Proceedings of the 12th Conference on Language Resources and Evaluation (LREC 2020), pages 4877–4882
Marseille, 11–16 May 2020

The Ontology of Bulgarian Dialects – architecture and information retrieval

Rositsa Dekova
Paisii Hilendarski University of Plovdiv
24, Tsar Assen Str, Plovdiv, Bulgaria
rosdek@uni-plovdiv.bg

Abstract

Following a concise description of the structure, the paper focuses on the potential of the Ontology of the Bulgarian Dialects, which demonstrates a novel usage of the ontological modelling for the purposes of dialect digital archiving and information processing. The ontology incorporates information on the dialects of the Bulgarian language and includes data from 84 dialects, spoken not only on the territory of the Republic of Bulgaria, but also abroad. It encodes both their geographical distribution and some of their main diagnostic features, such as the different mutations (also referred to as reflexes) of some of the Old Bulgarian vowels. The mutations modelled so far in the ontology include the reflex of the back nasal vowel /ѫ/ under stress, the reflex of the back *er* vowel /ъ/ under stress, and the reflex of the *yat* vowel /ѣ/ under stress when it precedes a syllable with a back vowel. Besides the opportunity for formal structuring of the considerable amount of data gathered through the years by dialectologists, the ontology also provides numerous possibilities for information retrieval – searches by dialect, country, dialect region, city or village, various combinations of diagnostic features.

Keywords: ontology, Bulgarian dialects, Protégé

1. Introduction

Although the geographic distribution and the characteristic features of the various dialects of the Bulgarian language have been the focus of many research studies through the years, little has been done in the direction of their digital archiving. Some of the few projects worth mentioning are the *Electronic Interactive Map of the Bulgarian Dialectal Division*[1], the *Bulgarian Dialectology as Living Tradition* website[2] and the attempt for storing and processing dialect data by using an XML-based model within the CLaRK system (Osvenova and Simov, 2007). While the first two are predominantly user-oriented and aim at making the overall information on the Bulgarian dialects visible to the general public as well as to specialists, the last one is focused mainly on the adequate computerization of the linguistic data for easy processing and portability.

Recent advances in employing ontologies and ontological modelling for various task-specific and more generalized applications have proven rather powerful and quite adequate in capturing also the specifics of the natural languages. For the purposes of this project, we use the term *ontology* in the sense of Gruber (2009) to specify a set of representational primitives for modelling a domain of knowledge, where the definitions of these primitives must include information on their meaning and constraints that ensure their logically consistent application.

The work presented in the current paper is a novel attempt at using the advantages of ontological modelling for encoding information, both linguistic and extra linguistic, about the Bulgarian dialects. The ontology incorporates data from 84 dialects and it encodes their geographical distribution, as well as some of their main diagnostic features – the different mutations (also called reflexes) of some Old Bulgarian vowels. The mutations modelled so far include the reflex of the back nasal vowel (ѫ) under stress, the reflex of the back *er* vowel (ъ) under stress, and the reflex of the *yat* vowel (ѣ) under stress when it precedes a syllable with a back vowel. Besides the formal structuring of ample data gathered by dialectologists across decades, the ontology also offers numerous possibilities for retrieving information related to dialect, dialect region, (combinations of) diagnostic features, etc.

[1] Available at: http://ibl.bas.bg//bulgarian_dialects/.

2. Technologies Implemented

Besides the concept of the ontological modelling as such, we have applied a number of different technologies to ensure the correct representation of the dialect data and to explore the various potentials for information retrieval.

2.1 Protégé

The Ontology of the Bulgarian Dialects was created with the open-source ontology editor Protégé, which is freely distributed by the Stanford Center for Biomedical Informatics Research at the Stanford University School of Medicine. Based on Java, Protégé is extensible and provides a plug-and-play environment that makes it a flexible base for both prototyping and application development (Musen, 2015). Finally, Protégé was favoured also because it is compliant with the W3C standards and fully supports the newest specifications of OWL 2.

2.2 The OWL2 Web Ontology Language

We used the OWL 2 Web Ontology Language (Hitzler, 2012) to express the extensive and multifaceted knowledge related to dialects, thus utilizing the wide potential of the classes, properties, individuals, and data values provided by the OWL 2 ontologies. We chose OWL 2 because it is a computational logic-based language that readily allows for the data encoded in it to be automatically checked for consistency, using an OWL Reasoner. Among other capacities for coding and structuring knowledge, OWL 2 provides also the option of including a rich variety of metadata as annotations that can be appended to classes and individuals as well as to properties and restrictions. This proved to be particularly useful in expressing dialect data.

2.3 The OWL API and the Hermit Reasoner

For the processing of the ontology and the retrieval of information, we used the class-based object-oriented programming language Java (JDK 1.8) and the Java compatible OWL API reference implementation for creating, manipulating and serialising OWL Ontologies (Horridge & Bechhofer, 2011). We also employed the Hermit OWL Reasoner (Glim et al., 2014) for consistency validation and for extracting implicit knowledge.

[2] Available at: http://bulgariandialectology.org/.

3. Building the Ontology

The overall architecture of the ontology follows the guiding principles and complies with the data standards for cataloguing cultural objects (Baca, 2006). Thus, it could be easily incorporated in larger projects for ontological modelling of knowledge on cultural and historical heritage. The data included in the ontology have been collected through the years and classified by the researchers at the Department of Bulgarian Dialectology and Linguistics Geography at the Institute for Bulgarian at the Bulgarian Academy of Sciences (Kochev, 2001).

So far, the ontology encodes information only on the geographical distribution of the Bulgarian dialects and some of their most prominent diagnostic features as described in Antonova-Vasileva et al. (2014).

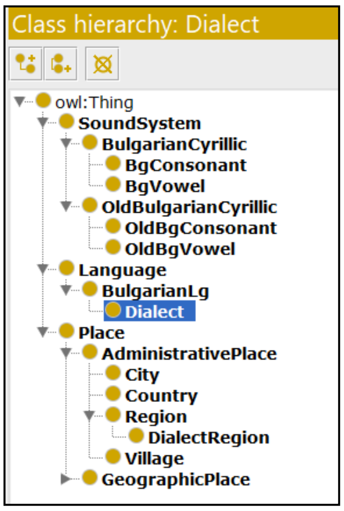
3.1 General Characteristics of the Ontology

The ontological model consists of a hierarchy of representational primitives (such as classes, properties, and relationships) designed to describe the specifics of the dialect knowledge.

Currently, the ontology contains 24 classes encompassing 384 individuals. We have defined 17 object properties and there are 1998 axioms. The data covers 102 dialect regions, 84 dialects, and over 150 administrative places (i.e. cities, villages, and countries). We have described 25 different mutations of the Old Bulgarian vowels /ъ/, /ѫ/, and /ѣ/. There are also 369 annotation assertions, which are mostly of the type <rdfs:label xml:lang="bg">, so that all the data can be readily visualised through a graphical user interface.

3.2 Specific Classes and Their Properties

Figure 1 shows the overall architecture of the ontology with respect to the hierarchy of classes and the hierarchy of the object properties is visualized in Figure 2, followed by a concise presentation of some of the most essential classes, together with their key characteristics (see Dekova (2019) for a more detailed description in Bulgarian).

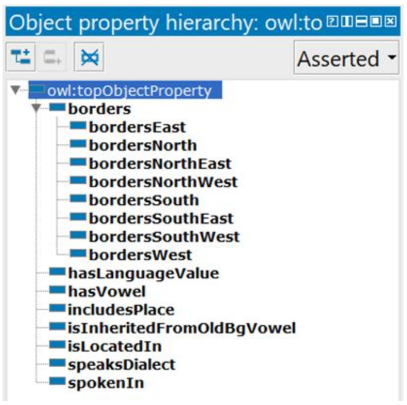


Class hierarchy: Dialect

- owl:Thing
  - SoundSystem
    - BulgarianCyrillic
      - BgConsonant
      - BgVowel
    - OldBulgarianCyrillic
      - OldBgConsonant
      - OldBgVowel
  - Language
    - BulgarianLg
      - Dialect
  - Place
    - AdministrativePlace
      - City
      - Country
      - Region
        - DialectRegion
      - Village
    - GeographicPlace

Figure 1: Class hierarchy.

The Bulgarian dialects are defined as instances of the class Dialect. The most important axioms for the class Dialect refer to the places where it is spoken (*spokenIn* some *DialectRegion*) and the mutations of the Old Bulgarian vowels which are specific for this dialect (*hasVowel* some *BgVowel*).



Object property hierarchy: owl:to

Asserted

- owl:topObjectProperty
  - borders
    - bordersEast
    - bordersNorth
    - bordersNorthEast
    - bordersNorthWest
    - bordersSouth
    - bordersSouthEast
    - bordersSouthWest
    - bordersWest
  - hasLanguageValue
  - hasVowel
  - includesPlace
  - isInheritedFromOldBgVowel
  - isLocatedIn
  - speaksDialect
  - spokenIn

Figure 2: Hierarchy of the object properies included in the ontology at present.

The object property *spokenIn* relates a dialect with an administrative or geographic location where this dialect can be found. Thus, each dialect is related to at least one place, but multiple relations of this type are also conceivable. For example, the Eastern Sub-Balkan dialect is related to three places: the cities Yambol and Sliven, and the Eastern Sub-Balkan dialect region.

The various dialect regions of Bulgarian described in the works of Kochev (2001) and Stoikov (1993) are defined as instances of the class *DialectRegion*. Since almost each dialect (and respectively its region) belongs to a larger group of dialects sharing similar features, we have modelled a hierarchy of the dialectal division as shown in Figure 3 (the colour coding complies with the *Electronic Interactive Map of the Bulgarian Dialectal Division*).

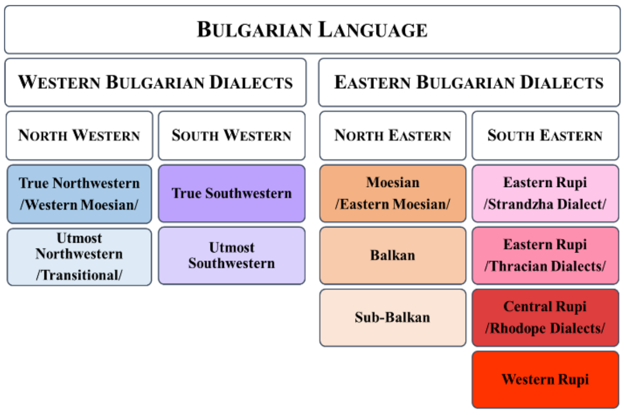
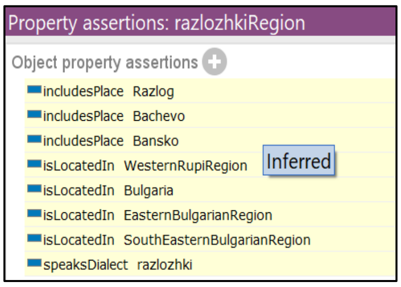

Bulgarian Language
Western Bulgarian Dialects
Eastern Bulgarian Dialects
North Western
South Western
North Eastern
South Eastern
True Northwestern /Western Moesian/
True Southwestern
Moesian /Eastern Moesian/
Eastern Rupi /Strandzha Dialect/
Utmost Northwestern /Transitional/
Utmost Southwestern
Balkan
Eastern Rupi /Thracian Dialects/
Sub-Balkan
Central Rupi /Rhodope Dialects/
Western Rupi

Figure 3: Hierarchy of the dialectal division of the Bulgarian language.

The transitive object property *isLocatedIn* and its reverse property *includesPlace* allow us to encode the relations of the various administrative and geographical locations within each other. Hence the assertion *includesPlace* some *DialectRegion* allows us to specify only the direct children of each larger region. Similarly, for the instances of type *Place* we assert only the direct hierarchical parent using the property *isLocatedIn* some *DialectRegion*.

Thus, with the assertions *spokenIn* and *includesPlace* and their respective transitive properties *speaksDialect* and *isLocatedIn*, we are able to include a lot of information in the ontology in a very effective way. Figure 4 shows an example of an instance of type *DialectRegion* where all the assertions are automatically inferred using the HermiT OWL Reasoner (Glim et al., 2014).


Property assertions: razlozhkiRegion

Object property assertions

includesPlace Razlog
includesPlace Bachevo
includesPlace Bansko
isLocatedIn WesternRupiRegion
isLocatedIn Bulgaria
isLocatedIn EasternBulgarianRegion
isLocatedIn SouthEasternBulgarianRegion
speaksDialect razlozhki

Figure 4: Inferred object property assertions for the Razlog dialect region.

To express the linguistic knowledge about some of the most prominent diagnostic features of the Bulgarian dialects we introduced the class *SoundSystem* with two subclasses *BulgarianCyrillic* and *OldBulgarianCyrillic* (see Figure 1). The different mutations (reflexes) of the Old Bulgarian vowels /ъ/, /ѫ/, and /ѣ/ were then defined as instances of the type *BgVowel*, where for each instance holds the assertion *isInheritedFromOldBgVowel* some *OldBgVowel*. Thus, we do not have a single instance for the sound /u/, for example, but we define the separate instances *UfromBigEr* and *UfromBigYus* and we use the assertion to relate them to their respective ancestors – the Old Bulgarian vowels /ъ/ and /ѫ/. The basic idea for this structure is the inherent relation between Old Bulgarian and modern Bulgarian sounds. These reflexes differ in the different dialects, i.e. one and the same Old Bulgarian vowel has mutated differently in the different dialects. For example, the Old Bulgarian back nasal vowel /ѫ/ has become /ъ/ in some dialects (e.g. in the Ihtiman, the Balkan, and the Sub-Balkan regions), while in others it has mutated to /a/ (e.g. Botevgrad, Vratsa, Pirdop), /u/ (e.g. Belogradchik, Negotino, Breznik), /o/ (e.g. the Devisilovo region), /ȏ/ (e.g. Smolyan, Shiroka Laka), etc.

Figure 5 below shows an OntoGraf of those instances of the type *BgVowel* which are related to their ancestor *bigYus* of the type *OldBgVowel*, i.e. all the mutations of the Old Bulgarian back nasal vowel /ѫ/ found in dialects.

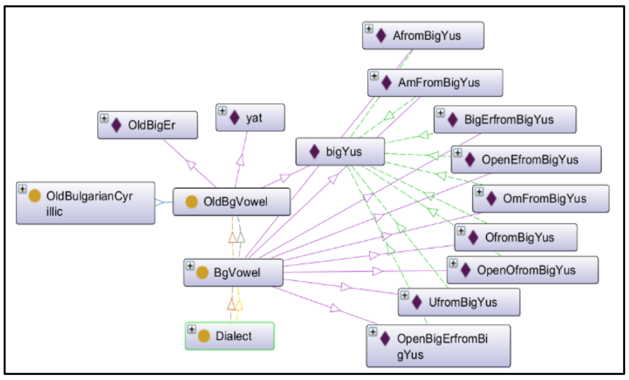
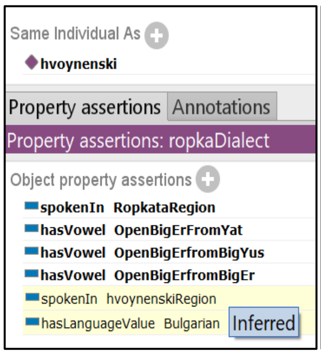


Figure 5: An OntoGraf of some BgVowel instances.

Hence, the knowledge for each dialect is encoded by a set of property assertions including the relations *spokenIn* and *hasVowel*, which enable us to generalize and extract further information.

Figure 6a offers an example of the amount and types of data presently available for most of the dialects in the ontology, while Figure 6b illustrates the inferred property assertions based on the relation *Same Individual As*, which indicates that the two dialects, i.e. the Ropka dialect and the Hvoyna dialect are one and the same individual. Similarly, *Same Individual As* relates the two dialect regions and entails the inferred property assertions *spokenIn*, which is also visible in Figures 6a and 6b below.


Same Individual As

hvoynenski

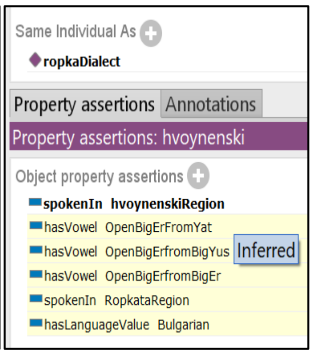
Property assertions | Annotations

Property assertions: ropkaDialect

Object property assertions

spokenIn RopkataRegion
hasVowel OpenBigErFromYat
hasVowel OpenBigErfromBigYus
hasVowel OpenBigErfromBigEr
spokenIn hvoynenskiRegion
hasLanguageValue Bulgarian

Figure 6a: Property assertions for the Ropka dialect.


Same Individual As

ropkaDialect

Property assertions | Annotations

Property assertions: hvoynenski

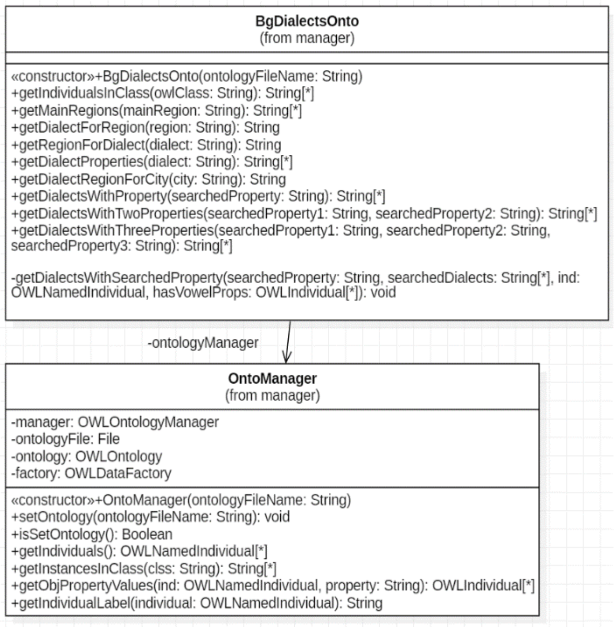
Object property assertions

spokenIn hvoynenskiRegion
hasVowel OpenBigErFromYat
hasVowel OpenBigErfromBigYus
hasVowel OpenBigErfromBigEr
spokenIn RopkataRegion
hasLanguageValue Bulgarian

Figure 6b: Property assertions for the Hvoyna dialect.

4. Ontology Processing and Information Retrieval

Since the ontology was not created with the sole objective of digital archiving but also with the intent of data visualisation and information retrieval oriented towards a broader audience, we developed the BgDialectsOntology project so as to take care of the queries and manage the data to be retrieved. The project was implemented in Java (JDK 1.8) in the IntelliJ IDEA programming environment[3]. We have also imported the libraries of OWL API (Horridge & Bechhofer, 2011) and the HermiT Reasoner (Glim et al., 2014), as they were not part of the standard Java libraries. For ease of portability, the ontology .owl file was imported in a separate package (*resources-Ontology*).


```
BgDialectsOnto
(from manager)

«constructor»+BgDialectsOnto(ontologyFileName: String)
+getIndividualsInClass(owlClass: String): String[*]
+getMainRegions(mainRegion: String): String[*]
+getDialectForRegion(region: String): String
+getRegionForDialect(dialect: String): String
+getDialectProperties(dialect: String): String[*]
+getDialectRegionForCity(city: String): String
+getDialectsWithProperty(searchedProperty: String): String[*]
+getDialectsWithTwoProperties(searchedProperty1: String, searchedProperty2: String): String[*]
+getDialectsWithThreeProperties(searchedProperty1: String, searchedProperty2: String,
searchedProperty3: String): String[*]

-getDialectsWithSearchedProperty(searchedProperty: String, searchedDialects: String[*], ind:
OWLNamedIndividual, hasVowelProps: OWLIndividual[*]): void

-ontologyManager

OntoManager
(from manager)

-manager: OWLOntologyManager
-ontologyFile: File
-ontology: OWLOntology
-factory: OWLDataFactory

«constructor»+OntoManager(ontologyFileName: String)
+setOntology(ontologyFileName: String): void
+isSetOntology(): Boolean
+getIndividuals(): OWLNamedIndividual[*]
+getInstancesInClass(clss: String): String[*]
+getObjPropertyValues(ind: OWLNamedIndividual, property: String): OWLIndividual[*]
+getIndividualLabel(individual: OWLNamedIndividual): String
```

Figure 7: An UML class diagram of the *BgDialectsOnto* and *OntoManager* packages and their methods.

[3] Available at: https://jetbrains.com/idea/.

The project comprises two main packages (*BgDialectsOnto* and *OntoManager*) that process the ontology and retrieve the necessary information (illustrated in Figure 7).

In addition, there is also an interface package (*IBGDialectsOntology*) which contains all the methods that might be used to extract information from the ontology, and defines the parameters that could be sent by the graphical user interface. Thus, we enable users to choose among different criteria for filtering and extracting information. These criteria are the parameters received by the interface, where the information required by the ontology is extracted by calling the appropriate method(s). Once the ontology has been processed, the knowledge matching the query is displayed by means of its associated annotation <rdfs:label xml:lang="bg">.

Thus, at the current stage of the project, users could search for and retrieve information on a dialect region and its sub-regions (if any); a place (city or village) and its dialect; a dialect and its diagnostic features.

More details on the potential for processing the ontology and data queries are discussed below in relation to the description of the respective methods in the packages *OntoManager* the *BgDialectsOnto*.

We divided the methods into two separate packages to obtain a higher level of efficiency. Thus, *OntoManager* encompasses all the methods that access the ontology directly and retrieve the necessary information using external org.semanticweb libraries, such as owlapi.model, owlapi.apibinding.OWLManager, owlapi.reasoner, and HermiT.Reasoner.

BgDialectsOnto, on the other hand, is designed to implement the methods from the *IBGDialectsOntology* interface and it handles only individuals. For any information accessible through ontology processing, it uses the methods of *OntoManager*.

4.1 Processing the Ontology

Besides the indispensable methods for setting and verifying the status of the ontology, we defined a number of methods, which are crucial for managing the ontological knowledge and retrieving relevant information.

4.1.1 Extracting ontology individuals

To obtain ontology individuals, we have defined the getIndividuals method. The method is public and implements the method getIndividualsInSignature() of the org.semanticweb.owlapi.model.OWLOntology class. As a result, it returns a list of the type OWLNamedIndividual, which can then be handled by the *BgDialectsOnto* methods.

4.1.2 Extracting instances by class

To retrieve the individual(s) that we want to visualise in the GUI drop-down lists so as to offer various searches by dialect, place or vowel mutation, we have defined the public method getInstancesInClass. It accepts a String variable that is set to specify the name of the class the instances of which we need to extract. The result is of the type List <String> and contains the Bulgarian labels of all the instances of the class. For example, for the String variable OldBgVowel the getInstancesInClass method returns the label annotations for the Old Bulgarian vowels included so far in the class, i.e. the back er vowel (ъ), the back nasal vowel (ѫ), and the yat vowel (ѣ) – [[задна ерова гласна [ъ], задна носова гласна [ѫ], ятова гласна [ѣ]].

4.1.3 Extracting object property values

To obtain the specific values of a particular object property, we defined the method getObjPropertyValues. It is public and it accepts two variables: one of the type OWLNamedIndividual and the String variable *property*. As a result, the method returns a set of the type OWLIndividual, i.e. the set of all values of the particular object property (e.g. *hasPlace, isLocatedIn, spokenIn, hasVowel*) assigned to a concrete object.

This method is implemented in almost all of the searches as it allows us to obtain all individuals with specific object property values and to extract those that match a particular query.

4.1.4 Extracting the <rdfs:label> annotation

As it is necessary to visualise the results of the queries in a format that is relevant and understandable to the users, we have designed the getIndividualLabel method to extract the annotations of the type <rdfs:label xml:lang="bg">.

The method accepts a variable of the type OWLNamedIndividual and returns a result of the type String, i.e. the name or the description of the individual in Bulgarian. Although currently Bulgarian is the only language included under this label, we plan to include label information for English as well.

4.2 Retrieving Information

One of the main strengths of the ontological modelling of knowledge lies in its potential for retrieving information of various types and even such that has not been explicitly encoded. The ontology presented in this paper was created as a distinct module of an ongoing project for building an intelligent system for the ontological modelling of the Bulgarian dialects (Dekova, 2018). Although the GUI module is still under construction, it was important for us to envisage and design a good number of search options already at this stage. These are implemented through the methods included in the *BgDialectsOnto* package and can be easily employed by the graphical user interface through the *IBGDialectsOntology* interface package.

4.2.1 Searches by Dialect or Geographic Place

The dialect search allows the user to select one of the dialects from a drop-down menu or to enter any string of characters in the search field. In the second case, the system responds interactively by correcting the dialect list, leaving only those dialects that include the user's input string.
To create the drop-down lists that are displayed to the user when searching by dialect or place, we defined the public method getIndividualsInClass. The method implements the getInstancesInClass method (described in 4.1.2 above), with the variable owlClass, which receives its value from the GUI input field, that is, the name of the class whose individuals we want to display in the drop-down menu and returns as a result a list of strings. This comprises the labels of those individuals, whether dialects, dialect regions, or locations like cities or villages.

Once the dialect is selected, its name is given as the value of the variable *dialect* of the public method getRegionForDialect. It implements the OntoManager methods getIndividuals and getObjPropertyValues, the

latter specified for the property *spokenIn*, and it returns information about the region where this dialect is spoken.
To return the specific characteristics of the dialect (its main diagnostic features), we have defined the public method getDialectPoperties. It accepts the variable *dialect* of type String and returns a list of type String as a result. This method gets its input parameter from the graphical user interface when the user selects a particular dialect for which they want to receive more information. As a result, a list of values of the hasVowel property for the particular dialect is returned, displaying their labels to the user. These are the specific features of the dialect chosen by the user.

For example, if we search for the Razlog dialect, we will get as a result the region where it is spoken (the Razlog dialect region) and the set of diagnostic features included in the ontology, i.e. the specific mutations of the Old Bulgarian vowels /ъ/, /ѫ/, and /ѣ/ in this particular dialect.
Searching for a geographic location employs the same principles as searching for a dialect, enabling the user to write the name of a city or a village. If the ontology has some information about this place, it will be displayed.
The method used for this search is the public method getRegionForCity, which behaves quite similarly to the getRegionForDialect method described above.

4.2.2 Retrieving the sub-regions of a dialect region

To retrieve the sub-regions of a specific dialect region, we defined the *getMainRegions* method – a non-recursive method which does not use automatically generated logic properties and outputs only the explicitly specified values set for the *inlcudesPlace* property of the selected dialect region. The *getMainRegions* method is public, accepts a *mainRegion* variable of type String and returns a list of type String as a result. This method expects a string as the input parameter, which is the name of the dialect region selected by the user through the graphical user interface and returns a list of strings – labels of the dialect sub-regions included in that region, which will be displayed as a result for the user.

4.2.3 Searching for dialects with specific properties

The search for dialects with specific characteristics enables the user to select from of a drop-down menu any of the possible mutations for each of the three Old Bulgarian vowels – the reflex of the back nasal vowel /ѫ/ under stress, the reflex of the back *er* vowel /ъ/ under stress, and the reflex of the *yat* vowel /ѣ/ under stress in a position preceding a syllable with a back vowel. This provides the users with the unique possibility to initiate a search by one, two, or all three criteria, and to receive as a result the set of dialects that meet these criteria.

To implement this type of search, we have defined three individual methods corresponding respectively to the user's decision to choose one, two or all three parameters.
The getDialectsWithPoperty method accepts only one String variable and is called in the cases where the user has selected only one parameter from any of the three drop-down menus. The method retrieves all instances of type *Dialect* that have the value selected by the user for the hasVowel property. The method returns as a result a list of labels of the dialects that meet the criterion.

The getDialectsWithTwoPoperties method accepts two variables of type String and is called when the user has selected two parameters from the drop-down menus. The method retrieves all instances of type Dialect that have the value selected by the user for the hasVowel property. Then only those that meet the second requirement are preserved. The method returns as a result a list of the labels of those dialects that have both characteristics.

The getDialectsWithThreePoperties method behaves similarly to the previous method but accepts three String variables and is called when the user has selected one parameter from each of the drop-down menus. The method returns as a result a list of labels of many dialects that have all three diagnostic features the user has selected.

All three methods implement the private void method getDialectsWithSearchedProperty with the four variables (String *searchedProperty*, List<String> *searchedDialects*, OWLNamedIndividual *ind*, Set<OWLIndividual> *hasVowelProps*). The first variable gets its value from the user's input. The second one is the list to which the dialects that have the specified characteristic are appended. The third variable is the specific individuality (dialect) for which the method verifies whether it meets the criterion, i.e. whether it has the desired characteristic. The last variable represents the set of values for the hasVowel property. Each element of type OWLIndividual from the hasVowelProps set is converted to an OWLNamedIndividual to check whether it matches the desired characteristics. If it does, it takes the value from the label annotation of the variable *ind* and adds it to the *searchedDialects* list.

In this way, the getDialectsWithSearchedProperty method undertakes a substantial part of the search and is used by the other three methods to achieve greater efficiency.

For example, a search with the following combination of criteria <AfromOldBigEr; AfromBigYus; EfromYat>[4] returns the set of dialects where each dialect has these three specific mutations – the reflex /a/ of the back *er* vowel /ъ/ under stress, the reflex /a/ of the back nasal vowel /ѫ/ under stress, and the reflex /e/ of the *yat* vowel /ѣ/ under stress when it precedes a syllable with a back vowel. As a result the user gets the following set of dialects: [vrachanski, shtipsko-radovishki, negotinski, iztochnosofiyski (elinpelinski), samokovski, veleshko-prilepsko-bitolski, razlozhki, dupnishki].

This search allows not only for retrieving and comparing dialects with similar characteristics but it also allows for discovering of previously undetected analogies. Therefore, it is foreseen as particularly valuable for specialists in the field of dialectology, but it can also serve users with non-specialist interests and the general public, more broadly.

[4] Here for convenience, we use the names of the reflexes and the names of the dialects as they are defined in the ontology. The user will see the respective annotation labels in Bulgarian.

5. Conclusions and Future Work

Although the data incorporated in the current stage of the ontology represent a tiny part of the knowledge related to the Bulgarian dialects, we believe that the ontological modelling of dialectal records has a number of significant advantages and it certainly deserves further development.

Our next steps include expanding the data related to particular phonological characteristics of the Bulgarian dialects, as well as expressing some of the more specific lexical and morpho-syntactic features, such as the dialectal phonetic variants of the morpheme denoting definiteness in singular masculine adjectives and nouns or the various of the future tense auxiliary verb form *ще* /ʃte/ (e.g. *ше* /ʃe/, *же* /ʒe/, *че* /tʃe/, *ке* /ke/, *за* /za/, etc.).

We are also working on the visibility of the data by creating a suitable graphical user interface that will allow a good visualisation for the processing and extracting information by the wider public.

Designed in compliance with the CCO data standards (Baca, 2006), the architecture of the ontology also allows it to be incorporated in different projects and applications related to the digitalization and exploration of cultural and historical heritage, such as the *Personal Internet of Things Tourist Guide* developed at the Faculty of Mathematics and Informatics, at the Paisii Hilendarski University of Plovdiv (Glushkova et al., 2018).

The conceptual modelling of data related to dialects is a novel usage and as far as we know, there are no similar studies on dialects of other languages. Although the features encoded in the ontology seem rather specific for the historical development of the Bulgarian language and its dialects, the overall design of the hierarchies of classes and properties can be easily adapted and applied to other languages. Therefore, it offers scope for valuable contribution that goes beyond national and regional borders.

6. Acknowledgements

This work was conducted using the Protégé resource, which is supported by grant GM10331601 from the National Institute of General Medical Sciences of the United States National Institutes of Health.

I gratefully acknowledge the anonymous reviewers for their valuable suggestions and my colleagues Maria Miteva and Milena Katsarska for their precious advice and friendly support along the way.

7. Bibliographical References

Alexander, R. and Zhobov, V. (2016) Bulgarian Dialectology as Living Tradition. Available at: http://www.bulgariandialectology.org (02.03.2020)

Antonova-Vasileva, L., Vasileva, L., Keremedchieva, Sl., Kocheva-Lefedzhieva, A. (2014) Dialektnata delitba na bulgarskiya ezik. http://ibl.bas.bg/Dialektna_delitba.pdf (12.03.2020)

Baca, M. (2006). *Cataloging cultural objects: a guide to describing cultural works and their images*. Chicago, American Library Association.

Dekova, R. (2018). *An Intelligent System for Ontological Representation of the Bulgarian Dialects: the Ontological Model*. MSc Thesis (in Bulgarian, unpublished), Faculty of Mathematics and Informatics, Paisii Hilendarski University of Plovdiv.

Dekova, R. (2019). An Ontological Model of the Dialectal Division of the Bulgarian Language. In V. Micheva, D. Blagoeva, S. Kolkovska, T. Aleksandrova and Hr. Deykova (Eds.), *Proceedings of the International Annual Conference of the Institute for Bulgarian Language Prof. Lyubomir Andreychin (Sofia, 14th – 15th May 2019)*. Prof. Marin Drinov Press of the Bulgarian Academy of Sciences, Sofia, pp. 309—318, (in Bulgarian).

Glimm, B., Horrocks, I., Motik, B., Stoilos, G., and Wang, Z. (2014). HermiT: An OWL 2 Reasoner, *Journal of Automated Reasoning*, 53:245–269.

Glushkova, T., Miteva, M., Stoyanova-Doycheva, A., Ivanova, V., Stoyanov, S. Implementation of a Personal Internet of Thing Tourist Guide, *American Journal of Computation, Communication and Control*, 5(2):39–51.

Gruber, T. R. (2009) Ontology. In Liu, L. and Özsu, M. T. (Eds.) *The Encyclopedia of Database Systems*, Springer-Verlag.

Horridge, M. and Bechhofer, S. (2011) The OWL API: A Java API for OWL Ontologies, Semantic Web Journal 2(1), Special Issue on Semantic Web Tools and Systems, pp. 11--21.

Kochev, I. (2001). *Bulgarski dialekten atlas. Obobshtavasht tom. Part. I–III Fonetika. Aktsentologiya. Leksika*. KIK Trud, Sofia.

Musen, M. A. (2015). The Protégé project: A look back and a look forward. *AI Matters. Association of Computing Machinery Specific Interest Group in Artificial Intelligence*, 1(4):4–12.

Osenova, P. and Simov, K. (2007). An Infrastructure for Storing and Processing Dialect Data. In *Bulgarian Islands on Balkans*, Figura publ., 2007, pp. 256--263.

Pascal, H., Krötzsch, M., Parsia, B., Patel-Schneider, P.F., and Rudolph, S. (Eds.). (2012). OWL 2 Web Ontology Language: Primer. W3C Recommendation. Available at: http://www.w3.org/TR/OWL 2-primer/ (27.11.2019)

Stoikov, S. (1993). *Bulgarska dialektologiya*. Bulgarian Academy of Sciences, Sofia.