Efficient search in hidden text of large DjVu documents*

Janusz S. Bień†

September 12, 2011

Abstract

The paper describes an open-source tool which allows to present end-users with results of advanced language technologies. It relies on the DjVu format, which for some applications is still superior to other modern formats including PDF/A. The DjVu GPLed tools are not limited just to the DjVuLibre library, but are being supplemented by various new programs, such as pdf2djvu developed by Jakub Wilk. It allows in particular to convert to DjVu the PDF output of popular OCR programs like FineReader preserving the hidden text layer and some other features.

The tool in question has been conceived by the present author and consist of a modification of the Poliqarp corpus query tool, used for National Corpus of Polish; his ideas have been very succesfully implemented by Jakub Wilk. The new system, called here simply Poliqarp for DjVu, inherits from its origin not only the powerfull search facilities based on two-level regular expressions, but also the ability to represent low-level ambiguities and other linguistic phenomena. Although at present the tool is used mainly to facilitate access to the results of dirty OCR, it is ready to handle also more sophisticated output of linguistic technologies.

1 DjVu technology and DjVuLibre

The DjVu technology, described by its authors as *an image compression technique, a document format, and a software platform for delivering documents images over the Internet* [Le Cun et al., 2001, p. 2] was originally developed by Yann Le Cun, Léon Bottou, Patrick Haffner, and Paul G. Howard at AT&T Laboratories in 1996. AT&T Laboratories acquired several patents for some aspects of the technology, but didn't offer any product using or supporting DjVu[1]. The broad rights to the patents have been purchased by LizardTech

*This is an updated version of the paper which appeared in Bernardi, Raffaella and Chambers, Sally and Gottfried, Björn and Segond, Frédérique and Zaihrayeu, Ilya (eds.), *Advanced Language Technologies for Digital Libraries*, Lecture Notes in Computer Science 6999, Springer Berlin / Heidelberg, pp 1-14, 2011, DOI 10.1007/978-3-642-23160-5_1 (http://dx.doi.org/10.1007/978-3-642-23160-5_1).

†Formal Linguistics department, University of Warsaw,Browarna 8/10, 00-927 Warszawa, Poland, jsbien@uw.edu.pl, http://www.klf.uw.edu.pl.

[1]Although the patents in question are valid only in USA, they definitely delayed the practical applications of the format (fortunately software patents are not allowed at all in European Union and a lot of other countries).

(it later became a part of Celartem Technology Inc., which in 2009 appointed Caminova Inc. "to develop, distribute and manage its DjVu document imaging technology", cf. `http://www.caminova.jp/en/`), which in 2001 allowed to use patented techniques in the software distributed under the GNU General Public License; as the wording of the statement was considered unprecise, in 2002 it was supplemented by an additional clarification. The implementation of the DjVu technology available on the GNU GPL licence is called DjVu-Libre. It is worth reminding that GNU GPL provides the user with 4 freedoms (`http://www.gnu.org/philosophy/free-sw.html`):

1. The freedom to run the program, for any purpose.
2. The freedom to study how the program works, and adapt it to your needs.
3. The freedom to redistribute copies so you can help your neighbor.
4. The freedom to improve the program, and release your improvements to the public, so that the whole community benefits.

In consequence, it is most appropriate for academic research.

DjVu has several features. First of all, it provides very efficient algorithms for image compression; the best of them are still available only in the form of commercial and quite expensive products. Secondly, it provides an efficient way to transfer the compressed images over the Internet, even on relatively slow lines. Moreover, it provides also an efficient way to display the image on the end-user's computer, using such tricks as progressive decoding (which decompresses only this part of the image which is to be displayed), downloading the next page in the background etc.

DjVu allows to store every page in a separate file and download only the pages which are really needed, which is of crucial importance especially for large dictionaries, which are not read in a sequential way. Another feature of crucial importance is the possibility to accompany the scans by the hidden text layer, which can be searched, copied etc.

From a user's point of view it is the DjVu viewer which is important. There exist several of them, both commercial and free, for various platforms, palmtops and cellular phones included. All the viewers profit from the DjVu design features allowing the viewer to simulate the operations on a paper document in comparable time, as illustrated by the table 4 in [Le Cun et al., 2001, p. 6]:

Action	Real-word equivalent	Acceptable delay
Zooming/Panning	Moving the eyes	Immediate
Next/Previous Page	Turning a page	< 1 second
Random Page access	Finding a page	< 3 seconds

From the very beginning, DjVu viewers allowed to highlight specified fragments of a remote text. For example, the address

```
http://www.leoyan.com/century-dictionary.com/04/index04.djvu?djvuopts=
&page=p2719.djvu&zoom=100&showposition=0.48,0.34&highlight=1084,
3451,1004,344
```

points to the entry *hardware* in the online edition of the famous *The Century Dictionary and Cyclopedia* (published from 1888 to 1891), referenced also later in the paper. The main part of the address describes the primary document

file, which in this case is just an index to the files containing individual pages of the 4th volume of the dictionary. The parameter `page` describes the page using its name which happens to coincide with the name of the file containing it. The `highlight` parameter specifies pixel coordinates of the rectangle to be highlighted, and the `showposition` part guarantees that the visible area of the page will contain the highlight.

This very useful feature was however very little used because there was no easy way to identify the coordinates of the area to be highlighted. Therefore in 2008 I asked Jakub Wilk (then a student of mine) to extend djview4 allowing to create such URLs conveniently after marking a region with a mouse. The patch has been submitted to the Sourceforge tracking system on 9th February and by 29th February it has been reimplemented more efficiently by Léon Bottou, the author of the program, who included it in the official distribution. I think this feature is extremely important for academic research, as it allows to quote a specific fragment of a digitalized work when including its image is technically difficult or not desirable.

When accessing a document with a highlighted fragment, the page is displayed in the default resolution and in the default position, so it could happen that the highlighted fragment is not immediately visible. The free but closed source LizardTech viewer for MS Windows had a solution to the problem in the form of the `ShowPosition` parameter. In May 2008 I asked for an identical feature in djview4 and just several months later (in June 2008) Léon Battou implemented it. So if you send an URL referring to a highlighted fragment of text, the receipient will see it exactly as the sender (with some minor exceptions).

2 DjVu and Portable Document Format

Portable Document Format (PDF) is an open standard (formally since July 1, 2008) for document exchange introduced by Adobe Systems in 1993. A subset of the specification is known as PDF/A and described in the international standard ISO 19005-1:2005 *Document management – Electronic document file format for long-term preservation – Part 1: Use of PDF 1.4 (PDF/A-1).*

Reportedly already version 1.0 of the specification allowed to create "sandwich PDF" containing both the scans and hidden text layers, predating in this respect DjVu, which however for years provided better compression (at present the compression ratio is comparable) and is still in many aspects more convenient.

Thanks to the open character of the PDF standard it became very popular, both as the output of scanning programs and stand-alone scanners, and as an input for printing, ranging from personal printers to professional devices. Moreover "sandwich PDF" is used also as the output format of many OCR programs, including the widely-used Abby FinerReader.

To have the best of both worlds, in 2008 Jakub Wilk created the first version of the pdf2djvu program, which he has since then actively maintained and developed; the software is hosted at `http://code.google.com/p/pdf2djvu/`. It is released under the terms of the GNU General Public Licenses and available in the package form in major free operating system (GNU/Linux and FreeBSD) distributions, such as Debian, Ubuntu and OpenSuse; it can be compiled also for MS Windows. The current version of the program is 0.7.10 (released on 20th

August 2011) and supports such features as

- compressing the scans the DjVu way, trying to split them into front and background;
- optionally preserving hidden text;
- optionally preserving the document outline;
- optionally preserving hyperlinks (with some limitation intrinsic for the DjVu format);
- optionally preserving and updating the document metadata.

The program is able in particular to preserve and update the metadata in the XMP format; XMP stands for Extensible Metadata Platform (`http://www.adobe.com/products/xmp/`) which is becoming more and more popular.

The expensive commercial DjVu document creators provide better compression than pdf2djvu, but are available only for MS Windows and include built-in OCR programs which cannot be controlled by the user. In consequence, *pdf2djvu* used alone or with an OCR program of choice is a viable competitor in many circumstances.

3 Searching the hidden text layer

Every DjVu viewer allows for searching the hidden text layer, but for large remote documents it is inefficient as it defeats the purpose of splitting the document into separate pages: to access the hidden text, all the pages have to be loaded, and if the search is repeated, they are reloaded multiple times. On the other hand, if the document is available locally, djview4 offers very efficient and convenient incremental search which seems to be absent in other viewers.

Hence, the optimal solution is to use some kind of index and a search engine. Yann LeCun, one of the creators of the DjVu format, implemented JSSindex (JavaScript Search Engine, `http://sourceforge.net/projects/jssindex/`), an interesting search tool for collections of documents in HTML, PS, PDF, and DjVu, but unfortunately oriented only at English language texts and very difficult to modify and extend. A simple search engine has been provided for *Century Dictionary Online* (`http://www.global-language.com/CENTURY/`) mentioned earlier. Although it looks like this is a special purpose software written for the specific task, this electronic edition created by Jeffery A. Triggs sets standards for an efficient and convenient access to DjVu documents. Another electronic edition prepared by Triggs is *Jamieson's Etymological Dictionary of the Scottish Language Online* (`http://www.scotsdictionary.com/`); it allows to choose between two search engines: Hunter and Amberfish. Hunter is commercial software developed by Alternative Output Inc. (`http://www.alternativeoutput.com/`), used by a few customers, one of them being Oxford University Press, which reportedly uses it for the online version of *Oxford English Dictionary*. Amberfish is an open source text retrieval system developed by Etymon Systems; the company seems to no longer exist, but the software is still available at `http://sourceforge.net/projects/amberfish/` and `https://github.com/nassar/amberfish`.

Although general purpose search engines are quite useful, there is a whole family of interesting software which treats texts as linguistic objects, namely corpus management software. One of the most sophisticated systems of this type is Poliqarp (*Polyinterpretation Indexing Query and Retrieval Procesor*), an open source tool developed in the Institute of Computer Science of Polish Academy of Sciences (`http://poliqarp.sourceforge.net/`). It has been in use for several years, now also for the National Corpus of Polish (`http://nkjp.pl/`); this should guarantee its continuous maintenance. An important factor is also a user community familiar with its query language. The maintainer of Poliqarp and implementor of the extensions designed primarily by Adam Przepiórkowski (cf. [Przepiórkowski, 2009]) was till recently Jakub Wilk.

The Poliqarp query language has been inspired by Corpus Query Processor, a component of Corpus Workbench developed at the University of Stuttgart (now an open source system, cf. `http://cwb.sourceforge.net/`, but it was not so when the development of Poliqarp started). The basic principle is to use two levels of regular expressions. One level is applied to strings representing the values of linguistic features of a word, the actual spelling of the word being one of them. The second level of regular expressions is applied to words or their sets defined with the first level expressions. In consequence the query language is very powerful (it seems that practically all queries available in e.g. Hunter and Amberfish mentioned above can be expressed in Poliqarp), but less user-friendly than in simpler systems.

The idea to use Poliqarp for searching hidden text of DjVu documents has been conceived by the present author in 2008 and formulated first as a term project for Computer Science students. The background and the results of this preliminary attempt were presented in [Bień, 2009a]. A research grant allowed to implement later a more efficient and elegant solution described below, and to support the development of some other tools mentioned in the paper.

The results of the search in the hidden text layer may be successful only if the text really represents the content of the scan. Usually it is not the case as the hidden text layer is created by 'dirty OCR', i.e. an unattended OCR process. Hence it is important to estimate easily the quality of the hidden text. Upon my request of May 2008 Léon Bottou in a few days included in djview4 the possibility to display hidden text for the scan fragment under the cursor; another added feature is the possibility to display the whole hidden layer at once. It allows e.g. to spot the OCR errors which are to blame if the search misses a target (such errors can be now corrected with the help of Jakub Wilk's program djvusmooth available in several Linux distribution including Debian Squeeze; the program is still under development, so it should become more convenient to use in the near future). On the other hand the same purpose can be served by graphical concordances mentioned below.

4 Poliqarp for DjVu

Poliqarp for DjVu, also known under the code name marasca-wbl, is an extension of Poliqarp allowing, at least in principle, to use the full power of the program to search hidden text in DjVu documents. Its development is one of the tasks supported by the Polish Ministry of Science and Higher Education's grant entitled *Text digitalization tools for philological research.* The source of the system is

available under the terms of the GNU GPL license at `https://bitbucket.org/jwilk/marasca-wbl`. It is worth noting that although at first the system was just a modification of Poliqarp, we contribute in return to the original project. Since March 2010 the National Corpus of Polish has used our version of the WWW Poliqarp client (`https://bitbucket.org/jwilk/marasca`).

Poliqarp for DjVu was implemented by Jakub Wilk according to the design of the present author. It has been available for testing since December 2009 at `http://poliqarp.wbl.klf.uw.edu.pl`. It operates by augmenting a standard Poliqarp corpus with information about the bounding box coordinates of the text tokens. The text and the coordinates are provided in hOCR format [Breuel, 2007] generated with the djvu2hocr program bundled with Jakub Wilk's ocrodjvu software (`http://jwilk.net/software/ocrodjvu`). Thanks to pdf2djvu it allows to apply Poliqarp for DjVu to the results produced by practically all important OCR programs. Moreover, recently a converter from the PAGE (Page Analysis and Ground-truth Elements) format [Pletschacher and Antonacopoulos, 2010] to hOCR has been developed, which allows Poliqarp to handle, at least in principle, numerous texts prepared in the very format by the so called library partners in the framework of the IMPACT project (*IMProving ACcess to Text*, `www.impact-project.eu`).

As of September 2011, four important Polish dictionaries are available for testing Poliqarp for DjVU:

- "Warsaw dictionary", more precisely *Słownik języka polskiego* (Dictionary of the Polish Language) by J. Karłowicz, A. Kryński and W. Niedźwiecki published in Warsaw in 8 volumes in 1900–1927. It has been scanned by the library of the University of Warsaw, which used Abby FineReader 8 for OCR; the results contain many mistakes but seem to be usable.

- *Słownik polszczyzny XVI wieku* (Dictionary of the 16$^{\text{th}}$ century Polish). The work started in 1949 and is still in progress. Its digitalization has complex history, which has been described elsewhere (cf. [Piotrowski, 2005] and [Bień, 2009b]). Since December 2010 all the 34 already published volumes have been available. Most of them are scanned and the OCR is, unfortunately, of rather low quality. Thanks to the sponsor of the dictionary, Foundation for Polish Science, which recently made publication on the Internet a formal requirement for further funding, the last two volumes are digitally born; the same files that were used for printing were converted by Jakub Wilk with his pdf2djvu program, so the physical and electronic versions have the same appearance and content. Two earlier volumes were preserved in the internal format of the typesetting system used; when typeset again, the resulting PDF files have slightly different appearance due to some minor changes in the system and fonts. As the content remained identical, these volumes are also available as digitally-born.

- Second edition of Linde's dictionary. *Słownik języka polskiego* (Dictionary of the Polish language) by Samuel Bogumił Linde were published in 4 volumes (two of them are split into two parts, so it makes actually 6 volumes) in 1807-1814, the second edition has been published in 1854-1861. This is one of the most important historical dictionaries not only from the Polish point of view, as all definitions are also given in German and there

is a lot of quotations from other languages (including Old Slavonic, Greek and even Hebrew) and dialects, some of them already extinct. The mixture of languages and scripts makes OCR extremely difficult; at present the hidden text layer has been prepared with Abby FineReader 10 set to Polish language. In consequence the fragments in Polish are of quite good quality, while the remaining parts are completely unusable; this is however already a sufficient help for readers trying e.g. to locate an entry, which are ordered according to rules which are different from contemporary ones. We have some plans to improve the quality of the hidden text, but this is outside the scope of the present paper.

- *Słownik geograficzny Królestwa Polskiego i innych krajów słowiańskich* (The Geographical Dictionary of the Polish Kingdom and other Slavic Countries), a gazetteer in 15 volumes of almost 1000 pages each, published in 1880-1914, extremely useful for genealogical research. The gazetteer covers Poland in its borders before the partitions between Russia, Germany and Austria, but due to the censorship it was impossible to state this explicitly in the title.

From a user's point of view, Poliqarp for DjVu enhances Poliqarp proper with functionalities present already in *The Century Dictionary Online* and *Jamieson's Etymological Dictionary of the Scottish Language Online*, namely with linking hits (keywords in the KWIC index) to the scans with highlighted hits. To quickly sort out false positives caused by the low quality of "dirty OCR", Poliqarp for DjVu additionally provides so called graphical concordances, i.e a KWIC index with the scan snippets created on the fly. Figure 1 shows a graphical concordance for a non-trivial query in Linde's dictionary. The purpose of the query is to find the occurences of the abbreviation *Syr.* meaning *Syryjski* (i.e. Syriac [language]). The problem is that the same abbreviation refers also to *Syreniusza zielnik* (i.e. Syreniusz' herbarium), but in such a case it is followed by a page reference in the form of a number. Hence regular expression

```
Syr "\." "[^[:digit:]].*"
```

specifies 3 tokens:

1. the character string `Syr`,
2. a full stop,
3. a token that does not start with a digit.

Before going into the details of the regular expression syntax let us note that most of the hits are obviously correct. Hit number 2 is a false positive due to an OCR error, the digit has been misinterpreted as a letter. Hit number 4 may seem incorrect, but actually this is a result of size limitation of the displayed snippet.

Let us have a look now at an example illustrating how the power of regular expressions can be used to circumvent the OCR errors. The following expression

```
("[CĆOGU]ze[sś]" | "[CO][z/]o[sa]") "\."
```

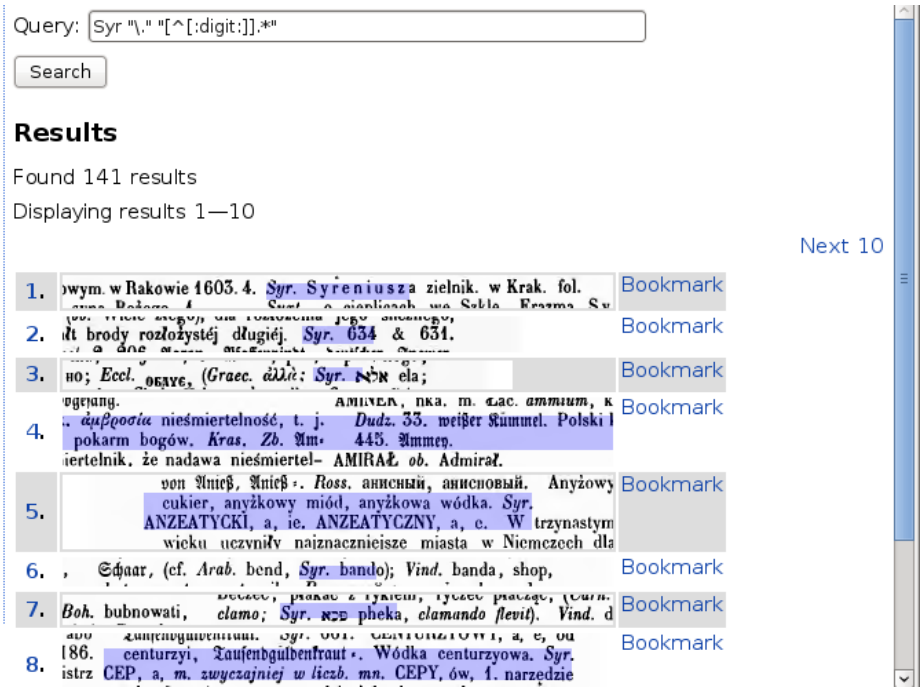

Figure 1: Graphical concordances in Poliqarp for DjVu

seems to match all the occurences of the abbreviation *Czes.* (meaning Czech language) in the Warsaw dictionary, which has been recognized as `Cześ`, `Gzes`, `Czos`, `Ozos` etc., as illustrated in figures 2 and 3.

Let us analyze the structure of the query. The top level of the query consists of three second level regular expressions and has the structure

```
(RE1 | RE2) RE3
```

which means that we are searching for `RE3` immediately preceded either by `RE1` or by `RE2`.

Expression `"\."` denotes simply a full stop ending the abbreviation. Because the full stop in regular expressions means "any character except new line" (in this meaning it occurs close to the end in the first example), it has to be escaped with backlash to recover its standard meaning. Quotes are needed to distinguish the levels of regular expressions.

Expression `"[CĆOGU]ze[sś]"` matches words consisting of 4 characters. The second and third one must be respectively `z` and `e`, the first and last may be any character from the respective bracketed list. If such a list starts with `^`, it means the the list specifies characters which are not allowed, as in our first example.

The bracketed list may contain also predefined names of character classes, as exemplified by `[:digit:]` in the first example. Another use of this construct is demonstrated by a query usefully applicable to the dictionary of the 16th century Polish:

```
"[[:upper:]]{3,}" within body meta orig=pdf
```

Figure 2: Graphical concordances for dirty OCR

It allows to search for headwords, always spelled in capitals. The query matches also the Roman numbers referring to centuries, but it doesn't do much harm and avoiding this makes the query much more complex. The results are presented in figure 4.

The top level regular expression is simple and consists of only one component, it is however supplemented by two clauses. The first clause limits the search to the section named `body`; sections are defined during the corpus building, in our case this sections refers to the part of dictionary containing the entries. The second clause refers to metadata assigned to the publications included in the corpus. In our case this is non-standard metadata which allows to limit our search to digitally-born volumes.

The second level expression consists of two parts: the character specification `[[:upper:]]` and the quantifier `{3,}`. The character specification is just a single element bracketed list, and the element is the name of a character class (also written in brackets); the class `[:upper:]` denotes, as expected, all upper case characters; the meaning of "all" depends on an operating system property called locale, but can be safely assumed to mean at least all characters present in the Basic Multilingual Plane of the Unicode standard (`www.unicode.org`).

The quantifier `{3,}` means that the preceding element has to occur in a word at least three times; in the case of our dictionary it means that we skip

Figure 3: Standard concordances for dirty OCR

the initials of authors (in the dictionary every entry is signed by its author) but match the head of entries longer than two letters. Other popular quantifiers are: `*` (the preceding element occurs any number of times or does not occur at all; the construct was used in the first of our examples), `+` (the preceding element occurs at least once), `?` (the preceding element occurs at most once).

The regular expressions are far from being user-friendly, they may be confusing even for an experienced programmer. Their use is however so ubiquitous that learning them is a good investment. On the other hand, there exist already various tools for editing and debugging regular expressions and we hope to adapt one of them in the future to Poliqarp. For the time being the best approach is to start with a simple general query and to refine the search by adding additional restrictions.

5 Lemmatization, morphosyntactic tagging and polyinterpretations

The standard linguistic corpus workflow includes two important steps: morphosyntactic analysis and disambiguation (cf. eg. [Przepiórkowski, 2004, p. 14]). Morphosyntactic analysis assigns all possible interpretations to a word, in par-

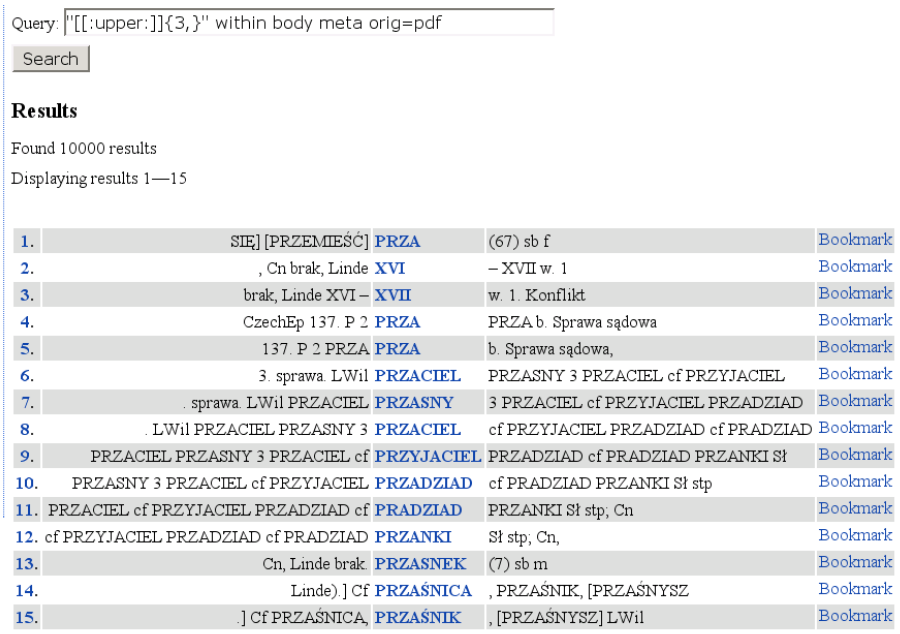

Figure 4: Standard concordances for digitally born texts

ticular all possible canonical forms of the word. For example, the Polish word *mam* can be a form of MIEĆ (*to have*), MAMA (deminutive of *mother*) or MAMIĆ (*to deceive*); moreover even for a fixed canonical form there are often different values of morphological categories possible. Disambiguation is usually performed by a program using stochastic rules to select the interpretation suitable for the given context; of course the results are sometimes wrong. Therefore Poliqarp allows to store and access all the interpretations and to compare them; for example, the user can search for words which where unambiguous already at the level of morphosyntactic analysis etc. (cf. section 3.5 of [Przepiórkowski et al., 2006]). This unique feature of `Poliqarp` is called polyinterpretation.

All this features related to language technology are present also in Poliqarp for DjVu. Moreover, some of them can be used on a lower level then originally intended.

When working with historical texts, e.g. with the quotations in the dictionary of 16th century Polish, we have different but analogical problems on the spelling level: letter *y* may mean contemporary *y* or *j*, letter *i* may mean contemporary *i* or *j* (so e.g. *przyymuiemy* is now spelled *przyjmujemy*) etc. Listing all possible interpretation of a letter can be considered an analogue of the morphological analysis, while reconstructing the contemporary spelling according to some inferred rules is an analogue of the morphological disambiguation by a stochastic tagger. The main difference is purely technical and consists in the fact that the latter interpretation processes do not operate on linguistic features, but on the canonical form field (in Poliqarp the features are represented differently than the textual and canonical forms). This possibility has not been yet used in practice, but we hope that it will be in due time.

For many purposes a simplified form of morphological analysis is quite use-

ful. So called lemmatization consists in assigning the canonical form, i.e. the lemma, to the given word form; the process is sometimes also called stemming. It is relatively easy for English, so it is often built into some search engines like JSSIndex mentioned earlier, but quite difficult and costly for inflectional languages like Polish. The tools used for the National Corpus of Polish do not seem to be directly applicable to historical texts, so it is still an open problem. One of the possible solutions is to organize collaborative work of volunteers who would enter the requested information by hand.

6 Towards collaborative proofreading and lemmatization

The important innovation of Poliqarp for DjVu is the ability to bookmark the hits with Firefox and other browsers based on the Gecko engine. The bookmark refers to the appropriate page of the DjVu document with the hit highlighted. The name of the bookmark is created by JavaScript code and contains the following elements:

- the abbreviated name of the dictionary,
- the text of the query,
- the timestamp (to distinguish different hits of the same query).

Additional information can be added by the user either by editing the name or by using, in Firefox and some other browsers, the tags field. Hence the user can not only bookmark hits easily for his own use, but also mark and correct OCR mistakes. The problem is how to organize sharing of this information.

From purely technical point of view, the tools are already available in the form of the Firefox Sync plug-in, which is to become a standard feature of the browser, and the Sync server (formerly called Weave) which collects information from the plug-ins. There is however a serious problem of privacy, because it is not possible to grant access only to the Poliqarp error-correcting bookmarks. The simplest solution seems to export the relevant bookmarks locally and submit them to the dedicated server by a special program.

We are of course aware of various specialised tools for collaborated proofreading, but our general philosophy is make OCR error report extremely easy for a casual user, and to move the burden to the receiving side. One of possible scenario is to convert the collected reports into annotations in a special copy of the document to be used later with Jakub Wilk's DjVu editor djvusmooth (`http://jwilk.net/software/djvusmooth`) mentioned earlier.

7 Concluding remarks

Poliqarp for DjVu is a powerful tool for searching the hidden text layer of DjVu documents, which can be created in particular by converting PDF files used for printing or output of OCR programs. Although the Computer Science principle *garbage in, garbage out* is generally valid, the sophisticated queries allowed in *Poliqarp* may partially alleviate the problem of bad quality OCR. For digitally born or thoroughly proof-read texts the program is even more useful.

8 Acknowledgment

The work described in the present paper is supported by the Polish Ministry of Science and Higher Education's grant no. N N519 384036.

References

Janusz S. Bień. Facilitating access to digitalized dictionaries in DjVu format. *Studia Kognitywne - Études Cognitives*, 9:161–170, September 2009a. URL `http://bc.klf.uw.edu.pl/160/`. (Referenced on p. 5).

Janusz S. Bień. Digitalizing dictionaries of Polish. In Krzysztof Bogacki, Joanna Cholewa, and Agata Rozumko, editors, *Methods of Lexical Analysis: Theoretical assumption and practical applications*, pages 37–45. Wydawnictwo Uniwersytetu w Białymstoku, Białystok, 2009b. URL `http://bc.klf.uw.edu.pl/71/`. (Referenced on p. 6).

Thomas Breuel. The hOCR microformat for OCR workflow and results. In *Proceedings of the Ninth International Conference on Document Analysis and Recognition*, pages 1063–1067. IEEE Computer Society, 2007. URL `http://madm.dfki.de/publication&pubid=4373`. (Referenced on p. 6).

Yann Le Cun, Léon Bottou, Andrei Erofeev, Patrick Haffner, and Bill W. Riemers. DjVu document browsing with on-demand loading and rendering of image components. In *Internet Imaging*, San Jose, January 2001. URL `http://leon.bottou.org/papers/lecun-2001`. (Referenced on p. 1 and 2).

Tadeusz Piotrowski. Digitization of Polish historic(al) dictionaries. *Review of the National Center for Digitization*, 6:95–102, 2005. URL `http://elib.mi.sanu.ac.rs/files/journals/ncd/6/d009download.pdf`. (Referenced on p. 6).

Stefan Pletschacher and Apostolos Antonacopoulos. The PAGE (Page Analysis and Ground-Truth Elements) format framework. In *International Conference on Pattern Recognition*, pages 257–260, Los Alamitos, CA, USA, 2010. IEEE Computer Society. URL `http://www.cse.salford.ac.uk/prima/papers/ICPR2010_Pletschacher_PAGE.pdf`. (Referenced on p. 6).

Adam Przepiórkowski. *The IPI PAN Corpus: Preliminary version*. Institute of Computer Science, Polish Academy of Sciences, Warsaw, 2004. URL `http://nlp.ipipan.waw.pl/~adamp/Papers/2004-corpus/`. (Referenced on p. 10).

Adam Przepiórkowski. TEI P5 as an XML standard for treebank encoding. In Marco Passarotti, Adam Przepiórkowski, Savina Raynaud, and Frank Van Eynde, editors, *Proceedings of the Eighth International Workshop on Treebanks and Linguistic Theories (TLT 8)*, pages 149–160, Milan, Italy, 2009. URL `http://nlp.ipipan.waw.pl/~adamp/Papers/2009-tlt-tei/`. (Referenced on p. 5).

Adam Przepiórkowski, Aleksander Buczyński, and Jakub Wilk. The National Corpus of Polish Cheatsheet, 2006. URL `http://nkjp.pl/poliqarp/help/en.html[Accessed2011-02-08]`. (Referenced on p. 11).