The Icecite Research Paper Management System

Hannah Bast and Claudius Korzen

Department of Computer Science, University of Freiburg, Germany
{bast,korzen}@informatik.uni-freiburg.de

Abstract. We present Icecite, a new fully web-based research paper management system (RPMS). Icecite facilitates the following otherwise laborious and time-consuming steps typically involved in literature research: automatic metadata and reference extraction, on-click reference downloading, shared annotations, offline availability, and full-featured search in metadata, full texts, and annotations. None of the many existing RPMSs provides this feature set. For the metadata and reference extraction, we use a rule-based approach combined with an index-based approximate search on a given reference database. An extensive quality evaluation, using DBLP and PubMed as reference databases, shows extraction accuracies of above 95%. We also provide a small user study, comparing Icecite to the state-of-the-art RPMS Mendeley as well as to an RPMS-free baseline.

1 Introduction

This paper is about *Icecite*, a new research paper management system (RPMS) that provides the following unique set of features:

(1) Automatic Metadata AND Reference Extraction: Icecite automatically extracts, with accuracies over 95%, bibliographic metadata (title, authors, year, conference, etc.) as well as references from academic research papers uploaded to the system.

(2) On-Click Download of New Papers: When reading a paper, other papers cited or listed in the reference section can be downloaded with a single click. Using the metadata from the reference extraction from (1), Icecite automatically searches the web for the correct PDF and uploads it to the system.

(3) Collaborative Annotation: Research papers can be annotated in the browser using the PDF standard. This ensures, that annotations remain modifiable in all standard (annotation-enabled) PDF viewers. Internally, annotations are kept separately from the PDF files. This enables collaborative annotation with other users in both online and offline mode (when annotating offline, annotations will be synchronized the next time the user goes online).

(4) Offline Availability: Icecite is web-based (no software download required), but papers can be read and annotated also when offline.

(5) Full-Featured Search: With Icecite, all the metadata, references, annotations, full texts as well as the underlying reference databases can be searched interactively (search as you type).

X. Lin et al. (Eds.): WISE 2013, Part II, LNCS 8181, pp. 396–409, 2013.

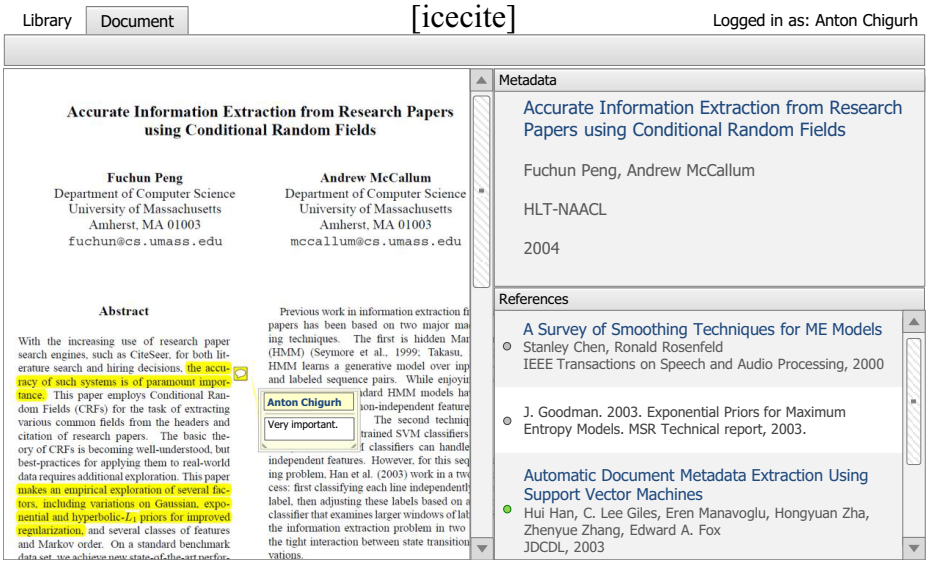

Fig. 1. A screenshot of the *Document View*. The left panel displays the PDF file, the right panels display the metadata (upper right) and the extracted references (lower right). The PDF file can be annotated in the browser using standard PDF annotations. The metadata and references panel can be arbitrarily resized, or hidden to display the PDF file in full screen mode. The references are listed with their full metadata. If no metadata record was found in the reference database, only the extract is displayed (as for the 2nd reference). The documents of the user are organized in a personal library (accessible by clicking the tab "Library" in the header). The colored bullet besides each reference indicates its availability in the user's library. A green bullet means: The document is already stored in the library and can be called by clicking it. A gray bullet means: The reference is not available in the library and can be clicked to import it.

The feature set described above looks quite natural and straightforward for a RPMS. However, *none* of the many existing RPMSs provides this combination of features. In fact, not one of these systems is able to provide even automatic metadata AND reference extraction (with acceptable accuracy). We provide an overview and comparison of fifteen RPMSs in Section 2.

Technically, Icecite combines known techniques in a (more or less) clever way to do what it does. The main idea behind the high-accuracy metadata and reference extraction is a combination of a rule-based recognition (of the passages in the text referring to metadata) with a fast index-based approximate search on a reference database. This is described in more detail in Sections 3 (metadata) and 4 (references). The results of our experimental evaluation, as well as a description of our reference databases are provided in Section 6.

The annotation and offline features are realized using the capabilities of the new HTML5 standard, namely its *Filesystem API* and the *Application Cache*. Annotations are merged using a standard text-based concurrent versioning

system. The fast and powerful search-as-you-type functionality is realized using CompleteSearch from [1]. These features are described in Section 5.

We have also conducted a first small user study (12 participants), comparing Icecite against the state-of-the-art system Mendeley, as well as against an RPMS-free baseline approach using Google Scholar for search and the local file system for storage. Study design and results are described in Section 7.

2 Related Work

2.1 Extraction of Bibliographic Metadata and References

Existing techniques for automatic metadata and reference extraction can be classified in two approaches: using *machine learning* and *rule-based*.

Typical techniques in the machine learning approach are: Hidden Markov Models (HMMs), Support Vector Machines (SVMs), and Conditional Random Fields (CRFs). As outlined in Table 1, machine learning approaches achieve good accuracies, of up to around 90%. However the generation of accurate labeled datasets, which are needed to train the models, is time-consuming and costly [9]. Further, machine learning approaches are usually expensive in terms of runtime of the extraction processes [2].

Table 1. Overview of the accuracies of selected machine learning doing metadata extraction (M. Ex.) and/or reference extraction (R. Ex.). The percentage marked * denotes the accuracy of only the title extraction.

Paper	*Model*	*M. Ex.*	*R. Ex.*	*Accuracy*
Seymore et al. (1999) [16]	HMM	✓		90.1%
Borkar et al. (2001) [3]	HMM		✓	87.3%
Han et al. (2003) [10]	SVM	✓		92.9%
Granitzer et al. (2012) [8]	SVM	(✓)		85.5%*
Peng et al. (2004) [15]	CRF	✓	✓	95.4%
Councill et al. (2008) [5]	CRF		✓	91.6%

Rule-based approaches consist of a set of rules, which are usually derived from human observations (e.g. regarding style information) to identify metadata fields in the headers or reference strings in the bibliography sections of research papers. Pure rule-based approaches are usually faster, but less accurate than machine learning approaches. Beel et al. [2] achieve an accuracy of 77.9% on extracting the titles of research papers by analyzing their font sizes. Guo and Jin [9] combine a rule-based approach with a metadata knowledge base to guide the extraction process of reference metadata. This approach yields a higher average extraction accuracy of 89.1% over all reference metadata fields.

2.2 Record Matching Techniques

Record matching (also called *record linkage*) is the problem of matching a given string (which could be an extracted title or reference string) to the "correct" record in a given database (of titles or references or whatever the application is). This process is usually affected by noisy factors like typing errors, alternative spellings or extraction errors. Most approaches therefore use fuzzy string comparisons to compute the similarity of the given string to selected fields of the database records. Typical similarity measures are: character-based (e.g. *Levenshtein distance* [13], *Smith-Waterman similarity* [17], etc.), token-based, and phonetic-based. See the surveys given in [6] and [11] for more details.

A brute-force comparison of the given string to all database records is usually too slow. Instead, a common approach is to use, in a first step, simplified criteria to quickly obtain a set of candidates of possibly matching candidates. This is often called *blocking* [11]. In a second step, the exact similarities are then computed only for the candidate records. Reasonable blocking strategies are mainly index-based, see the survey [4] for more details. Our approach taken for Icecite also falls in this category. A machine learning approach to blocking is presented in [14].

2.3 Related Applications

Table 2 summarizes and compares the feature sets of fifteen recent RPMSs as well as Icecite. In the following, we discuss a selection of the most powerful of these systems. We distinguish between desktop-based and web-based systems.

The **desktop-based applications** (upper part of Table 2) usually allow to organize research papers in a personal library, listed with extracted metadata. ReadCube is the only system that also provides metadata for bibliographic references of selected papers. However, the references are not actually extracted from the PDF files but fetched from special websites (like *digital libraries*, see below) if available. Automatic search and download of PDF files is supported only by ReadCube, EndNote and Citavi.

Annotating PDF files in a built-in PDF viewer is provided by EndNote, Mendeley, Qiqqa and ReadCube. However, annotations are either not displayed in the PDF when exported (EndNote, Qiqqa and ReadCube) or are "drawn" into the PDF (Mendeley) and thus can not be fully edited anymore in an external PDF viewer. Further, Mendeley does not support a built-in search in external sources to import new research papers easily.

The **web-based applications** (lower part of Table 2) can be further distinguished into (1) reference managers, (2) PDF annotation tools, and (3) digital libraries of academic publishers.

The main purpose of reference managers like *BibSonomy*, *CiteULike*, *EndNote Web*, and *RefWorks* is to manage collections of bibliographic metadata. Besides, BibSonomy and CiteULike also allow to attach PDF files to each record, but the automatic extraction of metadata and references from these PDFs is not supported. Furthermore, in CiteULike, annotating PDF files is only supported for (paying) premium users.

Table 2. Comparison of the feature sets of sixteen RPMSs, eight desktop-based (upper part) and eight web-based (lower part). If a feature is fully provided, it is denoted with "✓". If a feature is partially provided, it is denoted with "(✓)". The listed features are: *(EX-M)*: automatic extraction of metadata; *(EX-R)*: automatic extraction of bibliographic references; *(AUTO-DL)*: automatic search and download of PDF files; *(ANNOT)*: native and colored annotations; *(SHARED)*: data can be shared to collaborate with other users; *(OFFLINE)*: (parts of) the features can be used in offline mode; *(SEARCH)*: search in metadata, full texts, annotations and external sources. *(CLOUD)*: the data can be stored in the cloud to access them from multiple devices.

System/URL	*(EX-M)*	*(EX-R)*	*(AUTO-DL)*	*(ANNOT)*	*(SHARED)*	*(OFFLINE)*	*(SEARCH)*	*(CLOUD)*
Citavi www.citavi.com	(✓)	-	(✓)	-	(✓)	✓	(✓)	-
EndNote www.endnote.com	-	-	✓	(✓)	(✓)	✓	✓	✓
EverNote www.evernote.com	-	-	-	-	✓	✓	(✓)	✓
Mendeley www.mendeley.com	✓	-	-	(✓)	✓	✓	(✓)	✓
Papers www.mekentosj.com/papers	(✓)	-	(✓)	(✓)	✓	✓	(✓)	✓
Qiqqa www.qiqqa.com	✓	-	-	✓	✓	✓	✓	✓
ReadCube www.readcube.com	✓	(✓)	✓	✓	-	✓	✓	-
Zotero www.zotero.org	✓	-	-	-	✓	✓	(✓)	✓
BibSonomy www.bibsonomy.org	-	-	-	-	✓	-	-	✓
CiteULike www.citeulike.org	-	-	-	(✓)	✓	-	✓	✓
EndNoteWeb www.myendnoteweb.com	-	-	-	-	✓	-	(✓)	✓
RefWorks www.refworks.com	-	-	-	-	✓	-	(✓)	✓
A.nnotate www.a.nnotate.com	-	-	-	✓	✓	-	(✓)	✓
Crocodoc personal.crocodoc.com	-	-	-	✓	✓	-	-	✓
WebNotes www.webnotes.net	-	-	-	✓	✓	-	(✓)	✓
Icecite www.icecite.org	✓	✓	✓	✓	✓	✓	✓	✓

PDF annotation tools like *A.nnotate*, *Crocodoc* or *WebNotes* focus on annotating and commenting various file types (e.g. PDF files) in collaboration with other users. They are usually implemented in HTML5 or Flash and do not follow the PDF annotation standard, but again, "draw" annotations into the PDF while exporting.

Digital libraries of academic publishers like *ACM Digital Library*[1], *IEEE Xplore*[2], *SpringerLink*[3], etc. provide archives of scientific research papers, including extracted metadata and references. There are two caveats, however.

[1] `http://dl.acm.org/`
[2] `http://ieeexplore.ieee.org/`
[3] `http://link.springer.com/`

First, the techniques behind these services are neither published nor publicly accessible, and the extraction accuracy can only be guessed. Second, articles from the same publisher exhibit a homogeneous structure (and sometimes even include explicit meta information) which greatly facilitates the extraction task.

We did not include services like CiteSeerX[4] or Google Scholar[5] in our Table 2 above, because these are global archives and not really RPMSs. However, they also employ techniques for automatic metadata and reference extraction. Their task is harder though, because they lack a reference database. Accuracies reported in [7] are much lower than what we achieve for Icecite.

3 The Extraction of Full Metadata

We proceed in two steps. In the first step, we identify candidates for the title from the given PDF. In the second step, we approximately match these candidates against the titles of the records from our reference database, which is described in more detail in Section 6.

3.1 Title Identification

We use the open source Java tool *PDFBox*[6] to extract text along with characteristic properties like the position, the height, the width and the font of each character, word and text line from PDF files. These properties are then used to identify the title of a research paper.

Definition 1 (Emphasis Score). *Let l_i denote the i-th text line and $fs(l_i)$ denote the font size of l_i. The emphasis score $es(l_i)$ is defined by*

$$es(l_i) = fs(l_i) + \alpha(l_i) + \beta(l_i) \tag{1}$$

where $\alpha(l_i) = \begin{cases} 0.2, & l_i \text{ is printed in bold} \\ 0, & \text{otherwise} \end{cases}$ *and* $\beta(l_i) = \begin{cases} 0.1, & l_i \text{ is printed in italic} \\ 0, & \text{otherwise} \end{cases}$

Let $ES_j = \{l_i : es(l_i) = j\}$ denote the set of all text lines with emphasis score j. The most common emphasis score $es_\varnothing$ is then defined by

$$es_\varnothing = \arg\max_k \{|ES_k|\}$$

Let $T = \{l_i : l_i \text{ is member of the title}\}$ be the set of all lines belonging to the title. To identify T, the following assumptions are made:

(A1) All lines $\in T$ are placed in the header (the upper half) of the first page.

(A2) The emphasis score of all lines $\in T$ is $> es_\varnothing$.

Consequently, $es_\varnothing$ is computed for the first page and each line l_i with $es(l_i) \leq es_\varnothing$ is filtered. From all remaining lines in the upper half of the first page, stop words

[4] `http://citeseerx.ist.psu.edu/`
[5] `http://scholar.google.com/`
[6] `http://pdfbox.apache.org/`

(like "the", "and", etc.) are removed. Subsequently, the reference database is searched for candidate records, whose title contains (parts of) the remaining words. In the result, the candidate records are sorted by the number of words that they have in common with the extracted title words. Because there may exist candidates with similar titles, the related record is not necessarily the first candidate. To find the related record anyway, each candidate record is evaluated more precisely in the matching process.

3.2 The Matching of Titles

For each record of the top-100 candidates from the title identification, the following scores are computed.

First, the title score $s_t(r) = {sim_{SW}(t_r, ex)}/{sim_{max}(t_r)}$ where ex = the lines of the first page's upper half is computed. Here, $sim_{SW}(t_r, ex)$ is the Smith-Waterman similarity between the title t_r of the record r and ex, and $sim_{\max}(t_r)$ denotes the maximum achievable similarity for t_r. Note that $s_t(r) \in [0, 1]$ and $s_t(r) = 1$ if and only if ex contains t_r completely.

Second, the author score $s_a(r) = {\sum_{a_i \in A(r)} s_{a_i}(r)}/{|A(r)|}$ is computed, where $s_{a_i}(r) = {sim_{SW}(a_i, ex)}/{sim_{\max}(a_i)}$. $A(r)$ is the set of authors of record r. Note that $s_a(r) \in [0, 1]$ and $s_a(r) = 1$ if and only if ex contains all authors $\in A(r)$ completely.

Third, the year score $s_y(r) = 0.1$ (if the first page contains the year of r, otherwise $s_y(r) = 0$) and the venue score $s_v(r) = 0.1$ (if the first page contains the venue of r, otherwise $s_v(r) = 0$) are computed.

The total score $s(r)$ is given by $s(r) = s_t(r) + s_a(r) + s_y(r) + s_v(r)$. Finally, the research paper is matched to the record r with the highest score $s(r)$, as long as $s(r)$ exceeds a threshold of 1.5.

4 The Extraction of Bibliographic References

We again proceed in two steps: identification of the individual references from the PDF (again, using PDFBox), and matching of those references against the titles, authors and years of the records from the reference database. Due to the multitude of possible formats for the References section of a paper, the identification step is much more involved now.

4.1 The Identification of Bibliographic References

First of all, the extracted text lines are searched for a proper bibliography section header (e.g. consisting of the word "References", "Literature", "Bibliography", etc.). All lines following such a header are separated into logical blocks by analyzing the *line pitch* of each line to its previous line.

Definition 2 (Line Pitch). *Let l_i denote the i-th text line ($i \geq 0$) and let $y(l_i)$ be the vertical position of l_i in the page. Consider line pairs (l_{i-1}, l_i) sharing the same page. The line pitch $lp(l_{i-1}, l_i)$ between l_{i-1} and l_i is defined by*

$$lp(l_{i-1}, l_i) = y(l_i) - y(l_{i-1})$$

Let $LP_j = \{l_i : lp(l_{i-1}, l_i) = j\}$ *denote the set of all lines, whose line pitch to the previous line is* j*. The most common line pitch* $lp_\varnothing$ *is then defined by*

$$lp_\varnothing = \arg\max_k \{|LP_k|\}$$

Definition 3 (Type of a Reference Line). *Given a sequence of lines representing a reference. The first line is defined as the reference header, the last line as the reference end, and all other lines as the reference body.*

If $lp(l_{i-1}, l_i) > lp_\varnothing$, the lines l_{i-1} and l_i are separated into distinct blocks. If a block consists mainly of digits or if it is a caption (e.g. it starts with the word "Figure", "Table", etc.) it is not meaningful with respect to the reference extraction and is ignored. The type of each line in the remaining blocks is determined by checking it against the following rules:

(1) The line l_i is a *reference header*, if ...
   (a) l_i starts with a reference anchor (like `"[1]"`, `"(2)"` or `"[Smith95]"`); *or*
   (b) l_{i-1} is a reference end; *or*
   (c) l_{i-1} (or l_{i+1}) is indented compared to l_i; *or*
   (d) l_i starts with an author and l_{i-1} does not end with an author.

(2) The line l_i is a *reference* end, if ...
   (a) l_{i-1} and l_{i+1} ends up at the same horizontal position and l_i ends before l_{i-1} and l_{i+1}; *or*
   (b) l_{i+1} is a reference header.

(3) The line l_i denotes the *end of the bibliography*, if ...
   (a) l_i is the last line of the document; *or*
   (b) $\lfloor es(l_{i+1}) \rfloor > es_\varnothing$
   (rounded down to allow bold and italic lines within the bibliography).

(4) The line l_i is a *reference body* otherwise.

We assume that all references of a bibliography section share the same inner structure, so that the positions of the metadata fields within the references are consistent. Further, we assume that the authors are the first metadata field in a reference. Rule (1c) implies that, if indentations exists, reference headers are not indented, but references bodies and reference ends. Rule (1d) targets the fact that a listing of authors may cover multiple lines of the reference (a word is identified as a part of an author name if the reference database contains such an author name).

With rule (2a) we assume that the references are formatted as justified text if l_{i-1} and l_{i+1} share the same horizontal end position and that l_i denotes a reference end if it does not fill the whole line.

Once a reference string was identified, stop words are filtered and the remaining words are scanned for title words, authors and the year. Afterwards the reference database is searched for such records, which hold these metadata. Again, the resulting records are evaluated with a more precise scoring scheme.

4.2 The Matching of Bibliographic References

For each record of the top-100 candidates from the references identification, the scores $s_t(r)$, $s_a(r)$, $s_y(r)$ and $s_v(r)$ are computed as for the title matching described in Section 3.2 (with ex = the extracted reference string). Additionally, the pages score $s_p(r) = 1$ (if r defines page numbers and ex contains them, otherwise $s_p(r) = 0$) is computed. The total score $s(r)$ is given by $s(r) = s_t(r) + s_a(r) + s_y(r) + s_v(r) + s_p(r)$. Finally, the extracted reference is matched to the record r with the highest score $s(r)$, as long as $s(r)$ exceeds a threshold of 1.5.

5 Annotation, Offline Mode and Search

We enable the annotation of PDF files in the browser using the Adobe Acrobat Standard plugin. Javascript code is injected into the PDF files, such that annotations can be modified dynamically on opening the PDF file or when synchronizing with the server. The annotations are kept in text files, separate from the PDFs. This allows merging of annotations from different users (for the same paper) using a standard text-based versioning system (we use SVN). When online, Icecite periodically synchronizes with a server and automatically merges all annotations appropriately.

The offline mode is realized with the *Filesystem API* and the *Application Cache*, two features of the new HTML5 standard. The Filesystem API is used to store all library data (PDF files, metadata and annotations) locally on the file system of the client. The Application Cache is used to cache all specific web resources (like HTML files, CSS files, images etc.). If the resources have changed on the server, the browser downloads them and updates the cache automatically.

The search functionality of Icecite is implemented with *CompleteSearch* [1], which supports efficient search-as-you-type functionality. There is an index per user, which is automatically updated as soon as a paper or annotation is added. There is also one index for the reference databases (DBLP and PubMed). All searches can be directed to either of these, or to both at the same time.

6 Experiments

6.1 Experimental Setup

We evaluate the accuracy of our metadata and reference extraction algorithms on two reference databases: DBLP[7] and PubMed[8]. At the time of this writing, DBLP holds 2.1 million metadata records (with title, authors, year, venue, journal, etc.) of publications from the area of computer science and neighboring disciplines. PubMed is an order of magnitude larger, with 22 million metadata records of publications from the life sciences. We want to stress that nothing in

[7] `http://dblp.uni-trier.de/`
[8] `http://www.ncbi.nlm.nih.gov/pubmed/`

our approach is specific to these reference databases. We expect Icecite to work just as well with any other reference database.

Our test collection consists of 690 randomly selected research papers from DBLP and of 500 randomly selected research papers from PubMed. For all of these, we have determined the correct titles manually. For 91 papers of DBLP (containing 1,012 references) and 34 papers of PubMed (containing 1,235 references), we have also determined the correct reference strings. For each such title and reference string, we have further determined the key of its related metadata record (if available) in the reference database.

The code for the metadata and reference extraction is entirely written in Java, based on the Java library *PDFBox*. The index to browse the reference databases is written in C++. All the tests were run on a single machine with 4 Intel Xeon 2.8 GHz processors and 35GB of main memory, running Ubuntu 9.10 64-bit.

6.2 Extraction Accuracies

First, we have measured the accuracies (the percentage of correct results from the total numbers of results) of both the extraction and the matching algorithms. We have considered an extract as correct, if its Levenshtein distance to the expected extract ex_{gt} is $\leq 0.2 \cdot |ex_{gt}|$. Further, we have considered a matched metadata record as correct, if its key is equal to the key of the expected record or if no record was returned and there is in fact no expected record.

Table 3. Overview of the extraction accuracies of the metadata and references extraction on DBLP and PubMed. Column 3 provides the number of entities (document or reference) to process in the test collection. There were PDF files which could not be processed by PDFBox; subtracting these gives the numbers in Column 4. Columns 5 and 6 provide the absolute number of correct extractions as well as the percentage with respect to the value in Column 4.

		num.	max.	corr. extracts	corr. matches
Meta.	DBLP	690	679	672 (98.9%)	665 (97.9%)
	PubMed	497	490	474 (96.7%)	468 (95.5%)
Ref.	DBLP	1012	997	974 (97.7%)	951 (95.4%)
	PubMed	1235	1235	1179 (95.5%)	1166 (94.4%)

As shown in Table 3, we achieve very good extraction accuracies for both datasets. There are only few documents for which the extraction or matching process failed. We manually investigated the individual reasons for these few failures. For example, the title extraction failed, if (1) the title in the document was not emphasized compared to other text passages on the first page, (2) the title was not placed in the upper half of the first page, or (3) two titles were placed on the first page and the other title was extracted. The reference extraction failed, if (4) there was no bibliography header, (5) existing reference anchors were

not extracted or (6) title words were misleadingly identified as author words. (5) and (6) have led to the wrong identification of the reference line's type. Matching a title failed, if (7) there were multiple variants of a paper in the reference database (but published in an alternative journal) and there was no criterion to distinguish the variants, (8) more words than the title word were extracted (because of their emphasis score) such that records other than the related one were found in the reference database, (9) words were misspelled (mostly due to extraction errors), (10) the title in the document and the title of its related database record did not match exactly. Matching a reference string has failed, if (11) author names were misspelled, (12) the year in the reference did not correspond to the year of its related record, (13) the reference did not have a related record, but there was a record with a related title by the same authors.

6.3 Running Times

We have also evaluated the running times of both of our extraction algorithms.

Table 4. Overview of the runtimes of the metadata and reference extraction from research papers of DBLP and PubMed. The runtimes are broken down into the following subtasks: (1) *loading* a PDF file and analyzing the text lines, (2) *querying* the reference database, (3) *matching* an extract to its related record. The stated runtimes are per document (metadata extraction) respectively per reference (reference extraction; on average).

		total	loading	querying	matching
Meta.	DBLP	137.7ms	31.1ms (23%)	73.1ms (53%)	33.5ms (24%)
	PubMed	479.6ms	44.9ms (9%)	341.3ms (71%)	93.4ms (20%)
Ref.	DBLP	54.2ms	14.7ms (27%)	19.7ms (36%)	19.8ms (37%)
	PubMed	91.4ms	10.2ms (11%)	47.4ms (52%)	33.8ms (37%)

Table 4 shows that our algorithms are fast enough for an interactive experience, even on the very large PubMed reference database. Note that the given times for the metadata extraction are per document, while the times for the reference extraction are per reference. The time for *loading* is needed only once per PDF. The typical time for full metadata and reference extraction from a single PDF is therefore generally below 1 second for DBLP, and 1-2 seconds for PubMed.

7 User Study

We have implemented a fully-functional prototype for Icecite. To assess the user experience with our system, we have conducted a small user study with 12 participants: 1 female, 11 males, all aged between 22 and 30 years. All of them were familiar with web browsing and have not used Icecite before. One half of the participants were asked to compare Icecite against a plain RPMS-free baseline,

namely using *Google Scholar* to search for papers and managing the research papers manually on the local file system. The other half of the participants were asked to compare Icecite against a state-of-the-art RPMS, namely *Mendeley*. We have chosen Mendeley, because it considers itself to be "the world's largest social reference management system" [12]. Besides, we believe that Mendeley's feature set comes closest to the state of the art.

Each participant was asked to solve the following set of tasks twice. A participant from the first half would solve it once with Icecite and once with Google Scholar. A participant from the second half would solve it once with Icecite and once with Mendeley.

(T1) Download the paper X and store it in your system.

(T2) Find the paper Y in DBLP and store it in your system.

(T3) Open the first paper and add at least three annotations.

(T4) Log in on a second machine and annotate the paper from both machines.

(T5) As *(T4)*, but with one machine disconnected from the internet.

(T6) (Only to solve with Icecite and Mendeley) Export the PDF file and open it in an external viewer. Edit some annotations.

(T7) Choose the first ten references from the paper of *(T1)* and store the respective PDF files into your system.

(T8) Use the available search functions to search for the terms Z.

(T9) Identify the paper of *(T2)* and open it.

In total, there were three variants of this task set, each of them with different entities for X, Y, and Z. Each participant had to solve exactly two variants (one with the one system, one with the other system). The variants were assigned in a permuted form such that each variant was assigned equally often to a system and to the participants. For each task, the participants were asked to estimate the required time and to assign a score from $1-5$ indicating the (subjective) satisfaction on completing the task (1 = absolutely dissatisfied, 5 = absolutely satisfied). Table 5 summarizes the results of this quantitative feedback. Also, each participant had the opportunity and was encouraged to give (anonymized) general feedback in a free-text field.

Most Liked Features of Icecite. Using Icecite, all participants have enjoyed the automatic extraction of references, and the ability to download a citation or reference with a single click. The possibility to annotate research papers collaboratively in the browser was also positively mentioned by 10 participants.

Most Disliked Features of Google Scholar and Mendeley. On using Google Scholar and Mendeley, the references could not be extracted automatically and referenced papers could not be downloaded on click. Instead, the participants had to download all of them manually. That's why solving task *(T7)* with Google Scholar or Mendeley took much longer than with Icecite. Further, 3 participants disliked that Mendeley does not support multi-colored annotations, and 5 participants disliked that the annotations are not fully modifiable after

Table 5. Breakdown of the results of our user study. For each task *(T1)-(T9)*, the participant's subjective satisfaction and the required time for solving the task with Google Scholar, Mendeley and Icecite is stated. For each task, the best results are emphasized in **bold**.

	Google Scholar		*Mendeley*		*Icecite*	
(T1)	4.0	2.4 min	3.8	2 min	**4.1**	**1.3 min**
(T2)	**4.2**	1.4 min	3.7	1.8 min	**4.2**	**1.3 min**
(T3)	3.7	3.5 min	2.7	4.5 min	**4.4**	**2 min**
(T4)	1.2	-	3.0	7.2 min	**4.0**	**2.5 min**
(T5)	1.0	-	3.5	3.4 min	**4.5**	**2 min**
(T6)	-	-	2.2	5 min	**4.6**	**1.5 min**
(T7)	2.2	11.8 min	2.2	15.6 min	**4.1**	**8.3 min**
(T8)	2.0	4.1 min	**4.7**	**1.9 min**	4.0	2.1 min
(T9)	4.0	0.8 min	4.6	1 min	**4.8**	**0.7 min**

exporting a PDF file. It turned out, that the tasks *(T4)* and *(T5)* could not be solved with a reasonable effort using Google Scholar. That's why 5 participants have missed these tasks, if they were asked to solve them using Google Scholar (and that's why the required time for *(T4)* and *(T5)* is denoted by "-" in the respective column in Table 5).

Most Liked Features of Google Scholar and Mendeley. The search function of Mendeley was generally enjoyed and outperformed those of Icecite and Google Scholar *(T8)*. Every participant liked the possibility to jump directly to the position of a query-relevant text-passage in a PDF file. Further, Google Scholar was praised by 3 participants for its simplicity and its quality of search results.

Most Disliked Features of Icecite. 9 participants have complained about the minimal feedback the system is giving to users about what it is currently doing. They asked for more messages of the sort: "logging in", "saving documents/annotations", "synchronizing", etc. 6 participants expressed that existing messages could be more precise, e.g. by specifying the reason, why an import of a referenced research paper has failed. All participants but one were annoyed by small bugs of the search box: after sending a search query, the focus of the search box was lost such that it must be clicked again to modify the query. However, all of these revealed weaknesses are easy to address.

8 Conclusion and Future Work

We have presented Icecite, a fully web-based research paper management system (RPMS) with a unique feature set that has not yet been achieved by any other RPMS. This in particular applies to the automatic metadata and reference extraction, provided by Icecite with accuracies over 95%. We have also verified the benefits of Icecite in a small user study.

We have provided an error analysis of the missing few percents in accuracy. It appears that about half of these errors can be addressed by a further improved identification step (see Sections 3 and 4). The total extraction time per PDF is good (below 1 second for DBLP, 1-2 seconds for PubMed) but could be improved further. However, the current bottleneck here is not our algorithms, but the PDFBox library. Our small user study confirmed that the unique feature set of Icecite, in particular the automatic metadata and reference extraction and the one-click reference downloading, is of great practical value to users.

References

1. Bast, H., Weber, I.: The CompleteSearch Engine: Interactive, Efficient, and Towards IR&DB Integration. In: CIDR, pp. 88–95 (2007)
2. Beel, J., Gipp, B., Shaker, A., Friedrich, N.: SciPlore Xtract: Extracting Titles from Scientific PDF Documents by Analyzing Style Information (Font Size). In: Lalmas, M., Jose, J., Rauber, A., Sebastiani, F., Frommholz, I. (eds.) ECDL 2010. LNCS, vol. 6273, pp. 413–416. Springer, Heidelberg (2010)
3. Borkar, V.R., Deshmukh, K., Sarawagi, S.: Automatic Segmentation of Text into Structured Records. In: SIGMOD Conference, pp. 175–186 (2001)
4. Christen, P.: A Survey of Indexing Techniques for Scalable Record Linkage and Deduplication. IEEE Trans. Knowl. Data Eng. 24(9), 1537–1555 (2012)
5. Councill, I.G., Giles, C.L., Kan, M.-Y.: ParsCit: An Open-source CRF Reference String Parsing Package. In: LREC (2008)
6. Elmagarmid, A.K., Ipeirotis, P.G., Verykios, V.S.: Duplicate Record Detection: A Survey. IEEE Trans. Knowl. Data Eng. 19(1), 1–16 (2007)
7. Giles, C.L., Bollacker, K.D., Lawrence, S.: CiteSeer: An Automatic Citation Indexing System. In: ACM DL, pp. 89–98 (1998)
8. Granitzer, M., Hristakeva, M., Jack, K., Knight, R.: A Comparison of Metadata Extraction Techniques for Crowdsourced Bibliographic Metadata Management. In: SAC, pp. 962–964 (2012)
9. Guo, Z., Jin, H.: Reference Metadata Extraction from Scientific Papers. In: PDCAT, pp. 45–49 (2011)
10. Han, H., Giles, C.L., Manavoglu, E., Zha, H., Zhang, Z., Fox, E.A.: Automatic Document Metadata Extraction Using Support Vector Machines. In: JCDL, pp. 37–48 (2003)
11. Kan, M.-Y., Tan, Y.F.: Record Matching in Digital Library Metadata. Commun. ACM 51(2), 91–94 (2008)
12. Kraker, P., Körner, C., Jack, K., Granitzer, M.: Harnessing User Library Statistics for Research Evaluation and Knowledge Domain Visualization. In: WWW (Companion Volume), pp. 1017–1024 (2012)
13. Levenshtein, V.I.: Binary Codes Capable of Correcting Deletions, Insertions, and Reversals. Soviet Physics Doklady 10, 707–710 (1966)
14. Michelson, M., Knoblock, C.A.: Learning Blocking Schemes for Record Linkage. In: AAAI, pp. 440–445 (2006)
15. Peng, F., McCallum, A.: Accurate Information Extraction from Research Papers using Conditional Random Fields. In: HLT-NAACL, pp. 329–336 (2004)
16. Seymore, K., McCallum, A., Rosenfeld, R.: Learning Hidden Markov Model Structure for Information Extraction. In: AAAI 1999 Workshop on Machine Learning for Information Extraction, pp. 37–42 (1999)
17. Smith, T., Waterman, M.: Identification of Common Molecular Subsequences. Journal of Molecular Biology 147, 195–197 (1981)