8 calculus in order to handle imprecise information.

3 Facts in the database

In our knowledge database we store entities and facts. An *entity* is a notion for an individual object in the real world. One entity may have different names in different languages. One entity belongs to exactly one type. Entity types form a taxonomy. For example the city of *Warsaw* (a capital of Poland) with three possible names: *Warszawa* (Polish), *Warszawka* (Polish, informal) and *Warsaw* (English) forms one entity of the type *city* in our database.

A *fact* is a triple: subject entity, predicate name and object entity. A fact is the only way of knowledge representation in our database. Predicate name is

a name of the relation that holds between subject and object. Predicates are typed to form a taxonomy. An example of the fact that Warsaw is located in Poland is a triple: (city Warsaw, *is located in*, country Poland).

Our entities taxonomy includes 80 entity types, among which 45 are spatial entity types. The taxonomy is based on Geonames feature codes and on the general ontology of entities used in the Polint-112-SMS project described in [10]. Entity types are organized in a hierarchical way. The relation types taxonomy includes 15 relations among which 4 are qualitative spatial relations.

The creation of a constraint network from the set of facts is done by conversion of each fact subject and object to network node and transformation of the fact relation type to the corresponding calculus relation. We define, that a fact is **calculus transformable**, if the fact relation type is semantically equivalent to a calculus relation. The following transformations are set:

- *is located in* $\rightarrow PP$
- *overlaps* $\rightarrow PO$
- *is partially located in* $\rightarrow \{PP, PO\}$

We assume that the universal relation holds between entities for which no information is available. Following Gabrielli [3] we assume that spatial entities are approximated by regions in Euclidean topological space.

Some of the relations can be inferred from the world knowledge, thus we do not store them in the database. For example, we do not keep the fact that the continent of Europe is discrete with South America. Instead, we store the rules for indirect semantic relation. This includes granularity issues, like deciding that some of the entities may be considered point-like (e.g. one building can not be inside another building). Indirect semantic relations use only DR relation.

Linking entities with indirect semantic relations is executed using handcrafted rules. Exemplary rules are:

1. If subject and object entities are of the *building* type or any of its subtype, then they are discrete (DR),
2. If subject and object entities are of the same leaf type (have no subtypes in the taxonomy) then they are discrete (DR).

We use the following procedure to create a constraint network for a given calculus:

Create constraint network from the set of facts F

```
procedure CreateNetwork(F)
input:
 set of facts F
 temporary set of facts T := F
begin:
1. repeat
2.1. changed := false
2.2. for each fact f from T:
```

```
2.2.1. if f is calculus transformable then
2.2.1.1. transform f into a calculus relation e
2.2.1.2. add e to N; if new entity is added to N
         then set changed := true
2.2.1.3. search in the knowledge database
         for calculus transformable facts with
         identical subject as the subject or object of f;
         add them to T
2.3. for each pair of entities from N which are not related
     link them with indirect semantic relations.
2.4. set F := T and clear T.
3. until changed
4. return N
```

Potentially, the network that we create may contain a large number of entities. In fact we use only small local paths inside the network. This is due to the fact, that we expand our network only in "one direction" (see point 2.2.1.3. of the `CreateNetwork` procedure) and that we actually store facts in "one direction" (e.g. we store the fact that A is located in B, but not the fact that B contains A).

4 Adding facts to the database

In this section we describe the implementation of RCC-5 in the process of developing the spatial knowledge. We develop our knowledge database by integrating various data resources. A sketch of the algorithm for adding a fact to the database is given below:

Adds a fact to the knowledge database

```
procedure ProcessFact(f)
input:
 processed fact f (subject, predicate_name, object)
begin:
1. search the database for entities that are
   either candidate subjects or candidate objects.
2. select the best pair (s, o) of candidate entities.
3. if the corresponding constraint network is path-consistent,
   then assign the pair (s, o) to f.
4. otherwise go back to (2).
5. if no appropriate pair (s, o) has been found,
   try one of the following:
5.1. introduce a new entity for
     either subject or object of f
5.2. otherwise, introduce new entities for
     both subject and object of f
```

Candidate entities are those, whose names are consistent with the names occurring in the fact (e.g. the names have at least one word in common) and are of the consistent type (the types are equal or one is a predecessor of the other in the taxonomy).

To check consistency of the network for a given assignment of entities (s, o) to f, we use the procedure `CheckConsistency`:

Checks path-consistency for a new fact

```
procedure CheckConsistency(f, s, o)
input:
 processed fact f,
 candidate pair of attachments for subject and object (s, o)
begin:
1. create the constraint network assuming that s and o
   are assigned to f: use the CreateNetwork procedure.
2. use path-consistency method on the constraint network
3. if the resulting network path-consistent then succeed;
   otherwise, fail (<s, o> assignment to f is inconsistent
   with the knowledge database).
```

Consider the following example: a new fact *Drawa (type: river) overlaps Croatia (type: country)* is being processed. Assume that the database contains:

- the fact: Drawa (river, id:1) *is located in* Poland (country, id:2),
- the entity: Croatia (country, id:3).

In the first step the candidates are searched for in the knowledge database. The results are: one-element set {Drawa (river, id:1)} for the subject and one-element set {Croatia (country, id: 3)} for the object. The following candidate entities may be assigned to the names Drawa and Croatia occurring in the new fact:

1. Assign entity (id: 1) to *Drawa* and entity (id: 3) to *Croatia*
2. Assign entity (id: 1) to *Drawa* and introduce a new entity for *Croatia*
3. Introduce a new entity for *Drawa* and assign entity (id: 3) to *Croatia*
4. Introduce new entities for both *Drawa* and *Croatia*

The algorithm starts with checking the first possibility. An initial constraint network is created (see figure 2). Then, the final network is constructed (see figure 3). The network is verified for consistency and the result is negative (as we assume that two countries: Croatia and Poland should be discrete).

Introducing a new country entity for Croatia and assigning id: 1 to Drawa (possibility 2.) also leads to the same inconsistency (two countries should be discrete).

Finally, assigning a new entity to Drawa (possibility 3.) does not lead to inconsistency: there exist a river entity named Drawa that overlaps with Poland and a new river entity Drawa that overlaps with Croatia.

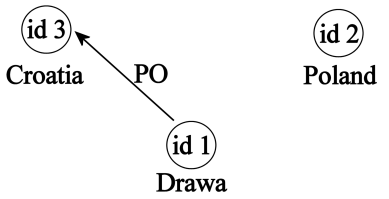

Fig. 2. Initial RCC-5 constraint network

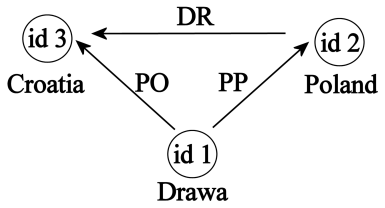

Fig. 3. Final RCC-5 constraint network

4.1 Analysis of the algorithm

In this section we show that the path-consistency method used in the *Check-Consistency* procedure decides consistency.

Algorithm attribute 1 *For any pair of nodes v and w of the constrain network N created by the function CreateNetwork (v, w) the constraint is one of:*

1. *the universal relation,*
2. *the relation created via transformation of the predicate into a RCC-5 relation,*
3. *the relation created via indirect semantic rules.*

Corollary 1. *A constraint network N used in the procedure* `ProcessFact` *uses only relations from the $\hat{H}_5$ relation set.*

As was said in section 2.2 Renz and Nebel shown that in the $\hat{H}_5$ path-consistency decides consistency, which leads to the final corollary:

Corollary 2. *The CheckConsistency algorithm decides the consistency of the network.*

5 Algorithm extension

We have observed that some of the knowledge sources processed by our knowledge acquisition system provide additional quantitative information (e.g. population, surface or elevation). We decided to use such kind of information in the process of disambiguation.

Our first approach was to compare the corresponding values and check if they are equal to those already stored in the database. This approach was not effective because most of the sources contain information, which values differ to some extent. For example the city of *Boston* (USA) has the population of 625,087 (according to Wikipedia) or 594,034 (according to MongoBay[3]).

In the extended algorithm we introduce the *approximate* relation, which allows to compare values with a certain tolerance level. Our goal is to model this relation in the RCC.

[3] http://world.mongabay.com/polish/population/pl.html

5.1 Q3 set of relations

We define the following relations:[4]

- $equal(x, y) \Leftrightarrow |x - y| = 0$,
- $aprox(x, y) \Leftrightarrow |x - y| > 0 \wedge |x - y| < \epsilon$,
- $notequal(x, y) \Leftrightarrow |x - y| \geq \epsilon$.

We define the C relation as follows:

$$\forall_{x,y \in \mathbb{R}} C(x, y) \Leftrightarrow |x - y| < \epsilon$$

Then, the following holds (see [12]):[5]

$$\forall_{x,y \in \mathbb{R}} DC(x, y) \Leftrightarrow notequal(x, y)$$
$$\forall_{x,y \in \mathbb{R}} EQ(x, y) \Leftrightarrow equal(x, y)$$
$$\forall_{x,y \in \mathbb{R}} ONE'(x, y) \Leftrightarrow aprox(x, y)$$

This allows us to use RCC-8 as a model for our *approximate* comparison.

5.2 Application to the disambiguation algorithm

In order to handle "approximacy" the disambiguation algorithm is modified as follows:

- we allow to add multiple facts at one time (all considering one subject),
- we define calculus transformable relations: *equal* and *approximate*, which are modeled in RCC-8 by EQ and EQ, ONE' respectively,[6]
- we define indirect semantic rules for the value entities with respect to the C relation defined in section 5.1.

We take the following assumptions about quantitative information:

- all values in the knowledge database are in the *equal* relation,
- values introduced by a new source are considered as *approximate*,
- each entity can have at most **one** value of each type.[7]

Reasoning is carried out using path-consistency algorithm. Remark that we do not need to apply extensive modifications to our disambiguation algorithm. We only need to specify the calculus transformable relations and indirect semantic rules.

Consider the following example: the following set of facts is being processed:

[4] The parameter ϵ is used to set a level of tolerance when comparing two values.

[5] ONE' is the short form of: $\{EC, PO, TPP, NTPP, TPPI, NTPPI\}$. The name refers to the RCC3 relation ONE

[6] Remark that the precise meaning of *approximate* relation is: *approximate or equal*

[7] Hence if the source contains value for the quantitative relation, which is already stored in the knowledge database, the new value is used only for disambiguation

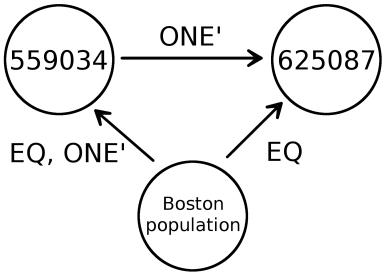

Fig. 4. Constraint network created when the name Boston refers to one city (the difference of population is in the degree of tolerance)

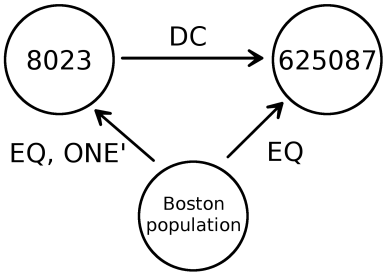

Fig. 5. Constraint network created when the name Boston corresponds to different cities (the population difference is too high)

```
Boston [city], is located in, USA [country]
Boston [city], population, 559,034 [value]
```

Assume that the following facts are stored in the knowledge database:

```
Boston [city] id:1, is located in, USA [country] id:2
Boston [city] id:1, population, 625,087 [value]
```

Remark that the entity *Boston* stored in the database is likely to be equal the new entity introduced by the processed fact. This is confirmed by the disambiguation algorithm, which in the first try assigns new fact entity *Boston*, to the entity with id 1. The corresponding quantitative knowledge constraint network is consistent (see figure 4), but no new facts are stored in the database (we do not store more than one *population* fact). Remark that if we do not allow for the *approximate* comparison we would end up with a new *Boston* entity (since it would be inconsistent with the fact stored in the database).

Now assume that we process the following fact set:

```
Boston [city], is located in, USA [country]
Boston [city], population, 8,023 [value]
```

In this case the new entity *Boston* and the entity *Boston* stored in the knowledge database are different cities. Remark that spatial relations are not sufficient to distinguish those cities (because both are located in the USA). However, using quantitative information, we end with the inconsistent constraint network (see figure 5). This way we identified that in this case the name *Boston* refers to a different city.

Table 1. Comparison of the number of acquired entities in the baseline version of HipiSwot and the final version

	Baseline version	Final version
Total number of entities	232,710	261,063
Number of spatial entities	111,891	147,085
Total number of facts	307,668	298,002
Number of qualitative spatial relation facts	191,715	186,829

6 Results and evaluation

We applied the above algorithms to our knowledge acquisition system HipiSwot. We processed 9 resources which are: an administrative division database (TERYT), Geonames project, data extracted automatically from Wikipedia categories and web services with touristic or geospatial information.

We compared the baseline version (without the disambiguation mechanism applied) with the final version. Both versions of the system integrated the same set of data sources into the knowledge database. Using the baseline version of the system we obtained $111,891$ spatial entities and $191,715$ facts concerning qualitative spatial relationships. Using the final version we obtained respectively $147,085$ spatial entities and $186,829$ facts. A significant increase in the number of spatial entities is due to the introduction of new entities when inconsistency with the knowledge database was identified. A little decrease in the number of facts obtained is due to applying a discarding mechanism for some data resources (some resources with lesser quality were blocked from introducing new facts). Table 1 shows the results for both versions.

In the consistency check experiment we computed the number of inconsistencies occurring in the baseline version. For each spatial entity in the knowledge database we created the constraint network for all facts attached to the entity. We checked if the network was path-consistent. If not, we marked entity as inconsistent. We identified that out of $111,891$ spatial entities about 21.6% ($24,170$ entities) were inconsistent. All these inconsistencies were removed in the final version, due to the implementation of RCC.

7 Conclusions

In this paper we proposed an implementation of RCC to disambiguate entities in the process of knowledge acquisition. We described an algorithm that allows for discarding ambiguities, which are not path-consistent with the already acquired knowledge. We created an initial version of a knowledge database for a Polish QA system by integrating various data resources. We evaluated our results by comparing them to an earlier version of the data set.

Our algorithms are calculus-independent and may be adopted to any constraint calculus. One needs only to define the rules for transformation of the

fact relation types stored in the knowledge database and the rules for indirect semantic relations between entity types for the chosen calculus.

References

1. Bennett, B.: Spatial reasoning with propositional logics. In: Doyle, J., Sandewall, E. and Torasso, P. (Eds.). Proceedings of the 4th International Conference on Principles on Knowledge Representation and Reasoning (KR-94). pp. 165–176. Morgan Kaufmann. Bonn, Germany (1994)
2. Bizer, Ch., Lehmann, J., Kobilarov, G., Auer, S., Becker, Ch., Cyganiak, R., Hellmann, S.: DBpedia — A Crystallization Point for the Web of Data. Journal of Web Semantics: Science, Services and Agents on the World Wide Web. pp. 154–165 (2009)
3. Gabrielli, N. Investigation of the Tradeoff between Expressiveness and Complexity in Description Logics with Spatial Operators. Ph. D. Thesis. Universita degli Studi di Verona. Dipartimento di Informatica (2009)
4. Hogenboom, F., Borgman, B., Frasincar, F., Kaymak, U.: Spatial Knowledge Representation on the Semantic Web. In: ICSC '10 Proceedings of the 2010 IEEE Fourth International Conference on Semantic Computing. IEEE Computer Society Washington, DC, USA (2010)
5. Randell, D. A., Cui, Z., Cohn, A. G.: A spatial logic based on regions and connection. In: Nebel, B., Swartout, W. and Rich, C. (Eds.) Principles of Knowledge Representation and Reasoning: Proceedings of the 3rd International Conference. pp. 165–176. Morgan Kaufmann. Cambridge (1992)
6. Renz, J., Nebel, B.: On the Complexity of Qualitative Spatial Reasoning: A Maximal Tractable Fragment of the Region Connection Calculus. In: IJCAI. pp. 522–527. (1997)
7. Renz, J., Rauh, R., Knauff, M.: Towards Cognitive Adequacy of Topological Spatial Relations. In: Spatial Cognition II. LNCS, vol. 1849, pp. 184–197. Springer Berlin/Heidelberg (2000)
8. Renz, J.: Qualitative Spatial Reasoning with Topological Information. LNCS, vol. 2293. Springer-Verlag. (2002)
9. Stocker, M., Sirin, E.: PelletSpatial: A Hybrid RCC-8 and RDF/OWL Reasoning and Query Engine. In: Proceedings of the 5th International Workshop on OWL: Experiences and Directions (OWLED 2009) Chantilly, VA, United States (2009)
10. Vetulani, Z., Marciniak, J., Obrebski, T., Vetulani, G., Dabrowski, A., Kubis, M., Osiński, J., Walkowska, J., Kubacki, P., Witalewski, K.: Language resources and text processing technologies. The POLINT-112-SMS system as example of application of Human Language Technology in the public security area. Adam Mickiewicz University Press. Poznań (2010)
11. Walas, M.: How to answer yes-no spatial questions using qualitative reasoning? In: Computational Linguistics and Intelligent Text Processing. LNCS, vol. 7182, pp. 330–341. Springer-Verlag. Berlin. Heidelberg (2012)
12. Walas, M.: Spatial-temporal reasoning in the Question Answering system. Ph. D. Thesis. Adam Mickiewicz University (2013)