CITATION INDEXING: A COMPARATIVE STUDY OF CITATION ACCESS IN THREE SERVICES

TOMÁŠ JAKL

School of Computer Science, The University of Birmingham, Edgbaston Birmingham, B15 2TT, United Kingdom
e-mail address: txj300@cs.bham.ac.uk

Abstract. We try to compare three citation indexing services: Google Scholar, Web of Science and Scopus, based on a quality of data they produce and the data's currency. Also, we take a look at how well the services cover basic literature of Computer Science and how user friendly they are.

Introduction

As there are more and more people working in science and more and more papers are being published every year, it is an important task to be able to recognise the papers that shine the amongst others and, also, to be able to find relevant information to the problems we are trying to understand.

One of the most important indicators about a paper is its list of citations. In theory, it shows the author's understanding of the topic and its history, and that the author is not doing something that has been already done, but rather that he is able to combine old ideas with new to extend human knowledge. Unfortunately, this is just a theory. Because scientists are often expected to write some amount of papers every year, it is inevitable that an important part of these papers is not of a great quality. And, moreover, in the low-quality papers the authors often cite their old papers or theirs friends paper to gain credit.

To understand how much are citation relevant and to help people to keep track relevant papers, several on-line *citation indexing services* were developed. Just having such tools is not good enough. It is also important to know their strengths and weaknesses.

In this paper, we present important properties of citation indexing services and show how one can measure them. To demonstrate the previous we will analyse the following three

1998 ACM Subject Classification: [**Information storage and retrieval**]: Content Analysis and Indexing—Indexing methods.

Key words and phrases: citation analysis, currency of citation indexing services.

The LaTeX style used is the one required by Logical Methods in Computer Science. The class file can be obtained on `http://www.lmcs-online.org/Information/style.php`.

LOGICAL METHODS IN COMPUTER SCIENCE DOI:10.2168/LMCS-???

services: Google Scholar (GS), Web of Science (WoS) and Scopus (Sc). We would like to answer the questions proposed by Peter Hancox in [3]:

(1) Are there significant differences in the citations retrieved by each of these services?
(2) How well does each service provide access to the core forms of literature of Computer Science?
(3) What is the currency of each the service?
(4) Subjectively, how easy is each of the systems to use?

We are aware of the fact that our resources are limited. The point is just to demonstrate the method and show how it can be used to understand real world data. For a full analysis, one would have to develop more automated systems for both data collecting and data analysis.

1. Methodology

For our experiment, we pick a prominent scientist in Computer Science, select five papers he published in 2010 and gather all the papers that cite these five. We do the last step for every service independently. Then, we provide a data analysis of the collected data and compare the three selected citation indexing services.

2. Papers selecting

We choose *Gordon Plotkin* as one of the most promiment scientists in the Computer Science. Next, we are suppose to pick five papers he submited in 2010 that are indexed by all three citation serviced we are interested in. However, GS, WoS and Sc agree only on the following four papers published by Gordon Plotkin in 2010[1]:

- *Paper 1:* Robin Milner, a Craftsman of Tools for the Mind [5],
- *Paper 2:* A language for biochemical systems: design and formal specification [4],
- *Paper 3:* A Model of Cooperative Threads [1],
- *Paper 4:* On Protection by Layout Randomization [2].

Therefore, we also need to pick one paper from 2009:

- *Paper 5:* Configuration structures, event structures and Petri nets [6].

One may object that the reason why we strugled with our choice of the five articles above is because we chose a retird or expired author. But, this is very far from the truth. In theoretical Computer Science it very uncommon to publish more than three papers per year (on avarage) and Gordon Plotkin is definitely in his best academic years.

[1]For GS, we first find the author and then list all the papers he published. For WoS and Sc, we get better results if we search for papers with author satisfying the query `"Plotkin, G*"`. This is necessary because Gordon Plotkin's middle name, David, is not always provided.

Remark. An another problem we had to tackle was that the citation services often show for a paper a different year of publication. Namely, Paper 3 was published in 2010 but presented in a conferencence in 2009 and Paper 4 was presented at a symphosium in 2010 but published at 2012. GS always considers different copies of a paper as just one, whereas WoS and Sc cosider different versions as different papers. Later, when we analyse citations of these papers, we take into account all citations of all versions of these papers (as GS does).

3. Citations gathering

The next step is to gather the most recent ten papers that cite the five papers we selected above. We decided to import the citations into BibTeX databases because, this way, we can use LaTeX itself and a script we wrote to help us analyse the data. The script is provided in Appendix A.

Getting a BibTeX database file containing all the papers that cite a particular paper is easy in WoS and Sc. Both tools provide an option to export search queries. GS, on the other hand, does not provide a such comfort. One has to select and export papers one by one.

To remove duplicates we use both LaTeX itself and the script we wrote. For the first, we created a LaTeX script (named `bibs.tex`) and by changing `\bibliographystyle` options (to `plain` or `ieeetr`, for example) we get different orderings of the BibTeX databases and we can search for duplicates. If we believe that our databases are in a good shape (i.e. the exported data are correct), we can do the same automatically with our script.

4. Analysis

4.1. **Retrieved data.** To try to answer the first question, we look at how many papers all three databases index and how many do they have in common:

	GS	WoS	Sc	GS ∩ WoS	GS ∩ Sc	WoS ∩ Sc	WoS ∩ GS ∩ Sc
Paper 1	0	0	0	0	0	0	0
Paper 2	10	8	10	4	5	6	4
Paper 3	10	1	4	1	1	0	0
Paper 4	10	1	10	0	5	0	0
Paper 5	10	9	10	3	5	4	2

In the table, every row corresponds to a paper, and columns GS, WoS and Sc show how many papers citing the paper corresponding to the row we could find in the particular database. The columns GS ∩ WoS, GS ∩ Sc, WoS ∩ Sc, WoS ∩ GS ∩ Sc show how many papers are in the corresponding intersection. For better understanding see Figure 1.

Although it looks like that there is very little similarity between the citation search engines, the converse is true. In fact, after inspecting all the data related to Paper 2 more carefuly (not just the last 10 citations but all of them), it is clear that GS indexes all the papers citing Paper 2, and WoS and Sc index just a subset of it.

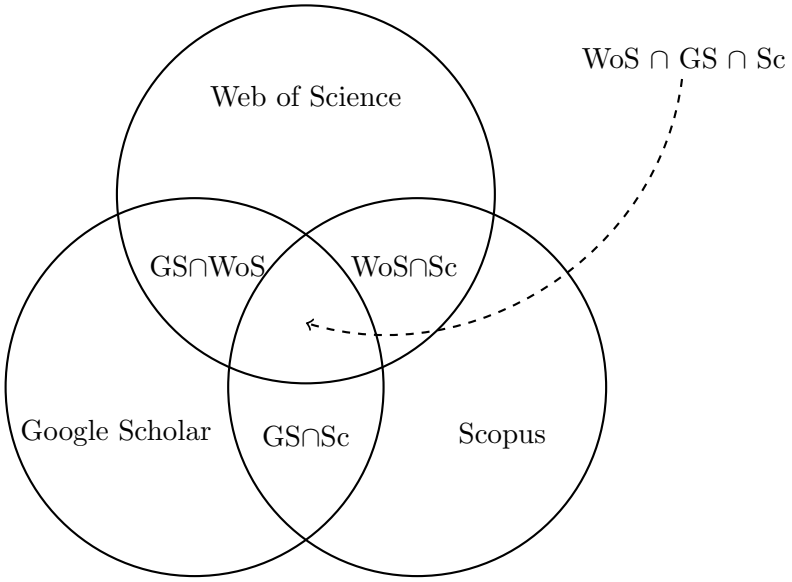

Figure 1: Intersections of indexings

4.2. **Forms of literature.** To decide whether an citation search engine covers all the core forms of literature, it is enough to inspect what is the proportion of cited articles, conference papers, technical reports, books and theses. Similarly to previous, we just inspect the data:

	Articles	From Conferences	Tech. Reports	Books	Theses	Unknown	Total
GS	30	5	1	1	2	1	40
WoS	9	11	0	0	0	0	20
Sc	27	8	0	0	0	0	35

It seems that all three services are having problems when indexing technical reports, books and theses. In the first two cases, however, one should not make such a strong statement. First, Gordon Plotkin's interests are very theoretical and we should not expect that many technical reports to even exists. Second, it does not make sense to write a book about a relatively new topic and, again, it may easily happen that no book citing any of the author's papers from 2010 was written in the past 5 years.

4.3. **Currency.** Simply by comparing the average year of publication, one can see that GS should index the most recent papers, and Sc should be the second in currency:

	GS	WoS	Sc
Paper 2	2013.200	2011.875	2012.400
Paper 3	2012.400	2012.000	2010.250
Paper 4	2013.556	2011.000	2012.636
Paper 5	2013.600	2012.200	2013.000
Average	2013.189	2011.769	2012.072

We get the same conclusion if we take a look at the data more closely and compare the total amount of papers published every single year. See Figure 2.

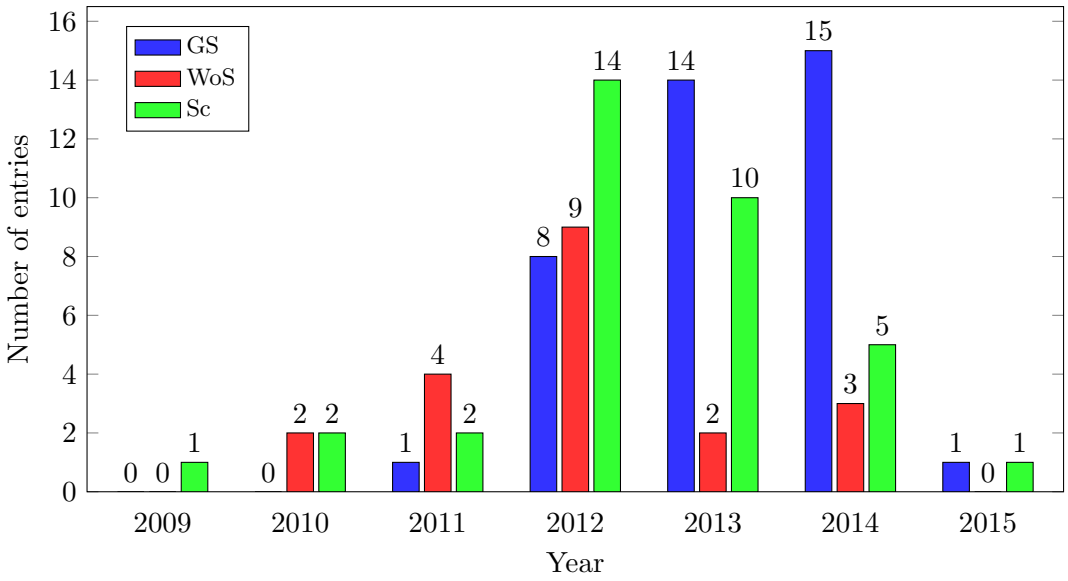

Figure 2: Number of citations per year

4.4. **Personal opinion.** The last question we would like to address is how user friendly the services are. This is clearly just a subjective comparison and, based on one's preferences, one may easily come up with a different result.

Google Scholar proved to be the best out of the three in data indexing but it is often too greedy. Many times it retrieve preliminary versions of papers from arXiv that were not even published. Also, the fact that it does not allow to export a search makes it useless for a meaningful data analysis. Moreover, query options of GS seems to be very limited. For example, ordering all citations by date shows only those that were published in the last year and there seems to be no way how to extend it to the previous years.

Web of Science, although it provides the most appealing user interface, is very slow and is sometimes hard to make it do what one would want to. For example, one can export searched citations to BibTeX only by going to the page of a particular paper and then listing the papers citing it (skipping the first step and going directly to the papers citing does not allow BibTeX export).

Clear winner in this category is Scopus. The interface is decent looking, easy to work with and the service does what the user asks for.

5. Conclusion

There is no clear winner. Google Scholar indexes much more papers than the other two but it does not provide such a good user interface and advanced tools as the others. Also, we did not aim to do a full analysis. For that, it is necessary to gather much more data.

Even with the little explored, it is clear that the current citation indexing services are still at the beginning and better tools need to be developed.

Acknowledgement

The author is grateful to Henning Thielemann for writing the Haskell library `bibtex` that allowed the analysis to be automatised.

References

[1] Martín Abadi and Gordon D. Plotkin. A model of cooperative threads. *Logical Methods in Computer Science*, 6(4), 2010.

[2] Martín Abadi and Gordon D. Plotkin. On protection by layout randomization. In *Proceedings of the 23rd IEEE Computer Security Foundations Symposium, CSF 2010, Edinburgh, United Kingdom, July 17-19, 2010*, pages 337–351. IEEE Computer Society, 2010.

[3] Peter Hancox. Assessment for students registered for PhD and MRes degrees. In *Research Skills Module*, pages 1–4. University of Birmingham, 2014.

[4] Michael Pedersen and Gordon D. Plotkin. A language for biochemical systems: Design and formal specification. *T. Comp. Sys. Biology*, 5945:77–145, 2010.

[5] Gordon D. Plotkin. Robin milner, a craftsman of tools for the mind. In *Proceedings of the 25th Annual IEEE Symposium on Logic in Computer Science, LICS 2010, 11-14 July 2010, Edinburgh, United Kingdom*, pages 58–59. IEEE Computer Society, 2010.

[6] Rob J. van Glabbeek and Gordon D. Plotkin. Configuration structures, event structures and petri nets. *CoRR*, abs/0912.4023, 2009.

Appendix A.

Here we provide the Haskell code we used for our analysis. It assumes that all the BibTeX databases are placed in folder `bibs` and named properly (see the end of the source code for the exact naming convention).

```
module Analyse where

import Text.BibTeX.Parse
import Text.BibTeX.Entry
import Text.Parsec.String
import Data.Char
import Data.List

data Type = Article | Conference | TechnicalReport | Book | Thesis | Unknown
    deriving (Show, Eq)

fromEntry :: String -> T -> [String]
fromEntry field entry
    = map (cleanBrackets . snd)
    $ filter ((== field) . lowerCase . fst)
    $ fields entry

yearFromEntry :: T -> [Int]
yearFromEntry = map read . fromEntry "year"

titleFromEntry :: T -> String
```

```
titleFromEntry = lowerCase . head . map clean . fromEntry "title"
    where
        clean = filter isAlphaNum

typeFromEntry :: T -> Type
typeFromEntry entry
    | typeName == "conference" || typeName == "inproceedings" = Conference
    | typeName == "incollection" || typeName == "article" || inJournal =
      Article
    | typeName == "book" = Book
    | typeName == "phdthesis" || typeName == "mastersthesis" = Thesis
    | typeName == "techreport" = TechnicalReport
    | otherwise = Unknown
    where
        typeName = lowerCase $ entryType entry
        inJournal = not . null $ fromEntry "journal" entry

lowerCase :: String -> String
lowerCase = map toLower

cleanBrackets :: String -> String
cleanBrackets ('{':s) = cleanBrackets $ reverse $ tail $ reverse s
cleanBrackets      s  = s

avrg :: [Double] -> Double
avrg l = (sum l) / (fromIntegral $ length l)

unique :: [String] -> [String]
unique = nub

intersections :: String -> [String] -> [String] -> [String] -> IO ()
intersections paper fromGS fromWoS fromSc = do
        putStrLn $ "- " ++ paper ++ ":"
        putStr   $ "    " ++ show g
        putStr   $ " & " ++ show w
        putStr   $ " & " ++ show s

        -- The rest is computed using Inclusion-Exclusion Principle
        putStr   $ " & " ++ show gwI
        putStr   $ " & " ++ show gsI
        putStr   $ " & " ++ show wsI
        putStrLn $ " & " ++ show (gwsU + gwI + gsI + wsI - (g + w + s))
    where
        card = length . unique
        iep2 first second together = (first + second) - together

        -- Unions
        g    = card fromGS
        w    = card fromWoS
        s    = card fromSc
        gwU  = card $ fromGS  ++ fromWoS
```

```
        gsU  = card $ fromGS  ++ fromSc
        wsU  = card $ fromWoS ++ fromSc
        gwsU = card $ fromGS  ++ fromWoS ++ fromSc

        -- Intersections
        gwI = iep2 g w gwU
        gsI = iep2 g s gsU
        wsI = iep2 w s wsU

yearsChart :: String -> [Int] -> IO ()
yearsChart name ys = do
        putStr $ "  " ++ name ++ ": "
        print chart
    where
        grouped = groupBy (==) $ sort ys
        chart   = map (\xs -> (head xs, length xs)) grouped

coreForms :: [[Type]] -> IO ()
coreForms typesXX = do
        putStrLn $ intercalate " & " [arts, cnfs, trps, boks, thes, unks, show
            count]
    where
        types = concat typesXX
        count = length types
        countType t = show $ length $ filter (== t) types

        arts = countType Article
        cnfs = countType Conference
        trps = countType TechnicalReport
        boks = countType Book
        thes = countType Thesis
        unks = countType Unknown

collectData :: String -> IO (Double, [Int], [String], [Type])
collectData name = do
    Right q <- parseFromFile file name

    let years  = concat $ map yearFromEntry q
        yAvrg  = avrg $ map fromIntegral years
        titles = map titleFromEntry q
        types  = map typeFromEntry q

    putStr $ name ++ ": "
    print yAvrg
    return (yAvrg, years, titles, types)

main :: IO ()
main = do
    (c2g, ys2g, t2g, ty2g) <- collectData "bibs/cit2.gs.bib"
    (c2w, ys2w, t2w, ty2w) <- collectData "bibs/cit2.wos.bib"
    (c2s, ys2s, t2s, ty2s) <- collectData "bibs/cit2.sc.bib"
```

```
(c3g, ys3g, t3g, ty3g) <- collectData "bibs/cit3.gs.bib"
(c3w, ys3w, t3w, ty3w) <- collectData "bibs/cit3.wos.bib"
(c3s, ys3s, t3s, ty3s) <- collectData "bibs/cit3.sc.bib"

(c4g, ys4g, t4g, ty4g) <- collectData "bibs/cit4.gs.bib"
(c4w, ys4w, t4w, ty4w) <- collectData "bibs/cit4.wos.bib"
(c4s, ys4s, t4s, ty4s) <- collectData "bibs/cit4.sc.bib"

(c5g, ys5g, t5g, ty5g) <- collectData "bibs/cit5.gs.bib"
(c5w, ys5w, t5w, ty5w) <- collectData "bibs/cit5.wos.bib"
(c5s, ys5s, t5s, ty5s) <- collectData "bibs/cit5.sc.bib"

putStrLn "\nAverage years:"
putStrLn $ "  GS: " ++ show (avrg [c2g, c3g, c4g, c5g])
putStrLn $ "  WoS: " ++ show (avrg [c2w, c3w, c4w, c5w])
putStrLn $ "  Sc: " ++ show (avrg [c2s, c3s, c4s, c5s])

putStrLn "\nYears chart data:"
yearsChart "GS" $ concat [ys2g, ys3g, ys4g, ys5g]
yearsChart "WoS"$ concat [ys2w, ys3w, ys4w, ys5w]
yearsChart "Sc" $ concat [ys2s, ys3s, ys4s, ys5s]

putStrLn "\nIntersections (GS, WoS, Sc, GS+WoS, GS+Sc, WoS+Sc, GS+WoS+Sc):
    "
intersections "Paper 2" t2g t2w t2s
intersections "Paper 3" t3g t3w t3s
intersections "Paper 4" t4g t4w t4s
intersections "Paper 5" t5g t5w t5s

putStrLn "Core forms of literature (Article, From a Conference, Technical
    Report, Book, Thesis, Unknown, Total):"
putStr "  GS: "
coreForms [ty2g, ty3g, ty4g, ty5g]
putStr "  WoS: "
coreForms [ty2w, ty3w, ty4w, ty5w]
putStr "  Sc: "
coreForms [ty2s, ty3s, ty4s, ty5s]
```