UNIVERSITY OF TARTU

FACULTY OF SCIENCE AND TECHNOLOGY

INSTITUTE OF COMPUTER SCIENCE

SOFTWARE ENGINEERING CURRICULUM

KHACHATUR HAMBARDZUMYAN

IDEAS MATCHMAKING FOR SUPPORTING INNOVATORS AND ENTREPRENEURS

Master's Thesis

Supervisors:

Fabrizio Maria Maggi, PhD, University of Tartu

Massimo Mecella, PhD, Sapienza – Università di Roma

Francesco Leotta, PhD, Sapienza – Università di Roma

Tartu 2016

Ideas Matchmaking for Supporting Innovators and Entrepreneurs

Abstract

In this paper we show a system able to crawl content from the Web related to entrepreneurship and technology, to be matched with ideas proposed by users in the InnovVoice platform. We argue that such a service is a valuable component of an ideabator platform, supporting innovators and possible entrepreneurs.

Keywords

Ideabator, crawling, text classification, keyword extraction

CERCS: P170

Ideede sidumine toetamaks uuendajaid ja ettevõtjaid

Lühikokkuvõte:

Käesolevas töös esitletakse süsteemi, mis on võimeline roomama veebist ettevõtluse ja tehnoloogiaga seotud andmeid, mida saab siduda kasutajate poolt InnovVoice platvormil välja pakutud ideedega. Selline teenus on ideabator platvormi väärtuslik osa, mis toetab ettevõtluse uuendajaid ja potentsiaalseid ettevõtjaid.

Võtmesõnad:

Ideabator, crawling, teksti liigitus, märksõna kaevandamise

CERCS: P170

Contents

1. Introduction

Generally speaking, “innovation is the application of better solutions that meet new requirements, unarticulated needs, or existing market needs” [1]; “this is accomplished through more-effective products, processes, services, technologies, or business models that are readily available to markets, governments and society. Therefore, the term “innovation" can be defined as something original and more effective and, as a consequence, new, that “breaks into" the market or society” [2]; innovation is generally considered to be the result of a process that brings together various novel ideas in a way that they have an impact on society. In business and economics, in particular, innovation can be a catalyst to growth. All organizations can potentially innovate, including for example hospitals, universities, and local governments. There are several sources of innovation: it can occur as a result of a focus effort by a range of different agents, by chance, or as a result of a major system failure. According to [3], “the general sources of innovations are different changes in industry structure, in market structure, in local and global demographics, in human perception, mood and meaning, in the amount of already available scientific knowledge, etc”.

As [4] states, “Another source of innovation, only now becoming widely recognized, is end-user innovation. This is where an agent (person or company) develops an innovative solution for their own (personal or in-house) use because existing products do not meet their needs”.

Today innovation is achieved in many ways, with much attention now given to formal research and development (R&D) for “breakthrough innovations". The more radical and revolutionary innovations tend to emerge from R&D, while more incremental innovations may emerge from practice - but there are many exceptions to each of these trends. ICT and changing business processes and management style can produce a work climate favorable to innovation. Notable examples are the one of Atlassian, which conducts quarterly “ShipIt Days"2 in which employees may work on anything related to the company's products, or Google employees that work on their own projects for 20% of their time (known as Innovation Time Off).

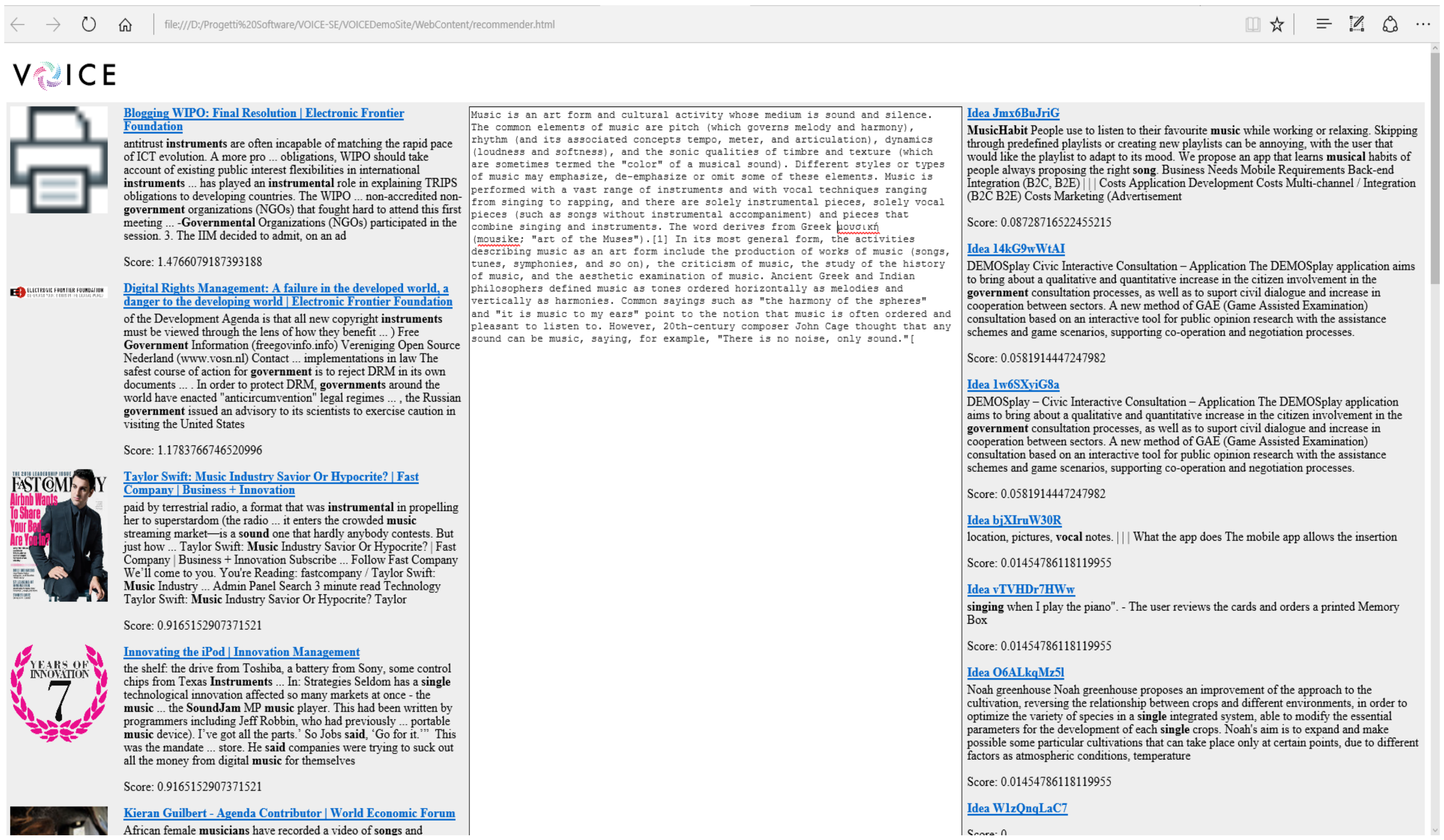
Nowadays, user-innovators may also choose to freely reveal their innovations, using methods like open source. In all these models, yet quite debated, two specific points emerge: (i) the importance of the ideation phase, and (ii) the enabling effect that innovation, and ideation in particular, can have on businesses and economics.

How can users validate their innovative ideas? How can they find people with the same entrepreneurial spirit?

In this context the InnovVoice ecosystem funded by EU commission (http://www.innovvoice.com) aims at empowering the crowd to innovate and prosper, by facilitating business idea development, enriching concepts, fostering partnerships, synergies and collaborations, in order to create a vibrant entrepreneurial community.

This can be achieved through the concept of ideabator, i.e., and incubator of ideas. Through the platform, ideation is supported, by helping users in conceiving, expressing, cooperating, validating and improving their ideas. This can be achieved by providing, among the many other functionalities offered by the platform, also a set of technologies allowing users to continuously check if the idea they are elaborating is somehow already present in the Web. As an example, the reader can consider the following scenario (see Figure 1): the user logins, starts describing his idea, sees in real-time relevant information retrieved and elaborated by the system about relevant webpages, blogs, news related to the proposed idea, which is refined as the writing goes on, and can decide if it is worthy to go on or change her business proposition. The user has his individual “Google of ideas".

In the following of this paper, after briefly introducing the InnovVoice platform, we describe some technical details on how to realize support to innovation and entrepreneurship, and we advocate that this use of Big Data technologies, namely crawling, Open APIs, NLP and text mining, combined in our specific setting, are an interesting example of data-driven innovation.



file:///D:/Progetti%20Software/VOICE-SE/VOICEDemoSite/WebContent/recommender.html

VOICE

[Blogging WIPO: Final Resolution | Electronic Frontier Foundation]

antitrust **instruments** are often incapable of matching the rapid pace of ICT evolution. A more pro ... obligations, WIPO should take account of existing public interest flexibilities in international **instruments** ... has played an **instrumental** role in explaining TRIPS obligations to developing countries. The WIPO ... non-accredited non-**government** organizations (NGOs) that fought hard to attend this first meeting ... -**Governmental** Organizations (NGOs) participated in the session. 3. The IIM decided to admit, on an ad

Score: 1.4766079187393188

[Digital Rights Management: A failure in the developed world, a danger to the developing world | Electronic Frontier Foundation]

of the Development Agenda is that all new copyright **instruments** must be viewed through the lens of how they benefit ...) Free **Government** Information (freegovinfo.info) Vereniging Open Source Nederland (www.vosn.nl) Contact ... implementations in law The safest course of action for **government** is to reject DRM in its own documents ... . In order to protect DRM, **governments** around the world have enacted "anticircumvention" legal regimes ... , the Russian **government** issued an advisory to its scientists to exercise caution in visiting the United States

Score: 1.1783766746520996

[Taylor Swift: Music Industry Savior Or Hypocrite? | Fast Company | Business + Innovation]

paid by terrestrial radio, a format that was **instrumental** in propelling her to superstardom (the radio ... it enters the crowded **music** streaming market—is a **sound** one that hardly anybody contests. But just how ... Taylor Swift: **Music** Industry Savior Or Hypocrite? | Fast Company | Business + Innovation Subscribe ... Follow Fast Company We'll come to you. You're Reading: fastcompany / Taylor Swift: **Music** Industry ... Admin Panel Search 3 minute read Technology Taylor Swift: **Music** Industry Savior Or Hypocrite? Taylor

Score: 0.9165152907371521

[Innovating the iPod | Innovation Management]

the shelf: the drive from Toshiba, a battery from Sony, some control chips from Texas **Instruments** ... In: Strategies Seldom has a **single** technological innovation affected so many markets at once - the **music** ... the **SoundJam** MP **music** player. This had been written by programmers including Jeff Robbin, who had previously ... portable **music** device). I've got all the parts.' So Jobs **said**, 'Go for it.'" This was the mandate ... store. He **said** companies were trying to suck out all the money from digital **music** for themselves

Score: 0.9165152907371521

[Kieran Guilbert - Agenda Contributor | World Economic Forum]

African female **musicians** have recorded a video of **songs** and

```
Music is an art form and cultural activity whose medium is sound and silence.
The common elements of music are pitch (which governs melody and harmony),
rhythm (and its associated concepts tempo, meter, and articulation), dynamics
(loudness and softness), and the sonic qualities of timbre and texture (which
are sometimes termed the "color" of a musical sound). Different styles or types
of music may emphasize, de-emphasize or omit some of these elements. Music is
performed with a vast range of instruments and with vocal techniques ranging
from singing to rapping, and there are solely instrumental pieces, solely vocal
pieces (such as songs without instrumental accompaniment) and pieces that
combine singing and instruments. The word derives from Greek μουσική
(mousike; "art of the Muses").[1] In its most general form, the activities
describing music as an art form include the production of works of music (songs,
tunes, symphonies, and so on), the criticism of music, the study of the history
of music, and the aesthetic examination of music. Ancient Greek and Indian
philosophers defined music as tones ordered horizontally as melodies and
vertically as harmonies. Common sayings such as "the harmony of the spheres"
and "it is music to my ears" point to the notion that music is often ordered and
pleasant to listen to. However, 20th-century composer John Cage thought that any
sound can be music, saying, for example, "There is no noise, only sound."[
```

[Idea Jmx6BuJriG]

MusicHabit People use to listen to their favourite **music** while working or relaxing. Skipping through predefined playlists or creating new playlists can be annoying, with the user that would like the playlist to adapt to its mood. We propose an app that learns **musical** habits of people always proposing the right **song**. Business Needs Mobile Requirements Back-end Integration (B2C, B2E) | | | Costs Application Development Costs Multi-channel / Integration (B2C B2E) Costs Marketing (Advertisement

Score: 0.08728716522455215

[Idea 14kG9wWtAI]

DEMOSplay Civic Interactive Consultation – Application The DEMOSplay application aims to bring about a qualitative and quantitative increase in the citizen involvement in the **government** consultation processes, as well as to suport civil dialogue and increase in cooperation between sectors. A new method of GAE (Game Assisted Examination) consultation based on an interactive tool for public opinion research with the assistance schemes and game scenarios, supporting co-operation and negotiation processes.

Score: 0.0581914447247982

[Idea 1w6SXyiG8a]

DEMOSplay – Civic Interactive Consultation – Application The DEMOSplay application aims to bring about a qualitative and quantitative increase in the citizen involvement in the **government** consultation processes, as well as to suport civil dialogue and increase in cooperation between sectors. A new method of GAE (Game Assisted Examination) consultation based on an interactive tool for public opinion research with the assistance schemes and game scenarios, supporting co-operation and negotiation processes.

Score: 0.0581914447247982

[Idea bjXIruW30R]

location, pictures, **vocal** notes. | | | What the app does The mobile app allows the insertion

Score: 0.01454786118119955

[Idea vTVHDr7HWw]

singing when I play the piano". - The user reviews the cards and orders a printed Memory Box

Score: 0.01454786118119955

[Idea O6ALkqMz5l]

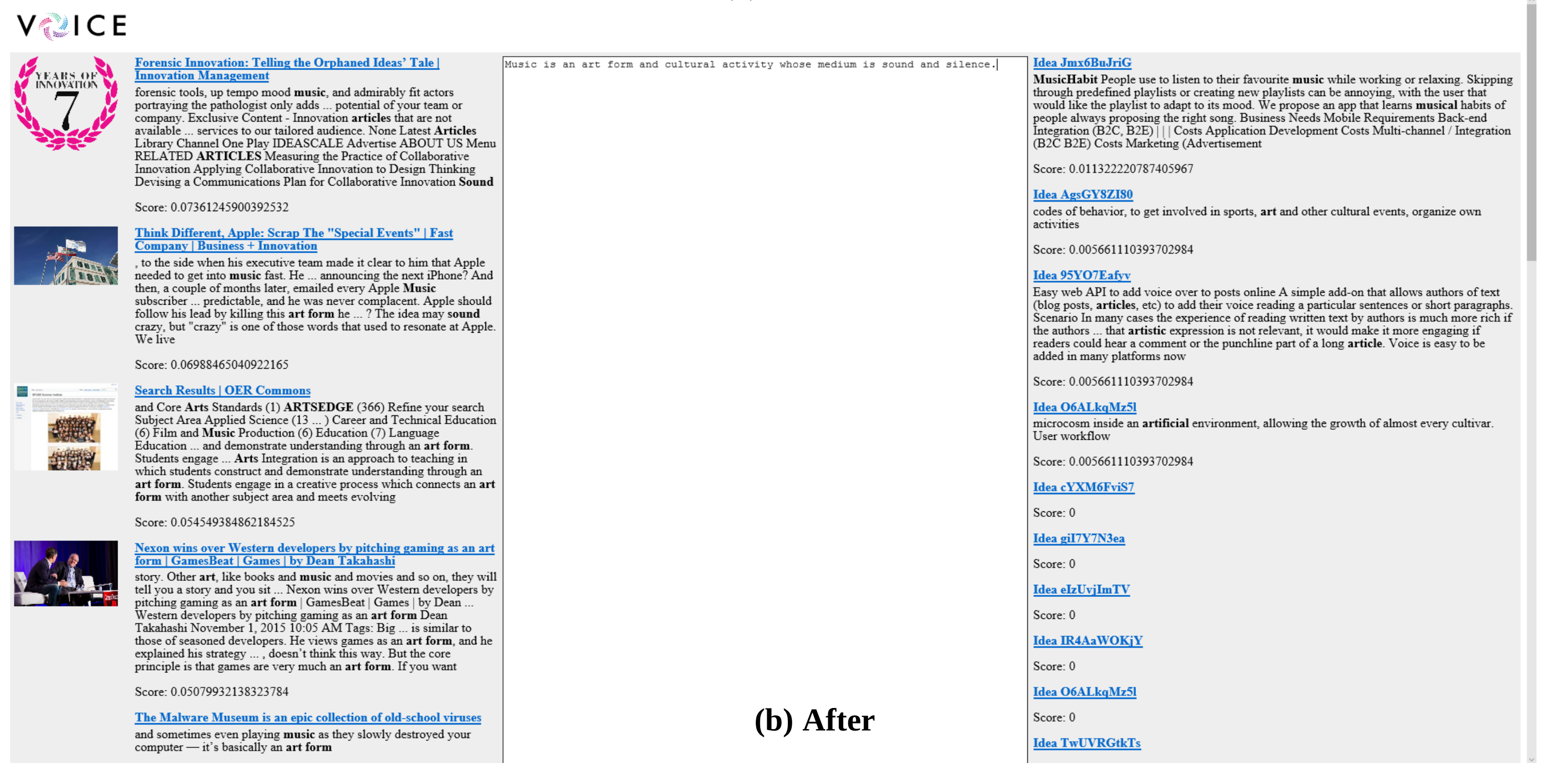
Noah greenhouse Noah greenhouse proposes an improvement of the approach to the cultivation, reversing the relationship between crops and different environments, in order to optimize the variety of species in a **single** integrated system, able to modify the essential parameters for the development of each **single** crops. Noah's aim is to expand and make possible some particular cultivations that can take place only at certain points, due to different factors as atmospheric conditions, temperature

Score: 0.01454786118119955

[Idea W1zQnqLaC7]

Score: 0

(a) Before



file:///D:/Progetti%20Software/VOICE-SE/VOICEDemoSite/WebContent/recommender.html

VOICE

[Forensic Innovation: Telling the Orphaned Ideas' Tale | Innovation Management]

forensic tools, up tempo mood **music**, and admirably fit actors portraying the pathologist only adds ... potential of your team or company. Exclusive Content - Innovation **articles** that are not available ... services to our tailored audience. None Latest **Articles** Library Channel One Play IDEASCALE Advertise ABOUT US Menu RELATED **ARTICLES** Measuring the Practice of Collaborative Innovation Applying Collaborative Innovation to Design Thinking Devising a Communications Plan for Collaborative Innovation **Sound**

Score: 0.07361245900392532

[Think Different, Apple: Scrap The "Special Events" | Fast Company | Business + Innovation]

, to the side when his executive team made it clear to him that Apple needed to get into **music** fast. He ... announcing the next iPhone? And then, a couple of months later, emailed every Apple **Music** subscriber ... predictable, and he was never complacent. Apple should follow his lead by killing this **art form** he ... ? The idea may **sound** crazy, but "crazy" is one of those words that used to resonate at Apple. We live

Score: 0.06988465040922165

[Search Results | OER Commons]

and Core **Arts** Standards (1) **ARTSEDGE** (366) Refine your search Subject Area Applied Science (13 ...) Career and Technical Education (6) Film and **Music** Production (6) Education (7) Language Education ... and demonstrate understanding through an **art form**. Students engage ... **Arts** Integration is an approach to teaching in which students construct and demonstrate understanding through an **art form**. Students engage in a creative process which connects an **art form** with another subject area and meets evolving

Score: 0.054549384862184525

[Nexon wins over Western developers by pitching gaming as an art form | GamesBeat | Games | by Dean Takahashi]

story. Other **art**, like books and **music** and movies and so on, they will tell you a story and you sit ... Nexon wins over Western developers by pitching gaming as an **art form** | GamesBeat | Games | by Dean ... Western developers by pitching gaming as an **art form** Dean Takahashi November 1, 2015 10:05 AM Tags: Big ... is similar to those of seasoned developers. He views games as an **art form**, and he explained his strategy ... , doesn't think this way. But the core principle is that games are very much an **art form**. If you want

Score: 0.05079932138323784

[The Malware Museum is an epic collection of old-school viruses]

and sometimes even playing **music** as they slowly destroyed your computer — it's basically an **art form**

```
Music is an art form and cultural activity whose medium is sound and silence.
```

[Idea Jmx6BuJriG]

MusicHabit People use to listen to their favourite **music** while working or relaxing. Skipping through predefined playlists or creating new playlists can be annoying, with the user that would like the playlist to adapt to its mood. We propose an app that learns **musical** habits of people always proposing the right song. Business Needs Mobile Requirements Back-end Integration (B2C, B2E) | | | Costs Application Development Costs Multi-channel / Integration (B2C B2E) Costs Marketing (Advertisement

Score: 0.011322220787405967

[Idea AgsGY8ZI80]

codes of behavior, to get involved in sports, **art** and other cultural events, organize own activities

Score: 0.005661110393702984

[Idea 95YO7Eafyv]

Easy web API to add voice over to posts online A simple add-on that allows authors of text (blog posts, **articles**, etc) to add their voice reading a particular sentences or short paragraphs. Scenario In many cases the experience of reading written text by authors is much more rich if the authors ... that **artistic** expression is not relevant, it would make it more engaging if readers could hear a comment or the punchline part of a long **article**. Voice is easy to be added in many platforms now

Score: 0.005661110393702984

[Idea O6ALkqMz5l]

microcosm inside an **artificial** environment, allowing the growth of almost every cultivar. User workflow

Score: 0.005661110393702984

[Idea cYXM6FviS7]

Score: 0

[Idea giI7Y7N3ea]

Score: 0

[Idea eIzUvjImTV]

Score: 0

[Idea IR4AaWOKjY]

Score: 0

[Idea O6ALkqMz5l]

Score: 0

[Idea TwUVRGtkTs]

(b) After

Figure 1: An InnovVoice user getting feedback on his idea. While he is writing, relevant content is matched and proposed on the basis of the content he is interested (note the differences in the provided content between the screenshots, due to the additional text provided in the second one).

2. State Of the Art

Since in this thesis we focus on matchmaking techniques we can review several platforms that have similar behavior in information crawling and result matching.

2.1 DesigNET

One of the projects is DesigNET (http://www.designet-italy.it/)[5]. It is a research and innovation project, commissioned by Italian companies in interior design, furniture, architecture and services that aims at investigating and developing Web-based e-business prototype supporting selling activities, by identifying, on the basis of the analysis of the Web, design market trends and new contract opportunities.

The purpose of the DesigNET project is to study and to develop a Web-based e-business prototype providing a decision support system regarding product development and selling activities by identifying, on the basis of the analysis of the Web, design market trends and new contract opportunities.
The information gathered using the platform (design trends, macroeconomics indexes and new contract opportunities, etc.) has value for the companies involved in the project. It gives opportunity for companies to discover highly qualified user interests and to predict future market trends and products evolution in different countries and adapt and offer products according to the new trends.
The platform focuses to improve prediction systems in order to apply them also in Small and medium-sized enterprises that cannot afford really expensive and cumbersome systems presently available on the market.

Architecture: The system has been organized into two main components:

- The spider, responsible for retrieving information available over the Internet accessing a variety of Web sources to collect information about contract opportunities, economic outlook, and design trends. This component can be seen as an integration layer that reads and parses heterogeneous Web sources, and stores the retrieved information in a relational DBMS.
- The business intelligence (BI), responsible for connecting to the spider database, perform the ETL process (extract, reorganize and historicize retrieved information) and store data in a data warehouse optimized to quickly return to users queries through the use of materialized views and indexes. Furthermore, it provides a Web-based user interface that displays information and analysis outcomes through a number of charts.

The whole system is based on a client-server interaction, where each component acts as a client when receiving information from actors in a higher level of the chain, and act as a server when providing information to clients in a lower level of the chain. Figure 2 shows how components and external actors interact each other and its data flow.

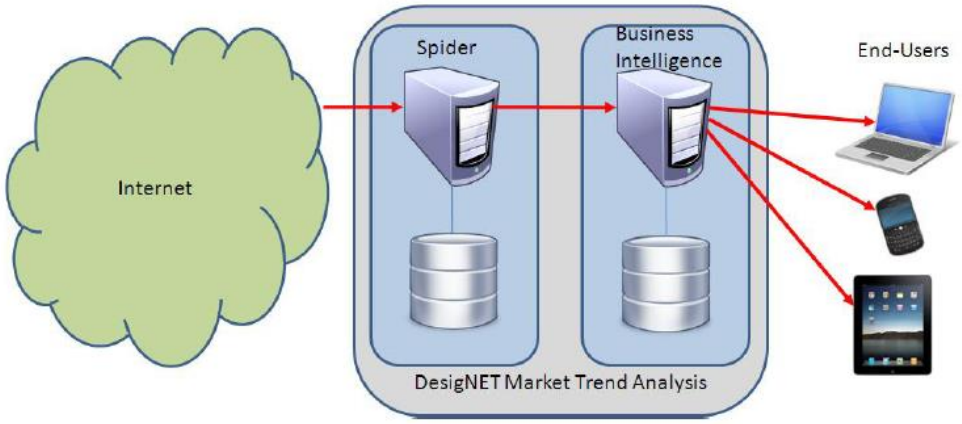


Fig.2

As shown in Figure, each component is characterized by different software modules, communicating each other or isolated, which perform the individual subtask that assemble any single service.

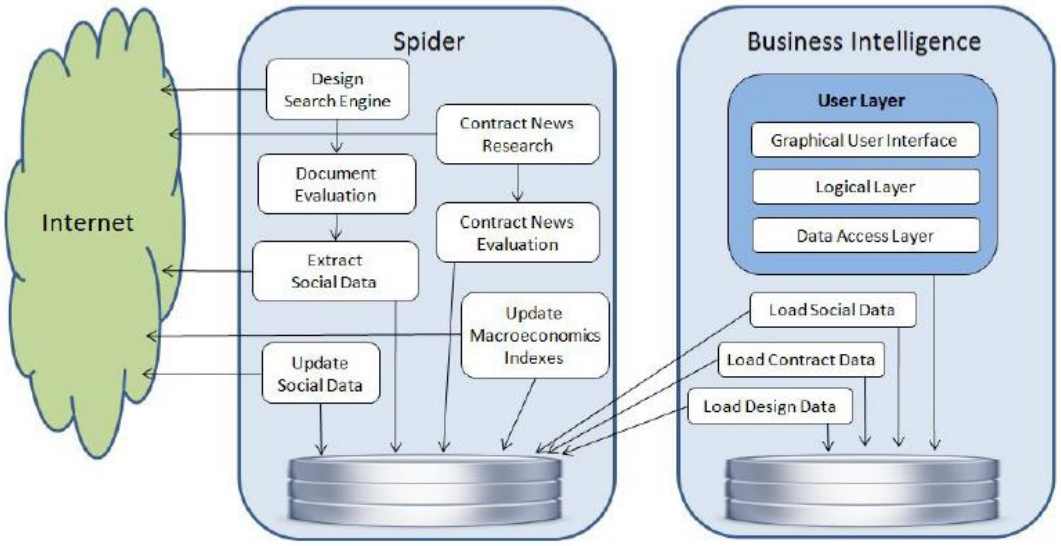


Fig. 3

The Spider Component
The Spider component is devoted to searching for new sources, analyzing and retrieving relevant information. Given the high quantity of data needed to achieve a statistical analysis, it runs at full speed 24/7 to analyze as many documents as possible. Regarding the macroeconomics indexes, system collects them with wrappers specific to the monitored sites and stores the information in a relational database. The wrapper, before updating macroeconomics indexes, checks if the specific source is actually online. Then, it checks if the downloaded page has changed since last visit, and just in case, proceeds with the information extraction and to updates the database. To retrieve new contract opportunities, the Spider accesses to web sites specialized on news regarding worldwide contract opportunities. Once documents have been retrieved, the system proceeds with documents analysis and categorization. Documents present information in a semi-structured form (unstructured text embedded in structured HTML template). In this case, to maximize the chance to properly classify the document, the parser navigates the HTML tree to extract the plain text news and then document classification is performed (using a Boolean Model and a thesauri). The system traverses HTML tree to reach the plain text, and then categorizes the document, according to a simple Boolean model, as relevant if at least one occurrence of the context thesaurus has been found. To increase performance, geographic reference categorization is performed only if the document is relevant. In case the document is relevant, it is saved and occurrence metadata useful for ranking results I shown in the user interface.

The Business Intelligence component is responsible for connecting to the spider database, perform the ETL process and store data in a data warehouse for subsequent analysis and visualization.

If we draw parallels with our platform we can see that it has similar crawling part, however the web spiders are different and configured uniquely to satisfy domain specific needs.

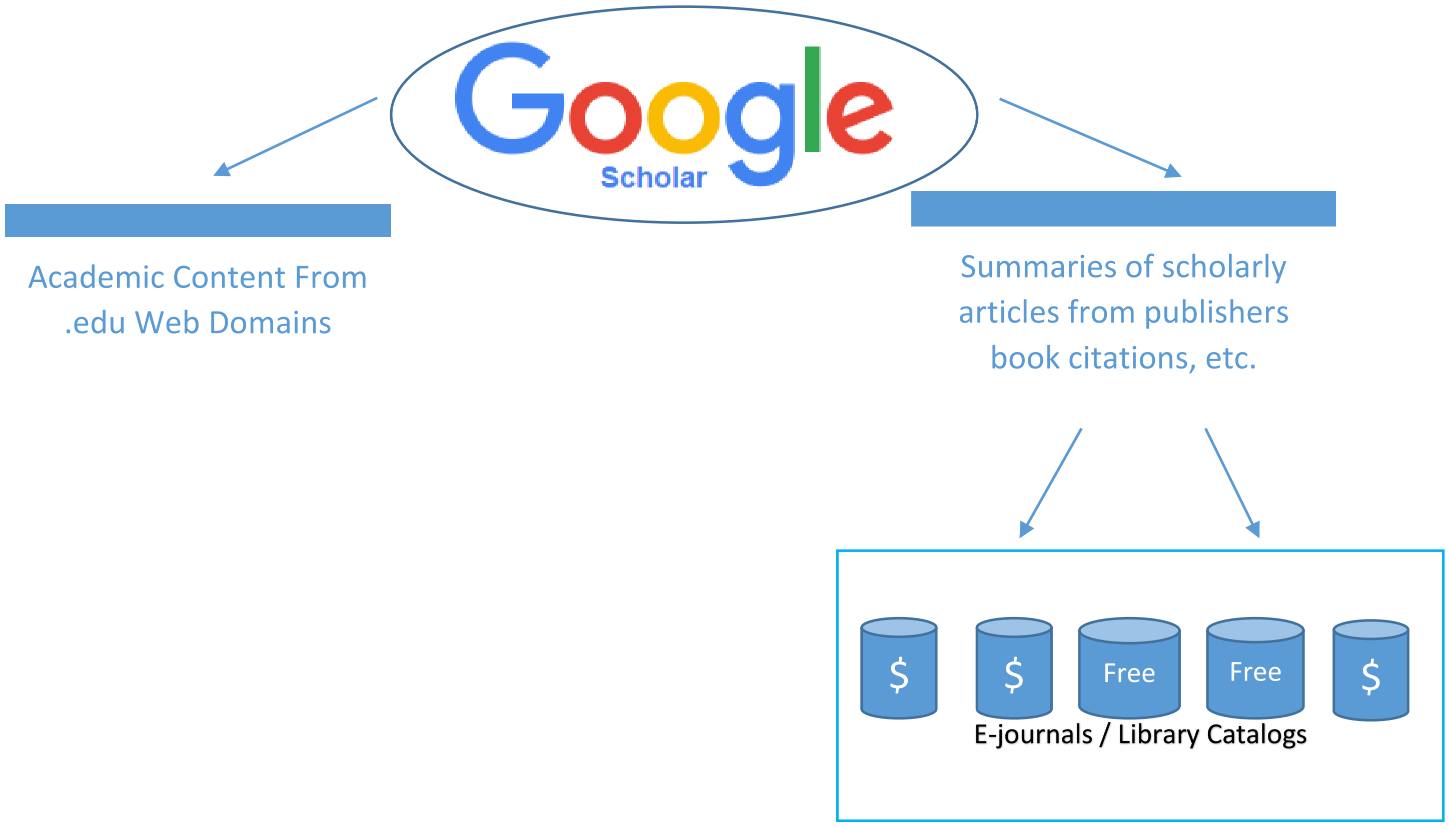
2.2 Google Scholar

Another system that can be of particular interest is well known Google Scholar (https://scholar.google.com/) which is a search engine for academic resources in all subject fields. Google Scholar web spider searches content in peer-reviewed journal literature, books, dissertations, academic society papers and technical reports.
Google Scholar describes its scope and content generally, and, unlike the major science and technology bibliographic databases (PubMed, Cambridge Scientific Abstracts, etc.), the search engine does not provide any source lists of publications searched or authority files for author names, journal titles, or controlled vocabulary for subjects. The search engine retrieves a large number of documents in a very short time.

Search Features

Simple Search: The Simple Search is a powerful tool in a number of ways. It automatically supports both Boolean and truncation operators. Instead of truncation symbols, Google Scholar uses *word stemming* algorithm, which returns documents with word variations based on specified keywords. For instance, a search on *stemming word* will retrieve documents with *stemming* or *stem* and *word* or *words*.

Advanced Search: The Google Scholar Advanced Search offers a number of search options for articles. It supports keyword and author searching and enables the user to restrict results published within a range of years, by name of publication, and by subject area. Keyword searching is more sophisticated than the Simple Search. It includes searching by **all words**, **exact phrase**, **at least one of the words**, **without the words**, and **where** the words occur in the document.


Google Scholar
Academic Content From .edu Web Domains
Summaries of scholarly articles from publishers book citations, etc.
$
$
Free
Free
$
E-journals / Library Catalogs

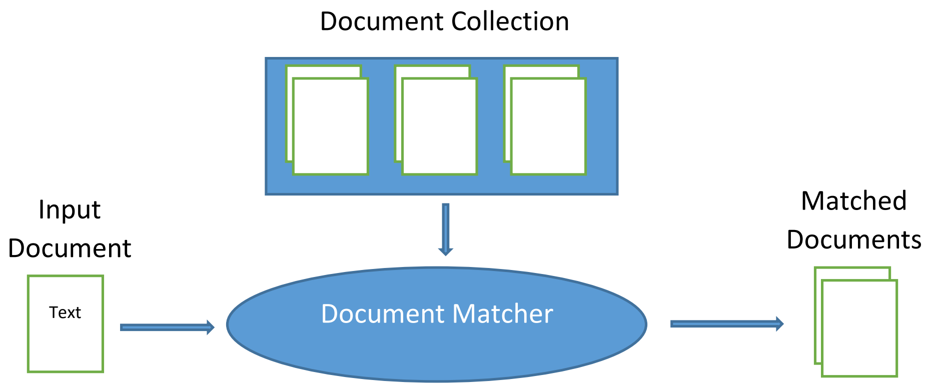
Google Scholar offers different search options for free academic resources on the Web. Its fast search engine and wide output results are compromise that must be considered considering against accuracy and preciseness in a literature search.

The main difference between the search engines mentioned here and our matchmaking system is that the number of keywords of query is limited for them, whereas our system allows to search by entire document of any length. There are plenty of platforms with their own crawling and search functions. The two above mentioned appeared to us more relevant. In our case we have “Google of Ideas”.

3. Background

3.1 Information Retrieval

[6] The general task of information retrieval is to return the documents matching the clues provided by us as a query.



Those clues represent keywords that help us to retrieve corresponding documents. In a typical case of information retrieval a few words are provided to a search engine, which are matched to the stored documents. The best matches are returned by the engine. We can generalize this process to a document matcher. In this case an entire document can be represented as a set of keywords. The document given as an input is matched to all stored documents, and the system retrieves the most relevant documents.

The main idea for information retrieval is assessing similarity between two documents. Even a query of a few number of words to a search engine is treated as a document that can be matched to others. The common theme is measuring similarity, and variations of these methods are fundamental to information retrieval. The data can be represented as a spreadsheet, and this model can easily be used for these tasks. The new document corresponds to a new row. The new row is compared to all the other rows, and the most similar rows and documents associated to them are the answers.

In order to mine a text, it is first necessary to preprocess and bring it to a special structure that can be used by data-mining tools. As noted before, this generally includes transforming features in a spreadsheet representation. Traditional data mining works with data that is highly organized.
We consider text mining unstructured due to the fact that it is far away from the spreadsheet representation that is required to process data for prediction. Yet, the transformation contents of the document from text to the spreadsheet representation can be highly methodical, and there is a carefully designed process to complete the spreadsheet cells. The first step is to come up with the nature of spreadsheet columns (i.e., the features). Some of the features can be easily obtained, such

as word occurrence in a text, others are much more difficult to determine, for example, the grammatical function of a word in a sentence (whether it is a subject or an object, etc.).

Collecting Documents: It's obvious that in text mining number one task is to gather the data. In a lot of cases, they may already be given or may be found in the problem description itself. For instance, if we have a Web page search engine for an intranet, it means that the relevant documents are the Web pages stored on the intranet. After the documents are determined, they can be fetched, and the major problem is to clean them in order to make sure that they have high quality. In some cases, the data may be acquired from document warehouses or databases. In these cases, it implies that data preprocessing was done before storing and documents are of the high quality.

In certain applications, a data collecting systems may be required. For example, for a Web application that consists of several autonomous Web sites, a software tool such as a Web spider can be deployed that acquires all the data. In some other scenarios, a logger application deployed on an input data stream can perform logging of data. Such an example is an application that performs email audit. It is able register all incoming and outgoing messages at a mail server and store in the log.

It happens that the number of documents is immensely large, and depending on the task data-sampling approaches can be used to select a set of relevant documents that can be handled. For example, documents may have a time stamp, and those that are more recent may be more relevant.

Another resource that we can take into account is the World Wide Web itself. Web spiders can produce collections of pages from a particular website, or on a particular topic. Given the size of the Web, collections produced this way can be huge. The key issue with this method is that the data might be ambiguous and need to be cleaned before it can be useful.

Standardization: The documents that are collected can be found in different formats. It depends on how these documents were generated. For instance, some of them may be created by a word processor with its own format (.docx, .pdf, etc.), others may be created in a simple text editor and stored as plain text and finally, there may be documents that are scanned and stored as image files. Obviously, for processing all those documents, it's more convenient to convert them to a common standard format.

Document standardization is very important. The main advantage is that the text mining applications can process them without considering how the document is generated. For collecting information from a document, it is not important at all what editor was used to create it or what format it had originally. Text mining tools should process documents just in a single standard format, and not in the many different formats they were generated initially.

Tokenization: Let's assume we have a set of documents stored in some format and we need to analyze the text to find out useful features. The first task in processing the text is to break it into words (tokenize). This is important for further analysis. Without tokenization, it will be hard task to mine higher level information from the text. Every token represents an instance of a type, thus the amount of tokens is much higher than the amount of types.

Tokenize a text is an easy task for someone who is familiar with the language structure. For the computer program, on the other hand, it is quite challenging and complicated task. The reason is the difficulty to identify delimiters. They can be different in different contexts. We consider characters space, tab, and newline to be delimiters and never be tokens. These characters are known as white space characters. The characters () <>!?" are always considered delimiters but may be tokens as well. Depending on the context the characters . : , ' - may or may not be delimiters.

Lemmatization: Once text is divided into a set of tokens, the next feasible stage is to bring all the tokens to a standard representation. This process is called stemming or lemmatization. This step usually depends on the application. It can be beneficial for example in document classification process. One of the positive outcomes of stemming is the reduction of the amount of separate types in a document and to make the frequency of occurrence of some particular types higher. For example, if we have several instances of token "plays", after stemming they will be transformed to token "play" and will be considered as instances of that type, along with instances of the tokens "play" and "played." In document classification where frequency is important, stemming can sometimes make a difference.

Inflectional Stemming: In many languages words appear in text in more than one form. It's obvious that the nouns "pens" and "pen" are two distinct forms of the same word. In many cases it is beneficial to normalize both words to the single form "pen". When we normalize the words in grammatical types such as present/past and singular/plural, it is called "inflectional stemming". In linguistics, this process is called morphological analysis. For an English language that has many irregular word forms and nonintuitive spelling, stemming is a very tough task.

3.2 Automatic Keyword Extraction

Automatic keyword extraction is the procedure to find a group of words (i.e. keyphrases or keywords) from a document that carry the meaning of the document [7]. And depending on the model, the extraction should be performed with either minimal or no human intervention. The aim of automatic keyword extraction is to use the power and speed of computation in challenges of access and discoverability, improving information organization and retrieval without essential costs and defects related to human indexers [8].
The manual procedure of keyword extraction is slow, expensive and bristling with mistakes. Therefore, there is a need in algorithms and systems that help people to perform automatic keyword extraction. Existing methods can be categorized in 4 groups:

1. simple statistics,
2. linguistics,
3. machine learning,
4. mixed approaches [8, 9].

Simple Statistics Methods: These approaches are simple and don’t have many requirements. Training data is not required either. They do not consider linguistic features of the text and instead examine frequency of term, position of a keyword in text, etc.. The statistical data of words collected from the document can be used to identify the keyphrases. The advantage of solely statistical methods is their simplicity of use and the fact that in most cases they produce good results.

Linguistics Methods: In these approaches linguistic features of the words, sentences and document are used. Methods which take into consideration linguistic features of the words (e.g. part-of-speech, syntactic structure and semantic qualities) are improving results, in some cases helping to avoid bad keywords. The use of linguistic features indicate the great improvement of the automatic keyword extraction. Some of these approaches are mixed methods, combining together some linguistic approaches and common statistical measures such as term frequency and inverse document frequency.

Machine Learning Methods: The machine learning mechanism works as follows. First a set of training documents is provided to the system, each of which has a range of human-chosen keywords as well. Then based on the gained knowledge a model is created which system uses to find keywords from new documents.[10]

Mixed Methods: Other methods of automatic keyword extraction are mostly combination of the previously discussed methods or apply some heuristic information in the application of keyword extraction (e.g. position, length, layout feature of the words, html tags around of the words, etc. [11].).

At present available tools for automatic keyword extraction require either training data or domain specific knowledge.

3.3 Crawlers

A classical kind of offline extraction tool is represented by web crawlers (also known as web spiders). A Web crawler is a program or component of search engine which repeatedly browses the Internet, in order to store browsed content in the database. Search engines and some online services use Web spider for updating their web content or indices of others websites' content. They are able to fetch all the pages they visit for later processing by a search engine which does indexation of the downloaded data. This enables users to do their search much more efficiently. In order to start Web crawler, we need to provide a list of URLs that it needs to visit. This list called the seeds. While the crawler is on specific website that is listed in the seeds file, it finds all the hyperlinks in the page and adds them to the URL list to visit. The list is called crawl frontier. Web crawler recursively visits the URLs from the frontier according to a set of defined rules.

If the crawler is doing archiving operation of websites, it copies and stores the data as it goes [12]. Due to the huge size of data only limited number of the Web pages can be fetched within a certain time frame. This means that downloads should be prioritized.
The way a Web crawler behaves during operation depends on the combination of rules or policies [13]:

- a selection rule which states the pages to download,
- a re-visit rule which determines the interval of checking for changes to the pages,
- a politeness policy that tells how to avoid overloading of server being crawled,
- a parallelization policy that states how to manage distributed (multiple) web spiders.

Crawlers are capable to fetch data much faster and in greater depth than human searchers are capable, which means that they can have a serious impact on the performance of a website. And it is evident, that if a single web crawler is sending a big number of requests in small period of time or fetching big files, a server would have problems with handling those requests from multiple crawlers.

Web crawlers are useful for a lot of tasks, but they are very demanding and costly in terms of resources. The costs of using Web crawlers include [14]:

- network resources, as crawlers need significant bandwidth and run with a high level of parallelism during a long time period;
- webserver overload, particularly if the frequency of requests to a given website is very high;
- badly implemented crawlers, which can lead servers or routers to crash, or which fetch data they cannot process;

This problem can be partially solved by using the robots exclusion protocol, also called robots.txt protocol that is a standard for crawlers. This file should be stored in the root directory of the web server (www.example.com/robots.txt). It contains information indicating which pages of the website should or should not be accessed by crawlers [15; 16]. Recently this standard started to include a restriction for the interval of visits to the same server, a special parameter called "Crawl-delay:" in robots.txt, which tells crawlers the time in seconds of delay between consecutive requests. This is the most effective way to avoid server being overloaded.
A big number of web pages are stored in the deep or invisible web. These pages are usually only reachable by querying data from the database, and regular spiders are not able to find them if there are no links that point to those pages.
Deep web crawling increases many times the amount of web links that should be crawled. Some crawlers only take some of the URLs in `<a href="URL">` form. In some cases, such as the Googlebot, Web crawling is done on all text contained inside the hypertext content, tags, or text.
The obstacles that may not allow web crawler to perform the operation may be divided into following categories [17]:

- Dynamic content: webpages which server returns as a response to a sent request or pages that are available only through a form, particularly if open-domain input elements (e.g. text fields) are used; those kind of fields are hard to process without proper domain understanding.

- Unlinked content: pages which have no links on other pages, which may prevent web crawlers from accessing the content. They are known as pages that have no backlinks. In addition, search engines do not always manage to identify all backlinks from processed pages.
- Private pages: sites that ask for registration and authorization to access the content (i.e. password-protected web sites).
- Contextual Web: pages that contain data which is not the same for different access conditions (e.g., ranges of client IP addresses or navigation order).
- Content with limited access: sites that restrict access to their pages in a technical way (e.g., using the Robots.txt exclusion protocol or CAPTCHAs, or nostore directive which restricts search engines to process the pages and save their cached copies).
- Scripted content: pages that can be only accessed through links generated using JavaScript as well as data dynamically downloaded from sites using Flash or Ajax technologies.
- Non-text content: textual data encoded in multimedia files, such as images or video, or other file formats that search engines are unable to maintain.
- Software: some specific data is purposely hidden from the ordinary web. It can be available using only specific applications (e.g. Tor, FAI, FreeNet, or similar darknet applications).
- Web archives: Web archival services enable users to see archived versions of websites across time, including sites which are not reachable anymore, and are not indexed by search engines.

3.4 Main Technologies Used in InnoVoice Platform

3.4.1 Apache Nutch [18]

Apache Nutch is a highly open source web spider software tool. Nutch is pluggable and consists of different modules. It has an extensible interfaces such as Parse, Index and scoring filters for custom implementations.

In the scope of VOICE project, it is used for crawling a set of websites that contain valuable information for users which is later stored in MongoDB. A separate package is responsible for creating indices from the database (Apache Lucene). Also it should filter content before storing into the database. The sources usually contain a lot of external and internal links which are not useful.

The main task was to call the Nutch crawler from the code with our own defined settings instead of running it separately from the console. It made the system more autonomous. Next, invent some kind of mechanism to focus only on the interesting contents of these sources. As mentioned before Nutch is very flexible in terms of configurations and allows usage of different 3rd party plugins. And another task is to find suitable one to make crawling more effective and fast by omitting unnecessary information on web sites. Nutch uses also regular expressions for filtering and we can limit the crawl to a specific domain or a special pattern. However, for using with different sources it is more difficult, because there not might a common pattern for them.

3.4.2 MAUI

As reported in [19], “Maui is able to automatically detect main topics in text documents. Depending on the task, topics can be tags, keywords, keyphrases, vocabulary terms, descriptors, index terms or titles of Wikipedia articles”.
Maui has the following features [19]:

- “term assignment with a controlled vocabulary (or thesaurus)
- subject indexing
- topic indexing with terms from Wikipedia
- keyphrase extraction
- terminology extraction
- automatic tagging
- It can also be used for terminology extraction and semi-automatic topic indexing.”

MAUI consists of several other projects. It includes following software in it:

Kea [19, 20]: According to [19] “Maui builds on the keyphrase extraction algorithm (Kea). In that it utilizes the two-step process of automatic indexing: candidate selection and filtering. Major parts of Kea became parts of Maui without any further modifications. Other parts, like feature computation, were extended with new elements.”

Weka: As written in [16] "Maui inherits from Kea the machine learning toolkit Weka for creating the topic indexing model from documents with topics assigned by people and applying it to new documents. However, while Kea only contains a cut-down version of Weka (several classes), Maui includes the complete library. This gives more opportunities to experienced users for tailoring Maui’s code to specific data sets”.

Jena [16]: According to [16] “In order to make Maui applicable for topic indexing with many kinds of controlled vocabularies, the Jena library is included. It reads RDF-formatted thesauri and stores them in memory for a quick access. Any vocabulary in RDF format (specifically SKOS) can be used.”

Currently MAUI is used in VOICE project. It requires data to be trained before it can be applied to the actual document. So depending on how well it is trained the output quality may vary a lot.

3.4.3 MongoDB [21]

MongoDB is a cross-platform document-oriented NoSQL database. Unlike the traditional table-based relational database structure MongoDB is based on JSON-like documents with dynamic schemas (the format is called BSON). Due to this fact in some types of applications the data integration can be done faster and simpler. This software is free and comes with open source code.
Platform uses this database to store crawled content from sources for later indexing it using Apache Lucene.

3.4.4 Apache Lucene [22]

Apache Lucene is used in VOICE system for indexing crawled content that is stored in MongoDB and later for providing results against our queries. It is a highly productive text search engine implemented completely in Java. This library is capable to suit nearly any application that uses full-text search. It is platform independent.

The tool is suitable for any task which requires full text indexing and searching capability. Apache Lucene is very powerful tool if used in the implementation of Internet search engines and local, single-site searching. The main idea behind the logical architecture of the tool is a document that contains fields of text, which allows Lucene's API to get rid of the file format constraint. Any type of document that contains searchable text (PDF, HTML, MS Word, OpenDocument, etc., (not images)) can be indexed as long as their textual information can be extracted.

Features (as mentioned in [22]):

- "ranked searching - best results returned first
- many powerful query types: phrase queries, wildcard queries, proximity queries, range queries, etc.
- fielded searching (e.g. title, author, contents)
- sorting by any field
- multiple-index searching with merged results
- allows simultaneous update and searching
- flexible faceting, highlighting, joins and result grouping
- fast, memory-efficient and typo-tolerant suggesters
- pluggable ranking models, including the Vector Space Model and Okapi BM25
- configurable storage engine (codecs)"

4 InnoVoice System Architecture

4.1 Introducing Voice Platform

InnovVoice is a unified combination of (i) a social media platform, (ii) an idea management platform, (iii) a collaboration platform, and (iv) a market place, with in addition (v) a content management system, and (vi) a Web observatory [9]. In particular, the specific features offered by the platform are training and mentorship services, crowd-evaluation of ideas and prototypes (scorecards, structured/free form questionnaires, idea/product summarization), innovation exposure (through expo rooms and an innovation map). This is supported through a rich toolset, including a Web observatory, a content library (consisting of both internal content, i.e., generated internally by the platform through its users, and external one, i.e., retrieved from the Web), and matchmaking techniques. The platform mainly consists of three subsystems, namely Voice Central (VCEN), Voice Content (VCON), and Voice Observatory (VOBS). VCEN is in charge of managing all the data and the application logic of the platform (users/ideas/comments/etc. management and persistence), and the user interaction; VCON is indeed our sub-system managing external content (i.e., crawled from the Web) and providing the matchmaking services; details on the VOBS can be found in [8].

VOICE Content can be broadly defined as pieces and collections of data and information that have a particular value to the users of the VOICE platform and the community in general. In VOICE, ideas are considered the most important class of contents. However, contents can be obtained from users providing their experience and thoughts in the form of suggestions and feedback, thus exploiting the so called "wisdom of the crowd". Information coming from users can be further processed in order to automate the extraction of metrics, which can be considered a form of derived contents. Another relevant source for content is represented by the internet. In particular content repositories and external services can be queried to extract information related to entrepreneurship. Additionally, as the realization of a project implies the involvement of people and companies, the platform will give to different kinds of professionals and companies the possibility to offer their services.

4.2 The Matchmaking architecture

Figure 2 shows the components of the InnovVoice platform that are in charge of managing the matchmaking (VCON). As previously introduced, in the overall platform VCEN is the sub-system managing the internal content (profiles of users, inserted ideas and related posts, etc.). As the figure shows, the VCON sub-system operates on both internal and external content through specific interfaces and components. Moreover, the VCON sub-system provides APIs (as RESTful services) to the remaining of the platform in order to be invoked for matchmaking (e.g., by the Web user interface, when presenting to the user relevant content matching to what she is writing, see the previous discussion). The sub-system is organized in layers:

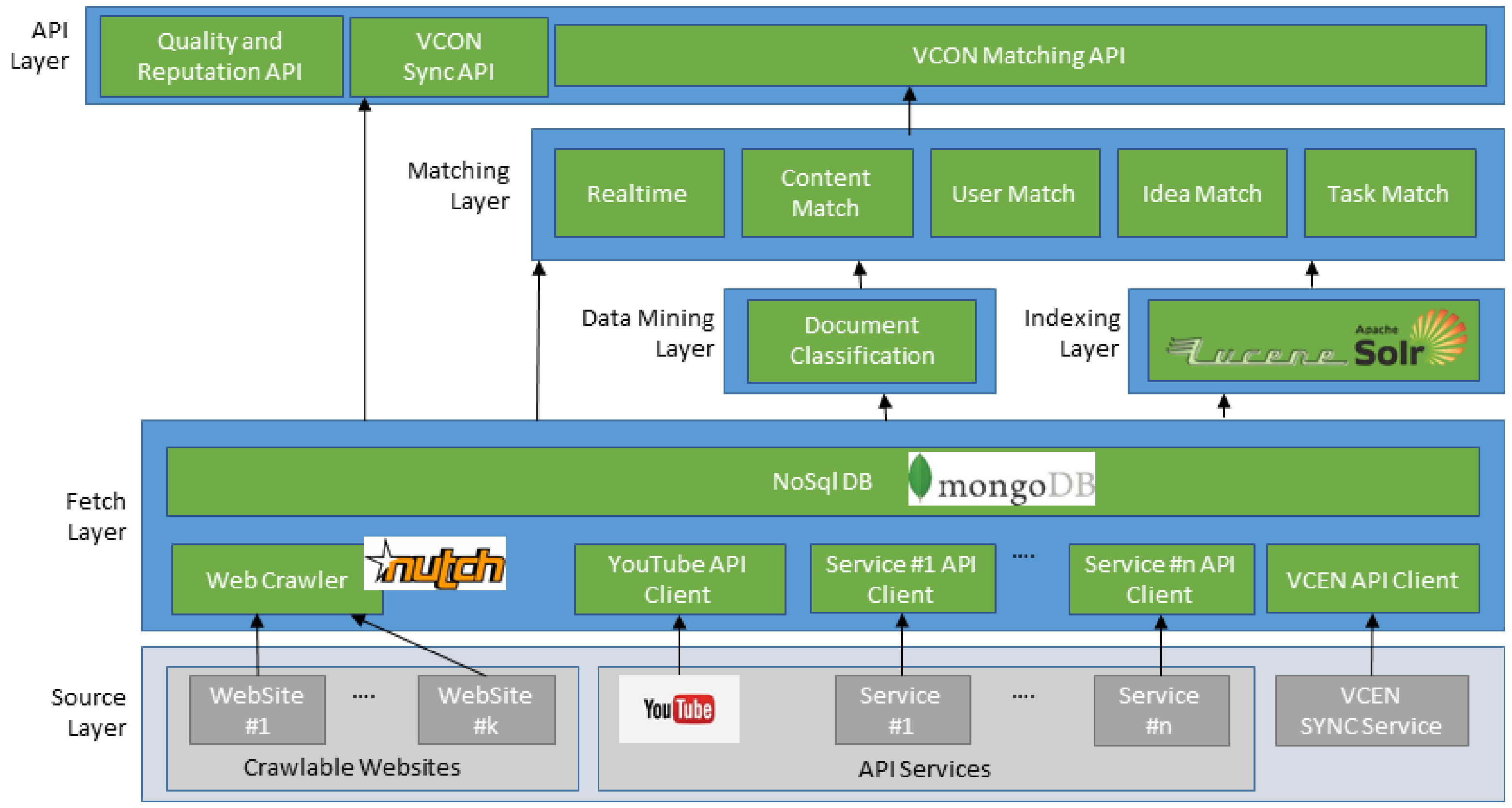
- The source layer represents the sources; in addition to the VCEN SYNC Service, provided by the remaining of the platform for allowing the retrieval of the internal content, sources include different Web sites to be crawled, and services accessed through specific APIs; as examples, currently the system accesses YouTube for relevant videos about entrepreneurship, and sites such as www.futureenterprise.eu (structured description of more than 100 courses/curricula delivered at a European as well as global level about entrepreneurship), steveblank.com/tools-and-blogs-for-entrepreneurs/ (a list of tools and blogs about entrepreneurship), ecorner.stanford.edu (online material from the e-corner of Stanford University for entrepreneurship creation), www.techcrunch.com and www.techradar.com (sites about technology and start-up, useful to compare ideas with already proposed similar ones by existing startups), etc. Currently 105 sites are crawled and information made available in our system: over 34.000 different pages/documents are retrieved from the Web respecting service modalities and intellectual property rights.

- The fetch layer includes the specific components in charge of retrieving information from the sources: crawlers, based on the Apache Nutch and specific modules invoking the APIs of the services. The information are then stored in the NoSQL database MongoDB.

- The data mining and indexing layers are where the information are processed in order to be later used; in particular, Apache Lucene is used for indexing documents, and MAUI[19] automatically finds and extracts main topics in text documents (tags, keywords, keyphrases, vocabulary terms, etc.),all of them to be used for matchmaking functionalities.
- The matching layer provides the specific matching components for ideas, users, tasks and content, addressing the specific needs of having the matchmaking techniques running in interactive way during content writing by the users.

- Finally, the matchmaking features are offered as RESTful services to the remaining of the platform through the API layer. As an example, the user interface previously shown interact with a RESTful service offered by the components in this layer.

The system also provides a set of additional services that aim at improving the quality of provided results. For example, a quality and reputation score is provided for users and contents. These scores influence the order matchmaking results are provided, thus promoting content that has been evaluated positively by users with a high reputation score through the end user interface of InnovVoice.



InnoVoice Architecture

4.3 The Offline Layers

The offline layers are those layers of the architecture that are executed offline, preparing the knowledge for the online functionalities provided by the upper layers. The source layer includes all the sources that can be used by the system to obtain content. These sources include contents available from the Web and contents edited inside the InnovVoice platform. The latter include ideas under incubation, documents edited inside the platform, user profiles and tasks connected to the development of a specific idea. On the other hand, contents on the Web can belong to many different categories including static Web pages, videos, online courses. The extraction of content is performed by the fetch layer. From this point of view, sources of Web contents can be mainly divided into two categories, i.e., those that allow crawling and those that instead provide (paid or not) Web services. Sources that allow crawling are explored through the employment of a Web spider (Apache Nutch in our case). A spider starts from a set of seed URLs and explores a Web site by following outgoing links. A spider can be configured to filter out specific URLs or Web pages according to rules concerning extensions and content patterns. Those sources that instead can be explored through APIs need a specific component to be developed in order to be explored. These components take care of authorizations and security and must respect the terms of service imposed by the specific API. At the current stage the only component developed is the one in charge of exploring YouTube videos by employing the API to obtain videos from channels specific to entrepreneurship (e.g., the Stanford channel about business). In this case, the extracted content is the description of the text, but it is possible, by paying specific fees, to access other information such as automatic transcription performed by YouTube. A particular component of the fetch layer is in charge of gathering content edited by the user on the InnovVoice platform by exploiting the API made available by VCEN. The VCEN API allows to query by last update date in order to avoid expensive reading operations. The indexing layer, implemented through the Apache Lucene search engine, is in charge of creating the indexes that will be used at runtime to

respond to user needs. It is important to note how the system takes many different indexes corresponding to the different kinds of content it handles.
The data mining layer is in charge of performing different text analysis tasks. The most important of these tasks is the automatic classification of documents according to a taxonomy of thematic subjects useful to entrepreneurship and innovation. This classification task is performed specifically on the documents that are crawled from the Web. Document classification is performed using libraries that are trained against a manually labeled dataset obtained through crowdsourcing. In particular, users of the platform are sporadically asked to classify contents, and the responses are employed to train the different classification algorithms implemented in Mallet (http://mallet.cs.umass.edu/), which is the library the system is currently employing for this task.

4.4 The Matching Layer

The matching layer works pretty much as a Web search engine. The indexes obtained from the different sources that the system integrates are searched against a query. This makes our system belonging to the class of application specific search engines. Differently from a Web search engine, the query is neither a sentence nor a sequence of keywords, but a content item. A content item is generically defined in InnovVoice as a source of information, thus including crawled Web pages, videos, ideas composed inside the platform and user profiles. In other words, the system is based on searching by content instead of searching by query. As Lucene, which is the search engine underneath the system, does not directly support this modality, the first step is to transform a content item into a query that can be handled by Lucene. In our platform, this step is obtained by first extracting important keywords from the text and then combining them into a textual query. Automatic extraction of keywords from a text is an open research field and the employment of simple statistical methods based on word count fails without an analysis of relevance of each single keyword. As an example, a single verb can be very frequent in a text (e.g., do, make) without being important for the semantics of the text. An alternative to the employment of a library for keyword extraction is the employment of advanced cloud based services for text mining such as AlchemyAPI5. A different approach to search by content is represented by the employment of word histograms, i.e., the matching of the word histogram of a content with the histograms of the contents available in the index. The system allows to optionally employ this modality, anyway, as it is not directly supported by Lucene, this kind of search is much slower and not suggested for a real time employment. In our system, we are currently using Maui as a keyword extraction technique [5]; differently from other approaches, Maui employs, beside statistical analysis, a learned model. The drawback of this approach is that tuning the system requires a set of manually labeled documents that strongly influence the behavior of the system at runtime. As an additional point, being the portion of the Web the system is aware of very limited, it is not possible to make an analysis of the authority of a content by analyzing links between contents themselves (e.g., using algorithms such as PageRank [7]); therefore, in order to compute authority, we focus on the social component of the platform by exploiting the so called wisdom-of-the-crowd. In particular, we assign a score to each user and each content item that allows to assess relevance and authority. A content item is evaluated according to evaluations provided by the users of the platform (tuned by the score assigned to the user providing the evaluation) and by the level of activity in terms of number of edits or likes of the content itself (where applicable). The score assigned to a user is instead obtained by monitoring its level activity in the platform and the scores assigned to the content they produced. The level of activity of a user is computed by taking into account the frequency of access, the frequency and number of comments it submitted, the number of teams it belongs to. As a consequence, the score assigned to content in the context of a search issued by content is a combination of the relevance to the issued query and the score assigned to content itself. This approach allows to provide users with relevant (by using keyword extraction) and high-quality (by employing user evaluations) content that can be useful while developing ideas.

Contents are proposed to users by the platform through a set of services. An important design aspect is indeed the way by which users, belonging to various stakeholder types and involved with different things, will be able to discover VOICE contents that will be relevant to what they are about to do. The high level services that the platform provides are the following:

- Matching profile skills and services with the activities (e.g., idea, projects) currently active in the platform). This service aims at providing information about professionals that can be helpful to the development of a given idea or project. It also aims at providing information about enterprises/companies operating (or wishing to operate) in areas related to a given idea or project.
- Matching contents with the development stage of an idea or the incubation phase of a project. This service aims at providing other contents (e.g., articles, videos, events, other ideas) that can be helpful to the development of a given idea or project. This information can be specific to the stage the idea is or to the incubation phase of a project. As an example, at a certain stage of the idea development, content about the creation of business plans will be provided/suggested.
- Matching similar ideas. This service aims at providing information about other similar ideas. This is useful if different teams working on similar ideas (potentially from different countries) that might wish to discover each other and, optionally, join their efforts.
- Profile Evaluation Metrics aim at evaluating the reputation of a user and his/her trends in the platform by employing the data coming from his/her profile, its Contents, and the comments of the other users to all of his/her activities.
- Team Evaluation Metrics. This service aims at extracting team metrics and showing them to VOICE Users. A team metric is intended to represent the quality of a team both from the point of view of the single members and of the team as a whole. In this sense, it also takes as input the results obtained by applying the User Evaluation Metrics service.
- Content metrics aim at measuring the value of a Content Item. These metrics are based on user comments and evaluations as well as on user reputation

The abovementioned services aim at providing knowledge to the users of the VOICE platform and to the community in general. This knowledge will also have an important role in the growth of the VOICE platform by satisfying the following needs:

- Need to show engaging content to casual, first time visitors of the VOICE ecosystem. Unregistered users visiting the ecosystem should not only get info about the VOICE platform and the site itself (so they are convinced about what VOICE can do for them), but also see some indicative content that will demonstrate the knowledge produced and provided within the ecosystem (so they are motivated to register and become active members of VOICE).
- Need to have initial content which will be indexed by commercial search engines.
- Need to inspire users to propose their own ideas by proposing interesting content according to their preferences.

Contents can be two types: internal and external.

The production of internal contents will be available to registered users. We will distinguish between those contents (i.e., Primary Content) that are produced by a team directly involved in a project (e.g. ideas, tasks) from those information (i.e., Community Content) that are instead provided by users not directly involved in projects and want to contribute their experience and thoughts.

External Content providers include:

- Open content resources (articles, lessons, etc.)
- Open data (governmental, etc.)
- Other social/technical networks (API based, on the fly)
- News aggregators (with license)
- Other sites (re-publications with license and attribution)

Another important thing to take into account in VOICE is the possibility to store external contents inside the platform; this is generally forbidden by terms of use but some exceptions do exist. Additionally, in the vast majority of cases indexing is not forbidden.

Extraction Mechanisms

In VOICE there are two main categories of extraction mechanisms. An offline extraction mechanism fetches resources independently from user runtime requests in order to promptly satisfy these latter once they are issued. An online extraction mechanism instead extracts a resource in the moment it is needed. The choice between offline and online extraction mechanism is not only matter of performance but it is also dependent from the kind of terms of service that the service provider declares.

Matchmaking Services

Matchmaking services may come in different forms, depending on the kind of concepts involved in the matchmaking task:

- Internal Content to VOICE User Matchmaking service. This service aims at providing information about professionals that can be helpful to the development of a given idea or project. This service also aims at providing information about enterprises operating (or wishing to operate) in areas related to a given idea or project.
- Internal Content matchmaking. This service aims at providing other contents (e.g., articles, videos, events, other ideas) that can helpful to the development of a given idea or project (or more generally of an internal content). This information can be specific to the stage the idea is or to the incubation phase of a project. Proposed contents can be for example educational resources.
- Idea-Idea matchmaking. This service aims at providing information about other similar ideas. This is useful if different teams working on similar ideas (potentially from different countries) want to discover each other and, optionally, join their efforts.

- Matchmaking services will mainly work by employing information coming from tags, content classes. Besides these metadata, the body of the content can be used itself matchmake different concepts. In order to do that the employment of natural language processing technique is needed to extract keywords.

Once to a specific object in the system a set of keywords and tags has been associated it will be possible to matchmake other objects by using clustering techniques on bags of words.
These services and mechanisms include both state of the art artificial intelligence techniques and ad-hoc approaches devised during the research. The techniques include data mining and machine learning algorithms for text analysis and classification. Moreover, a reference architecture is provided for the system that is be employed in the context of a European project.

4.5 The API Layer

The API layer is implemented through a set of RESTful services that make it possible to access the matchmaking services of the platform and other functionalities. Return values are objects represented through JSON (JavaScript Object Notation). The implementation of the services is obtained through Jersey over Apache Tomcat. The API layer provides three different API groups, namely the matching API, the VCON API, and the Quality and Reputation API. The matching API functionalities are available through the RESTful endpoints /api/match/{content-type}/{id} where {content-type} denotes the kind of content we want to match against the index and {id} is the id of the specific content inside the system. Valued values for the URL parameter {content-type} are (i) document for the content that is internally edited inside the platform, (ii) idea for innovative ideas that are currently under incubation inside the platform, (iii) user for users registered in InnovVoice, and (iv) task for tasks that are available to be taken in the context of the development of an idea. Calling the matching API returns a set of content items divided by category that match with the content required following the methodology introduced in Section 4. The returned categories are ideas, internally edited content, external crawled Web pages, tasks and users. As an example, issuing an HTTP GET method on the URL /api/match/user/12 will return:

- The other users that match with user 12 in terms of skills, interests, past experiences and curriculum.
- The ideas that can be interesting for the user 12 to follow or join according to its user profile.
- The internal edited documents that can be interesting to read.
- The Web pages that can be interesting to follow. These Web pages are returned together with a label representing their thematic subject.
- The tasks that the user can perform according to its skills and past experiences.

In order to match the last available information, the version of the InnovVoice content that is matched against the back-end is the most updated one by exploiting the API provided by the VCEN subsystem of the InnovVoice infrastructure. The matching API provides two additional endpoints that allow to return matching results with respect to a text edited in real-time (not stored inside the backend) or a specific query. The first one is accessible through the endpoint /api/match/realtime, whereas the second one through the endpoint /api/match/query. The difference between these two latter endpoints is that the first one undergoes the keyword extraction process described in Section 4 whereas the second one directly issues the query passed as argument to Lucene. The real-time endpoint is, for example, the one employed in Figure 1 where the user is editing an idea and the system provides, while the user is editing, new results according to the changes in the keywords

extracted by the automatic keyword extraction algorithm. The VCON API allows to access different information about crawled content from the Web, such as those ones discovered since a specific date, or those obtained from a specific source. This API additionally allows to access statistics about the external source employed for crawling.
Finally, the Quality and Reputation API allows to access the scores assigned to content and users as described in Section 4 and to post and get evaluations for different categories of content available in the system. Scores can be queried through the endpoints /api/quality/{content-type}/{id}/score where the parameters follow the specification already provided for the matching API. The evaluations can be retrieved (respectively posted) issuing a GET (respectively a POST) to the endpoint /api/quality/{content-type}/{id}/evaluation. All the described API endpoints are intended to be employed from the InnovVoice website or by external clients as part of the commercial exploitation of the platform.

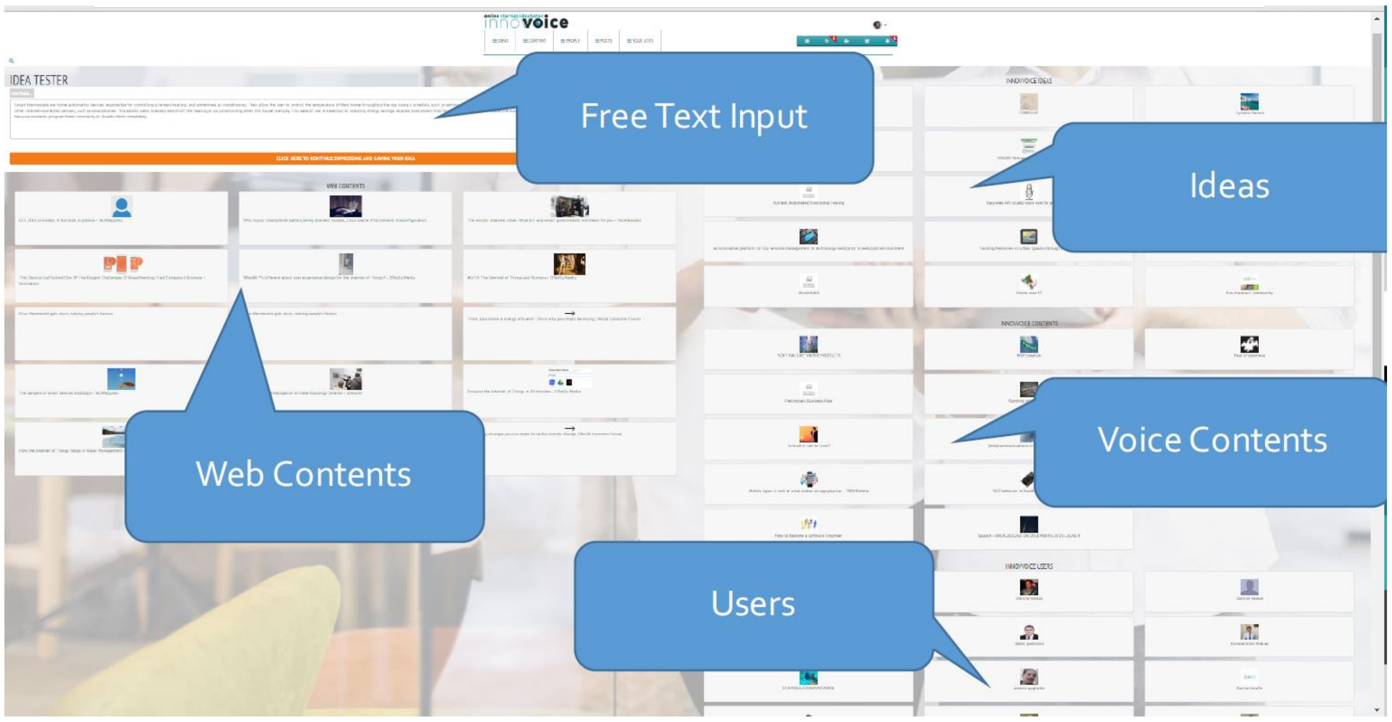


Fig. Innovoice interface

5 Implementation of Matchmaking System

In this chapter we will cover in details all the technologies used in the project and the algorithms they use to achieve the result, as well as other work done to bring the system online.

5.1 System Configuration

The initial step was to make the Voice Search engine configurable by developing a special configuration file. It is quite common for this cases to use XML language for its simplicity and usability. The configuration stores information about web clients that should connect to different services for accessing their content, such as Google, YouTube, LinkedIn, Dailymotion, AngelList and others.

The sample of this configuration is displayed below:

```
<youtube>
    <channels>
        <channel>
            <name>youtube_columbia_channel</name>
            <channelid>UCMt5mJXmE02QoIPhc2NtO9w</channelid>
            <indexingClasses>.Entrepreneurship.</indexingClasses>
        </channel>
    </channels>
</youtube>
```

It contains the id and the name of YouTube channel that we need to crawl. Similarly other resource information is stored in the xml file. It also contains all the access keys to the services that are required to authenticate the client on the service.

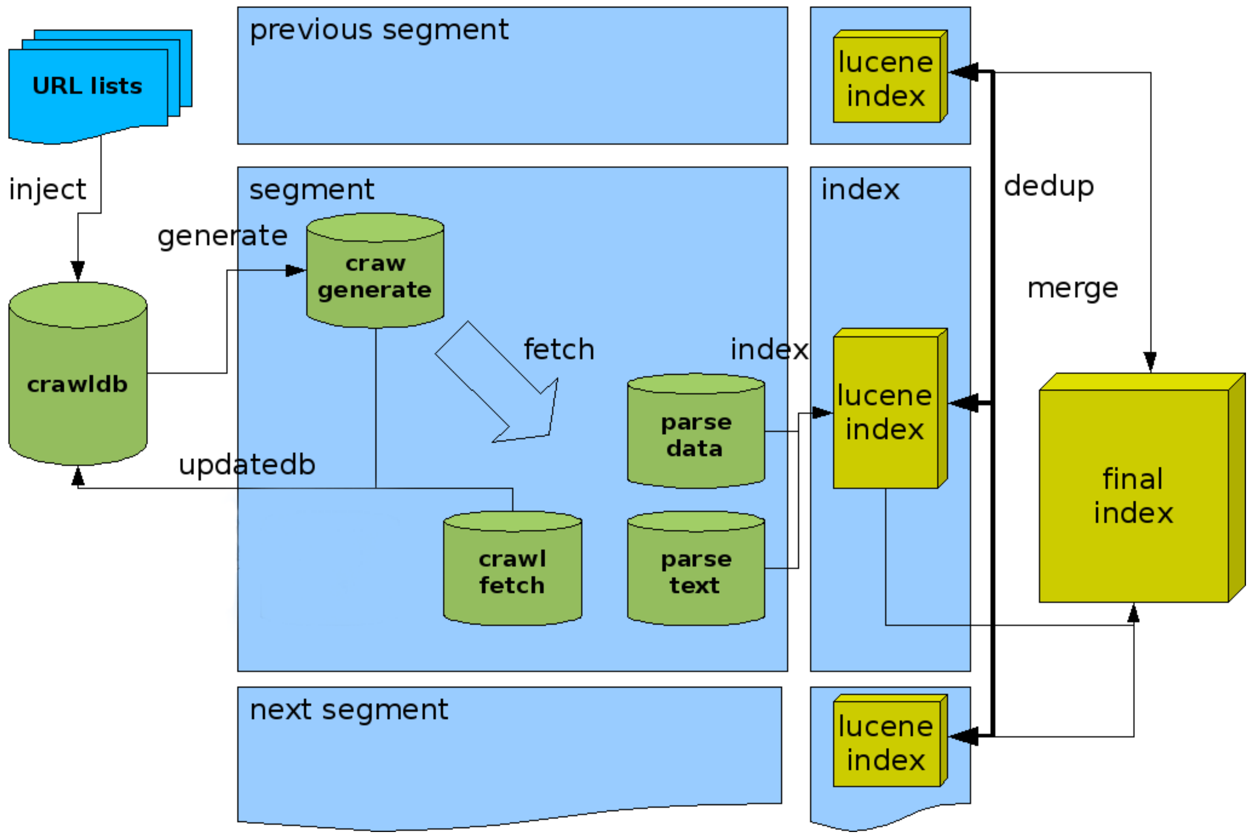
5.2 Apache Nutch

As mentioned in previous chapter Apache Nutch is open source web crawler. Apache Nutch is very flexible and it was customized to fulfill the needs of the VOICE project. By default, it is an autonomous application that can run from the console. But for needs of our project it was integrated into the system. Nutch provides and API which can be used for crawling the web. It contains configuration files where we can set all the parameters for crawling. Some of the parameters can also be changed from the code through the API.

To start crawling the Web, Apache Nutch needs to be provided with the list of URLs called *seeds.txt*. It injects the URLs into the database called *crawldb*. After that it generates a fetch list from the database.
It creates a fetch list of all of the pages due to be fetched. The fetch list is placed in a newly created segment directory. The directory is named by the time it’s created.

Main components that we use in this system are:

1. Injector
2. Generator
3. Fetcher
4. Parser
5. Updater



```
//1. Inject
String[] injectParams = {crawldbFolder, seedsFolder};
ToolRunner.run(conf, new Injector(), injectParams);

for (int i = 0; i < numRounds; i++){

    //2. Generate
    String[] generateArg = {crawldbFolder, segmentsFolder, "-TopN", "50000",
"-numFetchers", "2", "-noFilter"};
    ToolRunner.run(conf, new Generator(), generateArg);
    File segmentDirs = new File(segmentsFolder);
    String[] directories = segmentDirs.list(new FilenameFilter() {
        @Override
        public boolean accept(File current, String name) {
            return new File(current, name).isDirectory();
        }
    });
    Arrays.sort(directories);
    String segment = directories[directories.length - 1];

    //3. Fetch
    String[] fetchArg = {segmentsFolder + "/" + segment, "-noParsing", "-
threads", "50"};
    ToolRunner.run(conf, new Fetcher(), fetchArg);

    //4. Parse
    String[] skipRecordsOptions = {segmentsFolder + "/" + segment};
    ToolRunner.run (conf, new ParseSegment(), skipRecordsOptions);

    //5. Update
    String[] crawlDbArgs = {crawldbFolder, segmentsFolder + "/" + segment};
    ToolRunner.run(conf, new CrawlDb(), crawlDbArgs);
}
```

Another important aspect is the crawl depth also known as number of rounds. It is a number that defines how many levels should the crawler traverse down from the root page. While crawling it identifies the links on the page and on the next round crawler traverses them. If the number is big it may take a very long time to finish the task (up to several days). The number should be chosen carefully. In our case the number is set to 4. The crawler has an option to limit search on the links that point to pages within the same domain. So for each source from the list we insure that the crawler doesn't process external links (for example, skipping advertisement links).

In order to concretize what we need to crawl the system's configuration allows to use regular expressions filter which can narrow down the amount of documents which does not contain textual data or contain unnecessary information (login pages, terms and conditions, etc.). The crawler will skip all the pages or components that match regex rules. In the case of Voice project there are several rules set up that help the system to crawl more effectively. These rules helped to save some time while crawling and avoid unnecessary data to be stored in our database. We also filter out rss, xml, xls and other application specific extensions.

```
Regular expressions:
# skip folders
-cgi-bin
-images
-css
-login

# skip different directory listings
-.*\?C=(N|M|S);O=(A|D)$

# skip image and other suffixes

\.(gif|GIF|jpg|JPG|png|PNG|ico|ICO|css|CSS|sit|SIT|eps|EPS|wmf|WMF|zip|ZIP|pp
t|PPT|mpg|MPG|xls|XLS|gz|GZ|rpm|RPM|tgz|TGZ|mov|MOV|exe|EXE|jpeg|JPEG|bmp|BMP
|js|JS)$
```

Once we have all the documents crawled we now need to store them into the database. For this purpose we use MongoDB. A special class called *SegmentOutputParser* is responsible for storing all the content into the database which will be later indexed by Apache Lucene.

5.3 Indexing

Apache Lucene is responsible for indexing all the crawled content. A special class is created that uses Lucene API to index all the files and store the indexes in the MongoDB.

5.4 Keyword extraction

Keyword extraction is one of the most important features of the system. In the begging of project several libraries were chosen for this task. And the goal was to find out the one that extract the best keywords. After testing them on real documents, MAUI indexer was chosen for keyword extraction, since it was providing the best result. In the end of this section we will briefly cover other libraries used for extraction and explain why they were not chosen.

Maui indexer is one of few libraries that do keyword extraction. It provides a wide range of services. However, for the scope of this project only keyword extraction is applicable. After extracting keywords the indexer assigns score to each of them which shows how relevant is the keyword in the document. This information is useful, because in the future it will be used for querying the documents in Lucene. As mentioned in previous chapter, keyword extraction requires the indexer to be trained before it can be applied on documents.

Train data consists of two files collections: documents with similar topics, and files with keywords manually assigned to each document. Based on this data MAUI builds a model which later is used to extract keywords from documents.

There are several collections available on the web for training the indexer and building the model. Unlike collections with just one topic set per document, these collections contain topic sets assigned to each document by different people. So they have several keywords assigned to each document. This allows to measure the agreement between the people, which provides a direct comparison to the performance of the algorithm.

1. Keyphrase extraction model created using SemEval-2010 training data
2. FAO-30 data set
3. FAO-780 data set for term assignment
4. CiteULike-180 data set for automatic tagging [23]

CiteULike-180 is the only test set listed here that was created in natural environments. It has been automatically extracted from the large data set of tags assigned to the bookmarking platform CiteULike.

The resulting set contains 180 science articles from HighWire and Nature, with tags assigned by 332 voluntary taggers on CiteULike.

Maui indexer was trained using all the collections mentioned above. Then based on the generated model Maui was tested for test set of documents. The best results were achieved using keyphrase extraction model created based on SemEval-2010 training data. Maui was able to extract adequate keyphrases from the sample documents. However, it can be changed if new better model appears and proves to achieve better results. One of the main aspects of MAUI indexer that it is recommended to train it with the collections that has similar topics. In our case we have wide range of topics so SemEval-2010 training data is a compromise.

5.5 Other Keyphrase Extraction Mechanisms

In the initial stage of the project other keyword extraction mechanisms were tested for the system. However, none of them was fully able to cover all the requirements. They are listed below.

2. JATE

[24] Describes JATE (Java Automatic Term Extraction) as "toolkit a library with implementation of several state-of-the-art term extraction algorithms. It also provides a generic development and evaluation framework for implementing new term extraction algorithms".

JATE is implemented based on the common ground of most ATE/ATR algorithms, which typically conform the steps described below [24]:

- "Extracting candidate terms from a corpus using linguistic tools
- Extracting statistical features of candidates from the corpus
- Apply ATE/ATR algorithms to score the domain representativeness of candidate terms based on their statistical features. The scores give an indication of the likelihood of a candidate term being a good domain specific term".

4. Apache Open NLP

The Apache OpenNLP library [25] "is a machine learning based library for processing of natural language text. It supports the most common natural language processing tasks, including tokenization, sentence segmentation, part-of-speech tagging, named entity extraction, chunking, parsing, and coreference resolution. These tasks are usually required to build more advanced text processing services. OpenNLP also includes maximum entropy and perceptron based machine learning".

5. SNLP

The project developed by The Stanford NLP Group [26] provides statistical, deep learning, and rule-based natural language processing libraries for major computational linguistics tasks, which can be integrated into applications in the field of human language technology. These libraries are quite popular in industry, academic activities, and government sectors.

6. TreeTager

The TreeTagger [27] “is a library for annotating text with part-of-speech and lemma information. It was developed by Helmut Schmid in the TC project at the Institute for Computational Linguistics of the University of Stuttgart. The TreeTagger library is able to perform tagging of German, English, French, Italian, Dutch, Spanish, Bulgarian, Russian, Portuguese, Galician, Chinese, Swahili, Slovak, Slovenian, Latin, Estonian, Polish and old French texts and is adaptable to other languages if a lexicon and a manually tagged training corpus are available”.

7. ClearNLP

The ClearNLP [28] “project provides software and resources for natural language processing. The project started at the Center for Computational Language and EducAtion Research, and is currently developed by the Center for Language and Information Research at Emory University. This project comes under the Apache 2 license”.

8. RAKE

Rapid Automatic Keyword Extraction (RAKE) [29] is an algorithm to automatically extract keywords from documents. RAKE is a well-known and widely used NLP tool, but its concrete application depends a lot on factors like the language in which the content is written, the domain of the content and the purpose of the keywords.

The implementation in this library is mainly aimed at English. With additional resources, it is also applicable to other language. The library is inspired by a similar implementation in Python. Unlike MAUI indexer, it does not require any train data and can be applied on document as is. It needs to be initialized with list of stop words and 3 parameters

```
//Python implementation
rake_object = rake.Rake("SmartStoplist.txt", characters, max_words, word_appears)
```

- `characters` is the minimal number of characters that the word should have,
- `max_words` is the maximum number of words allowed in keyphrase,
- `word_appears` is the minimum number that word appears in the text.

Rake library was the only one that could perform relatively good results in comparison with other libraries tested excluding MAUI and it didn’t require any test data to be trained. It was applying statistical methods to extract keywords. However, those methods are not enough to achieve a good quality results. That’s why in the end we stopped on MAUI.

5.6 Porter Stemmer

Porter's stemmer [30] is the most used in information retrieval, probably because of its balance between simplicity and accuracy. Porter stemmer defines a five step algorithm applied to every word in the vocabulary. A word is defined as a succession of vowel-consonant pairs *[C](VC)m[V]*, where *C* and *V* are lists of one or more consonants and vowels respectively and *m* is the measure of the word.

Porter Stemmer Overview

- Excellent trade-off between speed, readability, and accuracy
- Stems using a set of rules, or transformations, applied in a succession of steps
- Around 60 rules in 6 steps
- No recursion

Porter Stemmer Steps

1. Get rid of plurals and -ed or -ing suffixes
2. Turn terminal y to i when there is another vowel in the stem
3. Map double suffixes to single ones: -ization, -ational, etc.
4. Process suffixes, -full, -ness etc.
5. Take off -ant, -ence, etc.
6. Remove a final –e

5.7 Lucene Queries

Although Lucene allows us to build our own queries through its API, it also has a custom query syntax for querying its indexes through the Query Parser, a lexer which creates a Lucene Query from the string using JavaCC. We use Query Parser to build a complex query.

5.7.1 Terms

Lucene query consist of terms and operators. Two type of terms are used to build a query:

- Single Terms – a single word such as “hello” or “world”.

- Phrases – group of words surrounded by double quotes (e.g. “hello world”).

To build a more complex query using multiple terms we can combine every term together by using boolean operators.

5.7.2 Query types used in the system

1. **OR**

In Apache Lucene the OR operator is the default conjunction operator. Which means that if there is no operator used between terms in the query, by default OR operator is used. The OR operator links two terms and returns a matching document if any of the terms exist in a document. This is equivalent to a union using sets. We can use also || symbol instead of the word OR.

To search for documents that contain either "Hello World" or just "World" the following query should be used:

"Hello World" World

or

"Hello World" OR World

Boosting a Term

Lucene provides the relevance level of matching documents based on the terms found. To boost a term, we can use the caret "^" symbol in combination with a boost factor at the end of the term we are searching. A boost factor is a number. The higher we set the boost factor, the more relevant the term becomes.

Boosting allows you to control the relevance of a document by boosting its term. For example, if we are searching for "Hello World" and we want the term " Hello " to be more relevant, we can boost it using the ^ symbol along with the boost factor next to the term. The query will have the following form:

Hello^5 World

This will make documents with the term Hello appear more relevant. Phrasal terms can also be boosted. For example:

"Hello World"^5 "What's up Universe"

If we don't specify any number after "^" symbol the default number is 1. Although the boost factor must be positive, it can be less than 1 (e.g. 0.5).

2. **Fuzzy Queries**

With Lucene we can do fuzzy searches based on Damerau-Levenshtein Distance. The tilde, "~", symbol corresponds to the fuzzy query, it should be placed at the end of a term. However, this operator can be only used with single terms. For example, if we want to search for a term similar in spelling to "fold" the following fuzzy query should be used:

fold~

This search will find terms like hold and folds. We can specify the maximum allowed number of edits by adding an optional numerical parameter after “~” symbol. The number should be between 0 and 2, For example:

fold~1

If no number is specified, 2 edit distances are used by default. In our case we use 1 edit distance in the query.

3. Wildcard Searches

Wildcard search allows to match strings based on character pattern matching between string specified in a query and words in documents that contain those character patterns. Asterisk (*) is used to perform the multiple character wildcard search. Multiple character wildcard searches look for 0 or more characters. For example, to search for test, tests, or tester, we can use the following search: test*.

4. Escaping Special Characters

In Lucene there are special characters that are part of query syntax. The following are the special characters:

*+ - && || ! ^ " ~ * ? : \ / ( ) { } [ ]*

Backslash character (“\”) should be used before each special character if we want to escape them. For example to search for *(a+b):c* the following query should be used: *\(a\+b\)\:c*

Lucene has more operators and modifiers, but during the tests the above mentioned were the only suitable for the correct system operation.

Now that we have all the operators and modifiers defined, we can build a simple query and see how it looks like.

Each query is built from the keywords extracted from previously crawled document using MAUI indexer. Keyword can be a single word as well as phrase. We add “OR” operator after each keyword. Then we boost each keyword by applying boosting coefficient based on the score assigned by Maui indexer. We normalize the score by applying primitive formula: *Lucene Score = Maui score * 10*. Maui score has 16 digits after the dot, and we round the number to decimal since Lucene doesn’t need that precision. Values assigned by Maui are between 0 and 1. The higher is score the more relevant is the topic. To boost the term in Lucene the score should be greater than 1 (though smaller number can be also used).

If the word is a single term we first stem the word where possible and then apply “^” operator which looks for similar words that are close to the word. It may help to avoid grammatical mistakes in some cases.

Let’s suppose we have the following keywords extracted and from the document and scored by MAUI indexer.

Topic	**Score assigned by MAUI**
Acronis	0.4435958897151936
Software	0.3922795866443547
True	0.30010875206803933
True Image	0.27028440952871347
Acronis True Image	0.1921594095287134
Image	0.17252496633122377
Disk	0.17218577793072223
Committing	0.16862527442624392
Computer	0.1394900883657769

The process looks as described below.

1. Add “OR” operator between each term.
2. Assign numerical coefficient to each term (i.e. boost the term).
3. Stem single terms where possible.
4. Use wildcard search for single terms (*).
5. Add fuzzy operator with numerical argument “2” to single terms.

Since we also have phrase terms in our query they need to be surrounded by double quotes to separate them from single terms. Otherwise, they will be considered by Lucene as single terms with *OR* operator between them. In this case \ symbol is used to escape the special character “ and build a correct query for Lucene.

```
public String buildQueryTextFromString(String text) throws Exception {
    List<Topic> topics = extractTopicsFromString(text);
    String foundTopics = "";
    for (Topic mt: topics) {
      if (!foundTopics.equals(""))
            foundTopics += " OR ";
        if (foundTopics.contains(" ")) //Phrase term
            foundTopics += ("\"" + mt.getTitle() + "\"" + "^" +
Math.floor(mt.getProbability() * 100) / 10);
        else
            Stemmer stemmer = new Stemmer();
        char[] charArray = mt.getTitle().toCharArray();
        for (int i = 0; i < charArray.length; i++)
            stemmer.add(charArray[i]);
        stemmer.stem(); //Stem the term
        String stemmed = stemmer.toString();
        foundTopics += ("\"" + stemmed + "\"" + "*~1^" +
Math.floor(mt.getProbability() * 100) / 10);
    }
    return foundTopics;
}
```

Final query using Query Parser tool will be the following:

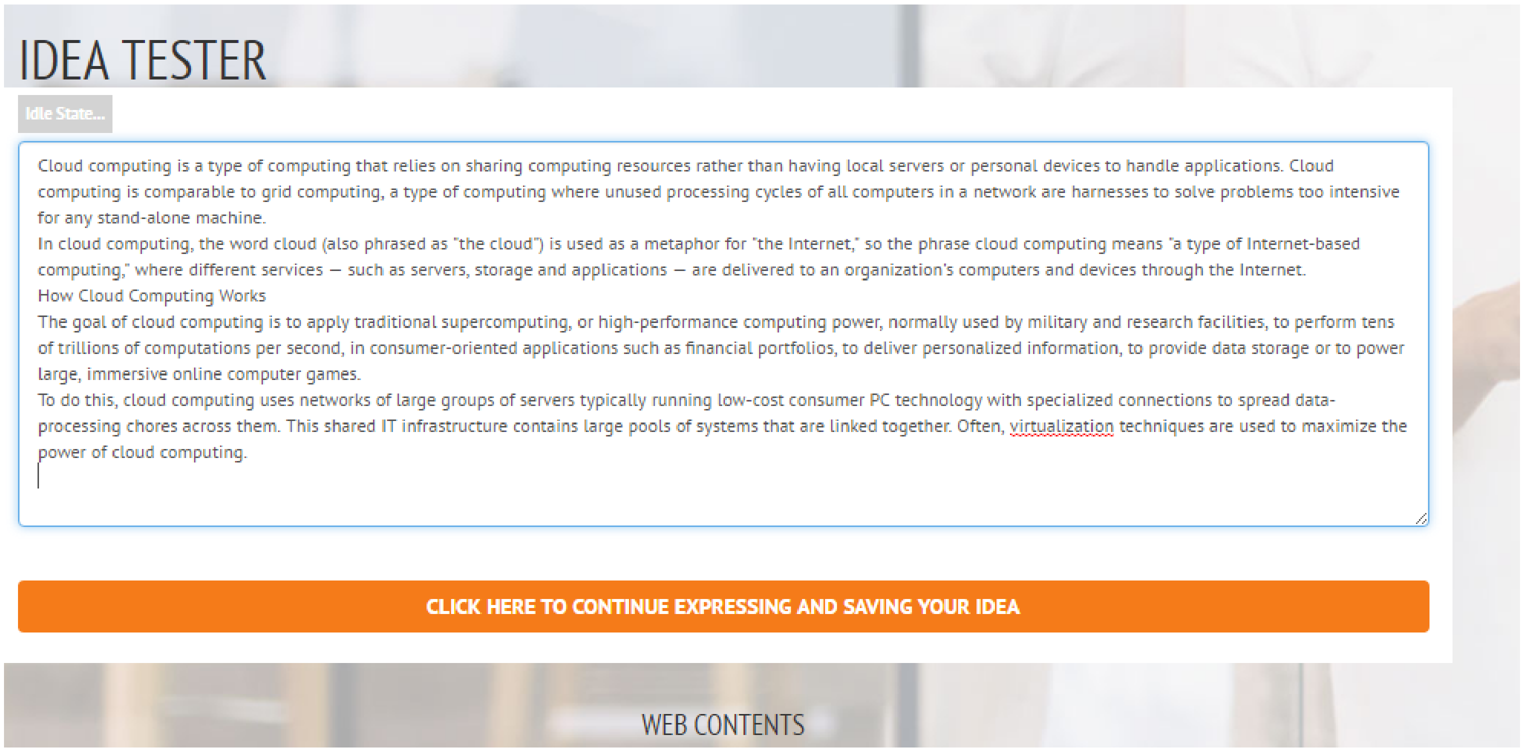
```
Query query = parser.parse("Acronis*^4.4~2 OR Software*^3.9~2 OR True*^3.0~1
OR \"True Image\"^1.9 OR \"Acronis True Image\"^1.9 OR Image*^1.7~2 OR
Disk*^1.7~2 OR Commit*^1.6~2 OR Computer*^1.3~2 ")
```

Note that in this example term “Committing” was stemmed to “Commit”.

Now when the query is ready we can submit it to Lucene. Lucene will find matched documents based on the term relevance and edit distance and return as an output all the ids of the documents stored in MongoDB. Lucene assigns score to each documents. The higher is the score the more relevant is document hence its ranking is higher.

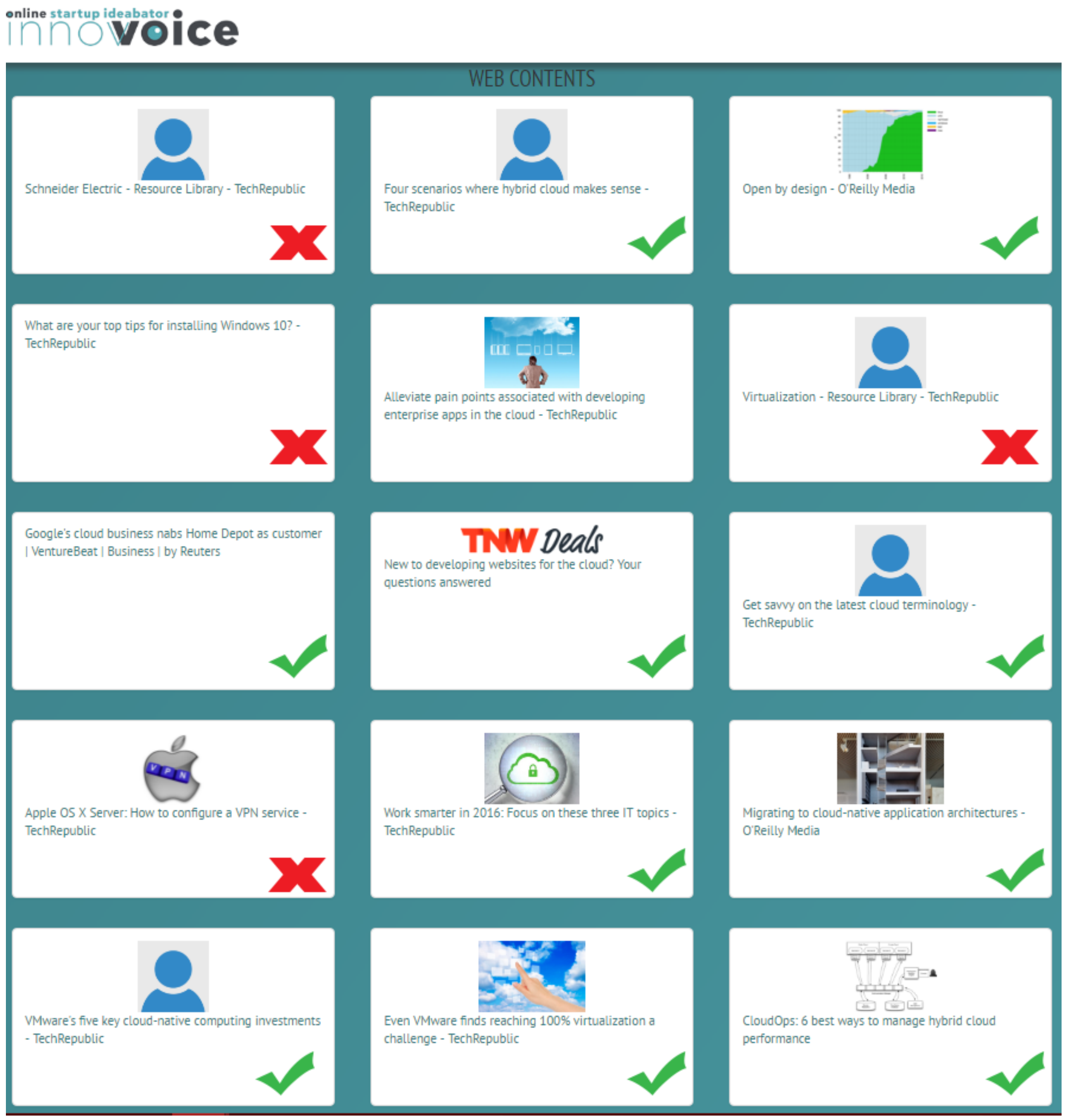
6 Validation

In order to assess how accurate results system provides several tests were designed. There are around 100 sources crawled and stored in our database. Most of them are related to business, innovation, startups, as well as different technical topics. We have randomly chosen from web around different 50 materials with mentioned topics and insured that these materials are not present in our database (Documents taken from web articles, Wikipedia pages, etc.). Each document was submitted to the system and corresponding output was analyzed. The platform is designed in a way that it provides 15 most relevant documents against our query (top 15 ranked documents by Lucene). Hence, for each document we calculated how many of them had similar topic with the input document. For every topic we have submitted multiple documents (at least 2). Tests have shown that for all documents at least the half of the output was correct. In many cases 15 out of 15 matched documents were correct.



Cloud computing is a type of computing that relies on sharing computing resources rather than having local servers or personal devices to handle applications. Cloud computing is comparable to grid computing, a type of computing where unused processing cycles of all computers in a network are harnesses to solve problems too intensive for any stand-alone machine.
In cloud computing, the word cloud (also phrased as "the cloud") is used as a metaphor for "the Internet," so the phrase cloud computing means "a type of Internet-based computing," where different services — such as servers, storage and applications — are delivered to an organization's computers and devices through the Internet.
How Cloud Computing Works
The goal of cloud computing is to apply traditional supercomputing, or high-performance computing power, normally used by military and research facilities, to perform tens of trillions of computations per second, in consumer-oriented applications such as financial portfolios, to deliver personalized information, to provide data storage or to power large, immersive online computer games.
To do this, cloud computing uses networks of large groups of servers typically running low-cost consumer PC technology with specialized connections to spread data-processing chores across them. This shared IT infrastructure contains large pools of systems that are linked together. Often, virtualization techniques are used to maximize the power of cloud computing.

Cloud Computing Article Submitted as an Input


online startup ideabator innovoice

WEB CONTENTS

Schneider Electric - Resource Library - TechRepublic ✗

Four scenarios where hybrid cloud makes sense - TechRepublic ✓

Open by design - O'Reilly Media ✓

What are your top tips for installing Windows 10? - TechRepublic ✗

Alleviate pain points associated with developing enterprise apps in the cloud - TechRepublic

Virtualization - Resource Library - TechRepublic ✗

Google's cloud business nabs Home Depot as customer | VentureBeat | Business | by Reuters ✓

TNW Deals
New to developing websites for the cloud? Your questions answered ✓

Get savvy on the latest cloud terminology - TechRepublic ✓

Apple OS X Server: How to configure a VPN service - TechRepublic ✗

Work smarter in 2016: Focus on these three IT topics - TechRepublic ✓

Migrating to cloud-native application architectures - O'Reilly Media ✓

VMware's five key cloud-native computing investments - TechRepublic ✓

Even VMware finds reaching 100% virtualization a challenge - TechRepublic ✓

CloudOps: 6 best ways to manage hybrid cloud performance ✓

Fig. Matched webpages on the topic of "Cloud Computing": Green – correct, Red – incorrect

We have grouped input documents by similar topics and have following results.

Most of the topics were selected from the field of entrepreneurship, startups, innovation, because the main focus of the system is to have this kind of materials. However for other topics results were excellent. All the documents and results are available on the following link.

Total for each topic:

Topics	Correct	Incorrect
Programming	23	7
Entrepreneurship/Startups	84	20
Marketing	33	12
Innovation/Leadership	55	20
Machine Learning	38	7
Cloud Computing	25	5
UI/UX Design	18	12
HTTP/WEP API/REST	28	17
SEO	18	12
IOT/Big Data	36	9
Mini PCs	18	12
Robotics	28	1
Artificial Intelligence	29	1
Cyber Security	22	8
Total	480	148

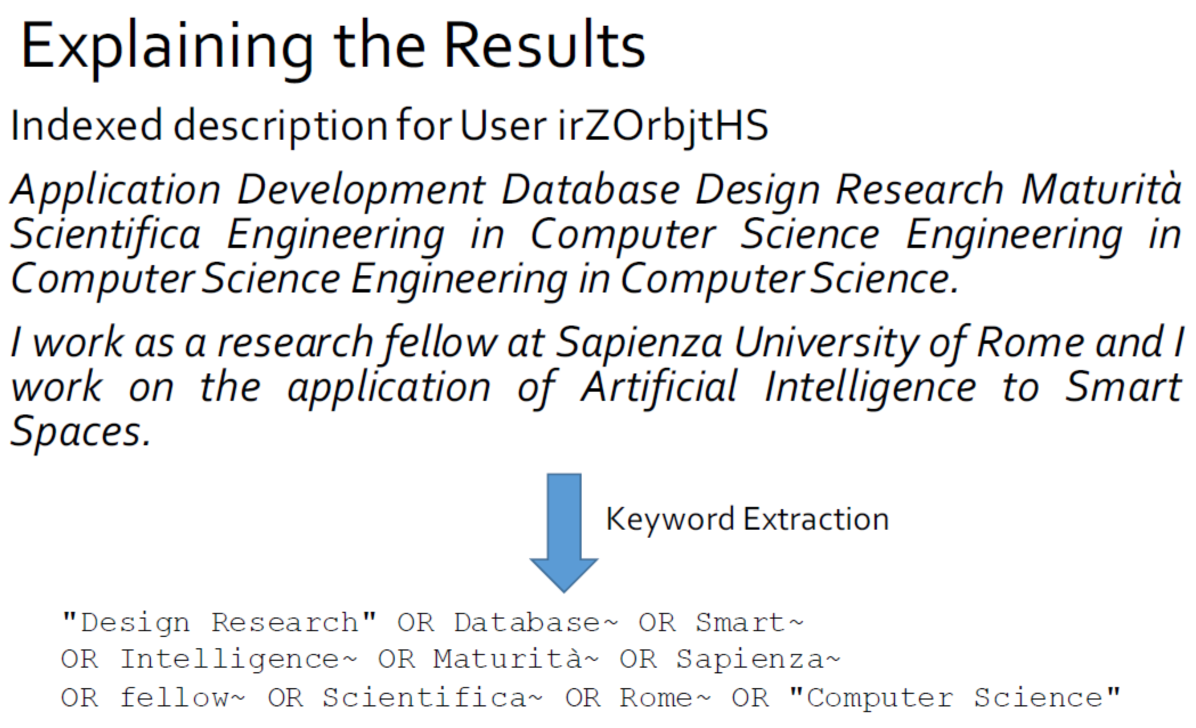
We can merge several topics in common fields. In the table below the performance of the system is listed in percentage for each field.

Category	Performance	Category	Performance
Overall	76%	Entrepreneurship/Innovation	77%
Marketing	73%	SW Development	74%
Artificial Intelligence	91%	Internet of Things	70%
Distributed Computing	81%	Internet Development	60%

Another test was done to see how accurate MAUI indexer extracts keywords from the documents before submitting the query generated from them to Apache Lucene. For this purposes a special web service was designed that returns the results by submitting a HTTP POST query. The query is in Json format and has the following structure:

```
{"text":"Some Text", "userId":”User”}
```

As an Output it returns in Json format the query and matched documents with score assigned and ranked according to that score. In all cases Lucene was able to extract keyphrases that could describe the topic. Of course, some of the extracted keywords were pointless, but the system accuracy didn’t suffer a lot. The minimal accuracy was 60%. Also system was able to output correct documents even if query keywords had grammatical errors, which proved Lucene Fuzzy search to be very effective. Another test was done to see how much the accuracy of the results depends on the amount of text in the input document. In most of the cases the bigger was text the better were results, because MAUI was able to identify keywords better since they occurred in text more. But in case of short text if the density of important keyphrases was high, system was again producing good results.



Explaining the Results

Indexed description for User irZOrbjtHS

Application Development Database Design Research Maturità Scientifica Engineering in Computer Science Engineering in Computer Science Engineering in Computer Science.

I work as a research fellow at Sapienza University of Rome and I work on the application of Artificial Intelligence to Smart Spaces.

```
"Design Research" OR Database~ OR Smart~
OR Intelligence~ OR Maturità~ OR Sapienza~
OR fellow~ OR Scientifica~ OR Rome~ OR "Computer Science"
```

Fig. Extracted keywords from user idea

Status of web contents extraction (as of 17-05-2016)

- **68262 web contents crawled from**:
    - 98 websites through crawling
        - 8 categories
            - IoT (2264), Tech (4637), Health (6110), Development (928), Entrepreneurship (50595), AI (2542), Pharma (94)
    - 6 YouTube channels
- **Quality check**
    - Filters on error pages
    - Main content identification
    - Clean-up of unreachable pages
- **The number of external contents grows day by day covering new sources and fresh contents**
    - Crawling started on a daily basis

7 Conclusion

In this paper, we have shown a sub-system of the InnovVoice platform, specifically focusing on the matchmaking of ideas with relevant content crawled from the Web. We argue that this is a valuable support for ideas proposers and innovators (and possibly entrepreneurs). Currently our subsystem collects information from over 100 sources on the Web related to entrepreneurship and technology (in particular AI - Artificial Intelligence, IoT - Internet-of-Things, and eHealth, etc.), and this set is continuously enlarged. The overall InnovVoice platform is running since a few weeks in a beta-testing mode, and therefore we will undergo a deeper validation of our approach in the following months. Future work includes improvements on all the presented layers by improving on the number of sources analyzed, and the intelligent techniques employed for keyword extraction and matching. This process will also take into account the feedback provided by users during the beta-testing evaluation of the InnovVoice platform.

8 References

[1] S. Maryville. Entrepreneurship in the business curriculum. *Journal of Education for Business* 68(1), pages 27-31, 1992.

[2] P. Frankelius. Questioning two myths in innovation literature. *Journal of High Technology Management Research 20(1)*, pp. 40-51, 2009.

[3] P. Drucker. The discipline of innovation. *Harvard Business Review*, August 2002.

[4] E. Von Hippel. *The sources of innovation*. Oxford University Press, 1988.

[5] Nunzio Giovinazzi1, Massimo Mecella1, *Information Retrieval, Market Trends Analysis and Forecast for Supporting Made-In-Italy: the DesigNET Prototype*, Sapienza - Universita di Roma

[6] Mining Sholom M. Weiss Nitin Indurkhya Tong Zhang, *Fundamentals of Predictive Text Mining Second Edition* 2015

[7] A. Hulth. Improved automatic keyword extraction given more linguistic knowledge. In Proceedings of the 2003 Conference on Emprical Methods in Natural Language Processing, Sapporo, Japan, 2003

[8] Michael J. Giarlo. *A comparative analysis of keyword extraction techniques*. Rutgers, The State University of New Jersey

[9] Chengzhi Zhang, Huilin Wang, Yao Liu, Dan Wu, Yi Liao, Bo Wang. *Automatic Keyword Extraction from Documents Using Conditional Random Fields. Journal of Computational Information Systems, 2008*

[10] I. Witten, G. Paynte, E. Frank, C. Gutwin, C. Nevill-Manning. *KEA: practical automatic keyphrase extraction. In Proceedings of the 4th ACM Conference on Digital Library, 1999*

[11] J. B. Keith Humphreys. Phraserate: *An HTML keyphrase extractor. Technical Report*. 2002

[12] Masanès, Julien (February 15, 2007). *Web Archiving*. Springer. p. 1. ISBN 978-3-54046332-0. Retrieved April 24,2014.

[13] Castillo, Carlos (2004). *Effective Web Crawling* (Ph.D. thesis). University of Chile. Retrieved 2010-08-03

[14] Koster, M. (1995). Robots in the web: threat or treat? ConneXions, 9(4).

[15] Koster, M. (1996). A standard for robot exclusion.

[16] Koster, M. (1993). Guidelines for robots writers.

[17] https://en.wikipedia.org/wiki/Deep_web_(search).

[18] Apache Nutch - http://nutch.apache.org/

[19] https://code.google.com/archive/p/maui-indexer/

[20] KEA - http://www.nzdl.org/Kea/

[21] https://www.mongodb.org/

[22] Apache Lucene - https://lucene.apache.org/

[23] O. Medelyan, E. Frank, I. H. Witten. 2009. Human-competitive tagging using automatic keyphrase extraction. To appear in Proc. of the Internat. Conference of Empirical Methods in Natural Language Processing, EMNLP-2009, Singapore.

[24] JATE - https://code.google.com/p/jatetoolkit/wiki/JATEIntro

[25] Apache OpenNLP - https://opennlp.apache.org/

[26] The Stanford Natural Language Processing Group - http://nlp.stanford.edu/software/

[27] TreeTagger - http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/

[28] ClearNLP - https://github.com/clir/clearnlp

[29] https://github.com/aneesha/RAKE

[30] http://tartarus.org/martin/PorterStemmer/

I. License