Competencies mining with LookinLabs

William Dedzoe, Jean Hany, Albert Benveniste

Competencies mining with LookinLabs

William Dedzoe , Jean Hany , Albert Benveniste

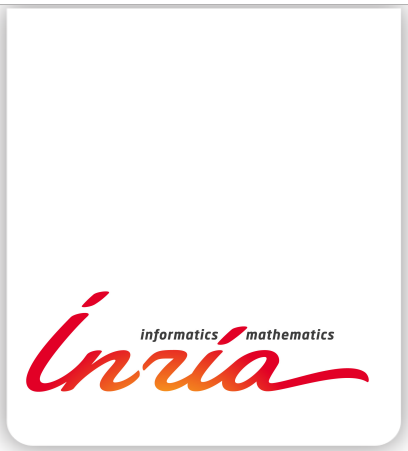
RESEARCH REPORT

N° 9158

March 2018

Project-Teams Hycomes

ISSN 0249-6399 ISRN INRIA/RR--9158--FR+ENG



Competencies mining with LookinLabs

William Dedzoe [*], Jean Hany [†], Albert Benveniste [‡] [§]

Project-Teams Hycomes

Research Report n° 9158 — version 1 — initial version March 2018 — revised version March 2018 — 27 pages

Abstract: LookinLabs is a tool that returns teams, individuals, and publications best semantically correlated with a query, submitted as a list of keywords or a text. In short, it performs competencies mining. The teams, individuals, and publications must belong to a defined institution or lab (e.g., Inria). The tool relies on the HAL open archive, complemented by administrative information bases offered by the institution or lab. This side information turns out to be essential for the quality of the responses, as meta-data (author's names and affiliations) are subject to many variations. To the best of our knowledge, no similar tool or service exists in France.

Key-words: competencies mining, text mining, Elasticsearch, HAL

This work was launched and supported by the Labex CominLabs. It is now jointly funded by CominLabs (https://cominlabs.u-bretagneloire.fr/) and Inria (https://www.inria.fr/).

[*] INRIA, Rennes, France
[†] INRIA, Rennes, France
[‡] INRIA, Rennes, France
[§] Corresponding author: albert.benveniste@inria.fr

RESEARCH CENTRE
RENNES – BRETAGNE ATLANTIQUE

Campus universitaire de Beaulieu
35042 Rennes Cedex

LookinLabs: un outil de fouille de compétences

Résumé : LookinLabs est un service de fouille de compétences. En réponse à une requête formulée sous forme de mots-clés ou texte, il renvoie les équipes, individus, et publications les mieux corrélés à la requête. Les équipes, individus, et publications sont ceux d'un univers bien défini (ex.: Inria). LookinLabs utilise les donées et meta-données de l'archive ouverte HAL, complétées par des données administratives fournies par les instituts ou laboratoires (personnel et affiliation en termes d'équipe). Ces informations complémentaires s'avèrent précieuses en raison de la piètre qualité des meta-données de HAL. Pour autant que nous sachions, il n'existe pas, en France, d'outil de ce type.

Mots-clés : fouille de compétences, fouille de texte, Elasticsearch, HAL

Contents

1 Motivation

Despite the advanced tools to search publications and more generally scientific information over the web (e.g., Google Scholar), it remains difficult to find the individuals, teams, or labs, best matching in competence a given topic.

For instance, Figure 1 shows the first page of the return, by Google Scholar, for the query *network security*. Focusing on people can be achieved by clicking on "profile", which returns the result shown on Figure 2. This is a significant imporvement over the previous response. The authors returned have mentioned *network* and *security*, either in their own description, or in

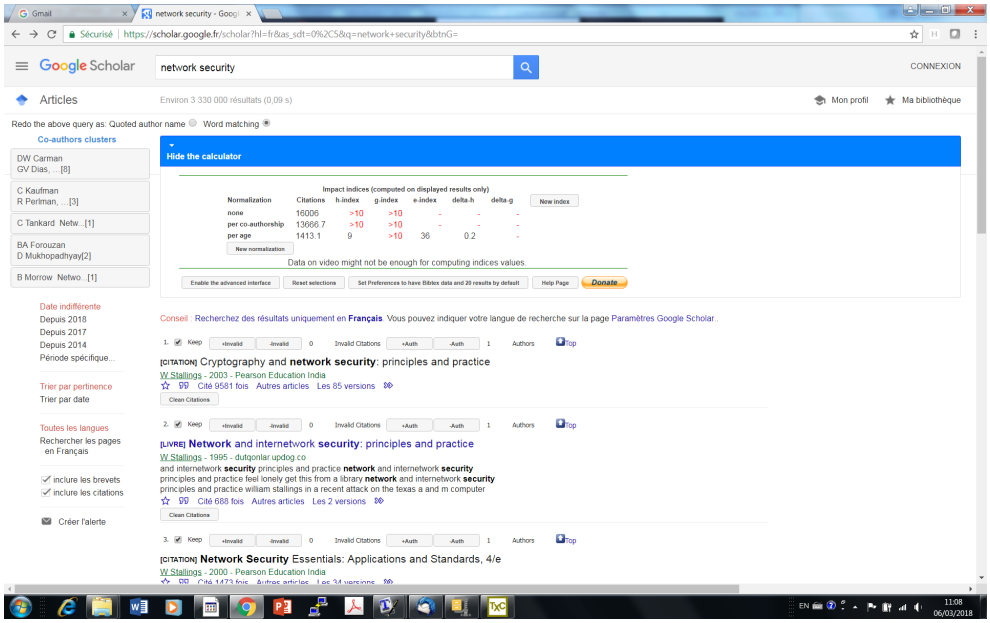


Figure 1: Google Scholar response to the query *network security*.

their list of keywords. Ranking is performed among the matching authors by the number of citations. However, nothing guarantees that the high ranking of the first author is due to a strong popularity in the considered topics — the problem can arise for authors having multiple topics of interest.

In addition, Google Scholar and similar services (such as, e.g., Research Gate https://www.researchgate.net/) respond in terms of individuals, but not teams or labs.

The objective of LookinLabs is to help users to find authors and teams best relevant to a given topic. The information used by LookinLabs consists of publications archives, and more specifically the HAL archive for science in France (https://hal.archives-ouvertes.fr/). We subsequently discuss the use of other archives for the same purpose. Finding best relevant authors for a given topic is performed by mining the publication data base. To return a response in terms of teams, we exploit the HAL meta-data, in combination with complementary information specific to a given subpopulation of authors — e.g., those affiliated to a given university, to CNRS, to Inria, etc.

As a comparison, we show in Figure 3 the response of LookinLabs, relative

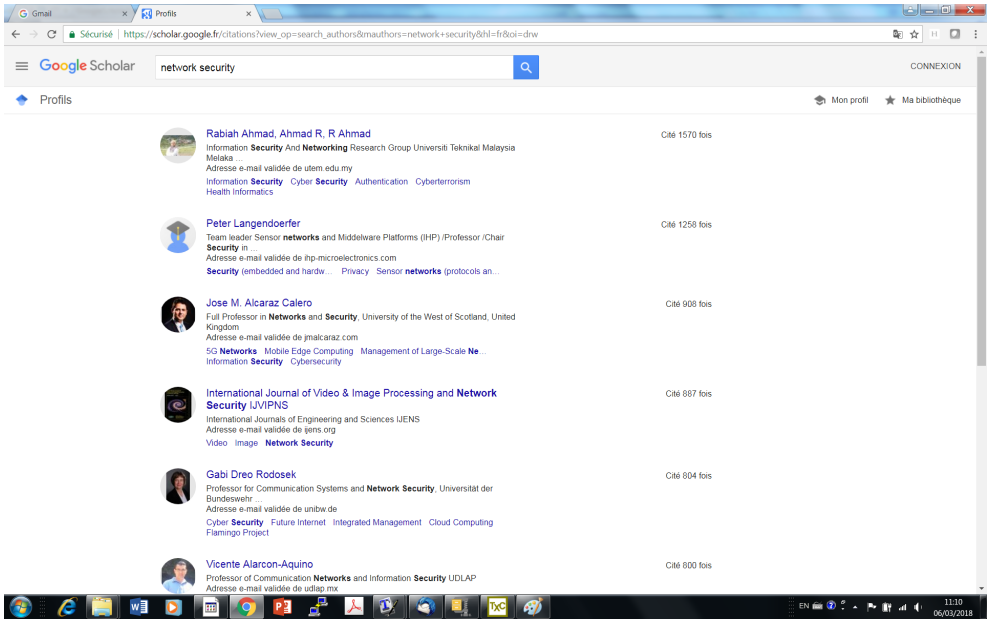


Figure 2: Google Scholar profile response to the query *network security*.

to Inria, for the query *network security*. The screenshot shows three columns, for the teams, individuals, and publications, best correlated to the query.

In the first column, the mention MASCOTTE→**COATI** corresponds to a situation in which 1) the tool has found the MASCOTTE team, 2) this team is no longer active (it is written in gray), 3) a team with about the same member followed with the name **COATI**. The **COATI** team could not be directly found by the tool because it is recent and thus has too small a publication record. The side information used by LookinLabs was found in administrative information bases of Inria, which are used to complement the HAL archive.

The "teams" column offers four buttons. The first one points to the publications of the team that are correlated to the query. The second one explains the reasons of the ranking in terms of explanatory keywords. The third one returns the team members who are correlated to the query. And the fourth button gives access to the social graph of the team (similar teams). Similar buttons with slef-explanatory meanings are found for the second and third columns.

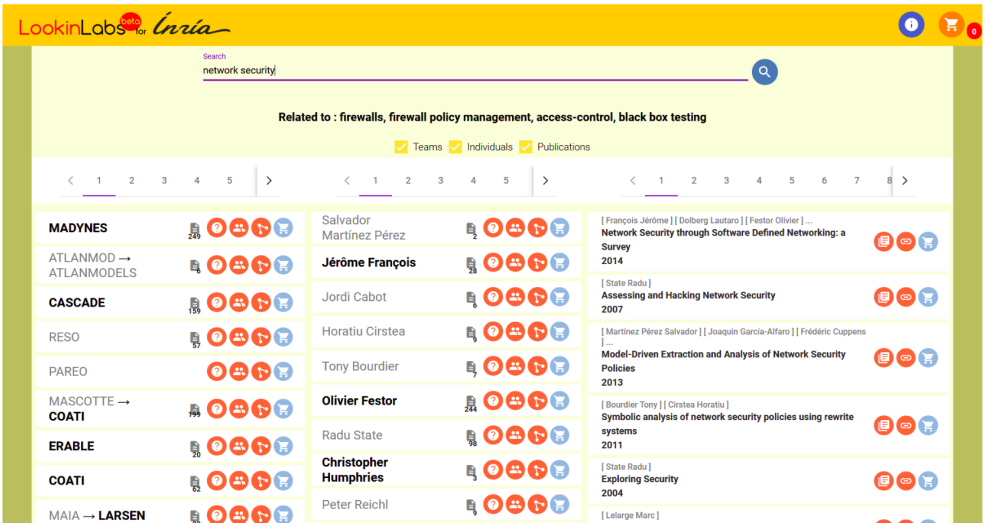


Figure 3: LookinLabs response to the query *network security*.

2 Service Description

2.1 LookinLabs Primer

LookinLabs for HAL-archives is more than a search tool. It allows you to find, among teams/individuals/publications. The tool exploits, as data, HAL-archives. No ontology is used. No data need to be manually entered (besides the users queries). The tool uses Elasticsearch as its core algorithm. This means that the matching is based on a distance between the query and the set of data attached, in HAL, to each team/individual/publication. Ranking is performed accordingly. Explanations are given for each returned item. Correlation graphs are given, allowing to navigate through teams or individuals that share common interests (they may or may not be co-authors).

How to use LookinLabs ?

As a simplest use, just enter your query. For example, if you enter network security, the tool will look for best matching with respect to 'network', 'security', and the digram 'network-security', with a bias in favor of the latter. If you really mean the digram only, you can specify this by adding quotes: network security. Then, the tool will look for the digram only and stop in case of the matching succeeds. In case of failure, quotes are removed

and the tool operates in its basic mode. The tool claims to perform semantic matching.

If, however, by network security, you have in mind the whole domain with its bench of associated keywords, this implicit context is not known by LookinLabs since LookinLabs makes no use of any kind of ontology in its current release. You can still benefit from sort of an adaptive ontology by using the tool stepwise as follows: (1) Enter your query; (2) Read the abstracts of the very first publications returned by the tool and select the one(s) you find best for describing your topic; (3) Copy this abstract and enter it as your refined query. This yields a rich query that will act as an ontology by enhancing your topic with all the words of the abstract.

Some screenshots of the application are shown in Figures 4 and 5. Figure 4 shows the team profile, consisting of informations related to the team (topics covered). Figure 5 depicts the author's Social Graph (similar authors).

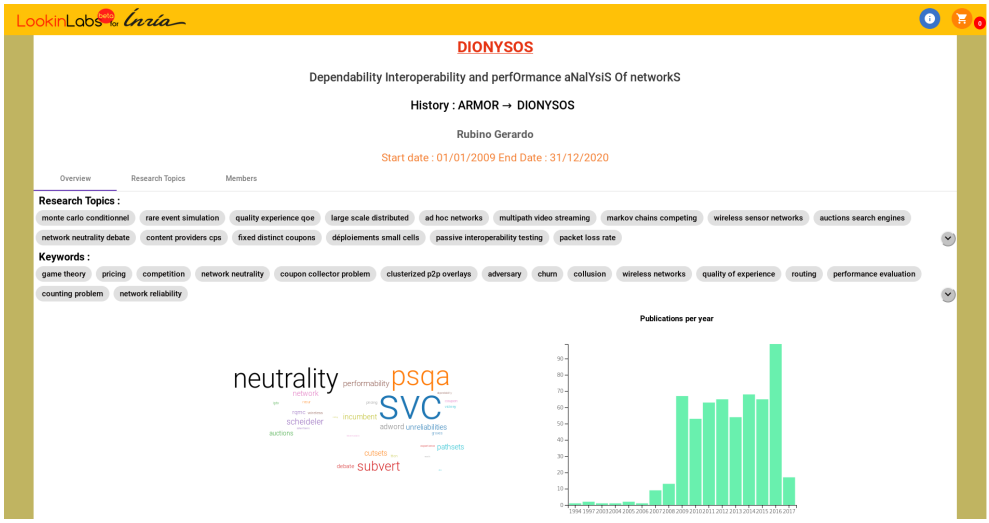


Figure 4: Team Profile

2.2 Making LookinLabs fully automatic

One strong requirement on LookinLabs is that it should not require any extra manual information. It should rather only rely on what can be found in the

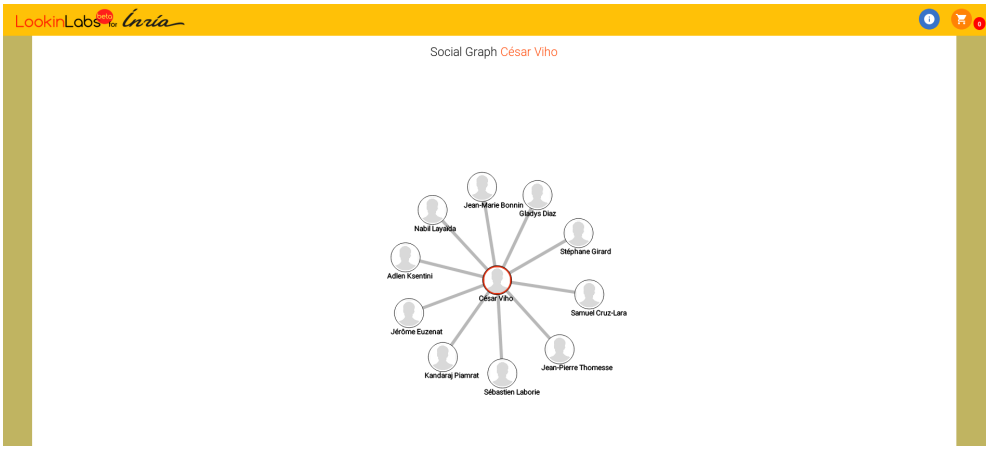


Figure 5: Author Social Graph

HAL archive, plus the reuse of some administrative information from existing information bases. No manual information or update is needed.

The keywords that are returned by LookinLabs to characterize the teams or individuals are not suggested by any kind of ontology. No ontology is actually used by LookinLabs. Such keywords combine those found in the meta-data of HAL documents with additional keywords synthesized by text mining techniques. The latter remain up-to-date and are not subject to obsolescence, in contrast to manually specified keywords.

Having the right list of teams and individuals is another difficulty. For LookinLabs dedicated to Inria, administrative information bases provided the needed information. LookinLabs benefits from the maintenance performed on those information bases, for other reasons. See Section 5.2 for details.

3 Related Work

Some related projects exist in France. We briefly review them now.

ScanR [7] is a search engine of the French ministry of education and research. It covers the whole set of French public and private labs, both academic and industrial (35,000 items). By searching over the web site of each institution, it selects and returns those mentioning the keywords of the query. General administrative information is returned, which is obtained by

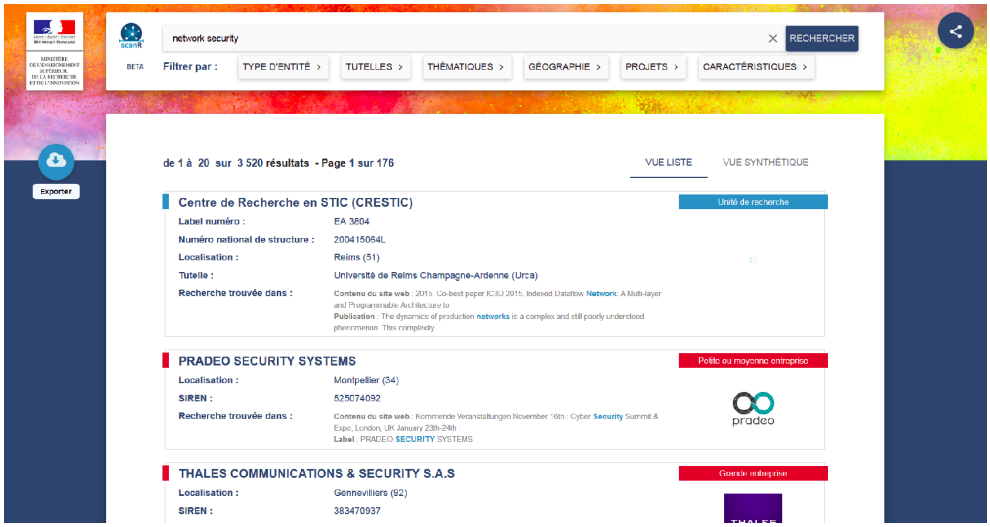


Figure 6: A screenshot of ScanR in response to the query *network security.*

the tool from open data sources. The service points the reader to the web sites of the institutions for further information. Figure 6 shows a screenshot of the return of ScanR for the query *network security*. The names of the returned institutions are hyperlinks to their web sites.

Plug in labs [16] is a similar effort undertaken by the regions Bretagne and Pays-de-Loire. When queried with *network security,* the tool returns two institutions. Figure 7 shows the information popped up by the tool for one of the two labs found by the tool. The informations displayed by this page were entered manually.

Gargentex [1] was developed under the umbrella of CNRS. The user must register and then sign in. Upon entering the query *network security* with HAL as archive, the service returns over 2,000 publications and provides some statistics (number per year,...). For this set of publications, some further services are provided. An analysis of the main words used in these publications (stopwords being listed but with a special flag) with their number of occurrences is returned. Also, co-occurrence graphs of words are provided, by which edges between nodes represent proximities of terms according to a specific distance between the documents; the conditional distance between the terms X and Y is the probability to have both terms X and Y in the same textual context; the distributional distance between the terms X and

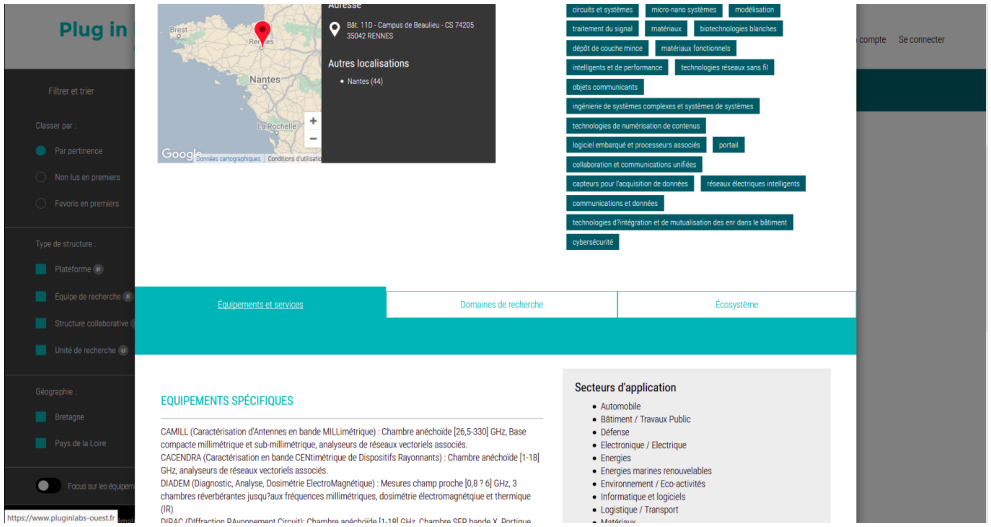


Figure 7: A screenshot of PlugInLabs in response to the query *network security.*

Y is the probability to have same others terms in the same textual context as X or Y. Returning this information takes a long waiting time (more than a few minutes).

Finally, AnHALytics [10] is a project aiming at creating an analytic platform for the HAL archive or other scientific Open Access repositories, exploring various analytic aspects such as search/discovery, activity and collaboration statistics, trend/technology maps, knowledge graph and data visualization (author's abstract).

4 Deployment of LookinLabs

In this section we review the status of deployment of LookinLabs to date March 2018.

4.1 Inria

LookinLabs4HALInria is the richest adaptation of LookinLabs4HAL by taking advantage of the following facts and data, that are unique to Inria:

- Inria is structured into teams only, with no other structuration like departments, etc. This makes it easier to return responses in terms of both individuals and teams.

- The web activity report of all Inria teams (nickname Raweb) provides a number of "certified" administrative information regarding the persons. For each Inria team, the list of members is given and the name of each member is guaranteed unique, with no variation throughout the report (this is checked by the web tool at the creation of the report). Each author of a HAL publication of the considered team is matched against the Raweb member list. This allows getting rid of the variations in the names.

- The BASTRI web site maintains the list of Inria teams together with a history of them (parent-child relations between teams, if any). This information is used by LookinLabs for pointing to a recent team, which otherwise would not appear in the responses due to a too small number of publications. Typically, a parent team is found by the tool and BASTRI is used to complement this information with the child recent team.

In addition, since both Raweb and BASTRI are maintained and updated by Inria, the link between Raweb, BASTRI, and LookinLabs allows for an automatic update of the administrative data. Updates are on-the-fly for BASTRI and annual for Raweb. In all cases, automatic annual updates of the administrative data can be established for LookinLabs. This makes LookinLabs4HALInria fully automatic, with no need for manual data loading. In relation with the DSI (Information Service Department of Inria), a wiki has been developed that would allow to hand over to Inria DSI the management of LookinLabs. The content of this wiki as well as the evaluation of the software development methodology are almost completed by the DSI.

4.2 Other labs partners of CominLabs

For other labs partners of CominLabs, who are not part of Inria, we cannot use the above complementary administrative data. Difficulties, therefore, arise regarding variations in the names of the authors and, mainly, the parent-child links between the different teams of the considered lab. Also, former researchers, who have left the lab, are no longer visible. This results in a

slight degradation in performance of the service regarding individuals, and a stronger degradation regarding teams. In turn, the labs are much smaller than Inria and, therefore, have a much smaller number of teams.

Dealing with labs of small or modest size

One specific difficulty arises when we deploy LookinLabs for a particular lab. The population of the lab is not large in general (from less than 100 to a few hundreds), which can be a problem for data mining algorithms. Three strategies can be considered when performing the first phase of the processing, namely the indexing of the publications data base:

1. Apply the data mining algorithms to the bibliographical data base collecting all the publications of the lab only.
2. Form the union of the populations of all labs of CominLabs at hand (including Inria in its whole, about 100,000 publications) and index it. This way, each publication is characterized by a set of tokens (keywords) that best identifies it in the context of the whole, large, data base. Then, a query is best matched to the individuals or teams that belong to the considered lab – other individuals or teams are filtered out.
3. Same, but take all publications of HAL, covering all topics (about 1,300,000 publications).

We chosed the first strategy because the response time was more than twice as long when we request data using strategies 2 or 3 and results returned by the three strategies were almost the same. The same issue arises when constructing the topics characterizing an individual or a team from a given lab. The same three strategies can be considered, with the same conclusions.

Making LookinLabs fully automatic

So far LookinLabs in its current form is not fully automatic, except for Inria. The reason is that the administrative information (correct list of individuals and teams) is generally not stored in a formal data base with clear API. The considered information was delivered to us manually, so we expect updates to be done manually. We believe this is a serious obstacle to the long term maintenance of the service. We believe it is essential to have it fully automatic (besides the maintenance of the software itself). We are also quite sure it can serve many other purposes for the lab.

4.3 Status of the deployment (march 2018)

LookinLabs4HAL is deployed for the laboratories partners of CominLab who have accepted a minimal set of conditions:

- Willingness to provide the list of lab members and teams;
- To have a person responsible for interacting with LookinLabs development team.

The links to the services deployed for the different labs are :

- Inria: https://lookinlabs4halinria.cominlabs.u-bretagneloire.fr
- Irisa: https://lookinlabs4halirisa.cominlabs.u-bretagneloire.fr
- LS2N: https://lookinlabs4halls2n.cominlabs.u-bretagneloire.fr
- Foton: https://lookinlabs4halfoton.cominlabs.u-bretagneloire.fr
- LTSI : https://lookinlabs4halltsi.cominlabs.u-bretagneloire.fr
- IETR: https://lookinlabs4halietr.cominlabs.u-bretagneloire.fr
- IMT/LabSticc: https://lookinlabs4halsticc.cominlabs.u-bretagneloire.fr

5 Service architecture

The overall architecture of the service is shown on Figure 8. We briefly list its components:

- Publications Database: the set of publications from the considered population (e.g., Inria);
- Administrative Information Base: the set of authors and teams from the considered population;
- Data Cleaning and Linking: Removing some unused fields and clean the remaining data for subsequent indexing of publications, authors and teams;
- Text Mining: we use various functionalities of the Elasticsearch tool [3];

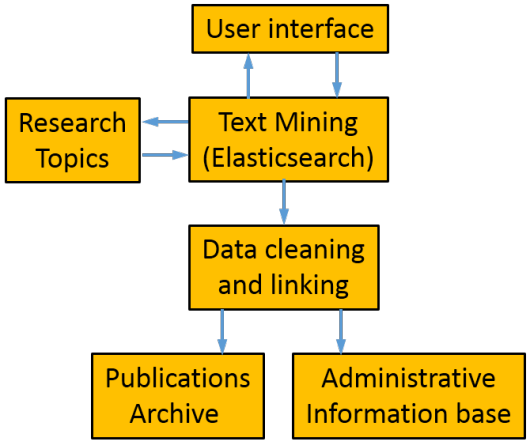


Figure 8: Components schema of LookinLabs.

- Topics Identification: This component uses text mining and natural langage algorithms to create topics for authors and teams;
- User Interface: to query the service and find publications, authors and teams.

The different components are detailed next.

5.1 Publications database

HAL [9] is the publication archive used by the application to get all the publications of the considered population. We use HAL in LookinLabs because it is an open archive and no restriction is applied to use its data and metadata. HAL stores datas in TEI format [15]. TEI is a standard format for representating texts in digital form. Following [15], the TEI *Guidelines for Electronic Text Encoding and Interchange* define and document a markup language for representing the structural, renditional, and conceptual features of texts. These guidelines are expressed as a modular, extensible XML schema, equipped with a detailed documentation, and are published under an open-source license.

Figures 9 and 10 show screenshots of the TEI taken from the list of authors and the list of affiliations, respectively. Corresponding data structures are used to store links between the publication, authors, and affiliations. Authors

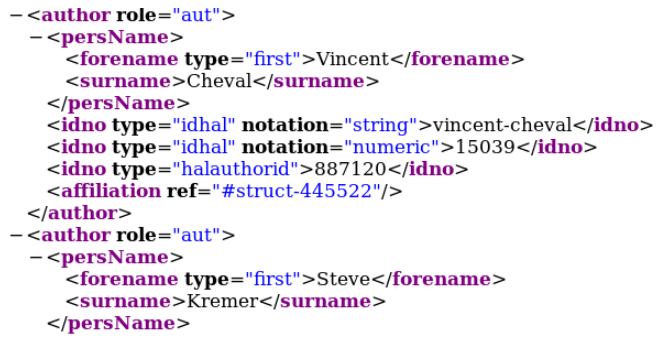


Figure 9: List of authors

Figure 10: List of affiliations

are linked to teams with the tag **affiliation** and the attribute **ref**. In the screenshot of **affiliation**, **xml:id** is equal to the attribute **ref** in Figure 9, so the author is linked to this research team. We see that this research team is linked to a laboratory (Inria Nancy - Grand Est) in Figure 10, and also to Inria. Inria will also appear in conjunction with other labs (e.g., Inria Rennes Bretagne Atlantique), thus suggesting a nesting between affiliations. Such a nesting can indeed be synthesized automatically.

This works well for Inria. Unfortunately, for most institutions, the variations on this nested structuring of the affiliations and the too often poor quality of their documentation result in wrong nesting. This procedure was tested but eventually not deployed.

5.2 Administrative Information Database

HAL meta-data, and particularly the affiliations and authors, are not well documented for most laboratories. Different laboratories use different attributes to store the same value in HAL (e.g., the acronym of the team may sit in different tags). Due to the poor quality in documenting meta-data in HAL, we complement the corresponding information by exploiting administrative information bases from the considered laboratory to create teams, authors, and the associated links. When HAL data will be documented with improved quality, the administrative information bases will become useless, except for history information (which team gives birth to which). Such historical data can also be used to know if researchers are still in the lab or team.

We deployed LookinLabs for different laboratories. Corresponding administrative information bases were different. For Inria, we used *Raweb* and *Bastri*. Raweb is a web site storing the yearly activity reports of all Inria teams. It is used to create the data base of authors. Authors can be crawled on the web site to get their name, team, and duration of stay in the successive teams. Bastri is a web site too. It is used to create the data base of teams. Teams are crawled to get their history, start and end dates. Thus, the administrative information displayed by Inria is rich. For IRISA, IETR, STICC, LS2N, LTSI, and FOTON labs, a CSV file is used to get names, team's authors and team's acronyms.

Administrative informations are not easy to get from most laboratories. For Inria, one has to crawl a web site. So, if the url changes or the web site structure changes, the part of LookinLabs code that deals with the crawling of administrative information must be modified. Another difficulty is to get informed about when informations are updated on the web site. For other labs, their administration had to create a CSV file and send it to us. These difficulties could be mitigated, if laboratories were publishing the date when their data get updated and were exposing a stable API to access these informations.

5.3 Data Cleaning and Linking

Data come from multiple sources, namely HAL and administrative information bases. For each publication, HAL provides under TEI some informations that are useless for our purpose. Relevant data for us are: title, abstract,

authors, date and affiliations. These informations are reformatted in JSON to prepare them for indexing.

Names of authors are subject to variations, particularly in case of special characters. HAL offers the possibility to associate a unique ID for each author. The OrcID (www.orcid.org) international ID can be used for this, or HAL itself offers to attach a unique ID to an author. This is, however, optional, and many authors do not use this possibility. We added a field to reformat author's names to mitigate variations in names.

5.4 Elasticsearch

We use Elasticsearch [3] for indexing data. Elasticsearch is a full text search engine, which implements and offers some useful algorithms for our use. Elasticsearch relies on Lucene [5].

Scoring

In Elasticsearch, fields are stored in columns and the data and meta-data attached to each publication are stored in rows. Elasticsearch computes the inverted indexes, consisting of: terms, term frequency and ID of the document [8], see Table 1.

term doc ID	network	security	test
doc1	term frequency	term frequency	term frequency
doc2	term frequency	term frequency	term frequency
doc3	term frequency	term frequency	term frequency

Table 1: Inverted index

Elasticsearch use terms frequency to compute the scores of the different entities (publications, teams, individuals), for a given query. The scoring formula is based on BM25 [11]:

$$\text{score}(D,Q) = \sum_{i=1}^{n} \text{IDF}(q_i) \frac{f(q_i,D)(k_1+1)}{f(q_i,D)+k_1\left(1-b+b\frac{|D|}{avgdl}\right)} \quad (1)$$

$$\text{IDF}(q_i) = \log \frac{N-n(q_i)+0.5}{n(q_i)+0.5} \quad (2)$$

In these formulas: $f(q_i, D)$ is the term frequency (tf/idf) in the document D; $|D|$ is the length of the document and $avgdl$ is the average length of the document; N is the number of documents in the data base and $n(q_i)$ is the number of documents containing q_i. In (1), the parameters for tuning are:

- k_1: This parameter controls how quickly an increase in the term frequency results in term-frequency saturation. The default value is 1.2. Lower values result in quicker saturation, and higher values in slower saturation.
- b: This parameter controls the weighting of a keyword according to the length of the paragraph that contains it. Its default value is 0.75.

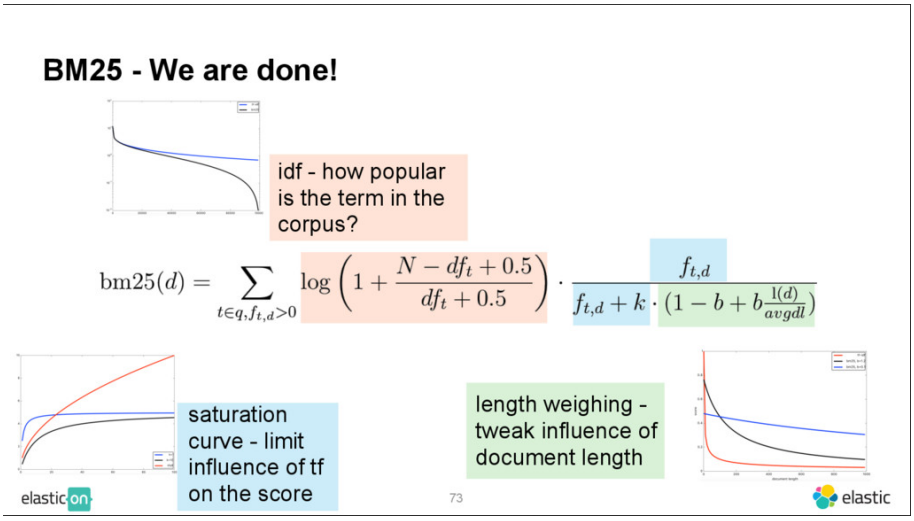


Figure 11: Sensitivity of different subterms of the right hand side of (1) when the parameters vary.

The effect of these two parameters on different subterms of the right hand side of (1) is explained on Figure 11.

BM25 is considered to be a state-of-the-art ranking function [19]. We use the default values for the parameters in LookinLabs. BM25 give less importance to fequently used words and more to words with lower frequency in the whole document data base. To compute the scores of the keywords for

a given query, BM25 use term frequency, inverse document frequency, and field-length normalization [17].

The comparison between tf/idf and BM25 is shown in Figure 12.

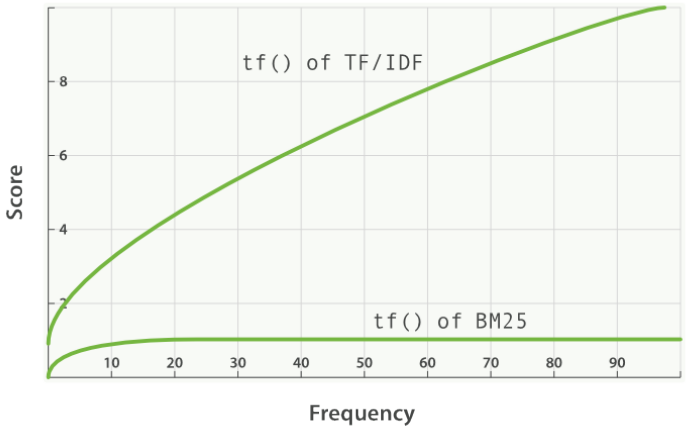


Figure 12: Difference between tf/idf and BM25

In the following subsections we document some specific features of how we use Elasticsearch and what else beyond indexing we obtain from it.

Special data structure

Data are stored in JSON format. For teams and authors, associated publications are directly attached to them, instead of being linked. This way of doing avoids performing joins when querying the database, and thus makes the response to queries much faster. Data of a team or an author are handled the same way.

Using analyzers

We use analyzers when we index the data in Elasticsearch. Analyzers are used to remove stop words and create bigrams/trigrams. To create bigrams, the analyzer proceeds as follows:

1. The entire text is converted to lowercase, to avoid distinguishing between lower/upper case writing of a word;

2. For French, elisions (l', qu', d', etc.) are removed [4];
3. Bigrams are created from the text and added in the inverted index as explained next on a small example.

The text *This is a bigram analyzer* is split in the following bigrams: *this is*, *is a*, *a bigram*, and *bigram analyzer*. All bigrams are stored. Clearly, all but the last one are irrelevant. However, we do not expect these bigrams to take place in a real life query. Using the analyzers, we can, in a response to a query that contains a bigram or a trigram, boost publications containing these bigrams or trigrams (with comparison to unigrams). Finally, analyzers used on the field *keywords*, manually entered by the authors in HAL, perform a cleaning and some normalization.

Boosting

Heterogeneous fields (abstract, title, keywords, and then bigrams and trigrams) are involved in the indexing and ranking tasks. Therefore the question arises of how to weight them to assign them the right value. We use analyzers to boost bigrams and trigrams. We finally used the following weights: abstract 1, title 4, keyword 10, bigram 50, trigram 100. It should be acknowledged that the tuning of these weights is quite robust: the actual rankings that result do not vary much.

Profiling

Profiling consists in characterizing teams or individuals with an ordered list of keywords synthesized from the mining of their publications in the context of the whole set of publications (we take the standpoint that author's keywords are not part of their profile). To this end we use the JLH function [12]. JLH is a function $term \mapsto score$, associated to a given subset of publications (typically, all publications of a team or all publications of an individual):

$$JLH = \begin{cases} (P_{fore} - P_{back})\frac{P_{fore}}{P_{back}} & if\ P_{fore} - P_{back} > 0 \\ 0 & otherwise \end{cases}$$

In this formula, $P_{fore}(term)$ is the frequency of *term* in the *foreground* (the considered subset) and $P_{back}(term)$ is the frequency of *term* in the *background* (whole set of publications).

The following is quoted verbatim from [12]: The numbers returned for scores are primarily intended for ranking different suggestions sensibly rather than something easily understood by end users. The scores are derived from the doc frequencies in foreground and background sets. In brief, a term is considered significant if there is a noticeable difference in the frequency in which a term appears in the subset and in the background.

The so obtained profiling is returned to the user as a word cloud. It is also used to enrich the set of authors's keywords.

Social graph

To get the *social graph* (graph of similar teams or individuals) Elasticsearch gets the terms with highest tf/idf of the input documents (set of publications of the individual or team). Then Elasticsearch queries the data base with this set of terms seen as a disjunctive query to retrieve similar authors or teams [2].

5.5 Research Topics

The *research topics* are generated with the help of Elasticsearch and NLTK (a Natural Language Toolkit) [14]. NLTK is a module written in Python for natural language processing. *Research topics* for teams or authors are obtained via the steps shown in the procedure of Figure 13, implemented by the algorithm of Figure 15. As an illustration we apply this algorithm to the text *We will get best colocations trigrams*. The result is shown in Figure 14.

5.6 User Interface

We use Angular [6] for the front-end. Angular is a framework that helps to create web applications. The interface is used by the end user to query the database and get authors, teams, and publications best correlated with the query. The user can query the database by keywords or text. The interface can display the profile of authors and teams and their similarity with others authors and teams. The interface is displayed differently in small resolution screen because we cannot display teams, authors and publications at the same time. The end user must choose one of them. nt.

1. Get the full set of publications for the considered entity (team or individual), consisting of abstracts and titles;
2. Remove stop words: the list found is given to Elasticsearch for removal;
3. Get bigrams with NLTK, keep the 200 best bigrams (ordered by their number of occurences, and bigrams occurring less than 3 times are ignored)
4. Get *significant words* from Elasticsearch using the procedure of Section 5.4 with the JLH score;
5. Filter the list of bigrams by removing the bigrams not containing any *significant word*;
6. Get trigrams with NLTK, keep the 150 best trigrams (ordered by their number of occurences, and trigrams occurring less than 2 times are ignored)
7. Filter the list of trigrams by removing the trigrams not containing any bigram obtained at step 5;
8. Remove duplicate or too close trigrams with the help of the *Levenshtein distance* [18].

Figure 13: Obtaining the *research topics* for a team or individual.

6 Future work

For some individuals or teams, different areas are covered (this is, for instance, the case for Albert Benveniste himself, with activities in signal processing, vibration mechanics, embedded systems design and programming, and web services). Currently, the topics characterizing such individuals or teams are organized as an ordered, but otherwise flat list, where topics related to different areas are interleaved. It would be desirable to structure topics according to semantic clusters instead. This is a task under investigation, with the following algorithms being experimented: K-means, Lingo, STC (Suffix Tree Clustering) [13].

So far LookinLabs makes no use of any kind of ontology. Our design choice was motivated by the principle of having no manual data to be entered in the service (ontologies are, in their classical forms, manually defined). However, the progresses made around Wikipedia have made some sorts of standard

1. Removing stop words yields: *get best colocations trigrams*;
2. Get bigrams: [('best','colocations'), ('colocations','trigrams'), ('get','best')]
3. Submitting this to *significant words* from Elasticsearch yields: 'trigrams';
4. Filter the list of bigrams by significant words yields one single bigram: [('colocations','trigrams')]
5. Getting trigrams yields: [('get','best','colocations'),('best','colocations','trigrams')]
6. Filtering trigrams by the list of bigrams leaves us with a single trigram: ('best','colocations','trigrams')
7. Removing duplicate trigrams and trigrams too close with the Levenshtein distance does nothing.

Figure 14: Example of the procedure of Figure 13 on the text *We will get best colocations trigrams*.

ontologies widely available. The most well-known instance is DBPedia, an ontology attached to Wikipedia. We can thus envision further cleaning our sets of keywords or topics by confronting them with such standard ontologies. Preliminary experiments have been undertaken. Final results are yet to come. An alternative approach would be to apply existing algorithms that build automatically some kind of ontology for a given data base of publications, e.g., by providing, for a given keyword, a set of neighboring keywords best correlated to it. An example of such an algorithm is Word2Vec.[1]

LookinLabs makes extensive use of Elasticsearch and is developed under Angular. These two tools are subject to updates, which impacts LookinLabs and thus requests a maintenance of the service to keep proper alignme

7 Discussion and conclusions

Our main finding is that bibliographical data bases (HAL for France, DBLP, IEEE Explore, Arxiv, etc) are probably not sufficient by themselves to deliver a service like LookinLabs with sufficient quality. The reason is the lack

[1] https://en.wikipedia.org/wiki/Word2vec

of quality in the metadata (authors and affiliations). We are aware that search tools exist that chase homonyms regarding individuals, but their robustness is still not satisfactory. Futhermore, they do not apply to groups of researchers such as teams or labs. A major difficulty is thus the need to synchronize the bibliographical data bases with administrative data bases, whenever available. Such data bases are, by definition, not universal. In our use cases, only Inria was able to offer the needed information in the due form.

Our initial plans included the extension of LookinLabs to HAL in its whole. The use of HAL was made mandatory by the ANR (Agence Nationale de la Recherche) and the AERES (the body in charge of evaluating universities, institutes, and labs, nationwide). But no clear specification of the metadata to be entered by the authors exists. We actually doubt that such a clear specification can be performed and then enforced, given the wide variability of teams or labs in their size, scope, and life, throughout the French research community. Therefore, the extension to HAL in its whole remains a challenge. It could become doable if progresses are made in our data mining algorithms and/or techniques to deal with homonyms in individuals or the variability of the nature of teams or labs.

The last question is: how far is our experience with LookinLabs reproducible in other contexts? We could envision large companies (the LookinLabs design team has ties with Orange, Safran, and SAP and some preliminary discussions were held). One difficulty is the nature of document data bases where the technical or scientific information sits. For academic researchers, we could rely on publications. In large companies; however, a large part of such information is not found in well structured texts, but rather in slidewares with lots of graphical material, not amenable of data mining techniques as we used. Maybe major consultant companies (Arthur Andersen, Mercer, etc) could be relevant targets, as these companies produce a huge amount of reports for their customers, stored in highly structured and labeled data bases. In such cases, however, issues of strict security and confidentality would occur, which we did not face in our case.

References

[1] CNRS. Gargantext. https://testing.gargantext.org/.

[2] Elastic Co. More like this query. https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-mlt-query.html.

[3] Elastic. Elasticsearch. https://www.elastic.co/products/elasticsearch.

[4] Elastic. Elision token filter. https://www.elastic.co/guide/en/elasticsearch/reference/current/analysis-elision-tokenfilter.html.

[5] The Apache Software Foundation. Lucene. https://lucene.apache.org/.

[6] Google. Angular. https://angular.io/.

[7] Gouvernement. Scanr. https://scanr.enseignementsup-recherche.gouv.fr/.

[8] Adrien Grand. Algorithms and data-structures that power lucene and elasticsearch. https://berlinbuzzwords.de/15/session/algorithms-and-data-structures-power-lucene-and-elasticsearch.

[9] HAL. open archive hal. https://hal.archives-ouvertes.fr/.

[10] INRIA. Anhalytics. https://github.com/anHALytics/anhalytics-core.

[11] Lucene. Bm25. https://www.elastic.co/guide/en/elasticsearch/guide/master/pluggable-similarites.html.

[12] Andr Lynum. How elasticsearch calculates significant terms. http://blog.comperiosearch.com/blog/2015/06/10/how-elasticsearch-calculates-significant-terms/.

[13] Prof. Neha Soni Pragna Makwana. Analysis and comparison of web document clustering algorithms with lingo. *International Journal of Engineering Research and Technology (IJERT)*, 2, 2013.

[14] NLTK Project. Nltk. http://www.nltk.org/.

[15] TEI. Text encoding initiative. http://www.tei-c.org/index.xml.

[16] UBL. Plug in labs. https://www.pluginlabs-ouest.fr/.

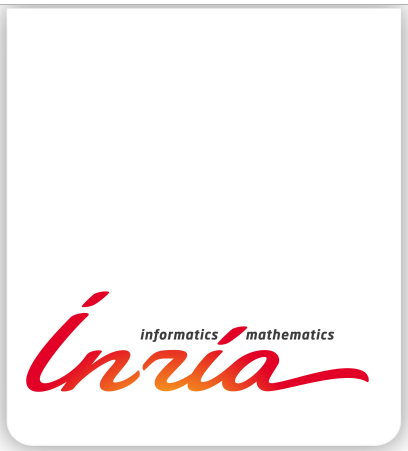
[17] Britta Weber. Bm25 demeystified. https://www.elastic.co/elasticon/conf/2016/sf/improved-text-scoring-with-bm25.

[18] Wikipedia. Levenshtein distance. https://en.wikipedia.org/wiki/Levenshtein_distance.

[19] Wikipedia. Okapi bm25. https://en.wikipedia.org/wiki/Okapi_BM25.

```
Step 1 : Get full text
D ← input documents
for d ∈ D do
    T ← concate abstract and title
end
Step 2 : Remove stop words
T_wst ← remove stop words from T with Elasticsearch analyzer
Step 3 : Bigrams
L_b ← use NLTK on T_wst to get a list of best bigrams with frequency > 3
Step 4 : Significants words
M_st ← get most significant terms with Elastisearch (JHL score)
Step 5 : Remove duplicate bigrams and filter by significant terms
L_bc ← list
for b ∈ L_b do
    if b not in L_bc and inverse(b) not in L_bc then
        L_bc ← add(b)
    end
end
L_bcf ← list
for b ∈ L_bc do
    if unigram b in M_st then
        L_bcf ← add(b)
    end
end
Step 6 : Trigrams
L_t ← use NLTK on T_wst to get a list of best trigrams with frequency > 2
Step 7 : Filter Trigrams L_tc ← list
for t ∈ L_t do
    if bigrams t in L_bcf then
        L_tc ← add(t)
    end
end
for t ∈ L_t do
    if t in L_bcf then
        L_tcf ← map trigrams with the bigram
    end
end
Step 8 : Remove duplicate and trigrams too close
for b, L_t ∈ L_tcf do
    bigram, L_t : list trigrams for t in L_t do
        S_t ← split(t) split trigrams in unigram
        for ts in S_t do
            if ts not in b then
                L_nb ← ts
            end
        end
        if len(L_nb) > 1 then
            for t_o ∈ L_nb do
                for t_t ∈ L_nb do
                    if distanceLevensthein(t_o, t_t) ≥ 3 then
                        L_t ← add to b the longest words and add to list
                    end
                end
            end
        else
            L_t ← add t
        end
    end
end
L_tc ← remove trigrams with duplicate bigram
```

Figure 15: The algorithm implementing the procedure of Figure 13.



RESEARCH CENTRE
RENNES – BRETAGNE ATLANTIQUE

Campus universitaire de Beaulieu

35042 Rennes Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399