Transactions on Machine Learning
and Data Mining
Vol. 11, No 2 (2018) 43-62
© ISSN: 1865-6781 (Journal),
ISBN: 978-3-942952-65-1
ONLINE-ISSN: 2509-9337

ibai Publishing
www.ibai-publishing.org

An Automatic Intelligent System for Document Processing and Fruition

Stefano Ferilli[1]

Department of Computer Science – University of Bari
Via E. Orabona, 4, 70125 Bari (BA), Italy
stefano.ferilli@uniba.it

Abstract. With the increasing number of documents available on-line, the need for intelligent digital libraries, that allow to automatize the document processing tasks and to suitably organize and make available the documents so as to provide personalized and focused access, becomes more and more pressing. This paper proposes an integrated system that merges intelligent modules covering all the phases involved in a document lifecycle, from acquisition, to processing, to information extraction, to personalized fruition for final users. The role and possible cooperation of Machine Learning and Data Mining techniques in the system is highlighted and discussed, along with their importance to provide effective support to both the building and the fruition of the Digital Library and the underlying knowledge base.

Keywords: Document Processing; Digital Libraries; Natural Language Processing; Information Extraction; Information Retrieval; Knowledge graphs

1 Introduction

The number of documents available on-line, and/or handled by companies, organizations and single users, is ever-growing, which leads to the well-known problem of *information overloading* (finding useful information items in the overwhelming amount of available information is nearly impossible using traditional means). Hence, the pressing need for automatic techniques that support various kind of users in managing, organizing and exploiting the documents and their information content.

Providing solutions to these problems involves several fields of Computer Science: e.g., Pattern Recognition is needed to process incoming documents in digital or digitized formats, and Digital Libraries are needed to suitably organize them. In this context, a fundamental area to obtain flexible, adaptive and personalized systems is Arti-

ficial Intelligence, and specifically its Machine Learning (ML) and Data Mining (DM) sub-fields, by which the systems may continuously improve the quality and effectiveness of their behavior and outputs to reach maximum user satisfaction.

While many general- or special-purpose DL and DM techniques have been developed in the literature and used for dealing with specific tasks, it may be interesting to investigate how they can be brought to cooperation in order to leverage their peculiarities to tackle several facets of more complex tasks. This paper proposes an integrated system, that includes several subsystems suitably connected so as to obtain an intelligent digital library: DoMInUS, for document processing and organization; ConNeKTion, for building a conceptual graph using the information provided by the processed documents, respectively; and GraphBRAIN, for knowledge base design, handling and fruition. The resulting overall system provides advanced management and consultation of both the documents themselves, and, which is a characterizing and novel feature, the knowledge contained in the documents themselves. It can automatize the document processing tasks and suitably organize and make available the documents, and the information they contain, so as to provide personalized and focused access. It represents a basic, general-purpose platform, to be tailored and extended in order to satisfy specific needs related to the particular context and environment in which it is to be used. While DoMInUS and ConNeKTion were already presented in the literature, GraphBRAIN was recently developed, specifically aimed at intelligent knowledge management, and integrated in the system.

The proposed system pervasively exploits ML and DM techniques, especially knowledge-based ones, to cover all the steps and functionalities involved in a document lifecycle (acquisition, processing, organization, information and knowledge extraction, personalized fruition by final users), for many possible uses (consultation, administration, research). The ML/DM functionality is provided by a module called the *Learning & Mining Server* (LMS), consisting of a suite of tools that come into play in different moments and with different objectives to support several tasks and modules of the other sub-systems. In particular, it includes tools from Inductive Logic Programming (ILP), the branch of Machine Learning based on (First-Order) Logic (FOL for short) formalisms and techniques. In addition to the tools, it also handles the *Learning & Mining Repository* (LMR), in charge of storing all the models learned by the ML/DM modules in the LMS, to be used for subsequent providing of services to the other systems. Two proprietary tools included in the LMS are:

- InTheLEx (INcremental THEory Learner from Examples) [1], an ILP system for supervised learning of classification rules. It may learn simultaneously multiple, inter-related concepts. It is a multistrategy learning system, that supports pure induction with deduction (to recognize concepts that are implicitly present in the descriptions), abstraction (to remove useless details from the descriptions), abduction (to guess needed but unknown information), and argumentation (to resolve cases of conflicting information). It may handle numeric, taxonomic, and sequential information in the descriptions.
- WoMan (WOrkflow MANager) [2], a process mining and management system. It defined a declarative language [3] and a novel mining approach

that proved able to learn very complex process models, also in application domains characterized by significant variability of behavior (such as learning people's routines or movements in an environment). In addition to the mining functionality, it also provides tools for analyzing the learned models, for supervision and conformance checking of new process executions, and for prediction of next activities and of the process under enactment.

A very important peculiarity of both InTheLEx and WoMan is their being fully and inherently incremental, meaning that they may start learning from scratch and progressively refine their models as long as new examples become available, without the need for re-training. This allows WoMan to exploit InTheLEx to learn pre- and post-conditions for the process model components.

Rather than discussing technical details of the single tools in the LMS, this paper focuses on the description of how they were brought to cooperation in order to support the task of automatic document processing and fruition. So, its contribution is two-fold: on one hand, presenting the integrated system, along with its features and functionality; on the other hand, describing how the various sub-systems were adapted for integration in the overall system, and how they rely on the ML/DM functionality provided by the LMS. Each of the next three sections will present one of the sub-systems, and how it is supported by the LMS. Then, the last section will provide some discussion on the overall system and conclusions about the work.

2 Digital Library Management: DoMInUS

DoMInUS (DOcument Management INtelligent Universal System) [4] is the system used as the intelligent digital library component, in charge of document management and organization in the proposed overall system.

2.1 System Description

DoMInUS can handle a set of digital libraries that can cooperate and/or be delivered in a unique, consistent and co-ordinated fashion.

Different kinds of user roles are available in DoMInUS. Each implies a set of prerogatives regarding document processing, management and access:

- Administrators control the general system and have the power to accept, include, suspend or expel digital libraries.
- Users that are in charge of managing a single library are called librarians. They must be enabled to play this role by an administrator or by another librarian of the same library, and have the power to accept, include, suspend or expel documents and users of the library. Each librarian can also enable other users to act as authors and/or maintainers of their library, and plays himself such roles.
- Authors can submit documents to a library.
- Maintainers are technicians that supervise the document flow inside the library, check that the automatic processing is correctly carried out, and, in

case it is, validate the intermediate or final results, or, in case it is not, make the proper corrections and then restart it from the last safe accomplishment.

- End-users, the basic role, has just consultation purposes. A query panel is available to end-users, in which they can combine different search techniques, mix search results, get help and suggestions for their search, bookmark and organize the documents of interest. Any newly registered user starts as an end-user, having access to public libraries only; private libraries may provide them access by enabling specific rights.

A Graphical User Interface (GUI) is provided for each role, that allows the users to comfortably manage the single documents, the overall collection(s) and all the processing steps. A single user can play several roles in several digital libraries handled by the system, and the same role in a given library can be played by several users. All kinds of users are required to register to the system in order to exploit its functionality because user traceability is deemed as fundamental to ensure a correct and controlled workflow.

The overall architecture of DoMInUS involves several components, possibly distributed on different computers. The main components are:

- the *Document Processing Engine* (DPE), in charge of carrying out the various tasks that are needed to extract useful information from an input document and to make it available to final users for search and consultation purposes;
- the *Dominus Data Repository* (DDR), a mix of (relational and NOSQL) databases and files (suitably grouped into folders) in charge of storing all the documents, the content and information extracted from them, and the associated meta-data.

Plus the LMS, used to support several tasks and modules of the DPE (e.g., document classification, component labeling, layout correction, block aggregation, etc.) and to provide subsequent user support and document fruition facilities.

A typical document lifecycle in DoMInUS starts with the document submission by an author. It currently accepts TXT, PS/PDF and raster formats, that are widespread standards for document interchange; extension to word processing formats, such as ODT, DOC, and DOCX, is planned for the future. The submitted document is saved in the DDR, and the DPE is activated to process it, by extracting its layout structure, assigning the document to one of the classes of interest to the library, and, based on this, tagging the document components according to the semantic role they play in that class. Then, the content of the most significant components only is extracted (which should improve the performance and quality of information extraction), in order to further process it. In doing this, the DPE stores all the results of the intermediate processing steps in the DDR. Subsequently, the document indexing and information extraction phases are started. Interesting objects are identified in images, while several kinds of Natural Language Processing techniques are applied to text (text categorization, keyword extraction, lexical and semantic indexing). All these steps will be described in more detail in the following paragraphs.

As a first thing, the input document undergoes a normalization process, that returns an XML internal representation thereof, independent of the original source format in which it was provided. The internal representation initially describes the document as

a set of pages made up of basic blocks directly extracted from the document source. Further information and meta-data will be added to this representation as long as the document processing steps are carried out by the system.

After obtaining the document's basic representation, the layout analysis step can be started. Based on several image processing and pattern analysis techniques, it is in charge of producing the frame-level representation of the document pages. For this purpose, a pre-processing phase may be required to aggregate the blocks in the basic document representation, often corresponding to single letters or fragments of words, into composite blocks corresponding to whole words, and then for further aggregation of words into lines. Finally, semantically related blocks are grouped into frames, each corresponding to a homogeneous and significant logical component in the document: first the Manhattan-shaped components are extracted; then, non-Manhattan sub-parts (if any) are searched in each frame obtained this way. An introduction to the pre-processing techniques used in these layout analysis step can be found in [5]. If different frames are merged, or single frames are split, in the automatically recognized layout structure, the maintainer may manually fix the problems by directly pointing and adding or removing background pieces. Given the final layout, the (textual or graphical) information in each frame is automatically extracted, stored in the DDR and added to the document internal representation.

The next step is Document Image Classification and Understanding, aimed at determining which frames are significant and should be further processed. New incoming documents are first classified according to their first-page layout, to determine what kind of processing they should undergo next. This step is crucial in digital libraries, where many different layout structures for the documents, either belonging to different classes or even to the same class, can be encountered. Then, each frame identified therein is associated to a tag expressing its role. In case of failure or wrong classification, the maintainer can point out the correct outcome, which causes the models to be suitably updated. So, after this step the final internal representation of the document contains the original document information, enriched with layout aggregations, class information and components content (text or images) and annotation.

The extracted information is then used for document categorization and indexing, based on a combination several and complementary techniques, and aimed at supporting effective content-based information retrieval by the end users. For this step, DoMInUS focuses on a subset of logical components deemed as particularly significant. Image processing techniques are applied to graphical components in order to identify known or relevant objects that can contribute to the understanding of the document content. Textual components are 'read', using a third-party OCR engine in the case of digitized documents, or extracting the text in the case of born-digital sources, and the extracted text is exploited for filing a record of the document in the DDR. A full set of NLP techniques is also applied to text frames, including both standard (tokenization, language recognition, stopword removal, PoS tagging, stemming) and advanced ones (syntactic and semantic analysis): while the former are often sufficient for document categorization and indexing purposes, the latter may support more precise and focused information extraction. An introduction to the NLP techniques used in these steps can be found in [5].

Specifically, document categorization is carried out from two different perspectives: a structured one, that associates the document to categories in pre-defined taxonomies, and an unstructured one, that extracts free keywords that are present in the text. Keyword Extraction is included both for extracting key terms to be used as metadata for the document, and to perform keyword-based information retrieval. Finally, Knowledge Extraction is carried out by exploiting ConNeKTion (see next section).

As to Information Indexing, several indexes may be computed and stored, differing for the indexing technique, and/or for the indexing parameters, and/or for the indexed subset of documents. DoMInUS includes both classical term-based and advanced concept-based indexing techniques, including a Vector Space Model based on TF-IDF [6] and a vector space based on Latent Semantic Indexing (LSI) [7] to capture hidden relationships among terms and documents. A module to suggest refined queries according to user relevance feedback is also supported, suggesting a different set of terms that turned out to be more characteristic of the documents that he singled out as interesting.

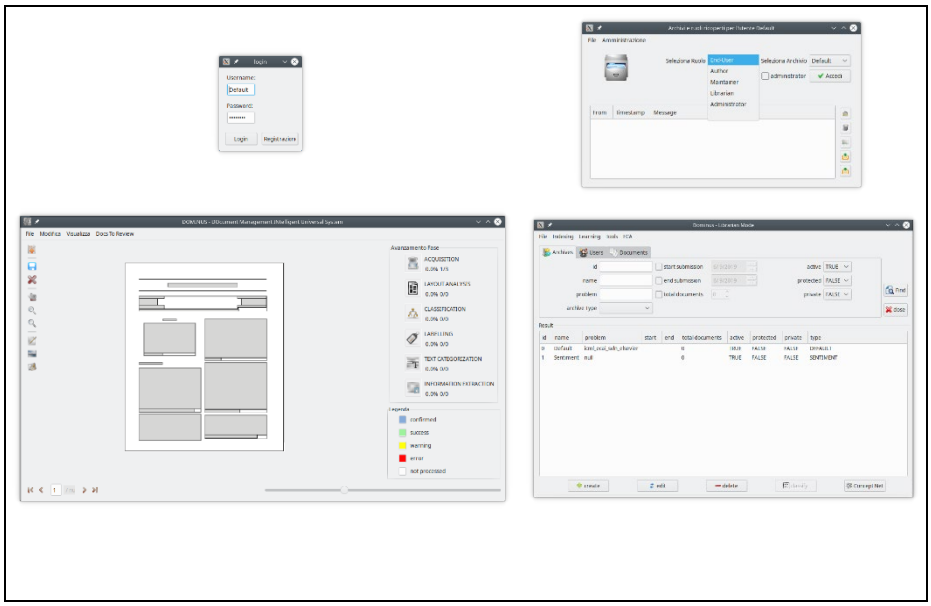

Fig. 1. DoMInUS Graphical User Interface

The user interface of DoMInUS is shown in Figure 1. After logging in (top-left screenshot), the user must choose the archive to work on and the role to act (top-right screenshot). The Maintainer's interface (bottom-left screenshot) allows him to upload a document and to start the layout analysis phase. The panel on the right-hand-side allows him to adjust the parameters for the various steps of automatic processing, to check the step at which the submitted document currently is, and to know whether the steps already carried out were completely successful or reported any warning or error.

In the screenshot, a scientific paper reached the 'Layout Analysis' step. Using the tools available on the left-hand-side and in the menus in the menu bar, the Maintainer may fix the errors (if any), starting in this way the automatic incremental learning of corrections for subsequent automatic exploitation, and let the document proceed towards next processing phases. The Librarian's interface (bottom-right screenshot) allows him to adjust the indexing and information extraction parameters, and to check the results of these steps.

2.2 Use of ML and DM techniques

Most of the steps in the previous sub-section exploit the tools in the LMS, which in turn exploit the models in the DMR. So, the LMS plays a fundamental role in the overall document processing architecture. It consists of a number of sub-modules, and it is exploited in two modes:

- The *learning* mode is used to continuously adapt the domain knowledge whenever new experimental evidence reveals inadequacies of the learned models or changes in the context. Since, in typical digital libraries, new documents continuously become available over time and are to be integrated in the collection, a predominant role is played by incremental learning techniques. The learning tasks are carried out off-line, so that the system can be in continuous operation, using the old models, in the meantime. Only after termination of the learning task the old models are replaced by the new ones, transparently to the users.
- The *classification* mode is exploited to apply the learned models to subsequent documents in order to automatically process them.

More specifically, the following functionality in DoMInUS is supported by Machine Learning and Data Mining:

- Recognition of the type of content (text, image, line, graphics, mixed) in the images of document blocks: using Decision Trees based on features such as density of black and white pixels, area, width and height of the block, etc. The block classification model was learned as specified in [8], using the benchmark dataset of 5473 manually labeled blocks from 54 distinct documents available in the UCI Machine Learning repository, reaching an average accuracy above 97%.
- Aggregation of basic blocks into lines: cast as a Multiple-Instance Learning problem and solved by applying the Iterated-Discrim algorithm [9]. The target concept is "the two blocks can be merged"; an example is a set of instances, each describing a reference block and its top-left, top, top-right, right, bottom-right, bottom, bottom-left and left Close Neighbor blocks. It is positive if at least one neighbor block should be merged to the reference block [5]. This task is especially important for multi-column documents, even more when the content blocks are not aligned in a grid, as in newspapers.
- Frame recognition correction: using ILP approaches, and specifically Markov Logic Networks. 786 manual layout corrections applied by the maintainer were used as examples: 523 examples of merging two pieces of

content erroneously split by the system, and 263 examples of splitting a block obtained by the system that erroneously merged two separate pieces of content. Each example describes the pieces of content involved by the correction and their neighboring blocks, along with their size and position in the document and the spatial relationships among them, both before and after the manual correction. Accuracy, as measured by the area under the ROC curve, is 96% for splitting operations and 99.2% for merging operations [10].

- Document clustering: based on ILP approaches, and specifically on a distance measure for FOL descriptions. Documents of unknown class are automatically grouped based on a relational description of their layout structure, expressing the features and spatial/topological relationships of the content blocks extracted from layout analysis, in order to obtain new classes. Experiments on scientific papers reported average precision, recall and purity all above 91% for this task [11].
- Document image classification and understanding, using ILP approaches, and specifically InTheLEx. Differently from clustering, this is a supervised learning task, where observations (layout of sample documents) are manually annotated by the maintainers to become examples for the learning system. When a maintainer identifies a new, previously unknown classes, InTheLEx can even autonomously extend the set of classes in the model (which most learning systems in the literature cannot do). The description of observations/examples is the same as for clustering, because the two tasks are aimed at the same result. Experiments on scientific papers resulted in 98% average accuracy for classification, and 95% average accuracy for understanding [11]. Note that the understanding task is more complex than the classification task, because, while rules defining classes are usually independent of each other, rules to identify logical components might be interrelated (e.g., the 'authors' frame might be defined as the frame placed just under the 'title' and above the 'abstract').
- Document categorization: both a supervised (using a mix of a similarity-based – Rocchio – and a probabilistic – Naive Bayes – technique, as in [12]) and an unsupervised (clustering-based) approach are exploited for a better organization of documents and their subject-based retrieval.
- Keyword Extraction. Two approaches are currently implemented [13], both taking into account the documents' logical structure to weigh differently the terms (e.g., the words in the title are weighted more than the words in the abstract, which in turn are weighted more than the words in the running text). The former approach is based on term occurrence statistics, and it is a global one (it works on the whole collection). The latter approach is a conceptual one based on WordNet technologies [14], and it is a local one (it works on single documents).
- Modeling of user interaction: using (declarative) Process Mining approaches, and specifically the WoMan system. The activities of the user during its interaction with the system are recorded in logs, that are used as examples to learn process models. Both general process models of the system's usage are learned, and specific models for single users and roles. Analysis of the model performances when used to supervise new interactions clearly shows a quick

convergence towards a standard behavioral model, whose final accuracy is above 97% for all models. Using the prediction functionality of WoMan, the system may also predict the kinds of users who are operating and their next activities, in order to suitably arrange the interface for them accordingly. The quality of predictions, based on how often the system makes a prediction, and on how often the correct activity is predicted, and on how confident the system is in the prediction (see [15]), is above 88% for all tasks.

Some of these tools are also exploited by other sub-modules of the integrated system.

It should be noted that the tasks of document summarization and novel document identification, also facing the issues of starting with only a few examples and learning the real classes when new samples arrive, were faced also in [16], where the application is left open it can be document from a store or a library or even images [17]. In both work conceptual clustering is considered for learning the document structure over the document data base, and specifically Fuzzy conceptual clustering as described in [18], since there is overlap in the documents. While the task and perspective is quite similar as ours, these works propose a case-based approach, while we aim at learning an explicit formal (and human-readable) model.

3 Conceptual Graph Learning: ConNeKTion

ConNeKTion (acronym for 'CONcept NEtwork for Knowledge representaTION') [19] is the sub-system aimed at learning conceptual graphs from text. It also provides functionality for consultation and exploitation of the learned graph.

Approaches to build taxonomies or ontologies by text mining can be classified depending on: (a) their using or not external resources (often, existing taxonomies) to fill the gap between the purely syntactic level and the semantic one; and (b) their needing or not human interaction as a support to the automatic processing step. As in a few other works in the literature, and in spite of some researchers claiming that fully automatic ontology acquisition from text is unrealistic (in their opinion, the system should be seen as a support for knowledge engineers [20]), ConNeKTion can work based only on what is expressed in the text, i.e., without using external resources and without human intervention. Compared to other works in the literature, ConNeKTion tries to build a conceptual graph relying on the whole set of concepts and relationships, rather than only on shared attributes as in a taxonomic representation of concepts, and does not restrict the nature/kind of relationships to a predefined set, since new relationships are created as soon as they are found in the text.

3.1 System Description

Working on (a collection of) plain text(s), ConNeKTion builds a knowledge graph that represents the contents of the collection, also identifying the concepts and relationships underlying it. In particular, it aims at: extracting the concepts expressed in the input text(s) and assessing their relevance; obtaining formal and human-readable

descriptions of the concepts underlying the terms; generalizing concepts in order to enrich and structure the knowledge base in a taxonomy or ontology.

To carry out its task, ConNeKTion brings to cooperation a mix of existing and novel tools and techniques. It adopts both propositional and relational concept descriptions, that allow to handle different levels of complexity and expressiveness in concept representation. While the implemented prototype works on English, the methodologies embedded in ConNeKTion are completely general and applicable to any language.

To build the knowledge graph, ConNeKTion requires a pre-processing of the input texts, based on a sequence of standard Natural Language Processing (NLP) tasks, the most relevant of which are:

1. *Anaphora Resolution*, to replace pronouns by the explicit nouns they stand for, so as to be able to associate to specific concepts the relationships applied to pronouns, in order to further enrich the conceptual graph.
2. *Syntactic Analysis*, to obtain the parse tree of each sentence along with the graph of the involved grammatical relations.
3. *Normalization*, to turn all words in the input text into their lemma (compared to stemming, this allows to distinguish their grammatical role and is more comfortable to read by humans).

In the integrated system, tasks 2 and 3 in the list above are carried out by DoMINUS when investing the documents, and reused by ConNeKTion.

In processing a (collection of) text(s) in natural language, ConNeKTion considers common nouns as concepts, proper nouns as instances, and verbs as relationships. When building the graph, nouns become nodes, and verbs become arcs (if representing relationships, their direction denoting the role of the source and sink nodes in the

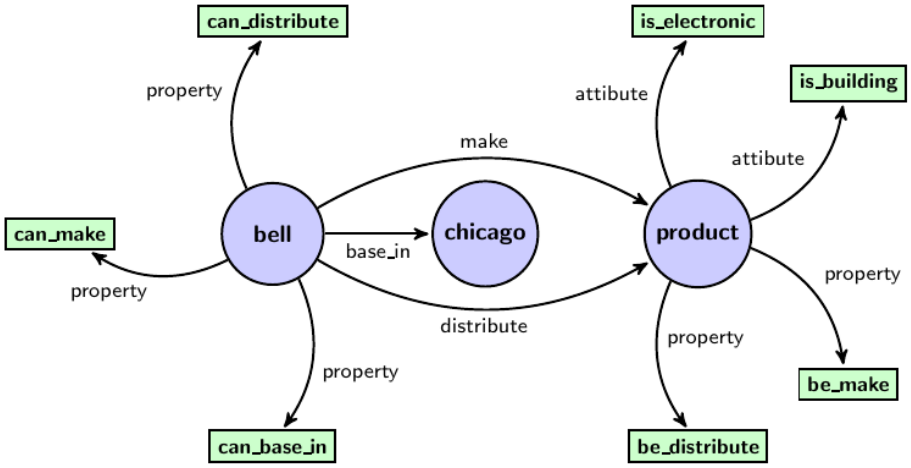

Fig. 2. Graphical representation of the output

relationship) or attributes (if representing properties). Since many words are polysemous, the current prototype uses the one domain per discourse assumption [21]

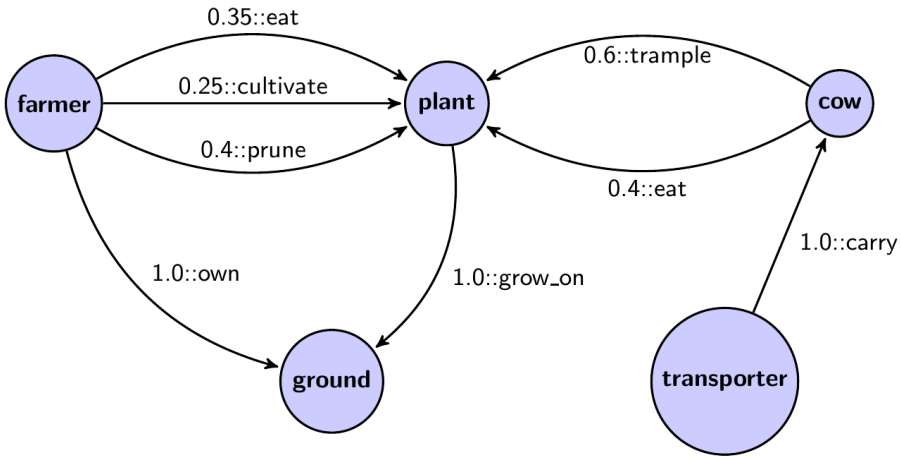

Figure 2 Sample portion of knowledge graph with relative frequencies of occurrence

Fig. 3. Sample portion of knowledge graph with relative frequencies of occurrence

("the meanings of close words in a text tend to refer to the same domain, which is probably the dominant one in that portion of text"). The resulting network can be considered as a formal, structured representation of the collection.

After building the basic graph representing the text, ConNeKTion further enriches and structures it in order to obtain a conceptual level, in the form of a taxonomy or ontology. For this purpose, it assumes that a concept can be defined by (a) the set of other concepts that interact with it in the world described by the corpus, and (b) the set of properties and attributes that describe it, expressed by verbs and adjectives. For this level, additional information is extracted using a purposely developed linguistic expert system, that produces the following relationships between concepts:

- *attribute(C,A)* attribute A describes concept C;
- *can(C,A)* concept C may perform action A;
- *be(C,A)* concept C may be the object of action A;
- *is_a(C1,C2)* concept C1 is a subclass of concept C2;
- *r(C1,C2)* relationship *r* holds between concepts C1 and C2;

where actions are expressed by verbs, and sentences involving verb 'to be' are used to obtain an initial sub-class structure for the taxonomy: e.g., "penguins are birds" yields is a(penguin,bird). A representational trick is adopted to treat indirect complements as direct ones, by embedding the corresponding preposition into the verb: e.g., "The cat jumped on the table" becomes jump_on(cat,table). Figure 2 shows the graph corresponding to sentence "Bell, based in Chicago, makes and distributes electronic, computer and building products".

Each arc in the graph is associated to the frequency with which the associated relationship was found in the processed corpus. More precisely, two frequencies are recorded: one concerning the relationship used in positive form, and the other concerning its use in negative form (i.e., with the verb negated). This indirectly expresses the

'typicality' of the relationships: depending on the balance of frequency for the two forms, the system can infer whether a given relationship is occasional, typical, mandatory, prohibited, etc. for the pairs of concepts. In addition, the frequency distribution for the different kinds of relationships between two nodes is reported, to understand which relationships are more relevant to those concepts. This lays the basis for applying statistical reasoning on the graph. Figure 3 shows a portion of a learned graph.

Note that this part is completely incremental: any time new texts are available, they can be processed separately and integrated into the existing graph, updating the frequencies of existing nodes/edges and possibly adding new nodes and/or edges.

The basic relation underlying taxonomies is generalization. While part of the taxonomy can be extracted directly from the text (as in the case of explicit sentences such as "penguins are birds", yielding arc "is_a" between nodes "penguin" and "bird"), most of its structure must be indirectly inferred by applying suitable reasoning/learning techniques to the non-taxonomic information in the learned graph. The general procedure adopted in ConNeKTion consists of two steps:

1. Grouping: the concepts in the graph are partitioned into separate groups;
2. Generalization: each group obtained in the previous step undergoes generalization, providing a higher-level concept (that may be new or already present in the graph).

The higher-level concepts are connected to the lower-level concepts in the corresponding group by "is_a" relationships. The procedure may be applied again and again to obtain higher and higher-level concepts in the taxonomy.

Of course, albeit learned without the help of external taxonomic resources, if available they can be added later to the resulting knowledge graph (e.g., [14]).

3.2 Use of ML and DM techniques

The FOL description of a concept/node is obtained in ConNeKTion by running a Spreading-Activation algorithm to select the neighborhood of the node. The selected neighboring sub-graph for a concept/node can be interpreted as a formal definition for it (the smaller the activation decay parameter, the larger the sub-graph, and the more refined and detailed the definition).

For the grouping step, ConNeKTion adopts pairwise agglomerative clustering: initially, each concept becomes a singleton cluster; then, clusters are progressively merged until a stop criterion is met. Specifically, ConNeKTion adopts a complete link clustering strategy [22]. If several pairs meet the condition, the pair having smallest (respectively, greatest) average distance for single components is merged. So, each cluster contains similar concepts that can be generalized in order to create new relationships. The generalization (that might correspond to an existing concept in the graph) is introduced as a super-concept of the generalized concepts adding corresponding is_a relationships from it to all of them. Depending on the representation formalism adopted for concepts, this general strategy must be properly adapted, and suitable techniques must be applied. ConNeKTion currently adopts two formalisms, a propositional and a relational one.

- In the *propositional approach*, each concept node *c* in a graph is described by a binary feature vector (v_1 ,..., v_n, v_{n+1}, ..., v_{n+m}), where C = {c_1,..., c_n} is the set of concept nodes in the graph, R = {a_1 , ..., a_m} is the set of relation types (arc labels) in the graph, and:
  - $\forall i = 1,\ldots,n : v_i = 1$ if there is at least one arc (relationship) connecting *c* to c_i, or $v_i = 0$ otherwise;
  - $\forall j = 1,\ldots,m : v_{n+j} = 1$ if there is at least one outgoing arc (relationship) from *c* labeled a_j, or $v_{n+j} = 0$ otherwise.

  Based on this representation, the Hamming distance [23] can be applied to compare pairs of concepts, and the distances between all pairs of concepts can be used to carry out clustering. Concepts described by all-0 vectors are ignored.
- The *relational representation* of the concepts in the learned graph starts from their definition as determined by the Spreading Activation algorithm mentioned above, and translates the resulting sub-graph into a conjunctive FOL formula in which edges are translated using binary predicates, and nodes as their arguments (the first argument representing the subject of the relation, the second argument representing the object).

  Expressiveness is significantly improved over the propositional solution, since not only the attributes and relationships directly associated to the root concept are considered (relation-centric description), but also those among its neighbors at various levels k of distance (concept-centric description). Concerning the clustering step, the relational similarity function presented in [11] can be applied to these concept representations.

A first concept taxonomy for the propositional representations is computed by applying Formal Concept Analysis [24] to the boolean vectors associated to concepts.

As regards generalization of clusters, based on the relational representation, the least general generalization operator proposed in [11] is exploited to generalize the concept descriptions in each cluster, and obtain a FOL description of their generalization, which is the subsumer of the cluster. The use of this operator can also support more advanced tasks, including retrieval of documents of interest.

There are no standard formal techniques to validate the quality of the learned conceptual graph; it can be assessed by the social consensus of domain experts or by its usability for given business objectives [25]. The different conceptual graph building and enrichment techniques embedded in ConNeKTion were evaluated using a purposely collected corpus of documents concerning social networks on socio-political and economic topics, involving 695 nouns and 727 verbs. The size of the dataset was deliberately kept small in order to have poor knowledge. Several experiments were run, varying the spreading activation and clustering parameters in order to obtain taxonomies at different grain-size of detail. 5 experts were asked to evaluate the final conceptual graphs by rating 100 concept definitions and 100 is_a relationships chosen at random from the resulting taxonomy. On average, based on their evaluation, 18% of the items could be considered as substantially correct, and 79% of them as at least

sensible. They also reported that the taxonomy and definitions might be a good starting point for manual refinement.

Other graph analysis tools are available in ConNeKTion, and will be described in the next section, since they are shared with GraphBRAIN.

4 GraphBRAIN

GraphBRAIN is a general-purpose system for the development, management and (personalized) fruition of a knowledge base. As its name suggests, GraphBRAIN adopts graphs as the knowledge base structure.

4.1 System Description

GraphBRAIN integrates several data mining tools for extracting relevant knowledge from the knowledge base and providing it to users and/or other systems, and a comfortable interface for expert to manually add, modify and consult ontological knowledge (both T-BOX, concerning terminology, and A-BOX, concerning specific instances) in the knowledge base. It can be seen as a more structured and controlled way to fill a knowledge graph, that may complement the more automatic and unstructured approach provided by ConNeKTion.

The underlying data management tool is a graph database, currently Neo4j [26]. In Neo4j, nodes and arcs in the graph may have associated attribute-value maps; nodes (representing individuals) may be labeled with any number of labels (usually representing classes), while each arc (representing a relationship) may be labeled with one type only. No schema handling is provided for, meaning that the user is totally free to use any type and/or attribute name for any single node and arc. While ensuring great flexibility, this does not allow one to associate a clear semantics to the graph items. For this reason, GraphBRAIN requires its users to work according to pre-specified data schemes, expressed in the form of ontologies. Thus, a characterizing feature of GraphBRAIN is bringing to cooperation a database management system for efficiently handling, mining and browsing the individuals, with an ontology level that allows it to carry out formal reasoning and consistency or correctness checks on the individuals [27].

Using a suitable tool, GraphBRAIN administrators may create, build and maintain ontologies by specifying the types of entities and relationships to be considered, each with its attributes and associated datatypes. The universal class is implicit, so the user must start the description of each ontology from the top-level classes, which are automatically considered as disjoint by the system. Each top-level class may be the root of a hierarchy of sub-classes, for which no assumption about disjoints is made. Several ontologies may be handled by GraphBRAIN; some classes and relationships may appear in different ontologies, but different ontologies may define different attributes for the shared classes and relationships, in order to reflect different perspectives on them.

In particular, in addition to various domain-specific ontologies, GraphBRAIN provides a top-level ontology defining very general and highly reusable concepts and relationships (e.g., Person, Place; Person.wasIn.Place; etc.). This top-level ontology plays a crucial role to interconnect the domain-specific ontologies, ensuring an overall connected knowledge graph. Indeed, there is a single, shared graph underlying all the domains. In the top-level ontology, currently two very general concept taxonomies are loaded: WordNet [14] and the Dewey Decimal Classification system hierarchy [28]. Note that the concepts in these taxonomies are not used as node labels in the graphs; they are reified and used as nodes, so that they may be related by arcs to other nodes and used by the graph browsing tools included in GraphBRAIN.

Thanks to the classes shared across different domains, this allows the system to reuse knowledge across domains, and thus to reach a wider range of outcomes for satisfying the user's information needs. So, if an individual is used by different ontologies, it acts as a bridge among those ontologies, allowing the users of a domain to obtain additional information coming from other domains, and fostering in this way cross-fertilization of knowledge. The tool automatically saves the ontologies in an internal format, used as a schema for the graph database, and may also export them into standard Semantic Web formats, and made publicly available for reuse. Currently, it can serialize them to Ontology Web Language (OWL) [29] format, so that it can be published and exploited for ensuring semantic access to the knowledge base and making it interoperable with other resources.

After setting up the ontologies, information is fed into the knowledge base by explicit interaction with users, or by automatic knowledge extraction from documents and other kinds of resources (e.g., the Internet) using ConNeKTion. The on-line consultation interface of GraphBRAIN is shown in Figure 4. The top-left screenshot shows the selection of a domain, while the top-right screenshot shows an overview of a portion of the overall graph, suitably selected depending on specific user queries and knowledge about his preferences, aims, background, etc. The bottom-right screenshot shows the interface for modifying and consulting the entities in the knowledge base, while the bottom-left screenshot shows the interface for modifying and consulting the relationships.

More specifically, the interactive interface consists of two form-based tabs, one for entities (see Figure 4, bottom-left) and one for relationships (Figure 4, bottom-right), allowing the user to insert/update/remove instances. The forms are automatically generated by the system starting from the specification of the ontologies provided by the administrators. For this reason, albeit GraphBRAIN may handle several ontologies, each specifying a different domain, the form-based interface for data management and querying requires the user to select one of the available domains in order to load the corresponding scheme/ontology to be used (see Figure 4, top-left).

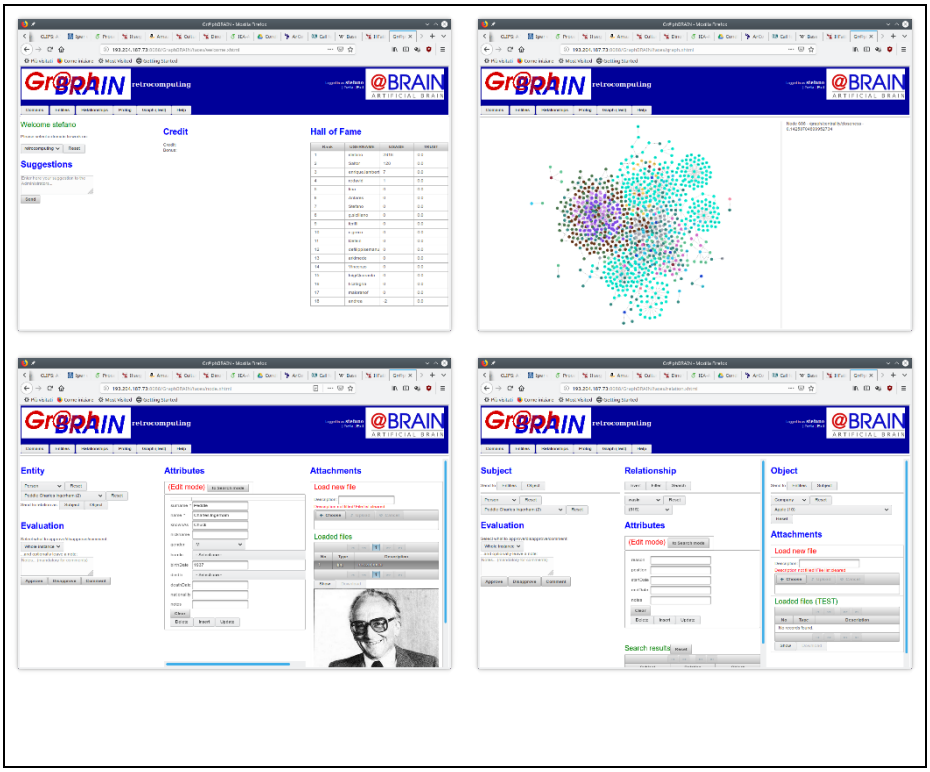

Fig. 4. Graphical representation of the output

Interesting additional functionality is also provided. First, users may also manage (add, show, delete) attachments for each instance. This is a very relevant feature, because in this way GraphBRAIN is not only a knowledge management tool, but it becomes a full-fledged digital library, whose content is indirectly organized according

to formal ontologies, and thus may foster interoperability with other systems. Second, users may add comments, or approve/disapprove, each entity or relationship instance, and even each single attribute value thereof. Since GraphBRAIN is a collaborative platform for knowledge base development, this feature can be used to ensure some kind of 'distributed' quality assurance on the content of the knowledge base, and to establish a trust mechanism for the users. Using the comments, the users may also provide useful suggestions to improve and extend the ontologies. Also, users are encouraged to provide knowledge, and high-quality knowledge, because using a combination of their number of contributions and trust they are assigned 'points' that they may spend in using advanced features provided by GraphBRAIN.

The same form-based interfaces can be used to query the knowledge base for instances of entities and relationships. The retrieved instances may be graphically displayed in another tab, as nodes and arcs in the graph (see Figure 4, top-right). This allows the user to continue his search in a less structured way, by directly browsing the graph (by expanding or compressing node neighbors). This is very useful to ex-

plore the available knowledge without a pre-defined goal in mind, but letting the data themselves drive the search. Thus, serendipity in information retrieval is supported, and the users may find unexpected information that is relevant to their information needs.

4.2 Use of ML and DM techniques

The knowledge management and extraction functionality provided by GraphBRAIN is supported by several analysis, mining and information extraction tools available in the LMS. They allow the users to obtain (personalized) indications on the relevance of single graph items or to extract suitable portions of the graph that may satisfy their information needs. Some of these algorithms are reused from the literature; others have been purposely extended to improve their ability to return personalized outcomes. This would ensure that each user obtains tailored information, which is another novelty introduced by GraphBRAIN.

More specifically, various algorithms for the following functionality are currently included:

- Learning and continuous updating of user-models, in order to capture their background, preferences and interests, so as to guide the knowledge extraction tools in identifying portions of the graph that are more relevant to each single user. It currently implements a decision tree model, based on features such as frequency of access/modification to the various domains, nodes, arcs and attributes.
- Assess relevance of nodes and arcs in the graph, and extract the most relevant ones, based on several centrality indexes provided by Neo4j, in order to capture different relevance perspectives: Betweenness, PageRank, Katz, Closeness, Harmonic.
- Extract a portion of the graph that is relevant to some specified starting points (nodes and/or arcs); the Spreading Activation algorithm is used for this purpose, suitably adapted to use different (personalized) decay parameters for different users, also based on the statistics previously mentioned for the user models.
- Extract frequent patterns of entities and relationships, and associated sub-graphs; the gSpan algorithm is used for this task, suitably modified to constrain the search for sub-graphs so as to include specific kinds of nodes, or even specific nodes.
- predict possible links between nodes, again using the tools provided by Neo4j.

Due to the peculiarities of the system, no ground truth was available for quantitatively evaluating the performance of these tools. We indirectly evaluated it by asking 41 users (undergraduate students aged 23-26, 23 male, 18 female) to use the system and compile a questionnaire to report their degree of satisfaction. All of them reported the system provided interesting and/or useful information in at least 80% of the times. In particular, 88% of them reported various degrees of satisfaction for the functionality that selects the initial portion of the graph to be displayed, and 76% appreciated the extraction of frequent patterns in the graph.

The set of mining and analysis tools in GraphBRAIN is constantly updated and extended, directly borrowing solutions from the literature, adapting or extending other approaches, or developing completely novel ones. E.g., a method for conceptual graph learning which might be relevant is described in [30], proposing the use of a frequent subgraph patterns mining approach to summarize graphs into groups of subgraphs to be used for further characterization, discrimination, classification, and cluster analysis of a collection of graphs.

5 Conclusions

The number of documents available in electronic format is ever-growing, which leads to well-known problems of organization and *information overloading*. This calls for intelligent digital libraries, that allow to automatize the document processing and knowledge extraction tasks, and to suitably organize and make available the documents and the knowledge they contain, so as to provide personalized and focused access.

This paper described an integrated system that merges intelligent modules covering all the phases involved in a document lifecycle, from acquisition, to processing, to information extraction, to personalized fruition for final users. The role and possible cooperation of Machine Learning and Data Mining techniques in the system is highlighted and discussed, along with their importance to provide effective support to both the building and the fruition of the Digital Library and the underlying knowledge base.

Quantitative and qualitative results show that both the single techniques, and their cooperation, may be effective in tackling the above issues and ensuring user satisfaction. Of course, different approaches are appropriate for different tasks, albeit based on the same representations. In particular, relational (First-Order Logic and Graph-based) representations and techniques proved to be very powerful and very useful.

Future work will be aimed at further improving effectiveness of the available tools and systems, and at extending the set of tools. Also, the design of decision support systems that cleverly exploit the knowledge in the learned graph to help users in carrying out their activities is envisaged.

References

1. Esposito, F., Semeraro, G., Fanizzi, N., Ferilli, S.: Multistrategy Theory Revision: Induction and Abduction in INTHELEX. Machine Learning Journal 38(1/2), 133-156 (2000).
2. Ferilli, S.: WoMan: Logic-based Workflow Learning and Management. IEEE Transaction on Systems, Man and Cybernetics: Systems 44(6), 744-756 (2014).
3. Ferilli, S.: The WoMan Formalism for Expressing Process Models. In: Perner, P. (Ed.) Advances in Data Mining – Applications and Theoretical Aspects. LNAI, vol. 9728, pp. 363-378, Springer (2016).

4. Esposito, F., Ferilli, S., Basile, T.M.A., Di Mauro, N.: Machine Learning for digital document processing: From layout analysis to metadata extraction. In: Marinai, S., Fujisawa, H. (Eds.) Machine Learning in Document Analysis and Recognition. Studies in Computational Intelligence, vol. 90, pp. 105-138, Springer, Berlin (2008).
5. Ferilli, S.: Automatic Digital Document Processing and Management – Problems, Algorithms and Techniques. Advances in Pattern Recognition, Springer, London (2011).
6. Salton, G., Wong, A., Yang, C.: A vector space model for automatic indexing. Communications of the ACM 18(11), 613-620 (1975).
7. Deerwester, S., Dumais, S.T., Landauer, T.K., Furnas, G., Harshman, R.: Indexing by latent semantic analysis. Journal of the American Society of Information Science 41(6), 391-407 (1990).
8. Esposito, F., Malerba, D., Semeraro G.: A Comparative Analysis of Methods for Pruning Decision Trees. IEEE Transactions on Pattern Analysis and Machine Intelligence 19(5), 476-491 (1997).
9. Dietterich, T.G., Lathrop, R.H., Lozano-Perez, T.: Solving the multiple instance problem with axis-parallel rectangles. Artificial Intelligence 89(1-2), 31-71 (1997).
10. Ferilli, S., Basile, T.M.A., Di Mauro, N.: Markov Logic Networks for Document Layout Correction. In: Mehrotra, K.G., Chilukuri, M., Oh, J.C., Varshney, P.K., Ali, M. (Eds.) Modern Approaches in Applied Intelligence – Part I. LNAI, vol. 6703, pp. 275-284, Springer (2011).
11. Ferilli, S., Basile, T.M.A., Biba, M., Di Mauro, N., Esposito, F.: A General Similarity Framework for Horn Clause Logic. Fundamenta Informaticae 90(1-2), 43-66, IOS Press (2009).
12. Ferilli, S., De Carolis, B., Esposito, F., Redavid, D.: Sentiment Analysis as a Text Categorization Task: A Study on Feature and Algorithm Selection for Italian Language. In: Gaussier, E., Cao, L., Gallinari, P., Kwok, J., Pasi, G., Zaiane, O. (Eds.) Proceedings of the IEEE International Conference on Data Science and Advanced Analytics 2015 (DSAA). pp. 1-10, IEEE (2015).
13. Ferilli, S., Biba, M., Basile, T.M.A., Esposito, F.: Combining Qualitative and Quantitative Keyword Extraction Methods with Document Layout Analysis. In: Agosti, M., Esposito, F., Thanos, C. (Eds.) Post-proceedings of the 5th Italian Research Conference on Digital Library Management Systems (IRCDL-2009). pp. 22-33 (2009).
14. Miller, G.A.: Wordnet: A lexical database for English. Communications of the ACM 38, 39-41 (1995).
15. Ferilli, S., Redavid, D., Angelastro, S.: Activity Prediction in Process Management Using the WoMan Framework. In: Perner, P. (Ed.) Advances in Data Mining. Applications and Theoretical Aspects, LNAI, 10357, pp. 194-208, Springer, 2017.
16. Perner, P.: Concepts for Novelty Detection and Handling based on a Case-Based Reasoning Scheme. In: P. Perner (Ed.) Advances in Data Mining. LNAI, vol. 4597, pp. 21-34, Springer (2007).

17. Perner, P.: Novelty Detection and In-Line Learning of novel concepts according to a case-based reasoning process schema for high-content image analysis in system biology and medicine. Computational Intelligence 25(3), 250-263 (2009).
18. Perner, P., Attig, A.: Fuzzy conceptual clustering Quality and Reliability Engineering International 26(8), 909-922 (2010).
19. Rotella, F., Leuzzi, F., Ferilli, S.: Learning and Exploiting Concept Networks with ConNeKTion. Applied Intelligence 42(1), 87-111, Springer (2015).
20. Maedche, A., Staab, S.: Mining ontologies from text. In: Dieng, R., Corby, O. (Eds.) International Conference on Knowledge Engineering and Knowledge Management. pp. 189-202 (2000).
21. Gale, W.A., Church, K.W., Yarowsky, D.: One sense per discourse. In: Markus, M.P. (Ed.) DARPA Speech and Natural Language Workshop. pp. 233-237, Association for Computational Linguistics (1992).
22. Defays, D.: An efficient algorithm for a complete link method. Comput. J. 20(4), 364-366 (1977).
23. Hamming, R.W.: Error Detecting and Error Correcting Codes. Bell System Technical Journal 26(2), 147-160 (1950).
24. Wille R.: Formal Concept Analysis as Mathematical Theory of Concepts and Concept Hierarchies. In: *Ganter, B., Stumme, G., Wille, R. (Eds.) Formal Concept Analysis – Foundations and Applications. LNCS, vol. 3626, Springer (2005).*
25. Hasegawa, R., Kitamura, M., Kaiya, H., Saeki, M.: Extracting conceptual graphs from Japanese documents for software requirements modeli*ng. In: Kirchberg, M., Link, S. (Eds.) Conceptual Modelling 2009, Sixth Asia-Pacific Conference on Conceptual Modelling (APCCM 2009). CRPIT, 96, pp. 87-96, Australian Computer Society (2009).*
26. Robinson, I., Webber, J., Eifrem, E.: Graph Databases. O'Reilly Media, 2nd edn. (2015).
27. Krötzsch, M.: Ontologies for knowledge graphs? In: Artale, A., Glimm, B., Kontchakov, R. (Eds.) Proceedings of the 30th International Workshop on Description Logics, CEUR Workshop Proceedings, vol. 1879, CEUR-WS.org (2017).
28. Dewey, M.: A classification and subject index for cataloguing and arranging the books and pamphlets of a library. Amherst (1876).
29. OWL Web Ontology Language Reference, W3C Recommendation, https://www.w3.org/TR/owl-ref/ (consulted May 6, 2019).
30. Perner, P.: Mining Frequent Subgraph Pattern Over a Collection of Attributed-Graphs and Construction of a Relation Hierarchy for Result Reporting. In: Perner, P. (Ed.) Advances in Data Mining. LNAI, vol. 10357, pp. 323-344, Springer (2017).