Serdica J. Computing **6** (2012), 163–174

Serdica
Journal of Computing

Bulgarian Academy of Sciences
Institute of Mathematics and Informatics

A DIGITAL LIBRARY OF FOLKLORE SONGS AND KEYWORD-BASED SEARCH ENGINE*

Kiril Kirov, Nikolay Kirov

Abstract. We present a full text search engine in a digital library of Bulgarian folklore songs—in the lyrics and scores of the songs. The deployment of the digital library on the cloud as well as the technical requirements for providing our data to Europeana are discussed.

1. Introduction. The Bulgarian folklore music is a valuable resource of cultural heritage. It is one of the main characteristics of the national identity of Bulgarian people. Throughout the 20th century the Bulgarian researchers of musical folklore wrote down hundreds of thousands of folklore songs in text and scores. Part of these song texts and music notations have been published, another part is preserved as manuscripts in specialized institutional or personal archives. The major part of the available music notations with Bulgarian national music however is on paper. In the second half of the 20th century researchers recorded

ACM Computing Classification System (1998): H3.3, H.5.5, J5.

Key words: digital library, search engine, cloud computing.

*This work is partially supported by Grant of the Bulgarian National Science Foundation under number DTK-02-54/2009 (see [22]).

– usually on magnetic tapes – performances of these songs by authentic singers. Today's information technologies give us the possibility to digitize the existing manuscripts and musical records [33], [34].

The search engine presented here was implemented in the Ruby programming language [29]. Its source code can be found at [15]. It can be used as a console or web application for keyword-based search in a library of authentic folklore songs. The folklore songs are provided as an index of digital content—lyrics, notes and images. This engine can be used by professionals in the field of folklore research for search of common motives, characters and similarities between folklore songs. These can be folklore songs from different parts of Bulgaria, variants of the same song or simply common keywords.

Hosting the digital library in the Amazon cloud [4] enable easy access from anywhere in the world. Cooperation with Europeana [9] acquaints a wider audience with our project and will contribute to promoting folk songs as a significant Bulgarian heritage.

2. Data Formats and Representation. Four types of files (sources) are used for representation of the songs in the digital library.

- LaTeX lyrics files. The texts of the songs in the library are written in the LaTeX typesetting system [24]. In addition to the song text, each LaTeX file also contains metadata for the song. This metadata is in the form of LaTeX commands or comments, and could be used both in compiling of the source and in generating the index.

- LilyPond score files. The scores of the songs are written in the LilyPond music typesetting system [25].

- MP3 digitized authentic performance files. These performances are digitized from magnetic tape libraries of the Archives of the Bulgarian Academy of Sciences. They have been recorded in various rural parts of Bulgaria during the 60s and 70s.

- JPEG digitized handwritten texts. The handwritten note-books included in this library were made by experts who worked with the authentic performers and then analyzed the collected data. They are a valuable source of information about the circumstances and different traditions associated with the performance of each particular song.

Examples of these file types can be found in [22].

3. Search Engine.

3.1. Input data. The basic data files can be divided into five types—four file types for representation of the songs in the library and the Ruby configuration file. The whole system must be configured from a Ruby config file. This file should contain the paths and file formats of all the content.

3.1.1. The Search Engine Index. The system uses Ferret [14] – a high-performance, full-featured text search engine library written for Ruby. It is inspired by Apache Lucene [6] and implemented in the C programming language. The `bin/index` command is used for building the search engine index, using the data provided in the configuration file (see Configuration). The search engine index has a table structure, with different fields provided by the different types of input files. The input files for indexing are lyrics and scores. The lyrics files can contain metadata, which is parsed and treated specially by the indexing engine. The metadata is preserved in the text fields of the index table, and can be used to form search queries.

3.1.2. Configuration. The search engine config file is `lib/folk.rb`—Ruby configuration file. The whole system must be configured from this file. It should contain the following data:

- Index path—the path to the index directory;
- LaTeX commands used for indexing—description of the LaTeX metadata commands that should be used for building the index;
- Google Maps API key—obtained from Google API access key;
- Number of CPU cores of the system—rebuilding the whole library could be a slow process. The system can speed it up by using different CPU cores for parallel recompilation.

3.1.3. Compilation. The compilation process can be started by `bin/lilypond` and `bin/latex`. It can take a relatively long time depending on the number of songs in the library, their complexity and the hardware parameters of the server. The compilation process generates the following output formats:

- LilyPond EPS and PDF engravings;

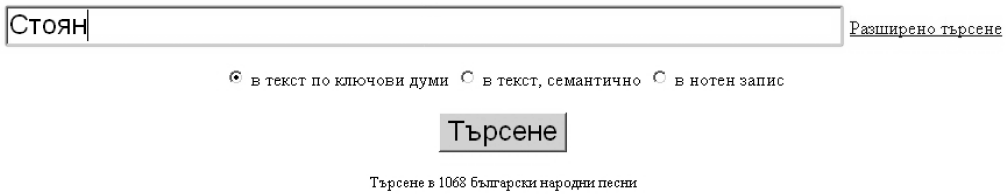


Fig. 1. Web interface for searching in the library

- LilyPond generated MIDI music;
- LaTeX PS and PDF lyrics.

3.1.4. Searching. This search engine provides a Google-like web interface that will be used for searching in the library (Fig. 1). It uses a search phrase (query) that must be written in the Ferret Query Language [13]. The search phrase can contain data as well as metadata, as defined in the configuration file. A short list of possible searches:

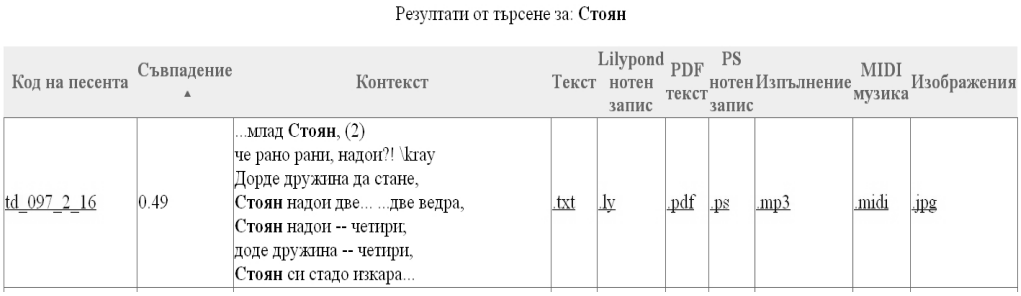
- `code:ba_002_2_04` – A code search. Every folklore song in the library has a unique code. The "code" is a separate field in the index table, so we specify a field using the syntax shown above.
- `Стоян` – A simple one word search for "Стоян" – a popular given name in Bulgaria. This search should return a result with all songs that contain this word. Note that "Стоян" could be the name of the singer or the name of the folklore character in the song.
- `content:"ожадня стоян за вода"` – A whole phrase search. Should return songs containing the given words in the given order. "content" is the lyrics field
- `ст*ян AND area\{ямболско\}` – A wildcard and boolean search. In the folklore songs the name "Стоян" is sometimes spelled "Стуян", so we want to match both of them. We also only want "ямболско" municipality, so we specify a metadata field which describes that area.
- `notes:fermata` – A score search. This search should return all songs which contains a `fermata` (an element of the musical notation) in their LilyPond coding.

3.2. Output data. The system presents data in web using an integrated Ruby web server stack. This stack includes several Ruby Gems [28]: Thin [32] (web server); Rack [27] (web server interface); Sinatra [30] (web development framework); HAML [19] (web page template system).

The output data are in two different categories – search result table and Google Maps visualization.

3.2.1. Search Result Table.

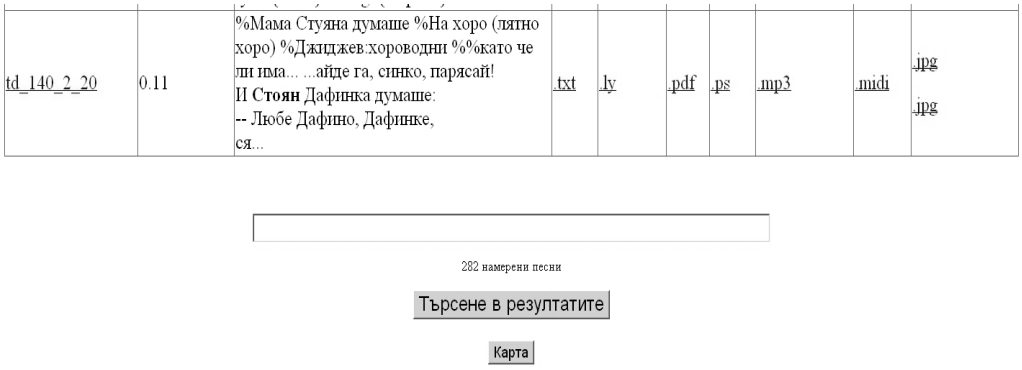
The search result table contains the songs that match a given search query (Figs 2, 3). Each song is represented by a row in that table, which contains all



Резултати от търсене за: **Стоян**

Код на песента	Съвпадение ▲	Контекст	Текст	Lilypond нотен запис	PDF текст	PS нотен запис	Изпълнение	MIDI музика	Изображения
td_097_2_16	0.49	...млад **Стоян**, (2) че рано рани, надои?! \kray Дорде дружина да стане, **Стоян** надои две... ...две ведра, **Стоян** надои -- четири, доде дружина -- четири, **Стоян** си стадо изкара...	.txt	.ly	.pdf	.ps	.mp3	.midi	.jpg

Fig. 2. The top of the search result table



td_140_2_20	0.11	%Мама Стуяна думаше %На хоро (лятно хоро) %Джиджев:хороводни %%като че ли има... ...айде га, синко, парясай! И **Стоян** Дафинка думаше: -- Любе Дафино, Дафинке, ся...	.txt	.ly	.pdf	.ps	.mp3	.midi	.jpg .jpg

282 намерени песни

Търсене в резултатите

Карта

Fig. 3. The bottom of the search result table

the metadata in the library about that song. Every song is identified by its unique code. By default the search result table is sorted by the relevance index given by the search engine – the best matches are shown first. In addition to that the user can sort the table by any field (column). This happens in the web browser using

a JavaScript sort table script. The context of the given match is also displayed in the search result table, so the user can for example see the specific stanzas, that contain the given word. Links to the lyrics, notes and images are provided. The user can also hear the authentic performance online, by clicking on the given MP3 link for the specific song in the search results. A compiled, MIDI version of the Lilypond source file can also be heard. That could be used as a reference between written scores and the authentic performance.

Because of the specific grammar of the Ferret Query Language [13], forming exact search queries could be difficult. So the user has an option to form a loose query, which could match a large number of songs, just to see how it works. Then he can refine the query further and run a new search, but this time not in the whole library, but only in the results of the first, broader search.

3.2.2. Google Maps Visualization. In every step of the search process a link that can visualize the resulting songs in Google Maps [18] is provided. The system extracts the relevant metadata from the index and forms a series of Google Maps queries that should return the exact location (or locations) associated with each given song. These queries are formed as strings containing the name of the town or village, where the song was performed or that it is associated with, and the municipality in which that town or village is located. Since Google uses keyword based search, that pair should be enough to distinguish between names of villages located in different municipalities (which is a common occurrence in Bulgaria). Google Maps queries return GPS coordinates and a JavaScript based map, on which each song location is visualized. By using this technique a user could figure out how a given song motive is spread across rural areas of Bulgaria. It can also be used to track the locations associated with different singers (Fig. 4).

4. Development and Deployment on the Cloud. This application could be deployed to a cloud computing platform. The Ruby web server stack presented above is supported on many cloud computing providers, for example it could run on Google App Engine [17], Heroku [20], Engine Yard [8] and etc. Maintaing a dedicated server hardware for this application is not cost effective. The nature of the application and the problem field prove that it is difficult, but not impossible to adapt it for the Cloud. Several deployment and development schemes were analyzed. It turned out that a combination between Amazon EC2 [4] and Amazon EBS [2] would be optimal solution, both technical and cost effective.

4.1. Amazon EC2. Amazon Elastic Compute Cloud (EC2) is a central part of Amazon.com's cloud computing platform, Amazon Web Services (AWS).

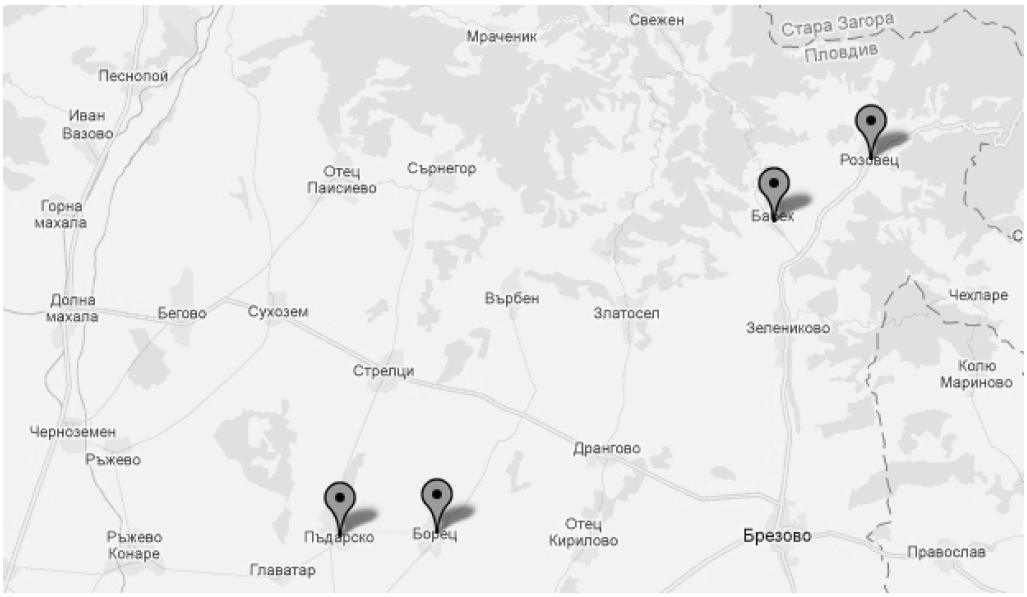


Fig. 4. Google Maps visualization

EC2 allows users to rent virtual computers on which to run their own computer applications. EC2 allows scalable deployment of applications by providing a Web service through which a user can boot an Amazon Machine Image to create a virtual machine, which Amazon calls an "instance", containing any software desired. A user can create, launch, and terminate server instances as needed, paying by the hour for active servers, hence the term "elastic". EC2 provides users with control over the geographic location of instances that allows for latency optimization and high levels of redundancy [5].

This application currently uses one micro instance running Ubuntu.

4.2. Amazon EBS. Amazon Elastic Block Storage (EBS) provides raw block devices that can be attached to Amazon EC2 instances. These block devices can then be used like any raw block device. In a typical use case, this would include formatting the device with a filesystem and mounting said filesystem. In addition EBS supports a number of advanced storage features, including snapshotting and cloning. Currently EBS volumes can be up to 1TB in size. EBS volumes are built on replicated back end storage, so that the failure of a single component will not cause data loss [3].

This application currently uses two 8 GiB instances.

4.3. Lilypond and LaTeX. Since Lilypond and LaTeX can be run on the cloud computing platform, this is not required. The application can use the local computer for compilation of the input files. However Amazon EC2 provides unlimited CPU usage, even for micro instances. That allows the compilation process to run inside the EC2 instance.

4.4. Scalability. Developed in the presented way, this project could scale infinitely. The computing resources provided by Amazon EC2 could be expanded, based on the application usage and needs. The storage devices on Amazon EBS could provide multiple terabytes. Developers and users could work in parallel, developing new features of the system, and domain experts could create and add new content. The application now have 6.7 Gb of data for about 1080 folk songs that consist of:

- EPS files – 1 Gb
- MIDI files – 1 Mb
- PDF files – 1 Gb
- MP3 files – 4 Gb
- JPG files – 500 Mb

5. Europeana. Europeana.eu [9] is an internet portal that acts as an interface to millions of books, paintings, films, museum objects and archival records that have been digitised throughout Europe. Mona Lisa by Leonardo da Vinci, Girl with a Pearl Earring by Johannes Vermeer, the works of Charles Darwin and Isaac Newton and the music of Wolfgang Amadeus Mozart are some of the highlights on Europeana. More than 2000 institutions across Europe have contributed to Europeana. These range from major international names like the Rijksmuseum in Amsterdam, the British Library and the Louvre to regional archives and local museums from every member of the EU. Together, their assembled collections let users explore Europe's cultural and scientific heritage from prehistory to the modern day [11].

5.1. Technology. Europeana provides a metadata format that should be used for providing collections of items for their site. This metadata contains the identifier, description, location and the type of the item. It is in XML format, validated with the Europeana Semantic Elements schema [12]. A collection of

record tags should be contained within the metadata tag. Each record has a text, image, sound or video type. We have a collection of folk songs. A folk song in our collection could be described like this [23]:

```
<?xml version='1.0' encoding='UTF-8'?>

<metadata xmlns="http://www.europeana.eu/schemas/ese/"
          xmlns:europeana="http://www.europeana.eu/schemas/ese/"
          xmlns:dc="http://purl.org/dc/elements/1.1/"
          xmlns:dcterms="http://purl.org/dc/terms/">

  <record>
    <dc:identifier>BA,001,2,09</dc:identifier>
    <dc:title xml:lang="bg">Снощи отидох, мамо</dc:title>
    <!-- <dc:alternative xml:lang="en">Bulgarian folklore song</dc:alternative>
    -->  <dc:subject xml:lang="bg">Българска народна песен</dc:subject>
    <dc:description xml:lang="bg">Лазарска народна песен (текст)</dc:description>
    <dc:contributor xml:lang="bg">Тодор Джиджев (записвач)</dc:contributor>
    <dc:contributor xml:lang="bg">Стоян Иванов Вандов (изпълнител)
                                                          </dc:contributor>
    <dc:coverage xml:lang="bg">Кортен</dc:coverage>
    <dc:coverage xml:lang="bg">Сливенско</dc:coverage>
    <dc:publisher>Magrathea Information Technologies</dc:publisher>
    <dc:type>Text</dc:type>
    <dc:format>text/pdf</dc:format>
    <dc:date>1908</dc:date>
    <dc:rights xml:lang="bg">Институтът за етнология и фолклористика с
                                          Етнографски музей</dc:rights>
    <dc:language>bg</dc:language>
    <europeana:object>https://folk.magrathea.bg/pdf/ba_001_2_09/0
                                                    </europeana:object>
    <europeana:provider>Bulgarian Academy of Sciences</europeana:provider>
    <europeana:type>TEXT</europeana:type>
    <europeana:rights>http://www.europeana.eu/rights/unknown/</europeana:rights>
    <europeana:dataProvider>Institute of Ethnology and Folklore Studies
                          with Ethnographic Museum</europeana:dataProvider>
    <europeana:isShownAt>https://folk.magrathea.bg/pdf/ba_001_2_09/0
                                                    </europeana:isShownAt>
  </record>
</metadata>
```

This XML metadata catalog is compiled using a Rakefile [26] task from the internal metadata representation of the projects. Since Europeana does not provide a dynamic site integration infrastructure, this task must be run and the compiled file sent to them every time a new song is added, or an existing song changed. Because our object folklore song consists of three parts – text, score

and sound, we can describe each part separately, for example in three XML files of the above type. Thanks to the robustness of Europeana Semantic Elements, there are many ways of describing our metadata. Different possibilities will be evaluated in the future. The above example is only one of the possible ways of system integration that passe the Europeana Content Checker [10] validation.

5.2. Searching Europeana. Users of Europeana can form search queries based on the provided metadata. The collection of search results is structured and visualized in the context of the Europeana site. There is a separate web page for each item, which provides a link to our system. None of the song data are hosted within Europeana. Currently the intelectual rights on folklore songs in Bulgaria are undefined, so we use the "Unknown" license field in Europeana Semantic Elements, and the data files are password protected. We believe that upon clarification of the intelectual right issues, we could provide open access to the folklore songs and license them to the general public using some version of Creative Commons License [7].

6. Future Development.

6.1. AST Analysis. During the LilyPond compilation, an Abstract Syntax Tree [1] is generated, based on the syntax of the input file. This syntax tree is interesting, because it provides all the necessary information for a possible automated song analysis and additional metadata generation. Luckily the LilyPond provides access to the AST by the means of a Scheme programming language API [31]. The system could provide a way to load an externally supplied Scheme script for automated analysis of the compiled LilyPond scores.

6.2. Use on the Internet. In its current state the folklore songs search engine library is highly experimental and meant to be used only by experts and folklore professionals. The web interface of the system could be improved and redesigned according to web standards. Such improvement would make the system easier to use for professionals in the field and also usable to the interested communities on the Internet. The system is in no way limited specifically for Bulgarian folklore songs. It could be used for archiving, digitizing and indexing of different collections of authentic folklore art in the world at large.

REFERENCES

[1] Abstract Syntax Tree. http://en.wikipedia.org/wiki/Abstract_syntax_tree

[2] Amazon EBS. http://aws.amazon.com/ebs/

[3] Amazon EBS from Wikipedia. http://en.wikipedia.org/wiki/Amazon_EBS

[4] Amazon EC2. http://aws.amazon.com/ec2/

[5] Amazon EC2 from Wikipedia. http://en.wikipedia.org/wiki/Amazon_EC2

[6] Apache Lucene. http://lucene.apache.org/java/docs/index.html

[7] Creative Commons. http://creativecommons.org/

[8] Engine Yard. http://www.engineyard.com/

[9] Europeana.eu. http://www.europeana.eu

[10] Europeana Content Checker. http://europeana-contentchecker.isti.cnr.it:8080/portal/

[11] Europeana from Wikipedia. http://en.wikipedia.org/wiki/Europeana

[12] Europeana Semantic Elements. http://www.europeana.eu/schemas/ese/ESE-V3.4.xsd

[13] Ferret Query Language. http://www.davebalmain.com/api/classes/Ferret/QueryParser.html

[14] Ferret Search Engine. http://www.davebalmain.com/trac

[15] GitHub repository. https://github.com/kirilk/folk

[16] Git. http://git-scm.com/

[17] Google App Engine. http://code.google.com/appengine/

[18] Google Maps, GIS. http://maps.google.com/

[19] HAML, HTML templates. http://haml-lang.com/

[20] Heroku. http://www.heroku.com/

[21] Heroku from Wikipedia. http://en.wikipedia.org/wiki/Heroku

[22] Information technologies for presentation of Bulgarian folk songs with music, notes and text in a digital library. http://nikolay.kirov.be/2010/folk

[23] Kiril Kirov's Weblog. http://blog.kirov.be/2012/02/04/europeana/

[24] LaTeX typesetting system. http://www.latex-project.org/

[25] Lilypond music engraving program. http://lilypond.org/

[26] Rake, A software task management tool. http://rake.rubyforge.org/

[27] Rack, web server interface. http://rack.rubyforge.org/

[28] Ruby Gems. http://rubygems.org/

[29] Ruby, Programming language. http://www.ruby-lang.org

[30] Sinatra, web framework. http://www.sinatrarb.com/

[31] The Scheme Programming Language. http://www.scheme.com/tspl3/

[32] Thin, web server. http://code.macournoyer.com/thin/

[33] Peycheva L., N. Kirov, M. Nisheva-Pavlova. Information Technologies for Presentation of Bulgarian Folk Songs with Music, Notes and Text in a Digital Library. In: Proc. of the Fourth Int. Conf. "Information Systems & Grid Technologies", Sofia, Bulgaria, May 28—29, 2010, 218–224.

[34] Peycheva L., N. Kirov. Bulgarian Folk Songs in a Digital Library. In: Proc. of the Int. Conf. Digital Preservation and Presentation of Cultural and Scientific Heritage (Eds R. Pavlov, P. Stanchev), 11–14 September 2011, Veliko Tarnovo, 60–68.

Kiril Kirov
Magrathea Ltd.,
e-mail: kiril@kirov.be

Nikolay Kirov
Department of Informatics
New Bulgarian University,
21, Montevideo Str.
1618 Sofia, Bulgaria
e-mail: nkirov@nbu.bg

Received March 2, 2012
Final Accepted April 19, 2012