Use of Locality Sensitive Hashing (LSH) Algorithm to Match Web of Science and SCOPUS

Mehmet Ali Abdulhayoglu[1,2](0000-0002-1288-2181), Bart Thijs[1](0000-0003-0446-8332)

[1]ECOOM, Center for R&D Monitoring, FEB, KU Leuven, Leuven (Belgium)
[2]Faculty of Business and Economics, Department of Managerial Economics, Strategy and Innovation, KU Leuven, Leuven (Belgium)
(mehmetali.abdulhayoglu✉, bart.thijs)@kuleuven.be

Abstract
The objective of this study is to find the most appropriate parameters and text components for item-wise matching the two large bibliographic datasets: Clarivate Analytics Web of Science (WoS) and Elsevier's Scopus. Our focus is on detecting exact matches, that is, no false positives are tolerated at all. To this end, we follow a twofold matching procedure. First, a locality sensitive hashing (LSH) algorithm [15] is applied, which provides fast approximate nearest neighbours and similarities, in order to obtain WoS-Scopus pair suggestions. We experiment with three different combinations of text components (i.e., only publication titles, titles + journal names, co-author names + titles + journals) as input for the matching process. In addition, different values for LSH input parameters (i.e., number of random vectors, number of different random vector sets, number of neighbours, similarity threshold) are tested. Second, for each suggested pair, different heuristics are applied to identify those pair of records that indeed refer to the same publication. For example, the pairs are classified as correct matches if the journal name, volume, issue and begin page do match. We achieved the best results when only titles were matched and 50-50-50-0.80 or 100-30-30-0.80 input parameters are used. We observe that at least 70% of WoS publications are also indexed by Scopus. Last but not the least, when the parameters leading to the best matching results were applied, it took just about an hour to match 1.6 million vs 2.2 million.

Keywords: locality sensitive hashing (lsh), character n-gram, information retrieval from bibliographic databases, bibliographic database overlap

1 Introduction

In a previous paper [1], we had suggested a string matching system for short texts to find out whether given bibliographic references are indexed in a bibliographic database (BDB). In that paper, we used character 3-grams for this task comparing them with an approach based word unigrams. We respectively obtained an accuracy of 96.0% and 94.7% for character 3-grams and word unigrams when a cosine similarity threshold of 0.60 was used. We searched about 8,500 references within more than 35 million publications indexed in a bibliographic database (BDB) namely Clarivate

Analytics Web of Science database (WoS). We showed that character 3-grams could be more useful to obtain correct matches when erroneous or misspelling texts appear. Decreasing the required heavy manual job, the system works well when thousands of references are searched in a BDB with several millions of indexed papers. For example, we could obtain the matching results in 1 hour when those references were matched with 1.8 million papers published in 2007. However, from our experiences, when the number of publications is increased up to 50 thousands to be searched within the same source, the computation time reaches about 8 hours or so. This is because complexity of our system is still $O(kn^2)$ even though we could decrease k to a certain degree.

As a result, that procedure is not applicable when millions of records need to be searched in a BDB, for example, a task of measuring the overlap between two BDBs. Similar overlapping tasks have already been addressed by IR researchers [16]. In bibliometrics, this task has been of interest and achieved either on the journal level or with a limited number of articles since the paper based approach is daunting due to the high volume of indexed articles. However, two BDBs may index different publications from the same journal issue as [14] stated and hence an article based approach may yield more reliable results or confirm the previous literature results. In this context, we carried out an article based overlapping process for WoS and Elsevier's SCOPUS which are the two most extensive BDBs.

With the recent technological advances in distributed computing, such previously tedious tasks can be accomplished. To fulfill this as a complementary approach to our previous procedure, we applied another method based on locality sensitive hashing (LSH), which is an algorithm aiming to find approximate nearest neighbors [9]. To this end, we used a Spark (http://spark.apache.org/) library, which can be found in https://github.com/soundcloud/cosine-lsh-join-spark. This library is an implementation of [15] where the authors have built their work on [10] and [3] who are the pioneers of this fast algorithm. The details about the algorithm are given in the subsequent section.

To our knowledge, our attempt is the first to measure the overlap between WoS and SCOPUS at the article level without drawing samples. Based on our application, we managed to match millions of records from WoS with another millions of records from SCOPUS in a manageable time. However, we need to point out that this study is a preliminary endeavour using only the papers from one publication year, which aims to show the possibility of managing high volume of bibliographic data. Making the comparison of the DBs for all their indexed records remains as our future work. On the other hand, we present different input values to be applied for the LSH algorithm and suggest the ones outperforming the other options in terms of matching accuracy and running time.

2 Related work

[7] gives a very comprehensive literature review about the topic while [8] briefly gives examples of database comparisons from the library and information science

field. The degree of overlapping between BDBs has some indications, that is, high overlap indicates that there is no need to have both sources which saves the funds whereas low overlap requires to have all the sources increasing the costs [7]. The cost issue is even more crucial when costly databases are in question such as Web of Science and SCOPUS [6] as the most extensive bibliographic databases and our main sources in this study.

Comparison between the two BDBs has been in question among researchers from library and information science [2, 5, 6, 13].

3 Data

We chose articles, reviews and proceeding papers published in 2011 and indexed in WoS or SCOPUS databases. As a result, 1,635,395 and 2,255,989 papers were respectively retrieved and used for our application.

4 Locality-sensitive hashing (LSH)

Matching records from two distinct bibliographic databases is considered as a special application of the nearest neighbor search problem. As opposed to approaches such as matrix multiplication optimization [12] for between record cosine similarity calculations we opt for locality-sensitive hashing (LSH) being another approach for matching tasks aiming to fulfill the process with significantly lower complexity. Indeed, the LSH algorithm we applied estimates these cosine similarities between input vectors with a significantly low computational complexity. When calculating the cosine similarity in a traditional way, one has to match all the items (*n*) having *k* features in corpus leading to $O(kn^2)$ complexity. This makes it impossible to apply when *n* is too large. To remedy this bottleneck, [15] apply some rules for their LSH application.

First, it represents each vector with bit streams (called fingerprints or signatures) via randomly chosen spherically symmetric vectors of unit length. The number of random vectors (*d*) is given as input. As a result, a higher dimension is reduced to a number of random vectors. Figure 1 depicts this process in a simple way. Note that this figure is based on two figures by Benjamin Van Durme & Ashwin Lall in their presentation for 48th Annual Meeting of the Association for Computational Linguistics to present their paper [17] and used here with their kind permission. We merged the related figures to present the idea into one single figure. As seen, there are two points (A and B) and six random hash functions in the figure. Point A stands above h1, h5 and h6 and below the other three functions. Similarly, point B stands above h1 and h5. As a result, A and B have the following respective fingerprints, [1,0,0,0,1,1] and [1,0,0,0,1,0]. Once all fingerprints are retrieved, the list is sorted to calculate the similarity between closest points.

Second, the algorithm applies a random permutation function a given number of times (*q*) for each fingerprint. We can explain this process as follows: In the given example above, *d*=6 random vectors are used to obtain fingerprints. However, for the same points, different fingerprints can be obtained when another 6 random vectors are used. This means that each fingerprint will have a different place surrounded by different neighbours in the sorted list whenever a different random vector set is used. The more random vector sets are introduced, the more likely it is to reach correct matches. For each sorted list, (*B*) closest neighbours are retained for each vector. Eventually, the algorithm calculates the cosine similarity between the vectors and their neighbours and retains only those matches exceeding a similarity threshold input value. These processes approximately reduce the computational complexity to *O(dn)* and it can be run parallel thanks to its random processing. As [15] states, the higher *d* and *q* values result in more accurate matches but with a higher calculation time. In addition, the authors also warn that proper input values (*d*, *q*, *B*) differ depending on the domain. Therefore, we tried and suggested different input values in our bibliographic reference matching application.

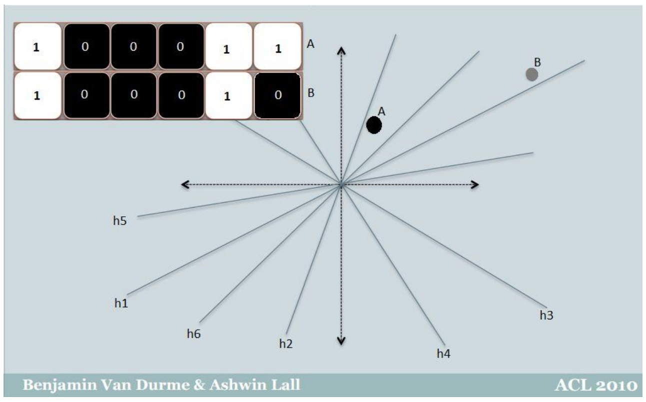

Figure 1. Obtaining fingerprints of input vectors via randomly picked planes.

5 Methodology

For the applied LSH code, we tried six different input sets in an arbitrary manner. The details are presented in the results section. We aimed to try different input values in order to observe the changes in the running time and the number of correct matches. To this end, we first chose some values leading to very fast computational time with a trade-off in accuracy. On the other hand, we tried some other values leading to significantly more correct matches with a trade-off in the running time. So in total, we used six different input sets to have an idea about the most optimum input values. Note that our main goal is to show the possibility of matching high volume bibliographic data and obtain as many correct matches as possible in a manageable time. Although obtaining the results as fast as possible would be so valuable for dynamic systems, this is not our priority in this study.

As the main sources, co-author names, publication title, source name (journal or conference name), publication year, begin page and end page from WoS or SCOPUS

DBs were used for our text matching procedure. We used three different scenarios as given below.

- Co-authors + title + source name + publication year + begin page + end page (ALL)
- Title + source (TI_SO)
- Title (TI)

The code we applied requires an input file in LIBSVM format [4]. It simply represents documents with indices and values where each index represents a *character 3-gram* in our case and the value stands for the frequency of the related 3-gram appearing in the document. For example, 1 1:3 2:7 12:1 14:5 … tells that reference 1 has a character 3-gram (e.g. *end*) indexed as 1 appearing three times, a character 3-gram indexed as 2 appearing seven times, a character n-gram indexed as 12 appearing once and so on. To this end, we created text files in this format for WoS and SCOPUS for different reference component combinations as given above.

Then for each scenario, two text files were merged into one. Moreover, the code was originally designed to find the duplicate records in a given input text file. Since in our case there was no need to check the pairs from the same BDB, we modified the code in a way that it only checked the pairs where one coming from WOS and the other from SCOPUS. Using one big text file for each of the three different scenarios, we ran our code and made our observations in a server with 36 processors. Figure 2 depicts how we prepared the input files for a clearer understanding. Note that for each reference combination (*all reference*, *title + source*, *title*) a different input file was created.

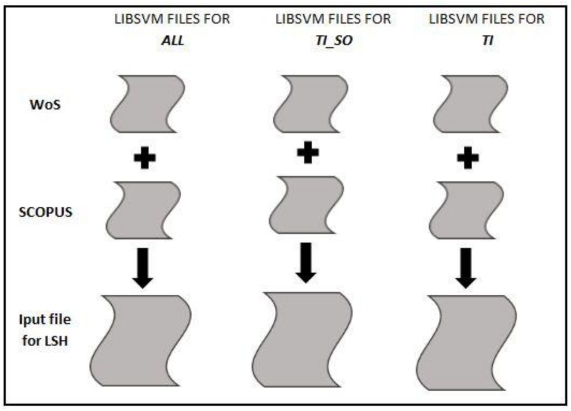

Figure 2. Preparing input files to be used in LSH code.

Once we retrieved the pair suggestions with a similarity score higher than a desired threshold, we followed some heuristics to confirm the results and obtained identical matches. This step is necessary since the derived similarity scores by LSH algorithm are estimated values thus it may offer wrong matches despite a high similarity score.

To this end, we classified pairs as correct matches if they comply with one of the following rules:

Rule 1.	Identical source name (so) (journal or conference name),volume (vl), issue (is) and begin page (bp) or
Rule 2.	Identical so, vl, is and title (ti) when bp is null or
Rule 3.	Identical so, is, bp and ti when vl is null or
Rule 4.	Identical so, is, and ti when vl and bp are null.

Note that above given scenario names (ALL, TI_SO, TI) and rules (Rule 1, Rule 2 so on) are used in Table 3 when presenting the results in detail.

Here we should point out two important facts. First, in the above given rules, we only considered identical titles for Rule 2, 3 and 4. By applying very high similarity scores, thousands of more correct matches could be retrieved. For example, consider the two following titles with a similarity score of 0.85 according to a 3-grams based edit distance measure [11].

A 95-nA, 523ppm/degrees C, 0.6-mu W CMOS Current Reference Circuit with Subthreshold MOS Resistor Ladder (WoS)

A 95-nA, 523ppm/Â°C, 0.6-Î¼WCMOS current reference circuit with sub threshold MOS resistor ladder (SCOPUS)

In our result set, there were around 10,000 pairs having a similarity score between 0.80 and 0.99 ignored by above given four rules. Even though we can intuitively count those pairs as correct, they still need to be checked manually especially for micro level bibliographic analysis. Since we focus on the possibility of matching two large datasets with a possible small manual effort in this study, we skipped that manual stage and did not involve those pairs.

Second, the databases may index some source names differently. Since checking all the source name pairs manually was not realistic, we reduced the number of pairs by retaining only those sources involving identical publication titles with identical begin pages and issue numbers. As a result, we ended up 3,640 source pairs to check manually. To ease this manual job, we leveraged similarity score between source names applying [11] and the frequencies of those pairs. Confirming the pairs having very similar names or co-occurring a lot was easier and this allowed us to complete the manual check within a work day. Table 1 gives some examples of such pairs.

Here we should mention that journals' International Standard Serial Numbers (ISSN) could also be exploited to match source names. However, through our approach, we already included all the ISSNs information and obtained even more records since ISSNs might change in time thus inconsistency might appear or they might simply be missing in BDBs. Nevertheless, this was valid for our data and it might be significantly valuable for other similar tasks. Similarly, Digital Object Identifiers (DOI) are very valuable to retain the correct matches since they are unique for each publication. However, BDBs might still be short of this information. For example, in our case, almost one-third of the publications' DOIs are not indexed in SCOPUS. Like ISSN, this information can be very valuable for other similar tasks.

Table 1. Examples of WoS-SCOPUS source name pairs with their co-occurrences and similarity
[Data sourced from Clarivate Analytics Web of Science Core Collection and Elseviers Scopus]

WoS source name	*SCOPUS source name*	*Similarity score*	*Co-occurrence frequency*
PHYSICS LETTERS A	Physics Letters, Section A: General, Atomic and Solid State Physics	0.24	449
LANCET	The Lancet	0.50	234
IIC-INTERNATIONAL REVIEW OF INTELLECTUAL PROPERTY AND COMPETITION LAW	IIC International Review of Intellectual Property and Competition Law	0.98	33
REVISTA BRASILEIRA DE ZOOTECNIA-BRAZILIAN JOURNAL OF ANIMAL SCIENCE	Revista Brasileira de Zootecnia	0.46	294
ADVANCED RESEARCH ON COMPUTER SCIENCE AND INFORMATION ENGINEERING	Communications in Computer and Information Science	0.44	41

The first three examples in the table are relatively easier to confirm compared to the last one since the source names are totally different and a detailed check was needed. Examining that pair in detail, we saw that in one source a more detailed conference or journal name was given while the other indexed a general series name. In our example, *Communications in Computer and Information Science* was the general name of the series which involved *ADVANCED RESEARCH ON COMPUTER SCIENCE AND INFORMATION ENGINEERING* amongst others. Confirming such pairs may cause the manual work last longer depending on the data. Nevertheless, in our approach the process was viable and valuable. So we accepted those manually checked source names as identical and involved them our matching process given above. As a result, we roughly gained 185,000 additional correct matches.

6 Results

We tried six different input values. Before giving the matching results, we would like to discuss the computational time based on the input values. Table 2 shows approximate elapsed times based on the used input values. The numbers are based on the results when only titles (TI) were used. For the other two scenarios, we did not observe significant deviations on elapsed times.

Table 2. Input values and elapsed time to obtain the matching results when using only titles

Number of random vectors (d)	*Number of neighbors (B)*	*Number of permutations (q)*	*Cosine similarity threshold*	*Elapsed time*
20	5	5	0.80	3 minutes
20	20	20	0.80	11 minutes
50	50	50	0.80	1.2 hours
100	20	20	0.80	31 minutes
100	30	30	0.80	58 minutes
100	50	100	0.80	9.3 hours

In Table 2, number of random vectors (d), number of neighbours (B), number of permutations (q) and cosine similarity threshold are given respectively starting from the first column. For example, we retrieved some matching results having a similarity score of higher than 0.80 in 3 minutes when 20 random vectors, 5 closest neighbours and 5 permutations were introduced. Note that when lower cosine thresholds were applied, computational time was affected significantly in a negative manner. For the lower thresholds, there appeared too many identical suggestions coming from different permuted lists which caused more computational time to sort those duplicates out. Experiencing this bottleneck, we opted 0.80 as the threshold for our procedure. As also seen in the table, increasing d and q input values results in more computational cost. Table 3 gives a detailed summary of matching results according to the different input values for each component combination. Moreover, number of correct matches for four different cases, as given in the methodology section, is detailed.

We observed that around 70% of WoS records and 51% of SCOPUS papers could be correctly matched in half an hour. In addition, we retrieved our results even faster when the High Performance Computing platform at our university was employed. For example, for the input values 50-50-50-0.80, it lasted more than 1 hour to get the results while it was less than half an hour in the cluster environment. Computational time can be decreased more if a cluster can be set with ideal configurations. This remains as our future work.

Our results showed that when pair suggestions were retrieved based on only titles (TI), many more identical matches were obtained. This might be because the components like source name or co-authors name introduce more noise leading to a similarity score lower than the given threshold. As we already mentioned, lowering the threshold would end up more computational time for the applied algorithm. As a result, we can suggest that for the given input values, using TI seems better to collect more identical matches.

The first striking result is that applying 20-05-05 input values, we could retrieve more than 560,000 (34.29%) and 424,000 (25.93%) identical results for WoS in three minutes (see Table 2) when TI and TI_SO were employed respectively. On the other hand, for the same input values, we could only obtain 92,000 (5.63%) identical values when applying ALL. Depending on bibliographic research aim, having more than half million papers indexed by both sources in such a short time might be quite valuable.

Table 3. Number of identical matches between WoS and SCOPUS based on different input values for different component combinations
[Data sourced from Clarivate Analytics Web of Science Core Collection and Elseviers Scopus]

LSH Scenarios	***Rule***	***20-05-05***	***20-20-20***	***50-50-50***	***100-20-20***	***100-30-30***	***100-50-100***
ALL	Rule 1	86,858	266,953	981,501	902,284	980,538	
	Rule 2	4,407	13,782	46,117	42,031	46,195	
	Rule 3	746	2,072	5,882	5,526	5,865	
	Rule 4	114	368	1,542	1,443	1,583	
		92,125	283,175	1,035,042	951,284	1,034,181	
	SCOPUS	**4.08%**	**12.55%**	**45.88%**	**42.17%**	**45.84%**	
	WoS	**5.63%**	**17.32%**	**63.29%**	**58.17%**	**63.24%**	
TI_SO	Rule 1	397,688	738,454	1,077,348	1,068,364	1,076,387	
	Rule 2	22,368	39,805	51,735	51,742	51,747	
	Rule 3	3,453	5,373	6,356	6,355	6,356	
	Rule 4	559	1,180	1,736	1,737	1,737	
		424,068	784,812	1,137,175	1,128,198	1,136,227	
	SCOPUS	**18.80%**	**34.79%**	**50.41%**	**50.01%**	**50.36%**	
	WoS	**25.93%**	**47.99%**	**69.54%**	**68.99%**	**69.48%**	
TI	Rule 1	530,038	909,013	1,091,480	1,088,727	1,091,649	1,094,164
	Rule 2	26,173	44,415	51,643	51,633	51,637	51,635
	Rule 3	3,562	5,686	6,348	6,347	6,350	6,350
	Rule 4	983	1,526	1,735	1,735	1,735	1,735
		560,756	960,640	1,151,206	1,148,442	1,151,371	1,153,884
	SCOPUS	**24.86%**	**42.58%**	**51.03%**	**50.91%**	**51.04%**	**51.15%**
	WoS	**34.29%**	**58.74%**	**70.39%**	**70.22%**	**70.40%**	**70.56%**

Second, among all the trials, we could automatically reach the highest number of identical matches when either 50-50-50 (70.39%) or 100-30-30 (70.40%) input values were used for TI. The same input values were also the best options for other two component combinations. Note that we could indeed retrieve slightly more identical records when 100-50-100 input values were used as given in the last column of TI section in Table 3. However, that procedure lasted more than 9 hours (see Table 2) and gave only 2,500 more records compared to our best results. Here we aimed to find some input values providing matching suggestions in a considerably short time. In this context, the higher input values like 100-50-100 make no sense since the trade-off does not seem valuable at all since the added value is too small with a very long computational time.

Aside from number of identically matched publications given by Table 3, we would like to discuss the cited publications. In this context, our WoS database has 1,127,239 (68.93%) cited publications in the type of article, review or proceeding where we detected that at least 1,002,478 (88.93%) of them are also indexed by SCOPUS. When highly cited publications are in question, similar results were obtained. For example, there are 304 and 9,652 publications cited more than 500 and 100 times, respectively where 264 (86.84%) and 8,741 (90.56%) of them were found to be indexed by both sources.

These results are in line with the related literature works that is a high overlap between two important sources has been observed. Furthermore, we grabbed more than one million joint publications which might pave the way for detailed paper based bibliographic analysis on common papers unlike previous works based only on journal comparison.

7 Conclusion

Two large bibliographic datasets (1.6 million vs. 2.2 million) from WoS and SCOPUS databases were matched at paper level using locality sensitive hashing (LSH) trying to find nearest neighbours. We experimented with different input values required by the algorithm. As a result, we found and suggested the ones providing the best results in terms of the number of identical matches and computational time. Based on the suggested approach, we automatically found that at least 70% of the publications (article, review or proceeding) indexed by WoS were also indexed by SCOPUS. Moreover, when we examined the matching results for those publications cited at least once, we observed more number of identical matches such that at least 88.93% of the cited publications in WoS were also cited by SCOPUS.

References

1. Abdulhayoglu. M. A., Thijs. B., & Jeuris. W. (2016). Using character n-grams to match a list of publications to references in bibliographic databases. Scientometrics. 109(3). 1525-1546. doi: 10.1007/s11192-016-2066-3
2. Bosman. J., Mourik, I. V., Rasch. M., Sieverts. E., & Verhoeff. H. (2006). Scopus reviewed and compared: The coverage and functionality of the citation database Scopus. including comparisons with Web of Science and Google Scholar. Utrecht University Library.
3. Charikar. M. S. (2002). Similarity estimation techniques from rounding algorithms. In Proceedings of the thiry-fourth annual ACM symposium on Theory of computing. 380-388. ACM. doi: 10.1145/509907.509965

4. Chih-Chung Chang & Chih-Jen Lin, (2011). LIBSVM : a library for support vector machines. ACM Transactions on Intelligent Systems and Technology. 2:27:1--27:27. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm. doi: 10.1145/1961189.1961199
5. Egghe. L. & Goovaerts. M. (2007). A note on measuring overlap. Journal of information science. 33(2). 189-195. doi: 10.1177/0165551506075325
6. Gavel. Y. & Iselid. L. (2008). Web of Science and Scopus: a journal title overlap study. Online information review. 32(1). 8-21. doi: 10.1108/14684520810865958
7. Gluck. M. (1990). A review of journal coverage overlap with an extension to the definition of overlap. Journal of the American Society for Information Science. 41(1). 43-60. doi: 10.1002/(SICI)1097-4571(199001)41:1<43::AID-ASI4>3.0.CO;2-P
8. Hood. W. W. & Wilson. C. S. (2003). Overlap in bibliographic databases. Journal of the American Society for Information Science and Technology. 54(12). 1091-1103. doi: 10.1002/asi.10301
9. Indyk. P. (2000). High-dimensional computational geometry. Doctoral dissertation. Stanford University.
10. Indyk. P. & Motwani. R. (1998). Approximate nearest neighbours: towards removing the curse of dimensionality. In Proceedings of the thirtieth annual ACM symposium on Theory of computing. 604-613. ACM. doi: 10.1145/276698.276876
11. Kondrak. G. (2005). N-gram similarity and distance. In International Symposium on String Processing and Information Retrieval. 115-126. Springer Berlin Heidelberg. doi: 10.1007/11575832_13
12. Kurzak, J., Alvaro, W., & Dongarra, J. (2009). Optimizing matrix multiplication for a short-vector SIMD architecture–CELL processor. Parallel Computing, 35(3), 138-150. doi: 10.1016/j.parco.2008.12.010
13. Meho. L. I. & Rogers. Y. (2008). Citation counting. citation ranking. and h-index of human-computer interaction researchers: a comparison of Scopus and Web of Science. Journal of the American Society for Information Science and Technology. 59(11). 1711-1726. doi: 10.1002/asi.v59:11
14. Pao. M. L. (1993). Term and citation retrieval: A field study. Information Processing & Management. 29(1). 95-112. doi: 10.1016/0306-4573(93)90026-A
15. Ravichandran. D., Pantel. P. & Hovy. E. (2005). Randomized algorithms and nlp: using locality sensitive hash function for high speed noun clustering. In Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics. 622-629. Association for Computational Linguistics. doi: 10.3115/1219840.1219917
16. Spink, A., Jansen, B. J., Kathuria, V., & Koshman, S. (2006). Overlap among major web search engines. Internet Research, 16(4), 419-426.
17. Van Durme. B. & Lall. A. (2010). Online generation of locality sensitive hash signatures. In Proceedings of the ACL 2010 Conference Short Papers. 231-235. Association for Computational Linguistics.