Interactive Text Analysis and Information Extraction

Tasos Giannakopoulos, Yannis Foufoulas(✉), Harry Dimitropoulos, and Natalia Manola

University of Athens, Greece and Athena Research Center, Athens, Greece
johnfouf@di.uoa.gr

Abstract. A lot of work that has been done in the text mining field concerns the extraction of useful information from the full-text of publications. Such information may be links to projects, acknowledgements to communities, citations to software entities or datasets and more. Each category of entities, according to its special characteristics, requires different approaches. Thus it is not possible to build a generic mining platform that could text mine various publications to extract such info. Most of the time, a field expert is needed to supervise the mining procedure, decide the mining rules with the developer, and finally validate the results. This is an iterative procedure that requires a lot of communication among the experts and the developers, and thus is very time-consuming. In this paper, we present an interactive mining platform. Its purpose is to allow the experts to define the mining procedure, set/update the rules, validate the results, while the actual text mining code is produced automatically. This significantly reduces the communication among the developers and the experts and moreover allows the experts to experiment themselves using a user-friendly graphical interface.

1 Introduction

Text mining of scientific publications is a very important task as it offers the tools for extracting useful information from their content that is of interest not only to researchers and research communities, but also to funders, publishers, policy-makers, etc. This information 'enriches' publications and allows the interlinking of relevant content; a process critical for allowing researchers to discover, share and re-use scientific results in context, and research administrators to assess research impact and investment in a transparent and efficient way.

In OpenAIRE[1], we tackle the problem of linking publications to projects and/or communities that are members of OpenAIRE. By initiating bilateral collaborations with national funding agencies, research and infrastructure initiatives, OpenAIRE is able to serve them with the OpenAIRE mining services and research impact suite of services. OpenAIRE mining services can automatically infer links from publication full-texts to datasets, projects, software, and

[1] https://www.openaire.eu/.

P. Manghi et al. (Eds.): IRCDL 2019, CCIS 988, pp. 340–350, 2019.
https://doi.org/10.1007/978-3-030-11226-4_27

research initiatives. Today such services are manually configured, based on a model of the concepts to be discovered by mining and a set of mining rules. This is a difficult text mining task because each new community/funder requires a different approach, so we actually need to customize the mining rules and run an almost new mining algorithm for every new funder/community.

Currently, an information extraction algorithm is developed following the steps below:

- Communicate with a field expert and exchange some information or data
- Produce an initial version of the algorithm
- Run the current algorithm version on a large 'test' corpus of publication full-texts
- Send the results to the expert for validation
- Update the algorithm using the experts' feedback.

The last three steps are in fact an iterative procedure. The algorithm is finalised when the quality of its results is satisfying. Since, each mining task is unique and the produced algorithm differs, this may be a time consuming method with a lot of communication between experts and developers.

In this paper, we provide the experts with an interactive mining platform which allows them to set up their mining rules, validate the intermediate results and update the rules accordingly. This platform covers the following requirements:

- It is user friendly, since its users are not developers
- It provides the users with enough configuration tools to produce their mining profiles
- It creates and runs automatically the mining algorithms on sample data selected by the user
- It supports reproducibility of the mining algorithms.

The rest of the paper is organised as follows: First, we describe the configuration tools and how users are able to produce and tune a mining algorithm. Then, we present the user interface of the platform. We explain how users can evaluate a produced mining algorithm using sample datasets and we present the reproducibility features of the platform. Finally, we discuss in brief the future work.

2 Configuration Tools

A complete set of configuration tools are available, so that users are able to produce and update their algorithms. These tools are divided in three main categories:

- Preprocessing tools
- Processing tools
- Evaluation tools.

Preprocessing tools are all the tools that a data miner uses during the preprocessing phase of a text mining procedure. Such tools are stemmers, tockenizers, normalizers, etc. During this step the user uploads the concepts that are to be mined. Each concept consists of a string (i.e., its name) and a set of n-grams/phrases that usually are used when an author mentions this concept.

Processing tools include the main tools that a user utilises during the extraction phase. The algorithm runs a scrolling window over the text searching for matches. The user defines the length of this window. S/he also selects positive and negative weighted phrases. These phrases influence the confidence level of a match. Finally, the user selects the matching policy: when higher recall rates are prefered then the matching policy is more soft; otherwise the policy is strict.

Evaluation tools include the tools that are required to calculate the precision of an algorithm. Users are able to run their algorithms on predefined datasets or define and upload their own datasets. Moreover, during a run they have access to a preview where the matching areas, the positive/negative phrases, and the actual matches are highlighted. This makes the validation/update process easier since users are able to tune the algorithm rules and view online the results of their updates, thus facilitating rapid algorithm development.

3 User Interface

The user interface and its usability is very important since it provides the users with all the available tools to create the algorithm. In this section, we go through the portal and show how a user can produce algorithms, experiment himself, set his rules, etc.

Users are landed in the home page of the interactive mining platform which is shown in Fig. 1. Here, the user is able to see the saved profiles or create new profiles. Some example profiles are already available so that users may use them to familiarize themselves with the platform.

Figures 2 and 3 illustrate how the user uploads/edits his mining concepts. After selecting/creating a mining profile, users are able to configure their algorithm. The first step regards the addition/deletion of mining concepts. Users can either use the online form to edit their concepts or drag and drop a tabular file. For each concept users upload an identifying string and some phrases/n-grams that the authors often use to refer to this concept.

Figure 4 presents the main configuration and experiment page of the portal. When the concepts are ready, users are able to configure their mining rules. They can define their mining strategy (choose between high recall or high precision), add positive/negative phrases/keywords, tune the preprocessing steps, and modify the length of the mining area size (i.e., the length of the text area before and after a match that the algorithm uses as context to decide if a match is a true positive or not).

Positive phrases are phrases or keywords that are very likely to be found in the vicinity of a match. Different weights can be used to specify the relative importance of the different phrases. Negative phrases are phrases that when

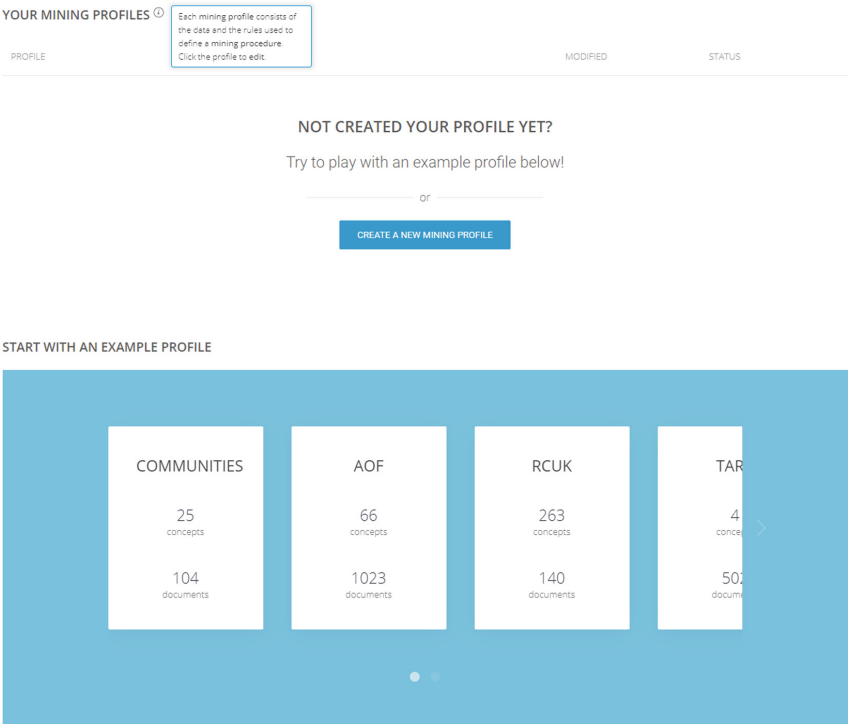

Fig. 1. Home page of the interactive mining platform

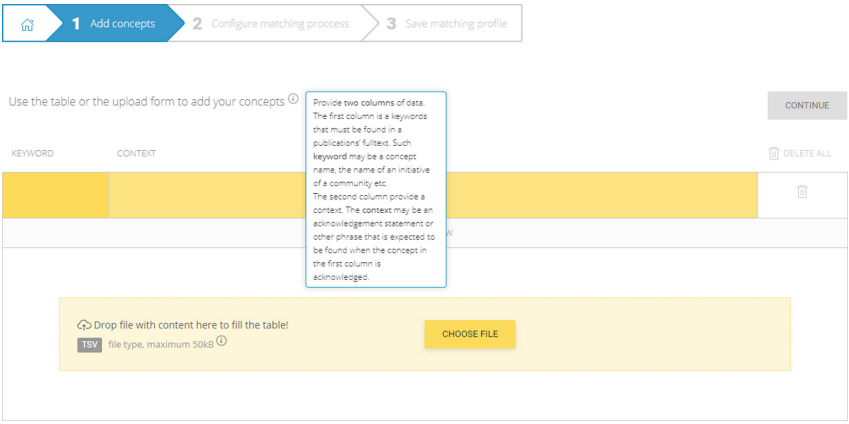

Fig. 2. Upload concepts

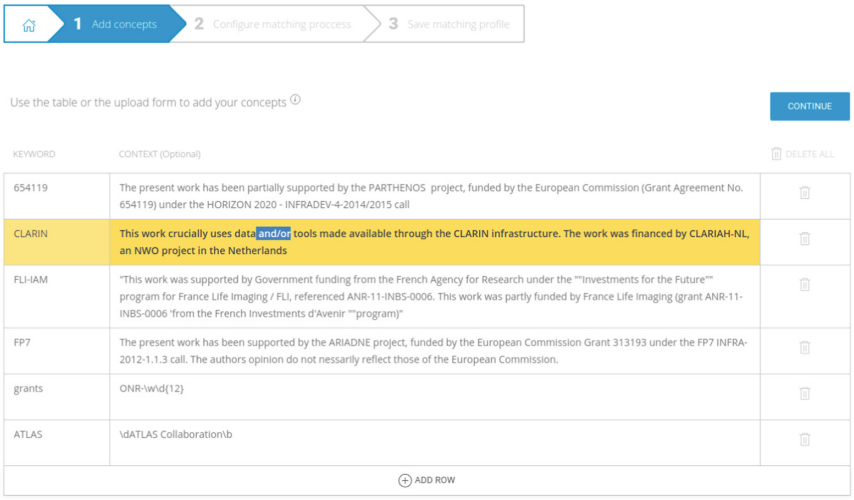

Fig. 3. Edit concepts

found near a match increase the likelihood of it being a true negative. As with positive phrases, users can specify different weights to assign relative importance to each phrase.

The preprocessing steps currently supported are stopword removal (removes common words such as articles like 'an', 'and', 'the', etc., and prepositions like 'after', 'to', etc.), punctuation removal, normalization (converting text to lower case), and word stemming (the process by which variant forms of a word are reduced to a common form, for example: connection, connections, connective, connected and connecting, are reduced to the stem 'connect').

Moreover, users can test online their mining rules against the selected datasets. The mining results are presented using annotated and highlighted text. The users edit their rules and run experiments repeatedly until they get valid results.

When users are satisfied with their mining rules they can continue to the last step of the mining configuration. Users can save their configuration for future use, as shown in Fig. 5.

When users return to the mining platform their profiles are available at their home page for further modifications and testing (Fig. 6).

4 Reproducibility Features

Since the purpose of the platform is to allow users to build and update their own mining algorithms, reproducibility features are highly important. Users should be able to revisit their algorithms, update their rules, and continue their experiments. To support this, a database file is stored for each user of the platform.

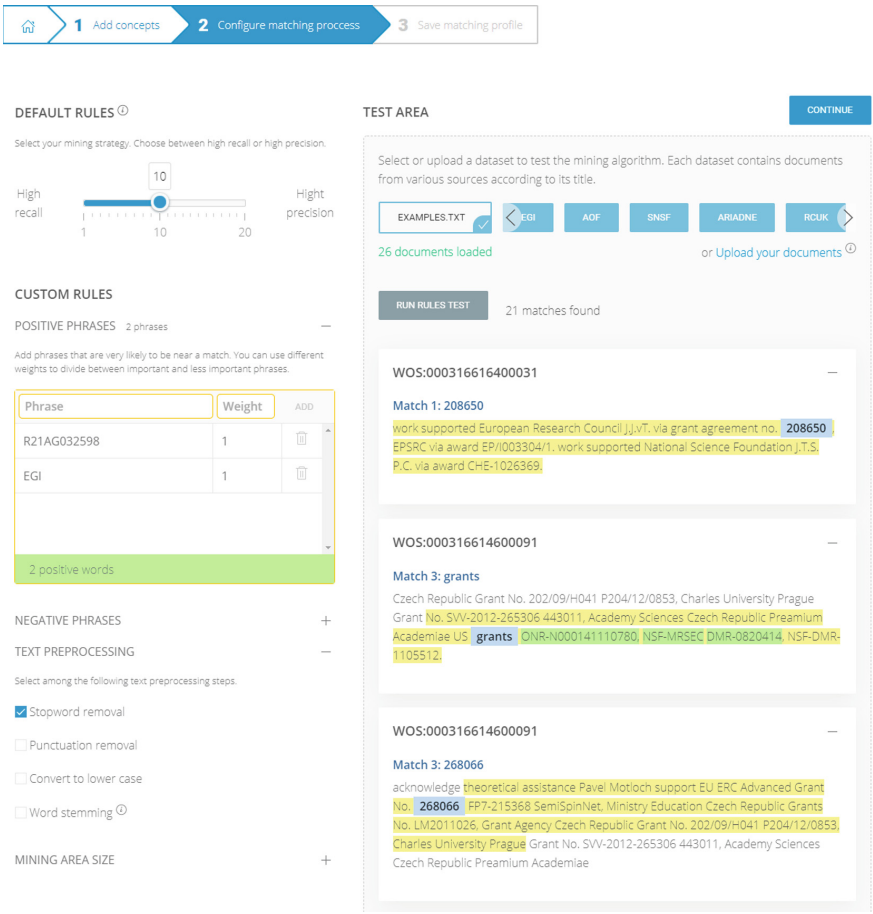

Fig. 4. Configuration Page. The page is split into two main horizontal sections. The left section is where users define their mining rules and choose between high recall or high precision with a slider. Here they can add or edit their positive and negative phrases/keywords, define the preprocessing steps to be used, and advanced users can also define the length of the text mining area size. The right section is the test area where mining results are presented using annotated and highlighted text, allowing users to tune their rules and run experiments repeatedly until they get valid results.

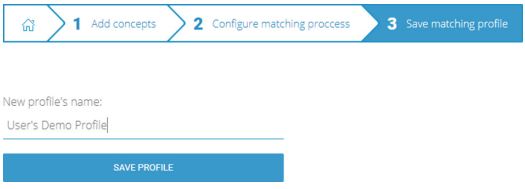

Fig. 5. Save a mining profile

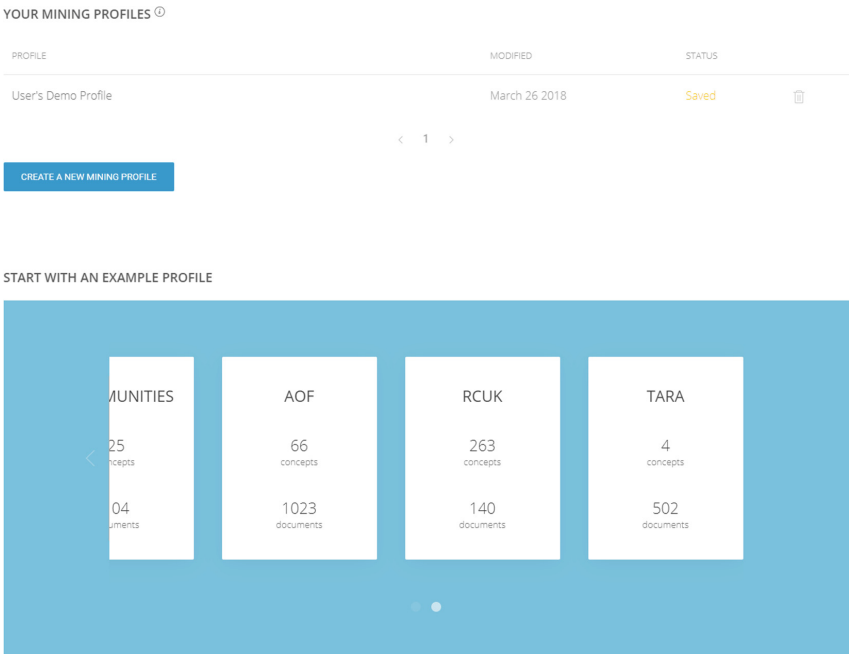

Fig. 6. When users revisit the mining platform their profiles are available at their home page for further modifications and testing.

This database contains all the saved algorithms per user. This list is shown to the users when they are connected to the portal. Users are able to select a profile and update it.

Another important feature is that the platform also maintains the history of the saved changes. So, users may visit a previous version of their algorithm. This is a very important feature that allows users to compare different versions.

5 Implementation Details

The mining platform is implemented using Python's Tornado Server[2]. Tornado is a Python web framework and asynchronous networking library using non-blocking network I/O. Its front end is implemented in Angular 4[3] and communicates with the back-end via a REST API.

The back-end is implemented on top of madIS [2], a powerful extension of a relational DBMS with user-defined data processing functionality. MadIS is built on top of the SQLite API[4].

[2] https://www.tornadoweb.org/en/stable/.
[3] https://angular.io.
[4] https://www.sqlite.org/.

MadIS allows the creation of user-defined functions (UDFs) in Python and it uses them in the same way as its native SQL functions. Both Python and SQLite are executed in the same process, greatly reducing the communication cost between them. This is a major architectural element and has a positive impact on joint performance.

MadIS is highly scalable, easily handling 10s of Gigabytes of data on a single machine. This benefit transparently carries over to distributed systems (e.g., Hadoop [3], Exareme [4]) which can use madIS in each node.

In madIS, queries are expressed in madQL: a SQL-based declarative language extended with additional syntax and user-defined functions (UDFs).[5] One of the goals of madIS is to eliminate the effort of creating and using UDFs by making them first-class citizens in the query language itself. To allow easy UDF editing with a text editor, madIS loads the UDF source code from the file system. Whenever a UDF's source code changes, madIS automatically reloads the UDF definition. This allows rapid UDF development iterations, which are common in data exploration and experimentation.

MadIS supports three types of UDFs:

- Row functions: Programmable row functions work in a similar way as standard SQL row functions such as abs(), lower() and upper().
- Aggregate functions: Programmable aggregate functions work in a similar way as standard SQL aggregate functions such as sum(), min() and max().
- Virtual table functions: These are actually functions that take parameters and output table like data. They can be used in the SQL syntax wherever a regular table would be placed.

All UDFs are written in Python and can use pre-existing Python libraries (NumPy[6], SciPy[7], etc.), thus inheriting features that are commonly used by data scientists [1].

The expressiveness and the performance of madIS allow developers to implement fast data analysis and complex text mining tasks [5–8]. The above were compelling reasons for choosing it as our processing engine.

The following is an example of an automatically produced query that text mines publications to extract links to research projects funded by the EC's seventh framework programme (FP7). The query preprocesses the text converting it to lower case and extracting its keywords, then joins the extracted keywords with the grants list and finally searches the context for positive words/phrases using pattern matching.

[5] MadIS, as well as the vast majority of other UDF systems, expects "functions" to be proper mathematical functions, i.e., to yield the same output for the same input, however, this property is not possible to ascertain automatically since the UDF language (Python) is unconstrained.

[6] http://www.numpy.org.

[7] https://www.scipy.org.

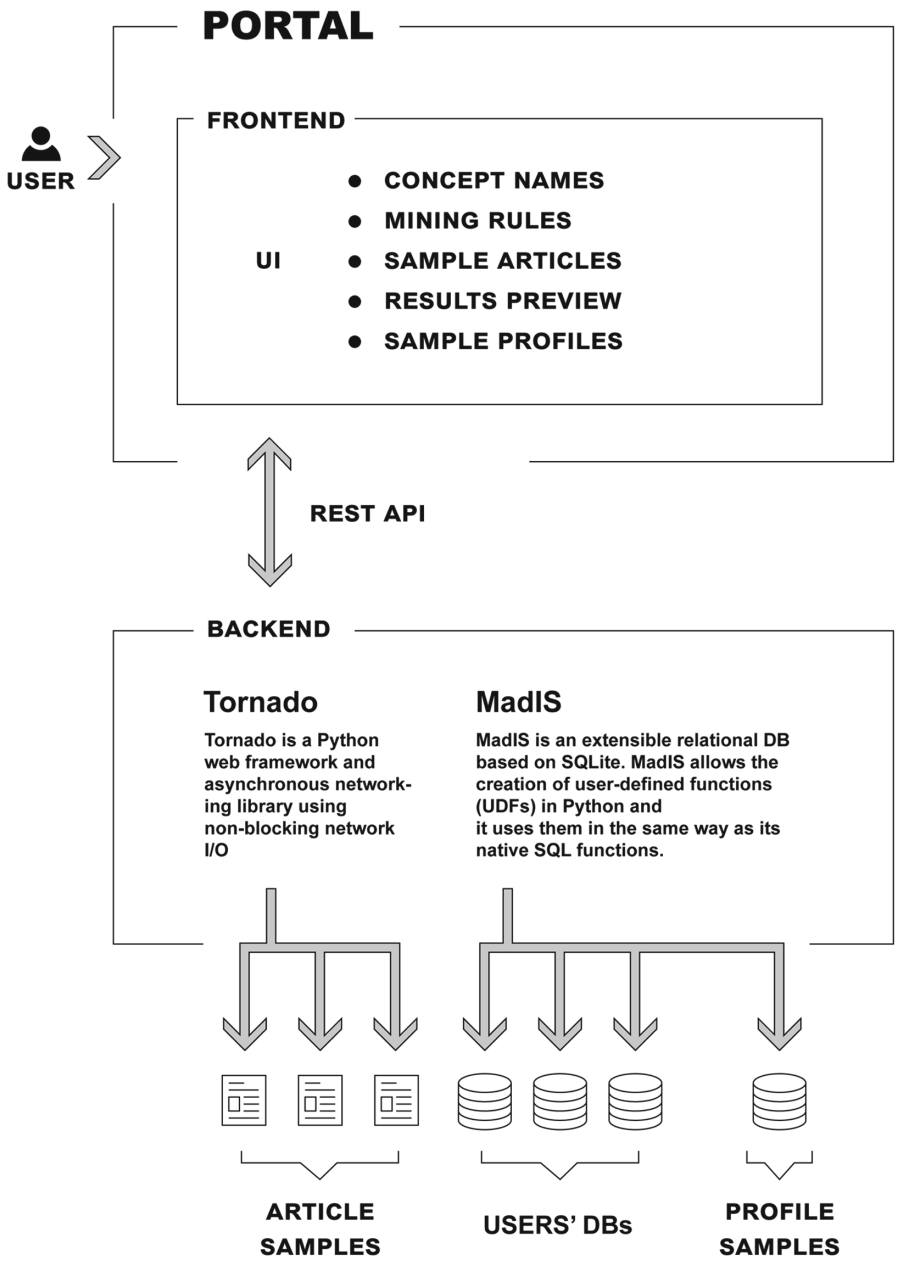

Fig. 7. Architecture of the platform

```
select docid, id, mining_category, mining_concept from (
    select * from (
        select
            document_id,
            textwindow(keywords(lower(document_text)), 20, 1, 20)
        from input_corpus
        ), concepts_list
        where mining_category = "FP7" and
              mining_concept=document_text_term and
              regexpr("fp7|support|grant|contract|funded|
                       european commission|acknowledge",text_window)
) group by document_id, mining_concept_id;
```

The user sets the parameters using the User Interface. These parameters are sent to the back-end where they are translated into an SQL query with the appropriate UDFs. All the processing is expressed within a single query. The query is run and stored for future use.

The architecture of the presented platform is shown in Fig. 7. The front-end includes the User Interface of the Platform and offers the necessary tools to the users. The back-end includes the Tornado Web Server and the processing engine (madIS). The storage layer contains the sample datasets, the sample mining profiles that are offered as examples, and one database file per user. The algorithms and other properties of each user are stored in this database.

6 Future Work

The future work mainly concentrates on utilization of machine learning techniques to make recommendations. Such recommendations are based on user's feedback. The platform could suggest both mining configurations according to already saved algorithms, and test datasets using topic modelling techniques.

Another useful functionality regards sharing of mining algorithms. A user may share his algorithm with the community so that an other user may validate it or even update it. History of all updates will be stored to support reproducibility.

Acknowledgements. This work is funded by the European Commission under H2020 projects OpenAIRE-Connect (grant number: 731011) and OpenAIRE-Advance (grant number: 777541).

References

1. Agrawal, R., Shim, K.: Developing tightly-coupled data mining applications on a relational database system. In: KDD (1996)
2. madIS, Lefteris Stamatogiannakis, Mei Li Triantafillidi, Yannis Foufoulas. http://www.github.com/magdik/madIS. Accessed 4 Oct 2018
3. Shvachko, K., Kuang, H., Radia, S., Chansler, R.: The hadoop distributed file system. In: 26th Symposium on Mass Storage Systems and Technologies (MSST) (2010)
4. Chronis, Y.: A relational approach to complex dataflows. In: EDBT/ICDT Workshops (2016)
5. Giannakopoulos, T., Foufoulas, I., Stamatogiannakis, E., Dimitropoulos, H., Manola, N., Ioannidis, Y.: Discovering and visualizing interdisciplinary content classes in scientific publications. D-Lib Mag. **20**(11), 4 (2014)

6. Giannakopoulos, T., Foufoulas, I., Stamatogiannakis, E., Dimitropoulos, H., Manola, N., Ioannidis, Y.: Visual-based classification of figures from scientific literature. In: Proceedings of the 24th International Conference on World Wide Web, pp. 1059–1060. ACM, May 2015
7. Giannakopoulos, T., Stamatogiannakis, E., Foufoulas, I., Dimitropoulos, H., Manola, N., Ioannidis, Y.: Content visualization of scientific corpora using an extensible relational database implementation. In: Bolikowski, Ł., Casarosa, V., Goodale, P., Houssos, N., Manghi, P., Schirrwagen, J. (eds.) TPDL 2013. CCIS, vol. 416, pp. 101–112. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-08425-1_10
8. Foufoulas, Y., Stamatogiannakis, L., Dimitropoulos, H., Ioannidis, Y.: High-pass text filtering for citation matching. In: Kamps, J., Tsakonas, G., Manolopoulos, Y., Iliadis, L., Karydis, I. (eds.) TPDL 2017. LNCS, vol. 10450, pp. 355–366. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-67008-9_28