GraphDBLP Released: Querying the Computer Scientists Network as a Graph

Mirko Cesarini\textsuperscript{1,2}, Fabio Mercorio\textsuperscript{1,2}, Mario Mezzanzanica\textsuperscript{1,2}, Vincenzo Moscato\textsuperscript{3}, and Antonio Picariello\textsuperscript{3}

\textsuperscript{1} Dept. of Statistics and Quantitative Methods, Univ. of Milano-Bicocca, Italy  
\textsuperscript{2} CRISP Research Centre, Univ. of Milano-Bicocca, Italy  
\textsuperscript{3} Dept. of Electrical Engineering and Information Technology (DIETI), Univ. of Naples-Federico II, Naples, Italy  
(discussion paper)

Abstract. In this paper we introduce GraphDBLP, a tool that models the DBLP bibliography as a graph, and enriches the DBLP data through semantic keyword similarities computed via word-embedding. GraphDBLP has been implemented on top of the Neo4j graph-database, and it can be queried through the Cypher query language. We also provide three meaningful queries for exploring the DBLP community to (i) investigate author profiles by analysing their publication records; (ii) identify the most prolific authors on a given topic, and (iii) perform social network analyses over the whole community. GraphDBLP is available on Github. To date, it contains 5+ million nodes and 24+ million relationships, enabling users to explore the DBLP data by referencing more than 3.3 million publications, 1.7 million authors and more than 5 thousand publication venues. Thanks to the use of word-embedding, more than 7.5 thousand keywords and related similarity values were collected.

1 Introduction and Related Work

Academic researchers and their interactions can be seen as a network, which includes different topics, interests, research products published, and venues, too. The discovery of researchers’ similarities, the estimation of these similarities, the identification of the authors that mostly contribute to a given topic, and the identification of similarities between different topics are important tasks with a practical significance in many fields, as in the case of community mining [14, 8], social networking [6, 16], and influence analysis as well [18, 2]. The idea behind GraphDBLP - firstly presented in [11] - is to organise the computer scientist network as a graph, using state-of-the-art word embedding algorithm to discover similarities between researchers, and to deploy the derived knowledge within a tool that anyone can use and improve over time.

To this end, GraphDBLP uses the DBLP [10] bibliography as a baseline. Several works have discussed the problems arising from analysing and monitoring the computer science community, such as expert-finding [13, 4], community detection [17, 2], community mining [9, 20], and keyword extraction [5]. Focusing on DBLP, the DBConnect project [20] aimed at exploiting random walks on the DBLP data model, measures the closeness between any two entities, to discover the community structure of the data,
and to recommend collaborations. The FacetedDBLP [5] tool performs a high-quality selection and review of DBLP keywords. At that point, then the user can search publications using these refined keywords as a starting point. On the other hand, DBLP has also been at the basis of several studies and tools, as in [7], which used DBLP to analyse a pre-selected list of conferences/journals in a specific research field, and to understand how the community evolved over time in terms of topics and research collaborations. In [13], instead, DBLP was used to learn the academic ranking of experts according to a specific topic. Finally, in [2], DBLP evaluated the influence of relationships among communities in dynamic social networks.

However, though all these approaches and projects are effective and provide a substantial contribution, they have only focused on a specific and pre-defined task. On the other hand, modelling the DBLP data as a graph would allow applying graph-based algorithms (graph-traversal, shortest-path, clustering coefficient, etc) to enrich the DBLP data through new relationships. Indeed, DBLP contents can be also analysed from a semantic perspective through word embedding, which enables semantic similarities among contents to be modelled and extracted.

As one might note, GraphDBLP also shares with the methods and tools discussed above most of the community mining purposes and features. However, it has some distinctive elements that deserve to be discussed.

(i) GraphDBLP computes (a) similarity metrics among venues\(^4\) and (b) metrics by weighting the authors’ scientific production w.r.t to topics (namely, keywords). The former (a) is computed by focusing on the authorship networks; The latter (b) focuses on an author’s publication records, estimating the weight of a keyword among all the author’s publications (i.e., the score) and the author’s prolificness on the whole DBLP community that is working on that keyword (i.e, the relevance). Hence, several interesting analyses may be computed e.g., investigation of author activities, comparison of author profiles of identification of influential authors;

(ii) GraphDBLP employs a language model (namely, word2vec) for the lexicon used in the titles of the papers to build-up a map of keyword similarities. The keyword similarity information is obtained (or updated) by automatically processing the DBLP data (rather than by using external dictionaries or taxonomies). These data are valuable as they improve the results of the analyses (e.g., “AI” recognised as similar to “Artificial Intelligence” with a score of 87% by learning lexical variants where they appear);

(iii) Knowledge retrieved from DBLP is represented as a graph, upon which Social Network Analysis (SNA) and graph-traversal queries can be performed;

(iv) GraphDBLP is implemented on top of an open-source graph database management system (i.e., Neo4j) and the whole source code for obtaining the GraphDBLP dataset has been made publicly available, so that anyone might contribute to the improvement of the graph instance of DBLP.

To date, our system allows performing community mining over 3.3M+ publications, 1.7M+ authors, 5K+ venues (including conferences and journals) and 7.5K+ keywords extracted from DBLP data through word embedding. The multi-graph generated

---

\(^4\) In this work, venues include conferences and journals
is composed of 5,173,049 nodes and 24,753,736 relationships. GraphDBLP has been released in 2018 and publicly available on GitHub.

2 GraphDBLP at a Glance

In essence, GraphDBLP is a tool that allows users to perform graph-based queries on the DBLP data and some derived knowledge from it through:

(i) Venue similarities computed on the basis of the author communities;
(ii) Keywords similarities identified through word embeddings on keywords inherited from the FacetedDBLP project;
(iii) Authors research topics computed by analysing the publication records of each author. It also estimates the weight of a given keyword in the author publications (i.e., the score) as well as the global prolificness of the author in the whole DBLP community working on that keyword (i.e., the relevance).

The data model of GraphDBLP is depicted in Fig. 1 using the four building blocks of the Neo4j graph-database, which are labels, nodes, relationships, and properties. Information that can be directly inferred from the DBLP dataset includes the labels Venues, Authors, and Publications. Since the graph-based model supports entity hierarchies (in contrast with the relational model), a publication can be either an Article or an InProc or both. Information about the Keyword label comes from the FacetedDBLP project keywords, while the Keyword_Sim represents additional keywords that have been found in the DBLP titles. Similarly, the solid lines in Fig. 1 represent relationships that can be derived from the DBLP and FacetedDBLP files. The dotted lines represent relationships created by reasoning over the DBLP graph, which are: (i) SIMILARITY, (ii) SIMILAR_TO, and (iii) HAS_RESEARCH_TOPIC. From a formal point of view, we modeled our graph-database as a directed multi-graph as follows.

Definition 1 (Directed labelled multigraph). A Directed labelled multigraph $G$ is a tuple $(N, E, L_N, L_E, F; 1, \nu, \xi, \sigma, \varsigma)$ composed of the following elements:

- $N$ is a finite set of nodes;
- $E$ is a finite set of edges;
- $L_N$ is a finite set of node labels;
- $L_E$ is a finite set of edge labels;
- $F$ is a finite set of property labels;
- $1$ is the incidence function that assigns to an edge $e \in E$ a pair of nodes $u_1, u_2 \in N$;
- $\nu$ is the node labelling function $\nu : N \rightarrow L_N$ that assigns a label $l \in L_N$ to a node $u \in N$;
- $\xi$ is the edge labelling function $\xi : E \rightarrow L_E$ that assigns a label $l \in L_E$ to each edge $e \in E$;
- $\sigma$ is the edge property function $\sigma : E \times L_E \times P \rightarrow D$ that assigns a value $v \in D$ to the property named $p \in P$ for the edge $e \in E$ with label $l \in L_E$. Notice that $D$ can be numbers, boolean, or strings;
- $\varsigma$ is the node property function $\varsigma : N \times P \rightarrow D$ that assigns a value $v \in D$ to the property $p \in P$ for the node $u \in N$.

Note that the incidence function is needed for connecting a pair of nodes through an edge, leaving the edge labelling function in charge of specifying which labels affect

---

5 fabiomerco.github.io/GraphDBLP/

6 Though the same result could be achieved adding a property on the node, the use of multiple labels allows one to immediately access to the nodes with the desired label.

7 A multi-graph is a graph where multiple edges between two nodes are permitted and might be specified through labels. Our notation was inspired by [3].
the node connections. The directed labelled multigraph is the formalism Neo4j uses to model a graph-store. Specifically, Neo4j is a property graph database, which means it can model attributed, labelled, directed multi-graphs. It is composed of four building blocks that one can easily map on the Def. 1, see [11] for details.

In the following we briefly describe how the additional knowledge has been derived from DBLP (i.e., the dotted relations of Fig. 1).

**Building the SIMILARITY relation.** The SIMILARITY relation estimates similarity between two venues based on the collaboration network that publishes there. Specifically, a collaboration network consists of a network of authors that publish on specific venues (i.e., the CONTRIBUTED_TO relation of Fig. 1). In this way, we can compute the similarities between two venues $v_1$ and $v_2$ based on the authors that they have in common. We decided to employ the Jaccard index computed on authors. Let $A_v$ be the set of authors that contributed to venue $v$, the Jaccard Index between two venues $v_i$ and $v_j$. Notice that the similarity between venues (they can be either journals or conferences) does not rely on keywords or on citations, but on the collaboration network that contributed to those venues. This is an important characteristic of our approach as it aims at discovering the implicit behaviour of authors, rather than the explicit, which emerges from the citations.

**Building SIMILAR_TO relation.** Research keywords are an important element that enables the differentiation of research communities. Since DBLP does not explicitly provide neither topics nor abstracts of the stored publications, we decided to exploit the words of publication titles to derive research keywords. To this end, we used the Faceted-DBLP project, which uses GrowBag graphs for identifying computer-science specific keywords [5]. Then, for each title, we computed a list of top-k most similar keywords through word embedding.

Specifically, vector representation of words maps each word of a given lexicon to a unique vector in the corresponding N-dimensional space. In our context, each word can be considered as the title of a research product. Here, an important contribution has been given by the Word2Vec algorithm [12], that computes the vector representations
of words by looking at the context where these words are used. Intuitively, given a word \( w \) and its context \( k \) (i.e., \( m \) words in the neighborhood of \( w \)), it uses the context \( k \) as a feature for predicting the word \( w \). After the Word2vec training on the lexicon, words with similar meaning are mapped to a similar position in the vector space. For example, “model_checking” and “formal_verification” are close to each other.

In GraphDBLP, as the first step, each title is pre-processed according to the following pipeline: (i) html tag removal, (ii) html entities and symbol replacement, (iii) tokenization (punctuation is removed and space used to tokenize), (iv) lower case reduction, and (v) stop words removal. Then, each pre-processed title is added to the list of words \( W \). Then, a \( N=500 \) vector representation of preprocessed titles is generated, and the similarities between each GraphDBLP keyword is stored within a Hash-table that assigns a list of similar n-gram to each GraphDBLP keyword.

**Building the \text{HAS\_RESEARCH\_KEYWORD} relation.** Here we describe how the research topics were computed for each author. To this end, we closely look at the authors (\( A \)), publications (\( P \)) and keywords (\( K \)) nodes of our model depicted in Fig. 1. This subgraph can be seen as a tripartite graph model (author-publication-keyword), as shown in Fig. 2a, where author \( a_1 \) has authored a publication \( p_1 \) that, in turn, contains a keyword \( k_1 \), and so on. Given an author \( a \in A \), we can compute three distinct elements.

- **Research Keywords** as the list of keywords used by an author in publications. In the example above, the \( a_2 \) research keywords list is \([k_1, k_2, k_3, k_4]\);

- **Research Keywords Score** gives an estimate of the weight of each keyword in the research publications of a given author. In the example above, focusing on author \( a_2 \), keywords \( k_2 \) and \( k_4 \) have a score of \( 1/3 \) each, while keywords \( k_1 \) and \( k_3 \) have a score of \( 1/6 \) each. Notice that the numerator of this value could be computed as the number of all distinct shortest paths starting from an author \( a_i \) and reaching the keyword \( k_j \).

- **Research Keywords Relevance** gives an estimate of the prolificness of the author \( a_i \) on the research community working on a given keyword. For a given author, keyword pair, this value is computed as the ratio between the number of publications authored by \( a_i \) having the keyword \( k_j \), and the total number of publications containing \( k_j \).

Fig. 2b shows the author’s relevance computed for each \text{HAS\_RESEARCH\_TOPIC} relationship. In this way, we can compute the \text{HAS\_RESEARCH\_TOPIC} relation and both the score and relevance for each author,keyword pair. Notice that these two attributes give two distinct views on the research publication records of an author. The score gives a rough estimate of how a given keyword ranks in the publication records of the author, whereas the relevance attribute estimates the extent to which the author contributed to keyword \( k \), by taking into account all the publications concerning that keyword.

### 3 GraphDBLP Usage and Examples

GraphDBLP can be downloaded freely from GitHub. It provides three built-in queries, though any graph-based queries can be performed by using the Cypher Query Language. Here, we introduce how these queries works and what is the rationale behind.

**Q1: Keyword Discovery.** The command \(-q1 \) keyword limit performs the keyword discovery query. This allows users to identify the most prolific authors in the DBLP community for a given keyword. This requires to specify also the keyword to be used
Fig. 2: (a) Working example of a tripartite graph with Authors (A), Publications (P) and Keywords (K) derived from the data model of Fig. 1, (b) HAS_RESEARCH_TOPIC relations computed on the graph of Fig. 2a. The values for the score/relevance properties are shown on edges separated by commas.

and the limit value for results. Example: -q1 ‘multimedia’ 10 will perform query 1 using multimedia as keyword and collecting top 10 results.

Fig. 3: Q1: Keyword Discovery

Q2: Author Publication Records Comparison. The command -q2 author-name-surname limit similarity-threshold runs the query. It begins from the keywords describing an author’s research activities i.e., the keywords connected through the has_research_topic relationship. For each keyword, the most proficient author in the field is identified, and the related data are retrieved: (prolific) author name, score, relevance, and related keywords. This requires to specify also the keyword to be used, the max number of researchers to be considered for each keyword and the similarity threshold value for similar keywords. Example: -q2 ‘John von Neumann’ 3 0.4 will perform query #2 profiling the publication record of John von Neumann and retrieving up to 3 top researchers for each keyword appearing the in profile of John von Neumann. Only keywords with a similarity value greater than 0.4 will be returned.

Q3: Local Clustering Coefficient. The command -q3 venue-name similarity-threshold enables query #3 for computing local clustering coefficient
on research communities [15]. This requires to specify the venue name and a threshold value for computing the similarity. Example: `q3 'sebd' 10 percent` will perform query 3 computing the community starting from sebd and considering venue with a similarity value with at least 10 percent. More specifically, here we are computing the weighted Local Clustering Coefficient (wLCC) \(^8\). Indeed, if on the one side Local Clustering Coefficient [19] computes the degree to which nodes in a graph tend to cluster together (aka transitivity coefficient), the weighted LCC explicitly takes into account the different degree of similarity between two nodes i.e., the triples are computed using all similarity relations on the basis of their degree. This allows a smaller neighbourhood of the node to be considered, which results in a tighter cluster of similar venues. Due to the space restrictions, examples of some of these queries have been omitted. The reader can found them at [11].

## 4 Conclusion and Future Work

We have presented GraphDBLP, a tool that has been recently released as open source tool to perform graph-based queries on the DBLP community. GraphDBLP was implemented on top of a Neo4j graph database providing a shell interface to interact with the graph-db. To date, GraphDBLP is the first attempt to organise and reason with the DBLP data as a graph. The source code and datasets used has been made publicly available on Github with the idea that anyone might contribute to the project in a schema-free fashion, by adding node labels, new semantic similarities and metrics, and new features through graph-based queries as well. We are actually working for including the Arnet Miner citation within GraphDBLP. Indeed, to date, DBLP does not provide any information about citation relationships between papers, and this prevents the use of DBLP for performing influence analyses, such as community influence.

---

\(^8\) it is inspired by [1] though they compute the weight of triples through arithmetic functions
References