## NetBibTEXing

Robert Tolksdorf

### 1 Introduction

BibTEX is the format of choice for cataloging and referring to literature with currently highest importance for scientific references. It defines a standard format for keeping meta-information on published material, a language to process these, and an implementation of the processor, the `bibtex` program.

Usually, the user collects bibliographic information in local BibTEX databases manually. With the widespread availability of the Internet, more and more bibliographic collections have been made available online via the Web. With the immense growth of the number of entries available, the need for services that help in locating reference information has increased.

An example is the "The Collection of Computer Science Bibliographies" at `http://liinwww.ira.uka.de/bibliography`. It collects over 1200 bibliographies that contain more than 940000 references and provides a search service on this data. The collections are well maintained by their respective authors and show a high timeliness.

In this article, we describe the design and implementation of NetBibTEX, a system that uses such a service online to retrieve bibliographic references based on a special kind of citations in a LaTEX document.

### 2 Overview

Figure 1 gives an overview on the files and processors involved. NetBibTEX contains a style-file that allows the inclusion of *netcitations* in the document source. Similar to a normal `\cite`, they require a bibliographic key, but also describe the reference by keywords for names fields, such as

```
\netcite{robert:lcs}{title=Coordination
 Laura,author=Robert Tolksdorf,year=1998}
```

The author will later interactively select a reference retrieved from the net based on this description — the *netreference*. For each netcitation, an entry in a special `.nbb`-file is generated that contains the information marked up with XML:

```
<netbibqueries value="Version 1.0">
<bibquery value="robert:lcs">
<title value="Coordination Laura"/>
<author value="Robert Tolksdorf"/>
<year value="1998"/>
</bibquery>
</netbibqueries>
```

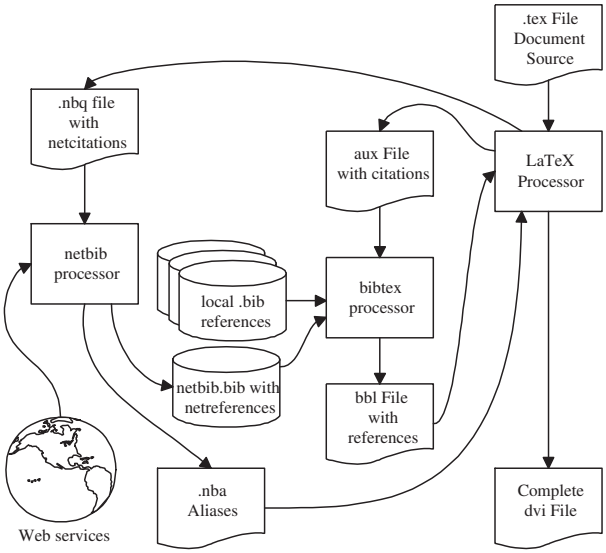


**Figure 1**: Overview of NetBibTEX

The choice of XML-format is motivated by the availability of the language WebL, a scripting language that allows easy extraction of information from XML/HTML marked up documents (**?**). We use it to implement the `netbib` processor. It performs a query to Web-services that collect bibliographic references for each netcitation. From the answers retrieved, it extracts the netreferences and asks the user to select one as shown in figure 2.

Each selected entry is written into `netbib.bib`. In addition, the key used in the netcitation has to be mapped to the actual key in the netreference. NetBibTEX provides a document style `bibalias` that allows the definition of aliases for bibliographic keys. `netbib` generates an `.nba` file that contains the appropriate definitions:

```
% Generated BibTeX key aliases by netbib
\bibalias{robert:lcs}{SCP::Tolksdorf1998}
```

With these mechanisms the queries from netcitations are matched by netreferences. These are stored in the generated bibliography and their keys are aliased with the keys used in the netcitations. Three LaTEX and one BibTEX runs are needed to produce the final, complete document.

```
grunge tolk 3 (~/bibalias): webl netbib.webl netbibtest

@Article{SCP::Tolksdorf1998,
  title =        "Laura---{A} service-based coordination language",
  author =       "Robert Tolksdorf",
  pages =        "359--381",
  journal =      "Science of Computer Programming",
  month =        jul,
  year =         "1998",
  volume =       "31",
  number =       "2--3",
  references =   "\cite{PPOPP::AghaC1993} \cite{ACMTCS::BirmanSS1991}
                 \cite{TOPLAS::BowmanDP1993}
                 \cite{ACMTCS::CarrieroG1986}
                 \cite{CACM::GelernterC1992}",
}

Accept this entry for citation "robert:lcs"?y
```

**Figure 2**: Selecting a reference found in the net

## 3 User interface

The user interface of NetBibTeX is very small — there is one LaTeX macro for netcitations and the `netbib` program.

### 3.1 Netcitations

A document using NetBibTeX has to include the package `netbib` with `\usepackage{netbib}`. It provides the macro `\netcite`, includes generated aliases at the start of the document and finalizes the generation of the `.nbq` at the end of the document. In addition, it changes the behavior of `\bibliography` to include the generated `netbib.bib`.

To use a netcitation in a document, one uses `\netcite{`⟨*bibkey*⟩`}{`⟨*query*⟩`}` as in the example above. ⟨*bibkey*⟩ is a bibliographic key for a reference, identical to the ones used with `\cite`. ⟨*query*⟩ describes the reference by a comma-separated list of keyword queries to fields. Note that this format is defined by NetBibTeX and is mapped to specific queries for Web-services by the `netbib` program. The fields defined are:

- `author`: The author of the cited work
- `title`: The title of the cited work
- `year`: The year of publication of the cited work
- `key`: The whole citation

In addition, two flags can be used in the query:

- `word`: Consider only complete words exactly
- `case`: Distinguish upper- and lowercase

Note the condition in this description: If the Web-service used by `netbib` does not provide the respective searching options, then the flags are ignored.

NetBibTeX will try to construct a "good" query to various services and favors conjunction of given keywords for the fields in order to narrow the set of possible matches as much as possible. The system could be extended to further control that behavior.

For each service used by NetBibTeX, a special routine has to be programmed that maps the net-query into a specific query to the respective Web service using the specific query syntax there. In the initial `netbib.webl` script, we demonstrate this for three services in the functions `queryGibbens`, `queryPPA`, and `queryCSBibColl`.

The extraction of BibTeX entries from the results of the queries is also dependent on the services used. In the three cases implemented, we pose queries that result in a single HTML page with a list of possible matches. It contains references in BibTeX syntax enclosed by the `<PRE>` tag in all three cases. The WebL function `Elem` returns a set of page fragments, each being one of the preformatted BibTeX entries.

Depending on the services used, the extraction could be implemented in a different manner. One could also program conversation routines if citation services return other formats than BibTeX.

Administrators of bibliographic collections can contribute to NetBibTeX by programming a respective query and extraction routine, or by documenting the syntax of their queries and the format

of the output in detail. E-mails with information on extending NetBibTeX with further services are highly appreciated by the author.

### 3.2 Interactive selection of netreferences

After writing out the queries for netcitations in the `.nbq` file, the `netbib` processor can perform a search on the Web for matching references. It is implemented in the scripting language *WebL*, which is an interpreter written in Java.

We have chosen Java as the underlying execution mechanism to implement platform independence of `netbib`. The current drawbacks in execution time are not relevant for `netbib`, as its speed is dominated by the external Web services that look for references and by the network latency.

In order to use WebL, you need a Java virtual machine and the WebL interpreter available free (including sources) at `http://www.compaq.com/WebL`. Follow the respective instructions for installation of WebL.

Assuming that there is some script `webl` installed that starts Java with the main class of `webl.jar`, the NetBibTeX processor is started for a document ⟨*document*⟩`.tex` with
`webl netbib.webl` ⟨*document*⟩

The program starts to extract the netcitations and searches for netreferences. For each one, the user is asked as in figure 2 whether to accept it.

The GUI shown is very clumsy and not very convenient to use. We will put an extended version with a graphical interface for the selection of references at the Web site mentioned below. Its implementation involves specific techniques to access Java-classes from WebL that are of no interest here.

The retrieval of netreferences is rather slow and the selection of a matching one can be very tedious if the query is not very precise and the service offered a long list of references.

In order to avoid unnecessary queries, the `netbib`-style put a

tag `<known/>` into each `bibquery` for which a netreference has already been retrieved. This is detected by testing whether the key used in the netcitation is an alias for a netreference. By using the option `-a` for the `netbib` program, this tag is ignored and all netcitations are (re-)processed.

## 4 Outlook

NetBibTeX is both expandable and dependent with respect to Web services that offer to search bibliographies and output results in BibTeX format. The implementation shown in the appendix might well lead to unpredictable results due to changes in URLs or forms. At `http://www.cs.tu-berlin.de/~tolk/netbib` you can find the homepage of NetBibTeX that includes the latest versions of the system.

◇ Robert Tolksdorf
Technische Universität Berlin
Fachbereich 13, Informatik
FLP, FR 6-10
Franklinstr. 28/29
D-10587 Berlin
Germany
`tolk@cs.tu-berlin.de`
`http://www.cs.tu-berlin.de/~tolk`

## A The Implementation

NetBibTeX consists of the two LaTeX stylefiles `bibalias.sty` and `netbib.sty`, and the WebL Script `netbib.webl` that are documented below.

### A.1 `bibalias.sty`

First, we introduce ourselves.

```
\ProvidesPackage{bibalias}
```

`\bibalias` For a key $k_1$ which is an alias for a key $k_2$, we define a label `ba@`$k_1$ that expands to $k_2$.

```
\newcommand{\bibalias}[2]{\@newl@bel{ba}{#1}{#2}}
```

`\@citex` Citations are expanded into the respective labels in the `\@citex` macro. The individual references are extracted from the comma-separated list in the second parameter and processed as `@citeb`. The first lines of the macro are copied directly from `latex.ltx`.

```
\def\@citex[#1]#2{%
  \let\@citea\@empty \@cite{\@for\@citeb:=#2\do
    {\@citea\def\@citea{,\penalty\@m\ }%
     \edef\@citeb{\expandafter\@firstofone\@citeb\@empty}%
```

Here we test whether the key is an alias for another one.

```
\@ifundefined{ba@\@citeb}{}%
```

Yes, it is an alias. We replace the content of `\@citeb` with the alias.

```
  {\typeout{\@citeb\space is an alias for \@nameuse{ba@\@citeb}}%
  \global\edef\@citeb{\@nameuse{ba@\@citeb}}}%
```

Note that we do not support aliased aliases here. The remainder of `@citex` is again a copy of the original LaTeX-code.

```
 \if@filesw\immediate\write\@auxout{\string\citation{\@citeb}}\fi
\@ifundefined{b@\@citeb}{\mbox{\reset@font\bfseries ?}%
  \G@refundefinedtrue
  \@latex@warning
    {Citation `\@citeb' on page \thepage \space undefined}}%
  {\hbox{\csname b@\@citeb\endcsname}}}}{#1}}
```

### A.2 `netbib.sty`

First, we introduce ourselves.

```
\ProvidesPackage{netbib}
```

We depend on bibalias for aliasing the keys of netcitations to the actual ones found in the net and `keyval` from the standard LaTeXgraphics bundle for dealing keyword-value lists.

```
\RequirePackage{bibalias}
\RequirePackage{keyval}
```

We now define the allowed set of keywords and their processing. When `\setkey` parses a list, it handles the keywords defined here and processes them by the commands in the third argument of `\define@key`. For each keyword, we write out a tag.

```
\define@key{netbib}{author}{\nb@writevaluetag{author}{#1}}
\define@key{netbib}{title}{\nb@writevaluetag{title}{#1}}
\define@key{netbib}{year}{\nb@writevaluetag{year}{#1}}
\define@key{netbib}{key}{\nb@writevaluetag{key}{#1}}
\define@key{netbib}{word}[true]{nb@writetag{word}}
\define@key{netbib}{case}[true]{\nb@writetag{case}}
```

`\nb@queryfile` refers to the `.nbq` file that contains the netcitations in XML markup.

```
\newwrite\nb@queryfile
\immediate\openout\nb@queryfile=\jobname.nbq
```

The following four handy macros write out XML tags.

```
\def\nb@writetag#1{\protected@write\nb@queryfile{}{\string<#1/>}}%
\def\nb@writevaluetag#1#2{\protected@write\nb@queryfile{}{\string<#1 value="#2"/>}}%
\def\nb@openvaluetag#1#2{\protected@write\nb@queryfile{}{\string<#1 value="#2">}}%
\def\nb@closetag#1{\protected@write\nb@queryfile{}{\string</#1>}}%
```

The XML startsymbol for our `.nbq` files is `<netbibqueries>`, thus we generate such a tag immediately and close it at the end of the document.

```
\nb@openvaluetag{netbibqueries}{Version 1.0}
\AtEndDocument{\nb@closetag{netbibqueries}}
```

`\netcite` `\netcite` is used with a key in the first argument and a keyword-value list in the second argument. `\setkeys` processes the keyword list and thus generates several tags. We encapsulate them with a tag-pair `<bibquery value="`*key*`">`. If there is already an alias for the citation key, then we generade a tag `<known/>` in the `.nbq` file to avoid unnecessary network queries. The final `\cite` will later cite an alias to a netreference, or generate a LaTeX message.

```
\def\netcite#1#2{%
```

```
\nb@openvaluetag{bibquery}{#1}%
\setkeys{netbib}{#2}%
\@ifundefined{ba@#1}{}{\nb@writetag{known}}
\nb@closetag{bibquery}%
\cite{#1}}
```

\bibliography

`\bibliography` has to include the generated `netbib.bib` for the netcitations. We redefine it to extend its argument appropriately and then leave the work to the original macro that we remember in `\oldbibliography`.

```
\let\oldbibliography\bibliography
\def\bibliography#1{%
  \ifx#1\relax \oldbibliography{netbib,#1}
  \else        \oldbibliography{netbib}
  \fi}
```

`netbib` generates an `.nba` file that contains the alias definitions for the netcitations. It has to be read at the beginning of the document.

```
\AtBeginDocument{\@input{\jobname.nba}}
```

### A.3 netbib.webl

```
// import some modules
import Url, Str, Files;

// if elem!=nil return the value attribute, nil otherwise
var valueOrNil = fun(elem)
  if (elem!=nil and Size(elem)>0) then return elem[0].value else return nil end;
end;

// if elem is empty, return nul
var trueOrNil = fun(elem)
  if (Size(elem)>0) then return elem else return nil end;
end;

// This service knows named fields - we construct the respective and-separated query
var queryCSBibColl = fun(author,title,year,key,word,case)
  var andString="", query="", case="off", partial="on";
  if (author!=nil) then query="au="+author;                andString=" and " end;
  if (title!=nil)  then query=query+andString+"ti="+title; andString=" and " end;
  if (year!=nil)   then query=query+andString+"yr=="+year; andString=" and " end;
  if (key!=nil)    then query=query+andString+"text="+key; andString=" and " end;
  // the advanced query that we use here does not support case and word
  var result=PostURL("http://liinwww.ira.uka.de/waisbib",
    [. database="local/bibliography", convert="bibtex", directget="1",
       sortmode="score", text=query, maxhits="170" .]);
  return Elem(result,"pre");
end;

// This service uses only keywords for the search
var queryGibbens = fun(author,title,year,key,word,case)
  var query="", type="substr";
  if (author!=nil) then query=author+" "      end;
  if (title!=nil)  then query=query+title+" " end;
  if (year!=nil)   then query=query+year+" "  end;
  if (key!=nil)    then query=query+key+" "   end;
  // case handling is unspecified by the service, words are handled
  if (word!=nil)   then type="exact" end;
  var result=PostURL("http://www.statslab.cam.ac.uk/cgi-bin/bibsearch.pl",
```

```
      [. header="~richard/misc/biblio/header", footer="~richard/misc/biblio/footer",
         files="~richard/misc/biblio/rjg.bib", term=query, field="all", type=type .]);
  return Elem(result,"pre");
end;

// This service uses named fields and expects a query starting with "find"
var queryPPA = fun(author,title,year,key,word,case)
  var query="find ", andString="";
  if (author!=nil) then query=query+"author "+author+" ";     andString=" and " end;
  if (title!=nil)  then query=query+andString+"title "+title; andString=" and " end;
  if (year!=nil)   then query=query+andString+year;           andString=" and " end;
  if (key!=nil)    then query=query+andString+key;            andString=" and " end;
  // word is ignored by netbib - we do not construct wildcards, case is ignored by the service
  var result=PostURL("http://wwwslap.cern.ch/cgi-bin/collective/bibsearch2.pl",
    [. query=query, output="BibTeX" .]);
  return Elem(result,"pre");
end;

// entries is a pieceset with each piece containing a bibtex record. select shows each to
// the user and prompts for a selection. This one is returned, or nil if nothing was selected
var select = fun(entries,key)
  every entry in entries do
    every t in PCData(entry) do  Print(Text(t)) end;       // write out the record
    Print("\nAccept this entry for citation \""+key+"\"?");
    var answer=ReadLn();                                   // ask for a selection
    if (answer=="Y" or answer=="y") then return(entry) end // return, if this one is accepted
  end;
  return nil;  // if no entry was selected, return nil
end;

PrintLn("This is netbib, Version 1.0");

// Process the command line
var fileName;
var queryAll=(ARGS[1]=="-a");    // check for -a option
if queryAll then fileName=ARGS[2] else fileName=ARGS[1] end;

// The names of the generated bibliography and aliases files
var entriesFile = "netbib.bib", aliasesFile = fileName+".nba", queryFile   = fileName+".nbq";

// Write out information to them
if (queryAll) then
  Files_SaveToFile(entriesFile,"% Generated BibTeX entries by netbib\n");
  Files_SaveToFile(aliasesFile,"% Generated BibTeX key aliases by netbib\n");
else
  Files_AppendToFile(entriesFile,"% Generated BibTeX entries by netbib\n");
  Files_AppendToFile(aliasesFile,"% Generated BibTeX key aliases by netbib\n");
end;

// Read in the query file and extract all netcitations as a pieceset
var queries = Elem(Files_LoadFromFile(queryFile,"text/xml"),"bibquery");

// The list of wrapper methods to query Web-services
var services = [queryGibbens, queryPPA, queryCSBibColl];

var selection, entries;
// process all netcitations
every query in queries do
  // If it has no matching netreference yet, or everything is reprocessed
```

```
    if (trueOrNil(Elem(query,"known"))==nil or queryAll) then
      selection= nil;
      PrintLn("Searching netreference for "+query.value);
100   // Query services until a netreference is selected
      while (selection==nil) and (Size(services)>0) do
        PrintLn("Quering a service");
        entries = (First(services))(valueOrNil(Elem(query,"author")),
                        valueOrNil(Elem(query,"title")),
105                     valueOrNil(Elem(query,"year")),valueOrNil(Elem(query,"key")),
                        trueOrNil(Elem(query,"word")), trueOrNil(Elem(query,"case")));
        services=Rest(services);
        selection = select(entries,query.value);
        if (selection!=nil) then
110       // extract BibTeX key from selection
          var bibkey=Str_Match(Text(PCData(selection)[0]),`^(\s|\S)*@\S*\{\s*(.*),(\s|\S)*`)[2];
          // alias the key used in \netcite to the one from the net
          Files_AppendToFile(aliasesFile, `\bibalias{`+query.value+`}{`+bibkey+`}`+"\n");
          // writeout entry into netbib.bib
115       every t in PCData(selection) do Files_AppendToFile(entriesFile,Text(t)) end;
        end;
      end;
    end;
  end
```

## Cartoon

by Roy Preston