# The expansion of image information using ontologies to enhancing the efficiency of data retrieval tasks

Martina Radilova [a,*], Patrik Kamencay [a], Roberta Hlavata [a] and Slavomir Matuska [a]
[a] *Department of Multimedia and Information-Communication Technology, University of Zilna, Slovakia*
*E-mails: martina.radilova@uniza.sk, patrik.kamencay@uniza.sk, roberta.hlavata@uniza.sk, slavomir.matuska@uniza.sk*

**Abstract.** This paper explores the enhancement of image information descriptions using ontologies to improve the efficiency of data retrieval tasks. Initially, the paper reviews current methods of describing image information in databases, private servers, and web documents. Based on this review, we propose an improved approach that provides more accurate and detailed descriptions. Our proposed method involves the use of a custom-designed ontology specifically created to semantically describe image information in web documents. This ontology enables a richer and more nuanced representation of image data, facilitating better understanding and retrieval. The proposed approach aims to significantly enhance the accuracy and efficiency of data retrieval in both databases and web search engines. And for that reason, at the end of the article, we described our application for verifying ontology deployment in a simple parser. After deploying the ontology, we found that after extracting the images from the web page, we got a much more accurate description of the animal than what was provided to us from the web document. The success of our parser in determining the more detailed description depended on the sample used of randomly selected web documents and the data extraction success rate was 88%, with 72% of outputs being correctly filtered. In addition, animal identification success was observed be up to 90%.

Keywords: Ontology, semantic description, Web Ontology Language, Resource Description Framework

## 1. Introduction

The exponential growth of digital images across various domains, from social media to medical imaging, has created a pressing need for efficient and accurate image retrieval systems. Traditional image retrieval methods often rely on basic metadata or manual annotations, which can be inconsistent and fail to capture the rich semantic content of images. As a result, there is a growing interest in developing more sophisticated techniques that can enhance the description and retrieval of image information. One promising approach to addressing this challenge is the use of ontologies. Ontologies provide a structured and formal representation of knowledge within a specific domain, enabling the encoding of complex relationships and semantic information. When applied to image information, ontologies can offer a more comprehensive and nuanced description of image content, which can significantly improve retrieval performance.

Nowadays the creation of descriptions of image information is possible in several methods. An increasingly used method for creating image descriptions is the neural network method. In spite of its good results, it still comes

*Corresponding author. E-mail: martina.radilova@uniza.sk.

1570-0844/$35.00 © 0 – IOS Press. All rights reserved.

only close to the description made by humans. A description of an image cannot be created semantically like the human brain. Therefore, humans are still irreplaceable in this process. Traditional methods of describing images rely heavily on metadata, which is often manually annotated and prone to inconsistencies and incompleteness. These descriptions are typically limited to basic attributes such as file name, date, and simple tags, which fall short of capturing the semantic richness of the image content.

This paper explores the integration of ontologies with image information to enhance the efficiency of data retrieval tasks. We begin by examining the current methods used to describe image information in databases and web documents. These methods typically involve manual annotation and metadata, which, despite their utility, have notable limitations in terms of scalability and semantic depth. Building the ontology itself is a very difficult process, so we decided to start with a smaller part of the ontology and test it gradually so that we know how to expand it further and what direction to take. Our first phase of research was focused on animals, so our ontology is based on animals. In order to build the ontology, we took as a model the hierarchical division of animals according to the encyclopedia BIOpedia [1]. The ontology is identified by a Uniform Resource Identifier (URI). The ontology has a hierarchical structure of classes, sub-classes and relationships between classes. We have successively divided the animal kingdom into the sub-kingdoms "unicellular" and "multicellular", tribes, sub-tribes, classes, orders, families, genera, and species. The whole ontology is created using Web Ontology Language (OWL), Resource Description Framework Schema (RDFS) and the Protégé environment [2]. Finally, we tested the built ontology on online image information available on HTML pages that already had a description. In the individual chapters of the paper, we will discuss in more detail the individual steps of building the ontology itself, as well as its testing. Finally, we evaluate the functionality of the ontology, its possible uses, and its extension in the form of self-learning. In the following sections, we delve into the specifics of our approach, starting with a review of current image description methods, followed by the design and implementation of our ontology, and concluding with discuss how to improve and direct our research. Through this work, we aim to demonstrate the potential of ontologies in enhancing image information retrieval and to pave the way for future advancements in this field.

Image retrieval remains a critical challenge, particularly in domains with vast and complex visual data. While numerous approaches have been proposed, including content-based image retrieval (CBIR) and text-based image retrieval (TBIR), significant limitations persist, particularly in bridging the semantic gap between low-level image features and high-level semantic concepts. Recent efforts have explored the use of ontologies to address this gap, enabling more semantic and context-aware image retrieval. However, many existing ontology-based image retrieval systems suffer from limitations in scalability, expressiveness, and the ability to effectively integrate visual and semantic information. They also tend to not be reproducible due to lack of focus on the concepts. This paper introduces a novel approach to ontology-based image retrieval that addresses these limitations by dynamically expanding image information within the ontology based on visual features and user feedback. Unlike traditional approaches that rely on static, pre-defined ontologies, our system [Explain how our system dynamically adapts or improves the ontology, connecting visual data with semantic knowledge]. This results in significantly improved retrieval accuracy, particularly for complex queries and ambiguous image content.

Our work differs from existing semantic image retrieval approaches in several key aspects:

- Point 1: We employ a novel method for automatically extracting and integrating visual features into the ontology, reducing the reliance on manual annotation.
- Point 2: Our system incorporates a feedback mechanism that allows users to refine the ontology over time, improving its accuracy and relevance.
- Point 3: We address the scalability issue by using a modular ontology design and optimized reasoning algorithms.

The remainder of this paper is structured as follows: Section 2 provides a more detailed overview of related work in ontology-based image retrieval, highlighting the limitations that our approach seeks to overcome. Section 3 describes our proposed system architecture and methodology, focusing on the novel aspects of our approach.

## 2. Related Work

The quantity of images and videos on the Internet, as well as on various private storage sites, is increasing annually. In the context of information retrieval, the sheer volume of data makes it challenging to navigate. To address this challenge, specialized techniques are employed to search for specific information, as discussed in [3]:

- Text-Based Image Retrieval (TBIR),
- Content-Based Image Retrieval (CBIR),
- Semantic-Based Image Retrieval (SBIR).

The search operations conducted in the TBIR and CBIR domains are based on the utilization of keyword or image elements. In the context of TBIR retrieval operations, a keyword is employed, whereas in CBIR retrieval operations, features extracted from the image are utilized, including those pertaining to color, shape, and texture. Such circumstances may result in the presentation of search results that are not aligned with the specified keyword or selected elements. While two images may exhibit identical color, shape, and texture characteristics, their underlying content may diverge. Consequently, images that do not correspond to the specified keywords or extracted features will also be displayed. The primary challenge inherent to TBIR and CBIR processes is known as semantic gap [4]. The semantic gap accounts for the differences between human and computer interpretation of an image. Methods such as machine learning algorithms [5], the use of different feature descriptors [6], the use of correlations between multiple data modalities [7], the discovery of semantic rules between humans and computers [8], and the use of interactive models are proposed to bridge this gap [9].

In this novel approach, exemplified by the Semantic Web, ontologies are employed to model concepts. The inferred meaning obtained through visual relations is transformed into a structure that can be processed by computers using ontologies. This structured representation is then employed to retrieve other images with the same meaning. Semantics-based image retrieval (SBIR) is the process of retrieving images from a database that exhibit analogous relationships between the objects depicted in the image and predefined objects. The relationships within the structure are transformed into a structure that can be utilized by computers to construct meaning through the use of ontologies, thereby enabling the retrieval of images that are similarly meaningful. This approach has been demonstrated to be effective in various studies, including those by [10], [11]. In [12], the authors propose a two-step methodology for developing a semantics-based image retrieval system with ontology support. In the initial phase, objects are identified within the image through the implementation of an object detection process, and a predicate that describes the relationship between two objects is established using the developed bidirectional recurrent neural network (Bi-RNN) model [13, 14]. Subsequently, in the second phase, the aforementioned relations are transformed into ontologies and utilized for the retrieval of images exhibiting a high degree of semantic similarity. In evaluating the performance of the developed model, a noteworthy 91% accuracy was attained, as evidenced by the Recall@100 (Top-5 accuracy) result. In [15], the authors present a query expansion algorithm based on an improved Aquila Optimization-based COOT (IAOCOOT), which extracts semantic aspects that match the query term. The proposed IAOCOOT algorithm offers flexibility for updating new information. The Modified Needleman Wunsch Algorithm (MNWA) is used to identify word similarity from both global and local perspectives. Local information is obtained using the symmetry property, while global information is obtained using the matching property.

Malik et al. [16] investigated the Query Expansion (QE) framework by developing a hybrid approach for biomedical literature retrieval. The method was analyzed to ensure that there was no vocabulary mismatch (VM). Word embeddings (WE) and clinical diagnosis information (CDI) combinations were developed to search related biomedical literature. PubMed, WordNet, and Wikipedia datasets were used in this analysis. The experimental results demonstrated that the scheme achieved a significant improvement in vocabulary mismatch rate and accuracy due to the integration method.

In their study, Wang et al. [17] proposed a novel approach to document quality feedback, namely Pseudo Relevance Feedback (PRF) . This technique combines semantic matching and relevance matching to address the inherent semantic gap in traditional retrieval models. In this approach, four datasets from the Text Retrieval Engineering Conference (TREC) are employed. Document and query representations are computed using Bidirectional Encoder Representation Transformers (BERT). The experimental results demonstrated that the proposed scheme exhibited the optimal retrieval performance based on the combined matching method.

Jafarzadeh et al. [18] proposed a semantic technique to predict query performance, establishing three predictors for retrieval under this scheme: query semantic shift, semantic distinctiveness, and semantic cohesion. The ClueWeb12-B, Robust04, and ClueWeb09-B datasets were used for this approach, and the experimental results demonstrated that the scheme achieved superior performance in demand prediction compared to existing methods.

The query expansion technique was developed by Dahir et al. [19] using DBpedia features and topic modeling techniques to predict the latent features in the semantics that were needed when using expansion. The Bose-Einstein statistic distribution method was employed to regroup the applicant's query documents. Related expansion terms were assigned to these documents using the Latent Dirichlet Allocation (LDA) method, with the TREC AP88-90 datasets utilized. The experimental results demonstrated that the scheme enhanced the multi-valued attributes, although it did exhibit some limitations, namely in terms of precision and recall in classification.

In the article, Wang et al. [20] proposed a new multimodal ontology model that integrates low-level image features and high-level text information to represent image content for image retrieval. By incorporating the ontology into the image retrieval system, they are able to realize intelligent image retrieval with high accuracy. They are currently conducting experiments on a dog in the animal domain. They built the ontology based on the low-level features and domain knowledge of dogs, and they have set up a prototype retrieval system to evaluate the performance. Finally, they compare the results with a traditional text-based image retrieval system and demonstrate the advantages of their approach.

The works described above deal with the description of images using various images and do not take into account whether or not they are already described in some way. For this reason, we were led to think about the problem and not to ignore the description already created for the picture. And therefore, since images and videos are often already briefly described in databases or on the web, we are led to create an ontology that provides more detailed information about the object in the image or video. Thanks to it, it is possible to open the door for a more accurate search for information in databases or a more detailed description of the objects in the image in various databases. Combining the ideas described in the articles above with ours could lead to good results, as it is important to use algorithms to describe images, especially in cases where the image is not described by any expression.

## 3. Description of Ontology

The term has previously been used exclusively by philosophers; however, it is now being applied in the field of computer science and knowledge engineering. Ontologies are defined as data models that delineate the concepts and the relationships between them [21, 22]. Individual ontologies may be employed to describe a range of situations and real-world relationships, including those pertaining to family structures, animal distributions, and other relevant phenomena. Large ontologies seek to maximize the number of occurring information and relationships between concepts, aiming to provide comprehensive descriptions and models of real-world situations. Such a system prevents the occurrence of multiple meanings and of terms. A basic representation of the ontology can be observed in Fig. 1, which depicts individuals and their relationships with each other [23].

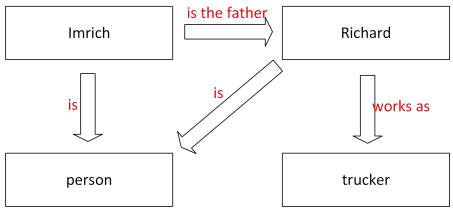


Fig. 1. A simplified outline of the ontology.

A multitude of ontologies are delineated through the use of ontology dictionaries, which elucidate the terminology employed. If an ontology is defined as a set of terms, then the addition of rules defining the constraints (restrictions) applied to these terms serves to clarify the ontology and facilitate its implementation. Ontologies may be represented

in a variety of ways, including as trees, dictionaries, editors, taxonomies, catalogues, thesauri, or axioms. The objective of an ontology is to provide a comprehensive description of a set of concepts. An ontology is designed to facilitate enhanced comprehension and communication between humans and machines, as is shown in Fig. 2. It represents a network of cooperating hierarchies, wherein the information is interconnected by logical relationships. Two concepts that are assigned a property are deemed to be semantically related [24].

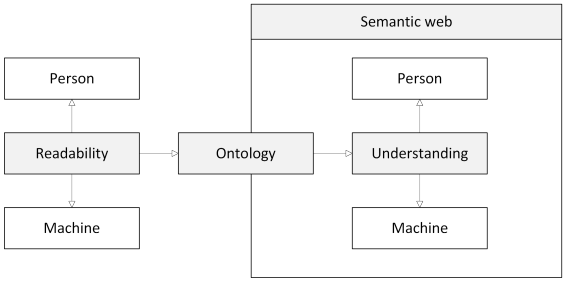


Fig. 2. Human-machine communication via ontologies.

## 4. Design of Proposed Ontology

The complete ontology design adheres to the hierarchical classification of animals as outlined in the encyclopedic reference BIOpedia [1] and is presented in the Slovak language. The ontology comprises a hierarchical classification of animals, comprising two principal sub-arrays, which in turn are divided into tribes. The sub-array designated "unicellular" comprises four tribes, while the sub-array "multicellular" comprises up to ten tribes. These are further subdivided into sub-tribes, which total 22 in the multicellular category and 7 in the unicellular category. In total, the animal kingdom comprises 27 classes, 53 guilds, 42 genera, and 515 species. The properties of each species are currently described by 32 object properties, 2 data properties, and 58 individuals within the ontology. Moreover, each species is linked to an annotation that provides a more detailed description of the animal in question, including information about its life or appearance. The Fig. 3 illustrates the hierarchical structure of the unicellular division.

### *4.1. Class Settings*

Ontologies comprise classes, which serve as the fundamental units of OWL ontologies. The initial, empty ontology contains a single class, designated "Thing." This primary class encompasses the entire set of subsequent classes that are derived from it.

The class Animals, which is an empire in the partition tree, has two subclasses, "Unicellular" and "Multicellular," which are related classes. Each subclass is further subdivided into tribes, classes, orders, and so forth. The hierarchy of classes is also referred to as taxonomy. Once the complete hierarchy of classes for the division of animals was established, the Properties class was created, which is a related class to the Animals class. The Properties class contains a set of individuals that provide further description of each species of animal. Consequently, it has no subclasses, only individuals. The disjointedness of the classes is determined by employing the "Disjoints" function. These are classes that are related to one another but do not share subclasses.

### *4.2. Setting of the Properties*

In the context of OWL ontologies, properties are defined as representing the relationships between classes. Properties can be classified into three principal categories

- object properties- associate a class with another class,
- data type properties - associate the class with a data literal of type integer and

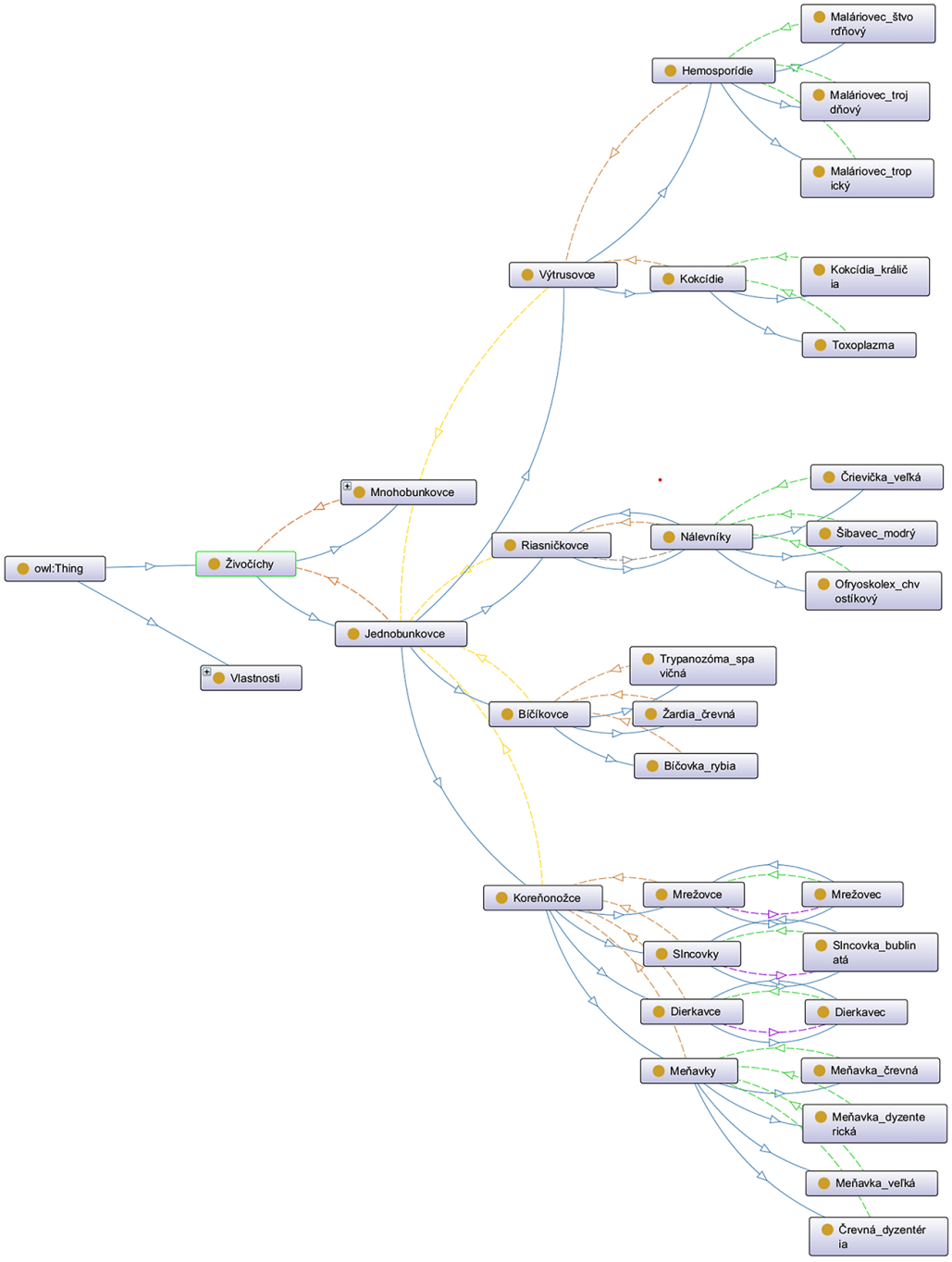


Fig. 3. Hierarchical structure of the Unicellular division.

– annotations - associate the class with a data literal of type string. They are used to add information, metadata (data about data).

The majority of the properties that have been defined and utilized within this ontology are object properties. As previously stated, object properties facilitate the linkage of one class to another. In ontology, the delineation of properties, in addition to their sub-properties, is feasible, thus facilitating the establishment of property hierarchies.

Sub-properties are specialized versions of super-properties, exhibiting a parallel to the class hierarchy. To illustrate, the property "has_mother" is a sub-property of the super-property "has_parent".

In the ontology, we defined 32 object properties and two data properties. Each property has been designated a name. In accordance with the hierarchical structure of "kingdom, sub-kingdom, phylum, sub-phylum, class, subclass, super-order, order, suborder, family, genus, species", we initially established a pair of object properties, "has_sub-kingdom" and its inverse property, "belongs_to_kingdom". These were employed to delineate the scope of the properties.

Subsequently, all 11 pairs of object properties were created in a sequential manner:

- „has_sub-kingdom“ - „belongs_to_kingdom“,
- „has_phylum“ - „belongs_to_sub-kingdom“,
- „has_sub-phylum“ – „belongs_to_phylum“,
- „has_class“ – „belongs_to_sub-phylum“,
- „has_species“ – „belongs_to_genus“.

With the exception of the initial pair, which represents the inverse, the remaining pairs represent ordinary properties due to the lack of strict adherence to the established hierarchy of animal classification. The species "Costia necatrix" is classified within the phylum "Flagellata", the sub-tribe "Unicellular", and the kingdom "Animals" If the established partitioning hierarchy were fully defined, the remaining 10 pairs of traits would also be inverted.

The animals were characterized using the data property "has_count_legs." The methodology was then applied to each animal species, with restrictions and the "Create Restriction" dialog box serving as the cardinality. The data type integer, object, and class from the ontology were assigned using the property "has_count_legs". The same process was repeated for the data property "has_number_of_cells", which was similarly assigned to all species of the sub-tree "Unicell".

### *4.3. Setting of the individual (instance)*

The next step was to construct the individual entities. Protégé facilitates the definition and characterization of individuals through the specification of their attributes pertaining to classes. By employing individuals, we were able to delineate the attributes of the ontology's objects (classes), including their habitat, physical characteristics, dietary preferences, legal status, and other pertinent details. We utilized "HasValue" constraints to achieve this objective. The total of 58 individuals were created.

### *4.4. Setting of the limits (restrictions)*

Once the requisite properties and individuals have been created, they can be employed to describe and define the relevant classes within the ontology of animal classification. Property constraints are identified, as are constraints by the addition of cardinality, and individual constraints.

#### *4.4.1. Restrictions using properties*

It should be recalled that properties are used to describe binary relationships between two classes. The newly created properties were then applied to the aforementioned classes via the "Create Restriction" dialog box.

The subsequent step is to switch to the "Classes" tab. The next step is to select the class that is to be described by a restriction. The "Asserted Conditions" section, located in the central region of the Protégé environment, provides the capability to establish constraints and limitations. To create a restriction, one must first press the "Create Restriction" button, which will open a window with the same name. In this window, one must select a property for the selected class in the "Restricted Property" window, select a restriction in the "Restriction" window, and in the "Filter" window, one must either write or select the class that is associated with the selected class by the currently selected restriction property and restriction.

The same procedure was employed to characterize all classes within the animal partition ontology. With the exception of species and the kingdom "Animals," each class was composed of two restrictions: a restriction property of the type "has_" and "belongs_to_," and a restriction "someValuesFrom".

#### 4.4.2. Limitations by cardinality

In the previous subsection, we delineated the constraints imposed by restriction properties. In this one, we examine how these limitations are qualified by cardinality (see Fig. **??**). The procedure is analogous to that employed for property restrictions. The sole distinction is that we select a data property as the constraint property, utilize the "cardinality" constraint, and assign a number to the chosen class in the "Filter" window, which is of data type integer.

In addition to the "has_number_legs" data property, the "has_number_cells" property was employed to establish constraints for the "Unicellular" sub-tree.

#### 4.4.3. Individual restrictions

The "HasValue" restriction, denoted by ∋, is implemented through the aforementioned process. Let us consider the question of determining the coverage of the animal body in the ontology. To illustrate, we may wish to state that the species "Canis_Lupus" has a body covered with a coat. We have already created individuals in the class "Properties," including the individual "coat." We may use the limitation "HasValue" in conjunction with this individual to specify what the animal has covered its body with. The process of creating limitations using instances is as follows.

In this manner, supplementary properties have been established within the "Properties" tab, which coupled with an appropriate instance, delineate the species within the ontology that has been constructed. Each species can be described by an unlimited number of individual constraints. Individual constraints are metadata that assist in the semantic characterization of the animals within the OWL ontology (see Fig. 4).

Fig. 4. Features and individuals to describe species.

## 5. Design of Proposed GUI

As previously stated in the introduction to this paper, our objective is to evaluate the ontology in a practical setting. To this end, we have developed a system (application) that extracts image data and associated descriptions from animal-oriented web documents, with the aim of enhancing the flood descriptions with more detailed information drawn from the animal ontology. In the following subsections, we will also present the technologies and libraries utilized in the development process, along with a concise overview and implementation details.

### 5.1. Design of the application

Python has been selected as the programming language for the application's development due to its suitability for data manipulation, web extraction, and the availability of a comprehensive library of tools that can be leveraged in its creation. As the objective of the application is to extract image data from online sources that feature animal-related content, including images and videos, it is essential to ensure the secure and effective functioning of several key operational aspects:

1. Create access to the target web document.
2. Correct processing and retrieval of web document content.

3. Analyzing the source code of a web document and extracting useful data.
4. Identification of animals from the extracted data.
5. Refining the description of identified animals using an animal ontology.
6. Presentation of results using the application's graphical user interface.

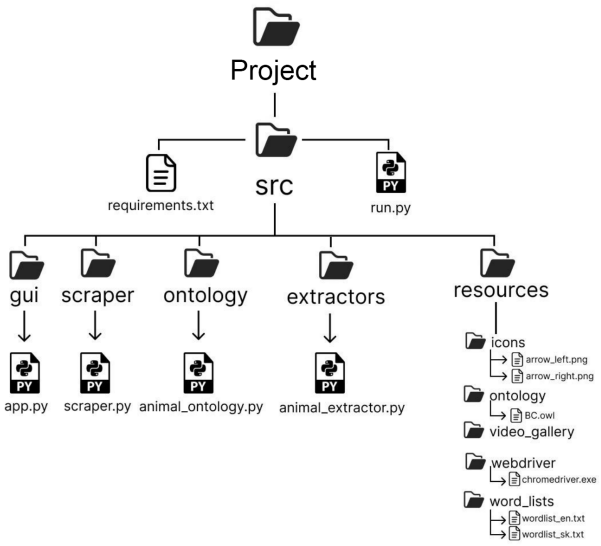


Fig. 5. File structure of the project.

The project was designed into a file structure (see Fig. 5) that begins with a root directory named "Project" and then continues to the "src" directory, where all the application source codes are located. These are divided on the basis of their individual functions. Additionally, two files are present at this same level of the directory. The initial file is a text file, designated "requirements.txt," which serves as a comprehensive list of all the essential Python modules utilized within the project. The subsequent file is the "run.py" file, which serves as the primary startup code for the entire application. In the directory designated "gui", which stands for graphical user interface, there is a file named "app.py". This file contains the source code for all the graphical elements that, when combined, constitute the complete graphical user interface that is an integral part of the application. Additionally, it contains the functions that are essential for the application's proper functioning. The directory named "scraper" contains the code representing the web scraper, which deals with accessing the document, retrieving web content, and extracting useful data from the document for further processing. The final two directories, "ontology" and "extractors", contain the source code for ontology and data analysis integration, which is used to extract animals from text. The "resources" directory contains non-source code files that are essential for the application's functionality, including icons, ontologies, webdriver, and text dictionaries.

Figure 6 illustrates a simplified flowchart of the proposed application. The fundamental premise is that upon execution, an input is obtained, specifically a link to a web document, which is then processed by the "scraper" class in accordance with the diagrammatic representation. Additionally, the output encompasses the transmission of error messages to the GUI in instances where errors occur.

Subsequently, the outputs from the scraper class should be processed in the main application class (app) using the remaining classes for filtering, extracting animals, and refining descriptions from the ontology. Finally, the results should be presented in the app.

In the following subsections, the functions of each class and the libraries used will be described in more detail, along with their implementation.

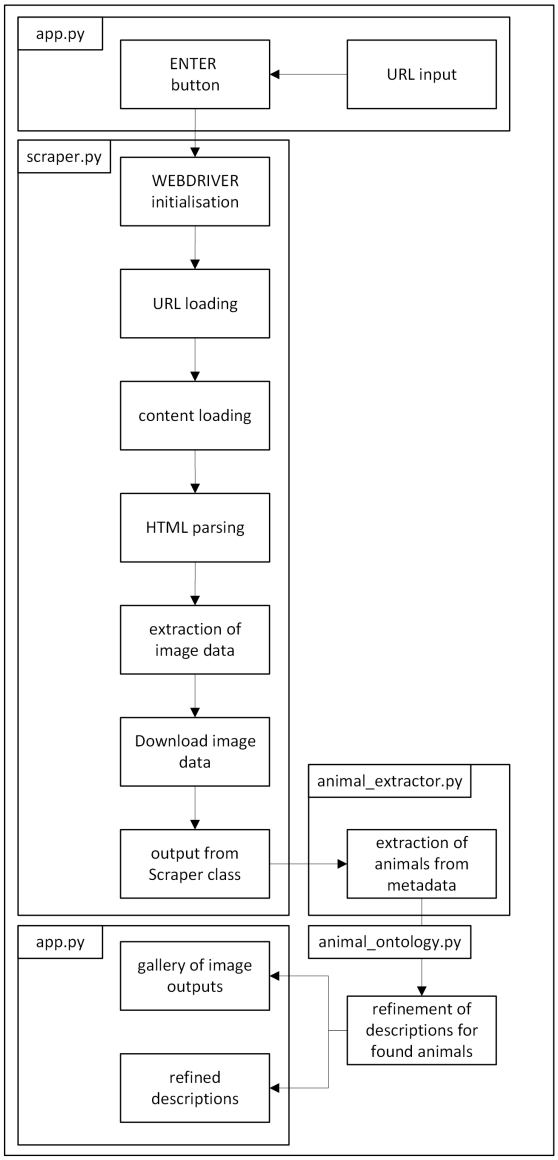


Fig. 6. Flowchart of the proposed application.

### 5.2. Design and implementation of the graphical user interface

A significant aspect of the application is to furnish an adequate portrayal of the extracted image data, necessitating the presentation of the results in a manner that accurately reflects the underlying image information. This entails the utilisation of outputs that encompass both the original image data and the extracted descriptions, which have undergone refinement by the system. The subsequent section will delve into the design of the graphical user interface (GUI) and its accompanying description, along with an exposition of the libraries and tools employed in its construction.

#### 5.2.1. Graphic design

In the design of the graphical interface for the application, the decision was made to employ an image gallery in conjunction with a pair of navigation buttons for the purpose of facilitating the presentation of results from the

mining process. This approach allows users to navigate between outputs with minimal effort. Other indispensable fundamental components of the graphical interface include an input field for a link to the target web document and a button to initiate the application process.

In summary, the graphical interface of the application should contain the following basic controls and display elements (see Fig. 7):

a) Input field for specifying a link to the target web document.
b) "Enter" button to start.
c) Navigation buttons to navigate through the image gallery.
d) Display field for extracted image information.
e) Display field for the created description.

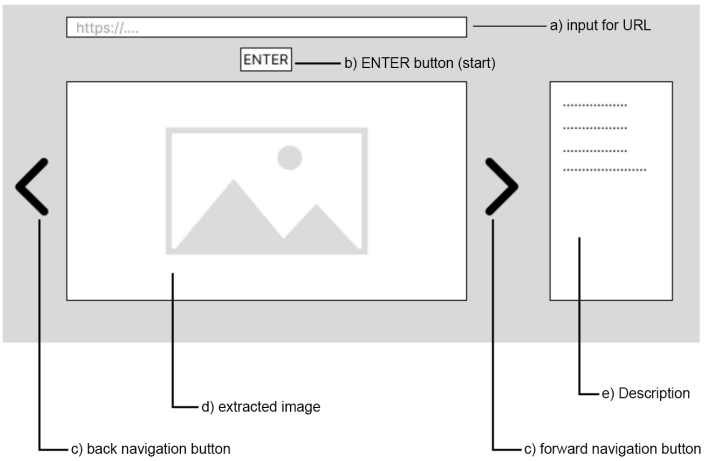


Fig. 7. Design of the graphical user interface of the application.

### *5.2.2. CustomTkinter Library*

The graphical user interface design was implemented using the CustomTkinter library, which is a Python UI (user interface) library based on the original Tkinter library, which is part of the standard Python module package.

The CustomTkinter library provides users with a set of new, modern, and fully customizable graphic elements that facilitate the creation of consistent user interfaces across all desktop platforms, including Windows, macOS, and Linux.

The resulting graphical user interface (GUI) layout incorporates all the fundamental control and display elements initially outlined in the design, augmented by the inclusion of a left panel that facilitates the selection of optimal application settings (see Fig. 8):

a) Option to enable - disable filtering of results.
b) Option to display output metadata.
c) Selection of allowed data formats for outputs.
d) Option to change the colour preference of the application (appearance option).
e) Button to exit the application.

Moreover, a component for displaying the progress of the application has been incorporated (Fig. 8(j)). This component presents the user with a graphical representation of the current status of the application's active processes, utilizing both the graphic element from the library and the text outputs situated beneath it. The latter are located in the central portion of the interface, adjacent to the gallery of image outputs. In the panel, which is located in the right part of the application, as well as according to the original graphic design, there is a place for displaying, whether the acquired descriptions of the image information, as well as a graphic element by means of which we display the

descriptions of the animal specified by us. The display of descriptions obtained from image data is implemented using simple graphic elements such as "CTk.Label" - these are elements into which we insert simple text obtained from image data, namely "Alt description" (g) - Fig. 8, which is the alt attribute of the HTML tag for the image information (<IMG>, <VIDEO>) and then the "Title" (h) - Fig. 8, which is obtained from the "title" attribute, again from the given HTML tag.

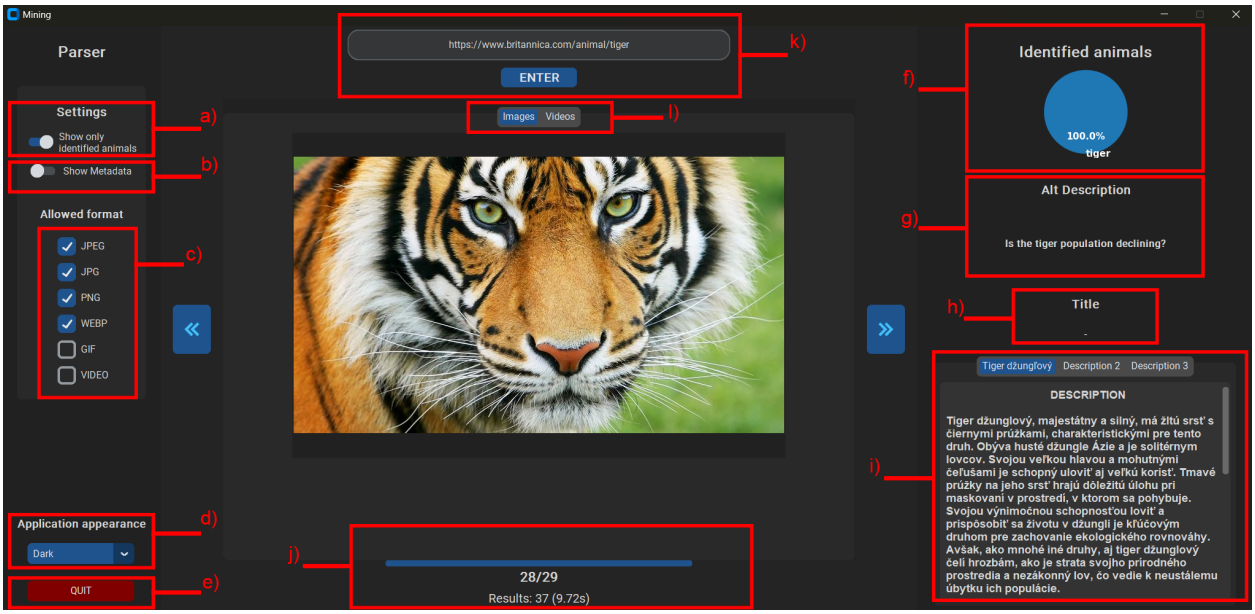


Fig. 8. The resulting graphical user interface created using CustomTkinter in a dark color preference.

To display the refined descriptions added by us (i) - Fig. 8, we chose to use a graphical element available in the CustomTkinter library, named "CTk.Tabview", which is convenient precisely because it allows us to insert multiple tabs that are created when displaying multiple results from the ontology file that contain the refined descriptions of the animals. Another added graphical element includes a pie chart (f) - Fig. 8, which graphically depicts for each output the frequency of occurrence of the animal in the metadata (percentage) on the basis of which the animal is identified, and according to which the selection for displaying the descriptions from the ontology is chosen. The original designs have been retained in their entirety, including the elements for entering the address of the target web document, the "ENTER" button to initiate (k) - Fig. 8, and the navigation buttons in the form of directional arrows to navigate between the image results on the right and on the left. Given that the application is primarily concerned with image data, namely images and videos, it was imperative to devise a mechanism for switching between disparate output formats. The method employed once more makes use of the graphical element, "Tabview", which facilitates the switching of content tabs. Accordingly, in Fig. 8 (l), two tabs, namely "Pictures" and "Videos", have been added, allowing users to switch the current output type view in the gallery between pictures and a video player.

The video player is set up to display the video after it is extracted from the web page. However, since web pages contain dynamic content, it is much more difficult to access the video using a parser, so our card is currently designed to display video embedded with <video>. In case of <video> detection, it considers the formats ’mp4’, ’mkv’, ’mov’, ’avi’ during the analysis. Unfortunately, none of the pages in our test sample contain any such videos, so it is currently not possible to visually present the video extraction outputs for this card in more detail.

### *5.3. Extracting data from web documents*

Given that a significant portion of the contemporary web is predicated on the generation of dynamic content through the utilisation of JavaScript libraries and tools, it is imperative that our application is capable of processing

this type of web and its associated content. However, this is complicated by the use of conventional methods for content retrieval and data extraction, such as a network request employing the "GET" method for the target web document [25]. For this reason, incorporated the Selenium library was selected into the data extraction component of the application, together with the use of a webdriver that enables the complete simulation of client interaction with a web document, even if its content is generated dynamically and not statically from the application environment itself. The application is structured into several classes, each of which fulfills a specific functionality. As previously stated, the GUI code is located in the "app" class in the project directory with the path "src/gui/app.py". The functionality for retrieving a web document and extracting useful data from it is included in the "Scraper" class in the folder with the path "src/scraper/scraper.py" (Fig. 9).

In addition to the primary function of extracting images and videos and their associated metadata from a document, the class also contains a number of auxiliary functions. These include a function for creating an image object from binary data, which involves transforming the extracted image into a format suitable for display in a graphical interface. The class also includes functions for filtering out image data based on format or resolution suitability, as this data will be further processed into output in the graphical interface. This is in accordance with the approach outlined by Berendt [26].

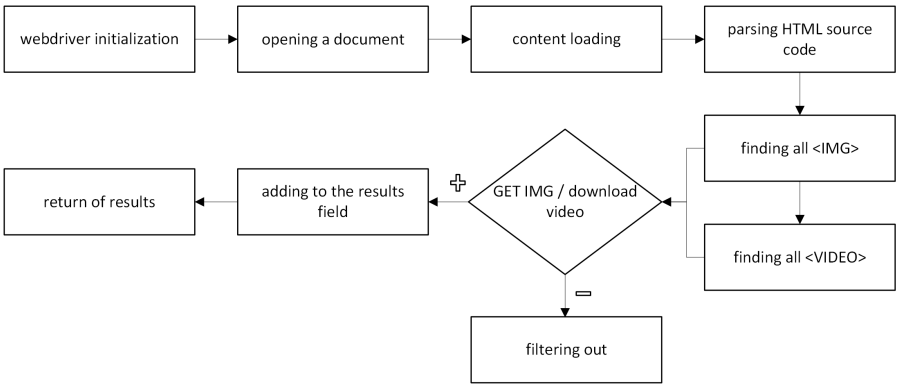


Fig. 9. Diagram of the process of extracting data from a web document of the Scraper class.

From the primary application class, only the "scrape" method is invoked, which has two input parameters. The first is the "url" of the string data type (text string) and the second is an instance of the "app" class for direct access to the application, thereby enabling the progress entry to be made available to the application. This method initiates the entire process, beginning with the utilization of the webdriver to access the target web document and subsequently obtaining the dynamic content of the web page. The subsequent step involves parsing and extracting the data deemed to be of value. The data that we need and focus on in the class after getting the loaded source code of the website is searched using functions from the bs4 library. These functions allow us to easily retrieve all images and videos based on their HTML tags, focusing on "<IMG>" tags for images and "<VIDEO>" tags for videos. The extracted elements are then grouped in arrays, together with their extracted metadata. Each extracted image information from the processed document is part of the created object, which contains all available data such as format, resolution, "alt" and "title" attributes. Such an object is subsequently stored in the result array of this process and will be returned as a result to the application for further processing and display.

### *5.4. Animal identification class*

In the process of designing the app, several significant objectives were established, each of which represents a crucial aspect of the app's development. One such objective is the ability to identify animals from the extracted image data. This class's objective is to leverage the fact that web authors utilize HTML tag attributes for image description, at least to some extent. This information is extracted in textual form in a manner described in the subsection on extracting data from web documents. A class designated "AnimalExtractor" has been developed for

this objective and is situated in the directory with the path "/src/extractors/animal_extractor.py". In addition to the animal identification function, the class also addresses the function for translation from a foreign language, given that our focus is on English web pages in addition to documents written in Slovak. The translation integration represents a crucial aspect of the application, as the results must be presented in Slovak. In addition to the previously mentioned methods for identification and translation, the class also contains a static method, "read_wordlist_from_file" (line 3), which is used to retrieve the dictionary of animals from the project files located in the "WordLists" directory. Two such dictionaries are included in the project. One contains animals in the English language, while the other contains animals in the Slovak language. The Slovak dictionary has been incorporated into the system as a contingency for instances where the content is in Slovak. This eliminates the necessity for utilizing the translator function, which consumes both data and memory resources. To identify the animals, we employ a method designated as "find_animals_in_text", which accepts the variable "txt" as an input parameter. The variable in question represents a text string, which is then parsed in this method using regular expressions (regex) from the Python module "re".

The process of identifying animals from text data begins with a preliminary step involving the modification of the input data using the "sub()" method from the module. This method is employed to return the string resulting from the replacement of the pattern occurrence in the original string with the specified substitutes. In the pattern specified by us (the first input parameter of the method), "[-/_]" is used to indicate the pattern to be searched for in the string. In order to express a set of characters, we utilize square brackets. In our case, the set comprises either a dash "-", a slash "/", or an underscore "_". We replace these with a space, which is the second input parameter of the method. In other words, we create a space instead of the original character. The final input parameter of the method is the string to which the pattern is applied, namely the input text string. The rationale for replacing these specific attributes is that the method employed for parsing the extracted metadata includes the web link for the image information where these attributes are found. This could potentially impede subsequent searches of the data. The subsequent step is to define the search pattern , wherein a pattern (variable "pattern") is established for the purpose of locating the animal that is included in the dictionary. The pattern that has been defined will form an expression that will find a match if the word in question matches any word in the list of animals stored in the variable "self.animals_wordlist". This variable contains the dictionary of animal names. The "s" that follows the variable name is an optional letter that can be included to find the plural form of the animal in the case of the English language. As a last step, a list (array) of hits identified in the data is generated using the method from the "re" module for regular expressions, "findall()". This method employs three input parameters, namely the pattern, text, and an additional optional flag. The "pattern" variable from the previous step is employed as the pattern, the text is the input variable in our "txt" method, and "re.I" is utilized as the flag to indicate that the method will not consider the case differences in the text. The output of this method is an array of objects that contain information about the animal identified in the text, as well as the numerical count of its occurrence. As previously stated, the objective of this course is to demonstrate the utilisation of translation methods for the purpose of displaying the output in Slovak. The selected method for this task is "translate_to_slovak," which accepts a text string intended for translation as its input parameter and generates the translated output. This method employs the translator of the "translate" module. The final method in the AnimalExractor class is "translate_animals_in_list, which employs the aforementioned method but utilizes a list (or array) as its input parameter. This is due to the fact that the animal identification method returns such an array.

### *5.5. Integrating animal ontology*

The principal objective of this study was to provide accurate and appropriate descriptions for the extracted image information pertaining to the animals in question. The descriptions presented in this work are primarily derived from the extracted metadata or auxiliary attributes of HTML (Hypertext Markup Language) tags, such as "alt" or "title," from the web document itself. In such cases, the application relies on the author to tag the image in an appropriate manner, with the aim of obtaining a more accurate description. In case we are able to extract directional information about the name of the animal present in the image from the extracted data, we are able to create a more accurate description with features for a given animal image using a set of animal-oriented ontologies. In this work, we will then utilize the ontology that has been created, which we will process using the available libraries for the Python programming language. The integration of the ontology takes place in the class "AnimalOntology," which is located

Table 1
Queries of the SPARQL query language.

| Clause | Description |
|---|---|
| **PREFIX** | Choosing a namespace |
| **SELECT** | Display format definition (Table) |
| **CONSTRUCT** | Creating new RDF statements |
| **WHERE** | Formulation of queries |
| **SORT BY** | Sorting Results |
| **OFFSET** | Limitation to a given subject number |

in the directory "animal_ontology.py" within the project in "src/ontology/animal_ontology.py". The objective of the class is to provide two crucial elements for the application:

a) Loading the animal ontology file.
b) Methods used to extract data from an ontology file.

The "Initialisation (___init___)" function is a python function that is called automatically when a class instance is created, otherwise known as a constructor. The input parameter of the class constructor is the variable "ontology_file_path" of the string data type, which represents the path to the ontology file that the class and therefore the application will use. Subsequently, the "Graph" instance of the class is initialized from the rdflib module, which is assigned to the "self.graph" variable of the AnimalOntology class in order to easily access the "ontology graph" object from within the environment of this class. In the class constructor, the next step uses the "parse" method of the graph object with the input parameter "ontology_file_path", which represents the path to the ontology file, and with the "format" parameter set to "xml" (Extensible Markup Language). This method reads the animal ontology file specified by us into a format suitable for further processing in the class, in our case in XML format.

As mentioned in the introduction of this subsection, the AnimalOntology class, in addition to loading the ontology file, is also responsible for providing methods that allow us to work with the data that is contained in the ontology, in our case it is text data describing animals.

In total, the class contains five methods, each of which is designed to retrieve a portion of the data needed to display a refined description of the animals. Each of these methods uses the Semantic Web Query Language (SPARQL) data query language, which is part of the RDFLIB module, to query the required data from the ontology file (Table 1). The "get_animal_matches" method is the only method that is called in the main class (app) of the AnimalOntology class and is responsible for obtaining the animal class identifier reference (URI). The input parameter is a variable "animal_name", which represents the animal we are looking for a reference to in the ontology. To search for a match of the desired animal in the ontology, we again use the module "re", i.e. regular expressions. If the animal is not found, the application will not provide a description in the user interface. However, if a reference to the desired animal is found, the remaining methods that query the data for the animal (using the found reference) are called further in this method. In total, the method only searches a maximum of three results (if the animal has multiple species) due to the limitations of displaying multiple results in the graphical interface of the application. The method returns an array of objects containing all the retrieved animal data, which is then used by the main application class to display in the application.

In the case of a successful discovery of an animal in the ontology set, we are able to provide the following textual data for application using the remaining methods:

a) A short textual annotation (commentary) on the animal.
b) Its position in animal taxonomy (animal hierarchy).
c) List of similar animals (placed at the same position in the taxonomy).

These results are then inserted in text form into the text windows of the tabs of the "Tabview" graphic item in the description part of the application.

### 5.6. Parsing the content of a web document

BS4 (beautifulsoup4) is a python library proposed for projects dealing with processing, retrieving and modifying the parse tree of web documents [13]. This library is built on top of popular python parsers like lxml, html5lib and also automatically converts input documents to UNICODE format, and output documents to UTF-8 format, thus relieving the user of the need to handle document encoding. By employing straightforward library commands and methods, it is possible to parse the content of a web document in a single line of code, thus facilitating subsequent manipulation of the processed content within the code itself. This may entail searching, modifying, and other operations.

The first extraction example, uses the "soup" attribute of the object (a reference to an object already parsed by BS4) to retrieve the title of the web document. If the "title" tag is not present in the document, the value "None" will be assigned. In the second example, we use the "find_all" method, which can retrieve an array of all the tags found in the document, in our case we have determined that we are searching for "p" tags, which usually contain text in web documents. The result of this method is returned in the "paragraphs" variable, which is then used in the "for" loop to display the text of each element of our paragraphs array, using the "text" attribute.

## 6. Experimental results

The following subsection will present the results of testing the application on web pages with a focus on animal content. The subsequent evaluation will compare the results with the image information present in these documents. The sample of test web documents (50 in number) was randomly selected from pages searched using animal keywords entered into Google's search engine (Google Search) in both Slovak and English languages. A total of five specific animals were identified, as we have previously created detailed descriptions for these animals within the ontology file. These descriptions are utilized by the application during the identification process, providing additional context to the output. Consequently, the target web pages were searched for the following keywords: "lion", "tiger", "cat", "dog" and "elephant". Each of the aforementioned animals is represented in a total of ten web pages (50 in total). Moreover, results in which the application was unable to access the content of the web page were excluded. In evaluating the comparative results, three types of statistics will be considered:

a) The success rate in the accurate extraction of relevant image data from a web document.
b) Filtering error of non-animal image information.
c) The success rate of correctly identified animals.

The application also records the processing time of the document during the data extraction process, but this information will not be included in the evaluation of the results because the processing time is directly dependent on the speed of the Internet connection currently used by the application, so it may be variable.For each type of statistic, a partial sample table is provided. Ideally, the application extracts the correct number of images and videos from the input web document, filters out the outputs that do not contain animals, and then correctly identifies the name of the animal. When identifying the animal, i.e. determining what animal it is, the application does not focus on species-level identification (if the image shows an Domestic cat, the correct identification result is considered to be "Cat"). Then the application provides us with outputs in the form of text descriptions for all cat species that are created in the ontology (maximum three due to display limitations), from which the user can choose and get more information about the animal or about similar animals (Domestic cat, Wild cat, etc.). In the sample web document with cat (see Fig. 10) there is one highlighted images showing cat, the application should be extract this image and then identify the animal that appear in them and assign appropriate description.

In Fig. 11 you can see the result of mining directly in the application, where the number of extracted results is in the number of one image, so the success rate of correct extraction of image data in this case is 100% as well as the identification of animals, which was correct. Then, once the animal was correctly identified, refined description from the ontology were retrieved and added, containing a short annotation, the location of the animal in the animal taxonomy, basic characteristics, and enumeration of similar animals.

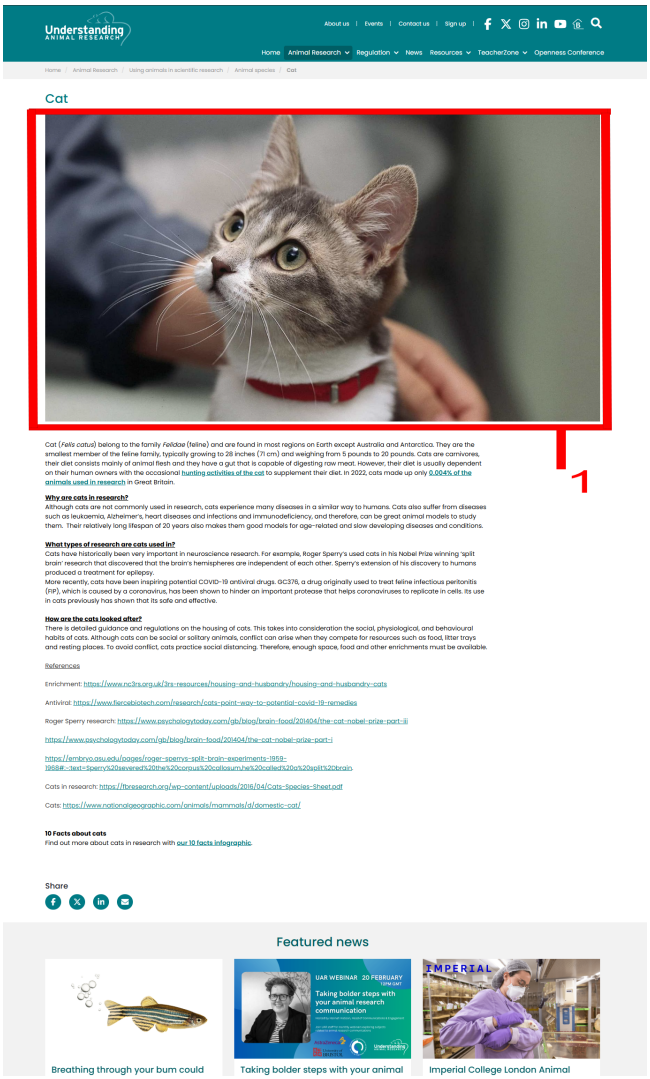


Fig. 10. Sample of web document with image of Cat (https://www.understandinganimalresearch.org.uk/ using-animals-in-scientific-research/animal-research-species/cat).

As can be seen in Fig. 11, the entry page to the application is the English web page from Fig. 10 and the outputs from it are already displayed in Slovak (Fig. 11 highlighted in red). Here we see that it is not necessary to have everything written in English, but it is enough to implement a translator and it will process the texts in the required form. So, even if the ontology is implemented in Slovak, the output from it can be inserted again through the translator so that the output is also displayed in English, or it would be a good idea to add a language option to the application so that the user can choose in which language he wants to display the descriptions.

### *6.1. Image data extraction success rate*

In order to produce statistics on the correct number of image data extraction from the target web document from a prepared sample of fifty animal-focused web pages, we focused on the percentage success rate of extracting images and videos. It is important to note that we are only concerned with the successful extraction of relevant image data, i.e., data that contains animals. In (Table 2) you can see the data collection record from of the first five random web pages from our database (Seq. No.). In the next colomn is number of data on the web (images/video) and in

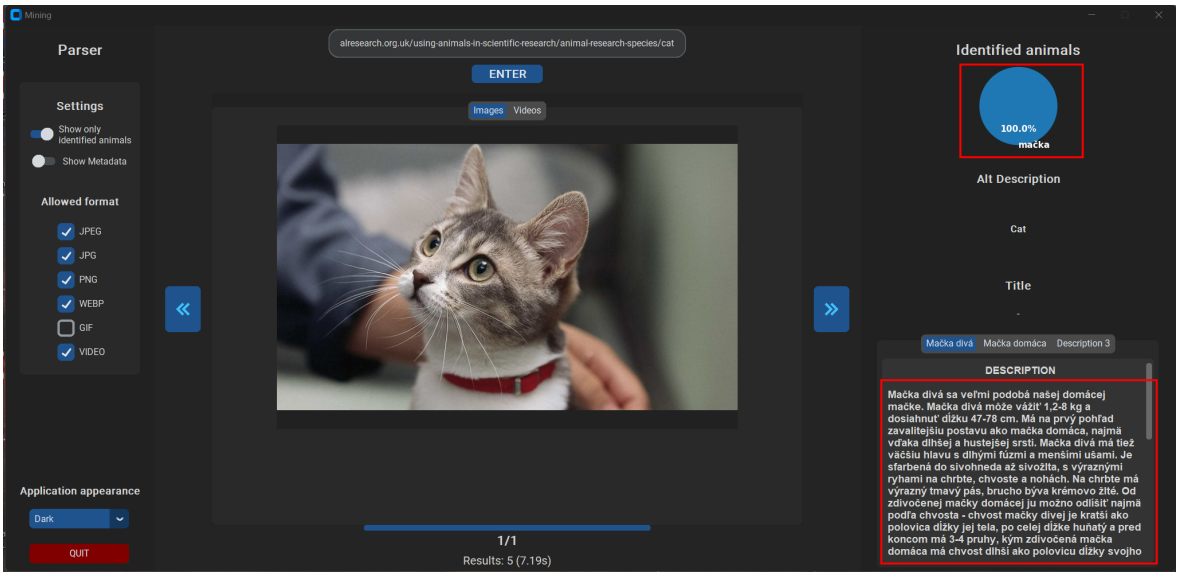


Fig. 11. Application output for the web document (Fig. 10).

Table 2

Data collection record for the number of image data extracted from the site.

| Seq. No. | Number of data on the web (images/videos) | Extracted by application (images/videos) | % |
|---|---|---|---|
| **1** | 6 | 6 | 100.00 |
| **2** | 2 | 2 | 100.00 |
| **3** | 7 | 6 | 85.71 |
| **4** | 13 | 12 | 92.31 |
| **5** | 31 | 31 | 100.00 |

third column are extracted data used our application. In the last column are represents results the percentage success rate extracted images data with detail descritpions obtain from ontologie. For the first row of (Table 2), the number of content image data on a given web document is six, and the application was successful in retrieving these six image data in the document processing phase. That is, for the first web page tested, the success rate of image data extraction was 100 %.

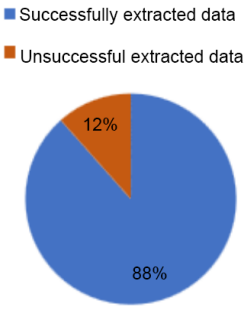


Fig. 12. Graph of image data extraction success on a sample of 50 web documents.

After testing the entire sample of web pages, we concluded that the application was able to extract approximately 88% of the content of the target web page (Fig. 12). These results excluded cases where the web page denied the

Table 3
Data collection record for the number of erroneous filtered image data from the web

| Seq. No. | Data extracted from the web | Error filtering | % |
|---|---|---|---|
| **1** | 6 | 0 | 0.00 |
| **2** | 2 | 1 | 50.00 |
| **3** | 6 | 0 | 0.00 |
| **4** | 12 | 1 | 8.33 |
| **5** | 31 | 0 | 0.00 |

application access to its content, for a total of six web pages out of the entire sample, i.e., twelve percent of the documents (Fig. 13).

Contributing to much of the omitted image data in this step, was the fact that video content is currently provided in documents using a number of advanced dynamic video delivery techniques for which the video extraction portion of the application is not sufficiently prepared.

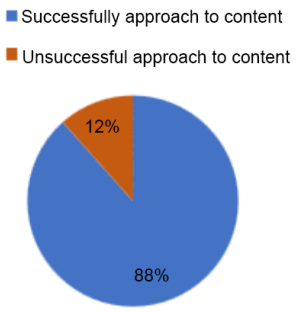


Fig. 13. A graph showing the success rate of access to web content.

### 6.2. *The error rate in filtering results*

The purpose of recording the filtering error rate, is to gain knowledge of how many results are erroneously filtered out in the filtering phase. Erroneous results include cases where image information is filtered out:

a) Containing the animal but has not been identified.
b) Finding an animal in pictorial information in which it is not actually.

The data collection table from the recording of this statistic (Table 3) contains a total of four columns and these are:

1. Seg. No.- The sequence number of the web page that is currently being analyzed.
2. Data extracted from the web- Number of image data extracted from the web (information from the previous table).
3. Error filtering - Number of erroneous filterings from the given extracted data.
4. % - Percentage evaluation of the error rate for a given line.

As a result, after completing the testing of a sample of fifty web documents, the average level of filtering error rate reached almost one-third of the poorly filtered results (Fig. 14). This means that the application lost approximately 28% of the results due to faulty filtering.

The bias was largely due to the fact that the results containing animal were not sufficiently textually tagged and did not contain any descriptions, and since the application uses regular expressions to identify animals when filtering the results, it is not possible to record these animals and not filter out the results that do not contain animals at all.

Table 4
Record of success from animal identification.

| Seq. No. | Filtered Outputs | Correct identification | % |
|---|---|---|---|
| **1** | 6 | 0 | 100.00 |
| **2** | 1 | 1 | 100.00 |
| **3** | 6 | 0 | 0.00 |
| **4** | 1 | 1 | 100.00 |
| **5** | 31 | 29 | 93.55 |

Another cause of errors, as already mentioned in the introduction of this subchapter, was the fact that the animal was identified in the image information where it was not present, and these were mostly cases of images of company logos and promotional graphics of the websites that published the image information.

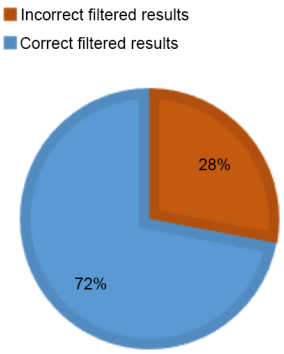


Fig. 14. Graph showing the error rate from filtering the results.

Error rates therefore vary in some cases by many percentages, as it very much depends on the way the site is designed and the extent to which it is appropriately tagged.

### *6.3. The success rate of correct animal identification*

The last step of the application, after extracting the image information and subsequent filtering, is the identification of the animal present in the image information. The aim of the following statistics is to get an overview of the success rate of determining the name of the animal from the extracted image data and descriptions from the previous steps. The table containing the record of identification success (Table 4), contains four columns:

1. Seg. No.- The sequence number of the web page that is currently being analyzed.
2. Filtered Outputs - Number of filtered retrieved outputs.
3. Correct identification- The number of correctly determined name of the animal.
4. %- Percentage evaluation of the success rate for the row.

When processing the data for this category, filtered outputs that misrepresented the animal were excluded, as these outputs were already accounted for when processing the filtering errors in the previous category.

The data obtained after testing the correctness of the animal identification shows that on average the application can identify up to nine out of ten of the obtained results correctly (Fig. 15). Thus, after correct identification, we are able in most cases to correctly enrich the results with the descriptions obtained from the animal ontology.

## 7. Discussion

The main objective of our research was to address the topic of ontologies and ontology design in order to provide a more comprehensive description of image information, whether stored in web databases or directly on web pages.

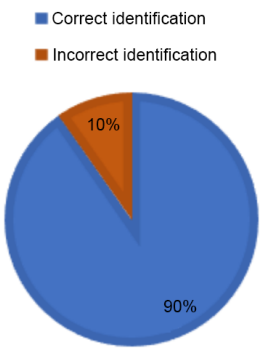


Fig. 15. Graph of success rate from animal identification.

In the initial section, we provided an overview of ontologies, their structural composition, and the methodology involved in ontology creation. We delineated the relationship between ontologies and the Semantic Web. The following section describes the structure and design of ontologies in Protégé. The primary objective of this section was to construct an ontology for the purpose of describing the animals depicted in the image and video. The animal classes were created and their position within the hierarchical tree partitioning of animals was determined. This was implemented according to the web encyclopedia BIOWEB. The division of animals permitted the assignment of a semantic description to each species. To illustrate, the species "African Elephant" is classified within a specific order, class, suborder, phylum, and kingdom. Additionally, an annotation was assigned to the species, providing a concise and explicit description. In the subsequent section, the characteristics of each class were assigned, thereby describing the relationships between them. Subsequently, the individuals that define the species were created. In the final stage of ontology construction, constraints were imposed on the classes, thereby completing the comprehensive relational representation of each ontology class. For illustrative purposes, a graphical representation of the unicellular ontology has been included.

The ontology was implemented in a basic web page browser, programmed in Python, to confirm the accuracy of the individual relations, individuals, and other assignments. A comprehensive transformation of the evolution of the World Wide Web is imperative, manifested in the form of Semantic Web ontologies. With the assistance of ontologies, the World Wide Web is gradually being populated with metadata. The use of ontologies in semantic search engines has resulted in significantly enhanced efficiency. As soon as search engines begin to prioritize web documents (pages) with applied metadata, competition for top search rankings will intensify. Evidence of support for semantic technologies can be observed in Firefox and Internet Explorer.

## 8. Conclusion

The aim of our research was to address the issue of improving the description of image information on the web. We developed a system for mining image data and their descriptions from web documents focused on animals. The application was tested and the results were compared with those obtained from the web. A sample of 50 randomly selected web documents was used and the data extraction success rate was found to be 88%, with 72% of the outputs being correctly filtered. In addition, the animal identification success rate was observed to be up to 90%. It was found that the results depend largely on the appropriate labeling by the document authors. In cases where these descriptions were not found, the application was unable to recognize the content of the image information and thus filtered out this data. On the contrary, in cases where at least partial labeling of the data was provided, the application achieved excellent results in terms of filtering and animal identification. This was despite the use of a relatively simple method, such as processing the text descriptions of this data on the web. In conclusion, the data filtering process turned out to be the weakest link in the whole procedure. To increase its efficiency, it would be advantageous to incorporate more sophisticated filtering techniques, such as machine learning and artificial intelligence, which will be our future

work. Our inspiration is also the publication by Wang et al. [20]. It is possible to supplement their ontology to our ontology and thus enrich it with data. Furthermore, this publication also serves as a confirmation of our knowledge and findings. At the same time, we will work on finding a way to improve the descriptions of image information for better and more efficient searching, whether on the web or in various database systems.

## Acknowledgements

This work was supported by Slovak Research and Development Agency under contract No. APVV-21-0502: BrainWatch: System for automatic detection of intracranial aneurysms and by the project PP-COVID-20-0100: DOLORES. AI: The pandemic guard system.

## References

[1] P. Pancik, Biopedia.sk: Jednobunkovce. (2023). https://biopedia.sk/zivocichy/jednobunkovce.

[2] J.H. Gennari, M.A. Musen, R.W. Fergerson, W.E. Grosso, M. Crubézy, H. Eriksson, N.F. Noy and S.W. Tu, The evolution of Protégé: an environment for knowledge-based systems development, *International Journal of Human-Computer Studies* **58**(1) (2003), 89–123. doi:https://doi.org/10.1016/S1071-5819(02)00127-1. https://www.sciencedirect.com/science/article/pii/S1071581902001271.

[3] J. Faritha Banu, P. Muneeshwari, K. Raja, S. Suresh, T.P. Latchoumi and S. Deepan, Ontology Based Image Retrieval by Utilizing Model Annotations and Content, in: *2022 12th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, 2022, pp. 300–305. doi:10.1109/Confluence52989.2022.9734194.

[4] S.H.A. Ibtihaal M. Hameed and B.M. Mahmmod, Content-based image retrieval: A review of recent trends, *Cogent Engineering* **8**(1) (2021), 1927469. doi:10.1080/23311916.2021.1927469.

[5] E. Hörster, R. Lienhart and M. Slaney, Image retrieval on large-scale image databases, in: *Proceedings of the 6th ACM International Conference on Image and Video Retrieval*, CIVR '07, Association for Computing Machinery, New York, NY, USA, 2007, pp. 17–24–. ISBN 9781595937339. doi:10.1145/1282280.1282283.

[6] R. Bahmanyar and M. Datcu, Measuring the semantic gap based on a communication channel model, in: *2013 IEEE International Conference on Image Processing*, 2013, pp. 4377–4381. doi:10.1109/ICIP.2013.6738902.

[7] H.-Y. Ha, F.C. Fleites and S.-C. Chen, Building multi-model collaboration in detecting multimedia semantic concepts (invited paper), in: *9th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing*, 2013, pp. 205–212.

[8] D. Bratasanu, I. Nedelcu and M. Datcu, Bridging the Semantic Gap for Satellite Image Annotation and Automatic Mapping Applications, *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* **4**(1) (2011), 193–204. doi:10.1109/JSTARS.2010.2081349.

[9] J. Tang, Z.-J. Zha, D. Tao and T.-S. Chua, Semantic-Gap-Oriented Active Learning for Multilabel Image Annotation, *IEEE Transactions on Image Processing* **21**(4) (2012), 2354–2360. doi:10.1109/TIP.2011.2180916.

[10] M. Cristani and R. Cuel, A Survey on Ontology Creation Methodologies, *Int. J. Semantic Web Inf. Syst.* **1** (2005), 49–69. https://api.semanticscholar.org/CorpusID:2175414.

[11] A. Kumar and R. Singh, Web Mining Overview, Techniques, Tools and Applications: A Survey (2017). https://api.semanticscholar.org/CorpusID:212594783.

[12] A. Gasi, M. Dagtekin and T. Ensari, Ontology supported semantic based image retrieval, *Semantic Web* (2024).

[13] D. Saraswat, Ontology Based Agriculture Data Mining using IWO and RNN, in: *2023 6th International Conference on Information Systems and Computer Networks (ISCON)*, 2023, pp. 1–8. doi:10.1109/ISCON57294.2023.10112187.

[14] T. Zengeya and J. Vincent Fonou-Dombeu, A Review of State of the Art Deep Learning Models for Ontology Construction, *IEEE Access* **12** (2024), 82354–82383. doi:10.1109/ACCESS.2024.3406426.

[15] E.F. Aminu, I.O. Oyefolahan, M.B. Abdullahi and M.T. Salaudeen, A Review on Ontology Development Methodologies for Developing Ontological Knowledge Representation Systems for various Domains, *International Journal of Information Engineering and Electronic Business* (2020). https://api.semanticscholar.org/CorpusID:216651445.

[16] S. Malik, U. Shoaib, S.A.C. Bukhari, H. El Sayed and M.A. Khan, A hybrid query expansion framework for the optimal retrieval of the biomedical literature, *Smart Health* **23** (2022), 100247.

[17] J. Wang, M. Pan, T. He, X. Huang, X. Wang and X. Tu, A pseudo-relevance feedback framework combining relevance matching and semantic matching for information retrieval, *Information Processing & Management* **57**(6) (2020), 102342.

[18] P. Jafarzadeh and F. Ensan, A semantic approach to post-retrieval query performance prediction, *Information Processing & Management* **59**(1) (2022), 102746.

[19] S. Dahir and A. El Qadi, A query expansion method based on topic modeling and DBpedia features, *International Journal of Information Management Data Insights* **1**(2) (2021), 100043.

[20] H. Wang, S. Liu and L.-T. Chia, Image retrieval with a multi-modality ontology, *Multimedia Syst.* **13**(5–6) (2008), 379–390–. doi:10.1007/s00530-007-0099-4.

[21] J. Wang, W. Zuo, F. He and Y. Wang, A New Formal Description of Ontology Definition and Ontology Algebra, in: *2009 Second International Symposium on Knowledge Acquisition and Modeling*, Vol. 1, 2009, pp. 363–366. doi:10.1109/KAM.2009.23.

[22] K. Kaneiwa and R. Mizoguchi, Formal Ontology and Extensions of Order-Sorted Logic, *Transactions of the Japanese Society for Artificial Intelligence* **20** (2005), 387–395. doi:10.1527/tjsai.20.387.

[23] R. Iqbal, M.A.A. Murad, A. Mustapha and N.M. Sharef, An Analysis of Ontology Engineering Methodologies: A Literature Review, *Research Journal of Applied Sciences, Engineering and Technology* **6** (2013), 2993–3000. https://api.semanticscholar.org/CorpusID:16954591.

[24] Z.Z. Zulkipli, R. Maskat and N.H.I. Teo, A systematic literature review of automatic ontology construction, *Indonesian Journal of Electrical Engineering and Computer Science* (2022). https://api.semanticscholar.org/CorpusID:252450072.

[25] M. LE, N. Dinh and T. Van, Developing a model semantic-based image retrieval by combining KD-Tree structure with ontology, *Expert Systems* (2023). doi:10.1111/exsy.13396.

[26] B. Berendt, A. Hotho, D. Mladenić, M. van Someren, M. Spiliopoulou and G. Stumme, A Roadmap for Web Mining: From Web to Semantic Web, in: *European Web Mining Forum*, 2003. https://api.semanticscholar.org/CorpusID:264279217.

[27] S. Haridy, R.M. Ismail, N.L. Badr and M. Hashem, An Ontology Development Methodology Based on Ontology-Driven Conceptual Modeling and Natural Language Processing: Tourism Case Study, *Big Data Cogn. Comput.* **7** (2023), 101. https://api.semanticscholar.org/CorpusID:258855924.

[28] J. Niu and R.R. Issa, Developing taxonomy for the domain ontology of construction contractual semantics: A case study on the AIA A201 document, *Adv. Eng. Informatics* **29** (2015), 472–482. https://api.semanticscholar.org/CorpusID:333067.

[29] I. Urbieta, M. Nieto, M. García and O. Otaegui, Design and Implementation of an Ontology for Semantic Labeling and Testing: Automotive Global Ontology (AGO), *Applied Sciences* **11**(17) (2021). doi:10.3390/app11177782. https://www.mdpi.com/2076-3417/11/17/7782.