DIGDUG: Scalable Separable Dense Graph Pruning and Join Operations in MapReduce

Manu Shukla, *Member, IEEE,* Dinesh Dharme, *Member, IEEE,* Pallavi Ramnarain, *Member, IEEE,* Ray Dos Santos, *Member, IEEE,* and Chang-Tien Lu, *Life Fellow, IEEE*

Abstract—Linking topics to specific experts in technical documents and finding connections between experts are crucial for detecting the evolution of emerging topics and the relationships between their influencers in state-of-the-art research. Current techniques that make such connections are limited to similarity measures. Methods based on weights such as TF-IDF and frequency to identify important topics and self joins between topics and experts are generally utilized to identify connections between experts. However, such approaches are inadequate for identifying emerging keywords and experts since the most useful terms in technical documents tend to be infrequent and concentrated in just a few documents. This makes connecting experts through joins on large dense graphs challenging. In this paper, we present DIGDUG, a framework that identifies emerging topics by applying graph operations to technical terms. The framework identifies connections between authors of patents and journal papers by performing joins on connected topics and topics associated with the authors at scale. The problem of scaling the graph operations for topics and experts is solved through *dense graph pruning* and graph joins categorized under their own scalable separable dense graph class. Experiments were performed on technical domains to validate the utility of the connections between interests and experts. Comparing our graph join and pruning technique against multiple graph and join methods in MapReduce revealed a significant improvement in performance using our approach.

Index Terms—Big Data, Distributed Graphs, MapReduce, Graph Joins

1 Introduction

Monitoring emerging topics [Joung and Kim, 2017] and connecting emerging topics to experts is crucial for determining trends in diverse domains such as insurance, investment and research [Furukawa et al., 2015]. This involves deep parsing of highly technical journals [Newman et al., 2014], patents [Zhang et al., 2015], and white papers to connect keywords and authors from them. Connecting experts through common and linked topics is of great value in scientific disciplines [Young et al., 2019]. Due to the large number of terms in each publication, tracking all terms and their connections rapidly devolves into a big data graph problem [Bordes and Gabrilovich, 2014]. Not only do the connections between key terms need to be maintained due to their co-occurrence in a document, the frequently connected terms must be eliminated in order to leave technical terms of high value that have direct or indirect connections through common terms. The resulting network of high-value technical and related keywords then represent emerging topics. These keywords, defined as *interests*, are then used to connect authors, or *users*, to each other.

Interest-to-interest as well as user-to-interest connections are modeled as graphs; Figure 1 shows a simple example. The left side graph depicts a typical interest-to-interest graph with a large number of nodes (only a few are shown) and edges, whose frequent nodes have been pruned. The right hand side shows the corresponding user-to-interest graph, which can also have a large number of interests for each user within a large corpus of technical documents. In the example shown, the term 'amylin' was pruned from the graph as it had a large number of edges to other terms with high connection strength that exceeded a set threshold, while the term 'sulfonylurea' was kept as it had fewer links with low weights to other infrequent terms. Terms that are connected to pruned terms without connections to each other, are connected with *derived* edges. The two graphs must be joined to find the highly connected users for key terms. The number of combinations of users that must be evaluated is often extremely high. In this case, the term 'sulfonylurea' and 'sorbitol'

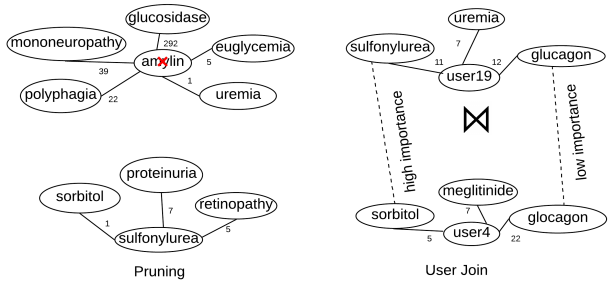


Fig. 1: DIGDUG example.

are considered more important in connecting two users due to their connection in the pruned interest graph than common interests between two users such as 'glocagon'. The scale of the data and connections contained in the graph and the number of operations that must be performed on them are key to solving this problem.

To represent the network of interests based on their co-occurrence in a document, a distributed graph approach has been applied here. Performing operations on the graphs introduces a classification based on the scalable graph class ($\mathcal{SGC}$) of problems [Qin et al., 2014]. However this class does not address how a self join should be performed for keys of a graph utilizing a second graph that contains relationships between value elements. Although $\mathcal{SGC}$ is efficient at handling iterative graph operations, this type of approach needs to be improved upon for iterative operations on dense graphs where in graph $G(V,E), \|E\| = O(\|V\|^2)$ and for joining across two graphs. $\mathcal{SGC}$ is not able to create user tuples for exploration to find similar user pairs based on their shared interests. Hence a new class of graph operations is needed to efficiently prune dense graphs and explore tuples of user pairs that are strongly connected based on shared interests.

Graph pruning is an efficient way to eliminate nodes with the

highest number of connections to other nodes as these represent frequent terms. Once the graph has been pruned and only key interests remain, the users and their associated interests can now be joined with other interests as well as with their connected interests graph. This join has to be performed at scale, where a user's interest has to be replaced with the other connected interests and aggregated before that user can be added to the join output. Since this approach uses key-value pairs as its main data structure, it can result in an extremely large number of keys and values being emitted, making the aggregation of values for a key untenable and the streaming of keys and its sub-values detrimental to performance. A new class of dense graph operations and joins must therefore be designed to enable these techniques to be run consistently on large numbers of terminology-heavy academic and technical documents. Several challenges need to be overcome to perform this task: 1) *Scaling the creation of graphs with a large number of edges and nodes.* With documents consisting primarily of infrequent technical terms, it is difficult to scale graph construction for large numbers of nodes and edges. Graphs that are built with users and terms based on their co-occurrence in a document can become very large. 2) *Similarity measures for large numbers of infrequent terms.* Similarity measures such as cosine similarity based on IDF or TF-IDF weights are not well-suited for this type of problem as large numbers of important terms are concentrated in just a few documents, which may not even occur frequently in these documents. Similarly, ranking documents against a set of query terms through an algorithm like PageRank does not show the true strength of the connection between the term and its related documents. The results merely identify documents that use the term more frequently. 3) *Graph operations on nodes with a large number of edges.* Existing graph methods in MapReduce do not take into account graph density [Karloff et al., 2010]. Where a very large number of interests are associated with other interests, this makes operations on the resulting dense interest-to-interest graphs extremely difficult to scale. Performing pruning operations in MapReduce at scale in these cases is not trivial, even by increasing cluster and node sizes. 4) *Scaling joins between graphs*. Joins in big data are well-studied, including similarity joins and self joins. However, joins in graphs to connect users by evaluating user pairs necessitates the exploration of cross product of users and their associated interests and greatly increases complexity of operations required in a big data environment.

To address these challenges, this paper proposes DIGDUG (Distributed Interest Graph Distributed User Graph), a framework that is specifically designed to build interest-to-interest and user-to-interest graphs at scale. The proposed system allows *horizontal scaling* to perform deep mining of increasing corpus sizes of journal articles and patents in a distributed paradigm by simply adding more nodes to the cluster [Hu et al., 2014a]. The graph of interests is iteratively pruned to eliminate highly-connected interests with the higest number of edges to other interests. All the edges that contain the pruned node can then be removed, after which the interests with the strongest connections to other infrequent interests are used to connect users with other users through the user-to-interests graph. The framework utilizes key-value pairs to build and update graphs with large numbers of technical terms. The resulting graphs connect users to interests and interests to other interests based on co-occurrence in the documents. The user-to-user mappings are generated by joining the user-to-interest graph with the associated pruned interests-to-interest graph. A join then connects users with the largest number of common and closely-connected infrequent interests. The key contributions of this paper are:

1) *Novel distributed framework to scale identification of key topics and experts:* Our proposed technique generates graphs where the nodes represent either users or interests. Links connect user to interests or interest to interests. A novel Scalable Separable Dense Graph Class ($\mathcal{SSDGC}$) is defined to perform operations on the graphs.
2) *Innovative dense graph operations to discover key technical topics as **interests***: Identifies terms that have strong connections to related terms. The operations remove widely connected interests and their connections to other interests in a dense interest graph and retains those that are infrequent and not connected to numerous other terms.
3) *Distributed graph joins on separable graphs to connect experts*: The join operations for connecting experts as **users** based on the strength of connections between interests associated with them are performed in a granular key-value pair paradigm. The joins do not in any operation aggregate all the interests for a pair of users, allowing horizontal scalability.
4) *Extensive experiments to validate the efficacy and performance of graph operations and joins:* Patents and academic journals in a scientific domain from several sources are used to demonstrate the high performance of our approach in finding connections between experts and critical topics. Analysts found topics and experts identified by DIGDUG were more relevant than those identified through document similarity.

The rest of the paper is organized as follows. Section 2 discusses existing work on user and interest graphs and joins in large scale data. Section 3 formulates the problem that the proposed system addresses. Section 4 provides details of the approach used in DIGDUG and the user and interest graph modeling and joins. Section 5 presents an overview of DIGDUG architecture and key-value pair based distribution algorithms for joins. Experiments and case studies are described in Section 6, and the overall conclusions are presented in Section 7.

2 Related Works

When it comes to finding experts, topics and connections between experts from social media and technical publications, three distinct domains are represented in the literature. Experts and emerging topics are often explored via social data. In some cases, graphs are built with experts and their topics and with terms extracted from social media. These graphs grow rapidly in both size and density so operations on them need scaling. This is most critical for joins across two graphs, as these tend to be the most computationally and memory intensive. Pruning is utilized to remove noise from graphs. For this work, we looked at the state of the art in identifying experts and emerging topics from social media and in scaling graph operations and joins.

Finding experts and evolving topics from social media: Identifying experts from social media within a large global organization has been studied [Guy et al., 2013] and a hypergraph learning approach has been applied to learn topic-based influencers from photo sharing sites [Fang et al., 2014]. A topic modeling based

approach to find interests of users in Mendeley has also been explored [Rossetti et al., 2017]. Social media mining has been a rich source of information on brands [Gundecha and Liu, 2012] and the use of social media for knowledge acquisition and validation is well known [Kondreddi et al., 2014]. Social media mining however focuses on identifying emerging topics and influencers and is not suited for finding connected researchers and topics from academic journals and patents that primarily contain domain specific terminology.

Linking new articles to generate evolving new stories is a popular approach [Tang et al., 2015]. The interactions of storylines in news has been explored [Hu et al., 2014b], as has building storylines of text, pictorial and structured data [Dingding Wang, 2014]. Storylines have been used to determine evolving events [Santos et al., 2016] and to exhaustively connect the dots in social media with MapReduce [Shukla et al., 2017b]. They have also been utilized to track brand perceptions [Shukla et al., 2016b] that could then be mapped back to postings that cause perception swings utilizing in-memory distribution techniques [Shukla et al., 2017c]. Storylines have been used to categorize events in terms of their theme, location and time [Shukla et al., 2016a] making it possible to identify key prospects as people and organizations [Shukla et al., 2017a]. However, none of these techniques can identify the surrounding topics and related users for a given topic when dealing with large sources of data. Unlike DIGDUG, these existing techniques do not combine users and interests from multiple data elements and multiple sources to create a set of connected users and interests.

Distribution of Graph Operations: A model for efficient algorithms in MapReduce paradigm has been proposed [Karloff et al., 2010], along with algorithms for massive, unordered, distributed computations that are equivalent in power to symmetric streaming algorithms [Feldman et al., 2010]. Graph operations in MapReduce that apply filtering to reduce the size of input in a distributed fashion to process on a single node have also been proposed [Lattanzi et al., 2011]. A scalable graph processing class and two types of join operators in MapReduce have been suggested [Qin et al., 2014] and a relational-graph data model with query based graph operations described [Gao et al., 2014]. MapReduce based algorithms have been applied for managing big Resource Description Framework (RDF) graphs [Cuzzocrea et al., 2017] and MapReduce has been used to enumerate maximal bipartite cliques [Mukherjee and Tirthapura, 2017] in large graphs. Finding connected components in large graphs in MapReduce has also been explored [Kiveris et al., 2014].

Distributed Joins: Self joins in MapReduce have been the focus of a great deal of research. For example, a 3-way approach to end-to-end set-similarity joins [Vernica et al., 2010], an efficient MapReduce based graph similarity join algorithm utilizing filtering-verification framework and bloom filters [Chen et al., 2014], and a partitioning strategy able to overcome memory bottlenecks in similarity self-joins [Baraglia et al., 2010] have all been proposed. Other researchers have developed a simple randomized algorithm for arbitrary theta-joins in a single MapReduce job [Okcan and Riedewald, 2011], a self-join approach to deal with high-dimensional vector data [Fries et al., 2014], mapping multi-way theta joins to a sequence of MapReduce joins [Zhang et al., 2012], and similarity joins using different metric distance functions [Das Sarma et al., 2014].

Graph Pruning: Mining graph data is a complex and varied field [Cook and Holder, 2006]. Pruning has been used extensively in trawling web communities [Kumar et al., 1999]. It includes iterative pruning and inclusion-exclusion pruning. Pruning is employed in frequent pattern mining in graphs including FSG for frequent subgraph mining [Borgelt and Berthold, 2002] [Huan et al., 2004]. Pruning is also utilized in graph matching to winnow the search space [Moy, 2005] in social network analysis. None of the pruning methods use a combination of node degree and edge weights to perform the pruning in a distributed manner.

All of these graph techniques, some MapReduce based, either solve specific graph problems efficiently or propose general techniques for graph operations and joins. DIGDUG builds on previous research by incorporating graph pruning and joins with nodes and edges as key-value pairs, which is not only more efficient than general techniques, but the resulting interest and user connections are more meaningful than those achieved using other techniques.

3 PRELIMINARIES

This section presents the information needed to conduct an in-depth investigation of distributed solutions to the problem of finding emerging interests and connections between the users associated with them. Section 3.1 formulates the problem to be solved and Section 3.2 presents the distributed computation framework needed to solve the problem.

3.1 Problem Formulation

First find interests that are not too common in a document and are not found in a significant number of other documents in the corpus. Second find highly connected interests based on their common and connected interests.

Identify non-frequent interests based on their connections: The first desired outcome is to find the infrequent terms (i.e., interests) connected to other infrequent terms, which can be done in two ways: (1) through their co-occurrence in the same document, or (2) through their connection with other infrequent terms across documents. To accomplish this, we remove the interests that are most frequently connected to other interests across all documents, leaving a smaller set that can better differentiate between documents.

$$
\begin{aligned}
&\textit{Documents } D_i, D_j, D_k, \ldots \in \textit{ corpus } \mathcal{D} \\
&\forall\, D_i \in \mathcal{D} \\
&\quad \textit{extract Interests } \{I_m, I_n, I_o, I_p, I_q, I_r, \ldots\} \\
&\quad \forall \textit{ interest } I_m \in D_i \textit{ connect each interest } I_m \rightarrow \{I_n, I_o, I_p, I_q, I_r, \ldots\} \\
&\qquad \textit{into interest graph } I(V,E) \\
&\textit{while graph contains nodes with high connectivity} \\
&\quad \forall\, I_j \in I \\
&\qquad \textit{if connectivity}(I_j) \textit{ is high} \\
&\qquad\quad \textit{delete } I_j \textit{ from } V \textit{ and all edges with } I_j \textit{ from } E \\
&\quad \textit{remaining pruned graph } I_{pruned}(V,E)
\end{aligned}
$$

This identifies strongly connected non-frequent interests across documents.

Find connected users based on the interests' strength: The second desired outcome is to find the most highly connected users

based on shared interests. Having a graph of users connected to their interests and the interest graph left with infrequent interests:

Documents $D_i, D_j, D_k, \ldots \in$ *corpus* $\mathcal{D}$
Interests $\{I_m, I_n, I_o, , \ldots\}$ *and Users* $\{U_j, U_k, \ldots\}$ *in* D_i
$\forall$ D_i *in* $\mathcal{D}$
Connect users $U_j, U_k \in D_i$ *with interests* $\{I_m, I_n, I_o, \ldots\}$ *of* D_i
as user to interest graph $U(V,E)$
Create and explore user tuples (U_j, U_k) *and their interests*
Find user pairs $(U_1, U_2), \ldots$*with strongest connection strength*
based on common and infrequent connected interests
where connected interests $(I_1, I_2) \in E$ *of* $I_{pruned}(V,E)$

The output is the set of connected users. The users that have the highest connection strength based on common and connected terms can be identified in the output.

3.2 MapReduce Framework

The DIGDUG architecture is based on the MapReduce Framework [Dean and Ghemawat, 2008]. The MapReduce programming model allows users to develop scalable, fault tolerant applications by providing a key-value pair based paradigm. Applications written in the paradigm run in a parallelized way on a cluster in a shared nothing environment. Each MapReduce job consists of three phases: Map, Shuffle and Reduce. The input data is stored in a distributed file system. The three phases transform as follows:

1) Map: This stage reads the input as key-value pairs $<k_1, v_1>$ and transforms them into another set of key-value pairs $<k_2, v_2>$ that are handed to the shuffle on each node.
2) Shuffle: The key-value pairs $<k_2, v_2>$ are taken from the map phase and distributed across all machines. This stage guarantees sorting on keys and all values for a key combined together before handing them over to the Reduce stage.
3) Reduce: This phase receives each key and all its values grouped together as $<k_2, <v_2, v_2', v_2'', \ldots>>$ from all the mappers across all the machines and allows the user to emit another set of key-value pairs $<k_3, v_3>$ to be processed in the next job.

The user can implement application specific operations in the Map and Reduce stages.

4 DIGDUG User and Interest Graphs, Pruning and User Connections

This section presents the creation of user and interest graphs from patents and academic articles. Section 4.1 and Section 4.2 describe the new dense graph class of problems and interest graph pruning, respectively. Section 4.3 discusses the connected users calculations, along with the user-to-interest and pruned interest graph joins. Section 4.4 describes the steps in the DIGDUG flow.

4.1 Scalable Separable Dense Graph Class

The density and separability of the user-to-interest and interest-to-interest graphs requires the definition of an additional $\mathcal{SSDGC}$, or scalable separable dense graph class of problems. The $\mathcal{SGC}$ class possesses the properties of *Scalability*, *Stability* and *Robustness* which accelerates the process as more machines are added, allowing the process to finish in bounded rounds and complete irrespective of the amount of memory in any of the cluster nodes. We extend these properties by adding the following:

1) Separability Exploitation: Graph algorithms exploit the separability and differences in cardinality of the number of nodes by type.
2) Node Edge Operation Combinations: Node and edge operations are combined in the same step.

Node edge combinations are useful in pruning where the interest-to-interest nodes and edges are operated upon such that the keys can represent both the graph nodes and the metadata about them, as well as the edges that represent the connections between nodes. Each iteration combines the keys for nodes and edges to determine which node to prune and then emit or drop edges that contain a pruned node.

Join operations in the $\mathcal{SSDGC}$ class maintain the separation between the interest-to-interest pruned graph and the user-to-interest graph. It exploits this separability to calculate the connection strength between users based on the combined strength of their common and connected interests. For user-to-interest and pruned interest-to-interest graph joins, in order to compute user connections, a very large number of user pairs and their interests must be explored. $\mathcal{SGC}$-based techniques with the $\mathcal{EN}$ *and* $\mathcal{NE}$ joins and their combination for finding user connections are less efficient than exploiting the separability of the graph. This exploitation of differences in cardinality of join nodes by node type as in $\mathcal{SSDGC}$ enhances the efficiency of joins and can be useful in additional graph operations. This necessitates a new class of graph MapReduce operations.

4.1.1 Indirect join operator in $\mathcal{SSDGC}$

This operator allows nodes to be joined that do not have an edge between them. An indirect join accepts as input a set of nodes of type u and i in graph G(V,E) where no nodes of type u have edges between them and $((u_i, i_j), (u_k, i_l), \ldots) \in$ E, outputting nodes as pairs (u_i, u_k) such that $f(i_j, i_l, \ldots)$ satisfies function h.

Indirect Join in MapReduce: This is done by collecting the nodes to be connected in the value of the key-value pair emitted in the mapper, and streaming through all the nodes in the reducer to be connected and emitting each of the pairs separately. Hence for graph G(V,E) where there is need to perform indirect joins between $u \in V$ for two types of nodes $\{u, i\} \in$ V, in order to create pairs of users u_1, u_2 we must emit i_1, u_1 and i_1, u_2 from the mapper. In the reducer, the values $<u_1, u_2, \ldots>$ are then processed in a streaming manner and pairs created. The final emission in reducer takes the format $<u_1; u_2, i_1>$.

4.1.2 Mixed join operator in $\mathcal{SSDGC}$

This operator enables nodes to be joined through a combination of node and edge operations in a key-value paradigm. In graph G(V,E) where $i \in V$ and $(i_1, i_2) \in E$, the operator allows operations to be propagated along the edges to be consolidated in a node and then emits a mix of node and edge propagated operations into a single set of nodes in same operation. This means that $i_1, (i_2, f(i_1, i_2))$ and $i_1, h(i_1)$ are emitted in the same operation that calculates h(i_1), where $f(i_1, i_2)$ is a function that represents the edge properties between i_1 and i_2 and $h(i_1)$ is a function that represents the properties of node i_1.

Mixed Join in MapReduce: In graph G(V,E) with nodes $i \in$ V and edges $(i_1, v_2) \in$ E, we perform operations such that both types $<i_1, h(i_1)>$ and $<i_1 : i_2, f(i_1, i_2)>$, representing node and edge tuples, are emitted from the mapper and reducer as key-value pairs. The format of keys and values allows the two sets of keys and values to be processed differently. This allows both edge and node propagations to happen in the same MapReduce.

4.2 Graph Pruning

Pruning the interest-to-interest graph is an iterative process that removes nodes that are highly connected to large numbers of other nodes with high connection strength. The graph pruning operation algebra can be defined as follows, with G(V,E) representing the interest-to-interest graph that is being pruned:

while *change in average_edge_weight and*
std_dev_edge_weight $> \theta$
$V_i \leftarrow \prod pruning\_status(V) = pruned$
$V \leftarrow V - V_i$
$E \leftarrow E - E_i$ *where* $E_i \in \{v_1, v_2$ *where* v_1 *or* $v_2 \in V_i\}$
$E \leftarrow E + E_j$ *where* $E_j \in \{v_3, v_4$ *where* (v_1, v_3) *or* $(v_2, v_4) \in E_i\}$
recalculate average_edge_weight and std_dev_edge_weight

The resulting pruned graph is then used to find connected users in the next step. Restoration of links between nodes connected through pruned node assures links between terms that are connected through common terms across documents are kept. V_i is the set of nodes that have *pruning_condition* evaluated as *true*. Here *pruning_condition* is true if:

$$degree > average\_degree\ of\ graph\ nodes\ OR \quad (1)$$
$$average\_edge\_weight > average\_edge\_weight(graph)\ OR \quad (2)$$
$$std\_dev\_edge\_weight > std\_dev\_edge\_weight(graph) \quad (3)$$

The pruning operation removes nodes with high degree, high average connection strength to other nodes and high variability in their connection strengths to other nodes. The first criterion eliminates terms that have high number of connections to other terms which would occur for terms that are in large number of documents with a lot of terms in them. The second criterion will eliminate terms that either occur frequently in one or more documents or have connections to many frequently appearing terms in one or more documents. The third criterion prunes terms that have high variability in the connection strengths to other terms that will be above the average threshold due to some terms having high connection strengths and other terms having very low connection strengths. The variability is an indication that the term is connected to a lot of terms that occur frequently in documents and also to some terms that occur very infrequently in documents which in combination with high mean of their edge weights and high degree is an indicator of their lesser relevance in connecting users in subsequent steps. The graph changes after each iteration with some nodes and edges removed and derived edges added which changes the degree and edge weights of remaining nodes and edges. Hence the *average_edge_weight* and *std_dev_edge_weight* of the graph needs to be recalculated after each iteration. The iterations stop when changes in *average_edge_weight* and *std_dev_edge_weight* are lower than threshold θ. In $\mathcal{SSDGC}$ pruning is done using a combination of a mixed join and an $\mathcal{EN}$ join, which accumulates information along the edges into the nodes. It works by iterating over the graph without accumulating the values for the key as nodes or edges of graphs. The aggregate characteristics of a node are kept separate from the edge entities during each iteration. Based on those characteristics, the edges are either kept or deleted and derived edges through the pruned nodes for terms across documents are created.

The MapReduce implementation of pruning operates solely on interest nodes and interest-to-interest edges in the graph G(V,E). Here, edge and graph nodes' transmissions are combined to determine whether an interest node's connectivity to other interests is above the average connectivity of all graph nodes, which indicates that the node must be pruned. The primary sort in keys in MapReduce is used to first determine if a node has been pruned and if so, all the subsequent edges with that node in it are also pruned in the same MapReduce job, which is why pruning is $\mathcal{SSDGC}$ operation.

4.3 User Connections

Users are connected from user-to-interest graph based on common interests and connected interests. Connected interests are defined in the pruned interest graph. This operation is performed as a combination of indirect and mixed joins in $\mathcal{SSDGC}$. The algebra for operation performed to connect users can be defined as follows, where G(V,E) is the user-to-interest graph and G'(V',E') is the pruned interest-to-interest graph:

$V \ni \{user_i, user_k, interest_m, interest_n, ...\}$ *and*
$E \ni \{(user_i, interest_m), (user_j, interest_n), ...\}$
$V' \ni \{interest_i, interest_j, ...\}$ *and*
$E' \ni \{(interest_i, interest_k), (interest_i, interest_l), ...\}$
$\|interest\| >> \|user\|$
$G \bowtie G' \rightarrow G''(V'', E'')$*where*
$V'' \leftarrow \prod user_i : user_j$ *where* $\|interests_{user_i} \cap interests_{user_j}\| \cup$
$\|(interest_k, interest_l) \in E'\| > \delta$
where $(user_i, interest_k)$ *and* $(user_j, interest_l) \in E$

The join operation assumes that the node type being joined has lower cardinality than the node type used to joined them. That difference in cardinality is used to build the sets of user pairs for the common and connected interests; the combined strength is used by DIGDUG to determine if the joined user pair is relevant or not. Relevance is determined by the combined strength of common and connected interests, referred to as *DIGDUG similarity measure* being higher than threshold δ. The final similarity score between users is simply a weighted sum of the strengths of their common and connected interests. The MapReduce jobs create user pairs with shared and common interests by collecting them based on the interest keys in values and building the pairs by combining keys as nodes representing interests and edges as interest pairs, making the operation a part of $\mathcal{SSDGC}$.

4.4 DIGDUG Flow

DIGDUG constructs User and Interest graphs, prunes interest-to-interest graph to find infrequently connected interests, and identifies users who are connected to other users through their highly connected infrequent interests. The first pass through a set of documents involves two steps: first find the connections between interests based on their co-occurrence in a document, and then find associations among users and interests as shown in

Figure 2. Subsequent operations are performed on the interest-to-interest graph and user-to-interest graphs. The interest-to-interest graph pruning operation requires iteration to ensure that all the nodes that have above a certain number of connections to other nodes are pruned.

The join between the user-to-interest graph and the pruned interest-to-interest graph connects users on the basis of shared or related interests. The join exploits asymmetry in the cardinality of node types in two graphs. The user-to-interest graph has far fewer user nodes than interest nodes, making it possible to perform the join by spreading the nodes evenly among cluster machines through a reverse value join as an indirect join. This type of join does not aggregate any values for a key, and utilizes secondary sorts and aggregations to scale performance to the number of machines in the cluster.

5 DIGDUG System

This section presents the architecture of the DIGDUG framework and provides details of the distributed algorithms. Section 5.1 describes the architecture of the system and Section 5.2 gives the algorithms used.

5.1 Architecture

The DIGDUG framework is composed of a sequence of MapReduce jobs that run on AWS (Amazon Web Services) EC2 (Elastic Compute Cloud) clusters utilizing S3 distributed file store. The choice to preform the operations in MapReduce rather than Spark [Zaharia et al., 2012] was due to the regularly scheduled batch nature of the jobs and the need to repeat the runs consistently on commodity low cost machine clusters. MapReduce was also preferred over other similar distributed systems such as Apache Flink and Apache Storm as these are better suited for streaming data and over MPI(Message Passing Interface) with CUDA on HPC(High Performance Computing) grids due to the easier programming interface and more efficient distribution paradigm. The system is designed to easily scale to large amounts of data from multiple sources that must be combined and cross referenced for people and topics such that the topics can be connected to each other and users can be connected to other users through large graph joins. The modules are divided into 3 groups:

User and Interest Graph Construction The first module in the architecture flow reads raw data and generates a list of authors as users and their interests from each document. It then associates interests to other interests in the document and combines these across documents. It also associates users to interests within individual documents and aggregates users' interests across documents. These serve as the starting interest-to-interest graph and user-to-interest graph for the corpus. A Natural Language Processing (NLP) Parts of Speech tagger [Baldwin and Dayanidhi, 2014] is utilized to extract meaningful entities as interests and remove stopwords. An interest must also appear in at least two documents for it to be kept in user-to-interest and interest-to-interest graphs.

Pruning interest graphs The second module prunes the interest graph by removing interests with a large number of edges to other interests, high weights on edges and high variability in edges weights and removing the edges to the pruned nodes from all other connected nodes. Derived edges are optionally created between nodes connected to the pruned nodes if they are from different documents with no common documents. The weights of the connections is a fraction of the combined weight of the edges of the two nodes to the pruned node.

User connections by joining the user-to-interest and interest graphs The third module joins user and interest graphs. Starting with the user-to-interest graph, it connects users that have shared interests. In each connected interest pair, one interest is associated with one user and the other with the other user based on common interests between users and connected interests from pruned graph. The result is a series of user pairs with sets of common and connected interests where the strength of the connection between users is based on the number and strength of their connected and common interests.

5.2 Distributed Algorithms

The algorithms used in DIGDUG are described in this section. Their purpose is to build user and interest graphs, prune interest graphs and find user connections. The MapReduce architecture is shown in Figure 3. A detailed key-value pair based description of pruning and join algorithms is provided in Appendix.

5.2.1 Build graphs

The tasks involved in generating the graphs prevent the construction of a single set of combined interests for a user because including all their interests (and all the interests connected to other interests) could easily make objects too large even for a node and its edges. Each user node is connected to all the interests not only in that document but across all the documents in the corpus. Each interest node is therefore connected to every other interest node on the document, but only directional links are made i.e. the terms that appear earlier in the document have outgoing links to subsequent terms. Links in the reverse direction are not made.

User graph is built in MapReduce by first creating $user_1$: $interest_1$ keys and their connection strength as value for a document and then combining across documents. For interest graph the keys $interest_1$: $interest_2$ and values connection strength are used to aggregate interest pairs and their connection strength across documents. The connection strength between user and interest represents the number of times that interest appears in the document where user is the author or inventor. Connection strength between interests is a product of the number of times the two terms appear in the document, so for $document_1$, the connection strength between $interest_1$ that appears a times in the document and $interest_2$ that appears b times is $a*b$. These first 2 MapReduce jobs for building the graphs are considered simple and omitted from detailed description.

5.2.2 Prune interests graph

This section describes pruning the interest graph to remove the high edge weight nodes and their associated connections. The pruning is applied to the entire set of nodes in a single pass, calculating the number of edges. After each iteration, the count is updated and the nodes with the highest number of connections, highest weights on edges and highest variability on edge weights are removed. The connections between nodes that are terms from different documents are created as derived connection after their edges to pruned terms are deleted. This process is repeated until the average degree of nodes, average weight on edges and variability in edge weights converges or does not reduce more than a threshold. The process of counting the edge connections for each node is performed with a mixed join that combines

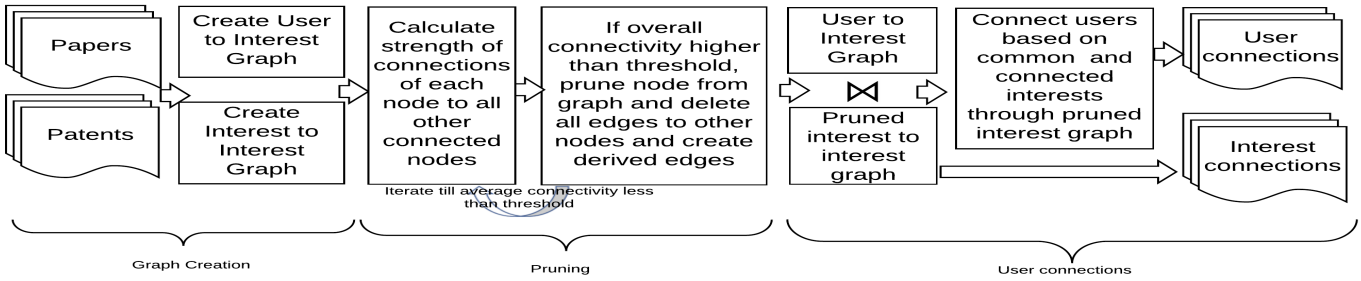


Fig. 2: DIGDUG design and flow.

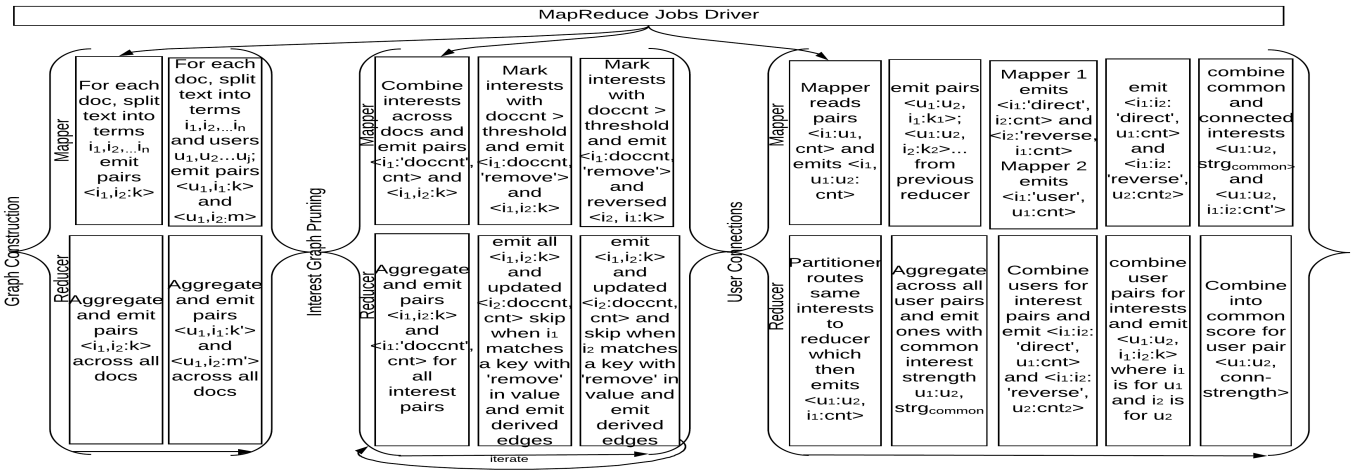


Fig. 3: DIGDUG user and interest graphs, interest pruning and user connections.

counting the number of connections for each interest node with emitting all the edges in the same MapReduce operation. The nodes whose connectivity exceeds average connectivity of graph are marked for removal and their edges are simply not emitted in the reduce step. The connection counts for the interests are then updated accordingly. The MapReduce steps for pruning are shown in Algorithm 1.

DIGDUG performs the pruning operations by combining the nodes and edges modeled in the keys and using streaming with partitioners to prevent the aggregation of values for a key in a reducer. It starts by modeling the interest-to-interest graph with the interest and document-id in the key, along with the other connected interests it has edges to, in the value for that document. It then generates the total weight of all edges for the interest using streaming with a partitioner in the reducer and also emits the node keys with edges to other interests. In the next MapReduce step, the nodes whose weights exceeds the threshold in Mapper are removed. It then goes on to eliminate all edges with the same starting node in the key in the reducer by not emitting them while saving the top k of them that are then connected with each other and emitted as derived edges if there are no common documents between them. The documents with an interest pair are stored as bitmaps to limit size of value of the key. It then reverses the edge in the next MapReduce job and removes the edges that have the end node of the edge as the pruned interest. Next, the average connection strength of all the nodes are recalculated and iterated until no node falls above threshold. MapReduce 4 and MapReduce 5 represent the mixed join operation. The use of streaming with partitioners prevents the aggregation of values for a key in order to determine removal of a node or an edge with the interest as the node during each iteration step as in $\mathcal{SGC}$, which significantly improves the performance.

5.2.3 Connect users

Connections between users are calculated by joining the user-to-interest graph with the pruned interest-to-interest graph. Algorithm 2 describes the join between users in a user-to-interest graph and the associated interest-to-interest mappings. This join emits all users that are connected through shared or strongly connected interests, with the interest strength above a previously determined criterion used to keep user connections that would otherwise be discarded. The join for the user-to-interest and interest-to-interest graph post pruning is performed by using interest as the key in MapReduce. However, the join key is the user. This reverse emission allows the aggregation of users and their grouping as pairs of users using the key sort in shuffle rather than aggregating them together as a value in the reducer. MapReduce 8 and MapReduce 9 utilize an indirect join operation to generate user pairs that are then tested for connection strength. This allows the operation to complete with a large number of users and an extremely high number of interests associated with those users without ever needing to aggregate them in memory.

The inversion of user pairs from value to key allows more voluminous interests to be included without the need to aggregate in the reducer, which could cause memory errors. The interest keys are suffixed so as to allow the sort order of the the keys to indicate when the interest as key switches from the user pair to some other connected interest as the value. Partitioners allow the same interest

Algorithm 1 Interest Graph Pruning

while σ >threshold **do**
  {Loop on the drop in average degree, average edge weight and average standard deviation in edge weight for all interests in graph σ from previous iteration not falling below a threshold}
  MapReduce3
    Read interest pairs, their strength and docs in mapper
    Stream through counts for $interest_i$ across all connected interests in reducer
    Update mean and std dev of interest strengths
  MapReduce4
    Read interest pairs, their connection strength and shared docs in mapper
    Read interest total edge count, mean edge weight, std-dev edge weight
  if mean_edge_weight >graph_mean_edge_weight OR std-dev_edge_weights >graph_std-dev_edge_weights **then**
      mark interest for removal
  end if
    Route all interests to same reducer in partitioner
    Drop interests marked as removed in reducer and keep top k connected interests in reducer
    Output connected interest with its connected interest and strength in reducer
    Output indirectly connected interests as derived interests
  MapReduce5
    Read interest pair and strengths in mapper
    Reverse the interests so as to remove the edges with pruned interest in value of previous job
    Read interest total count, mean_strength and std-dev in mapper
  if degree >graph_avg_degree OR mean_edge_strength >graph_average_mean_edge_strength OR edge_strength_std-dev >graph_average_edge_strength_std-dev **then**
      Output interest with suffix 'remove'
  else
      Output interest, count,mean_strength,std-dev
  end if
    Route all interests to same reducer in partitioner
  if <value>has 'remove' and key continuing in reducer **then**
      create list of top k derived connections nodes
  else
      Output interest pair, strength and document list
      Emit interest with its connected interest and strength
      Emit derived interest pairs if lists not null
      Output derived interest pairs
      null lists
  end if
  Combine all counts for all interests and count remaining interest and calculate average connections per interest
  Calculate difference between previous average connectivity for graph and current average as σ
end while

Algorithm 2 Connected Users with User-to-Interest and Interest-to-Interest Graph Joins

MapReduce7
  Read in all the users interest and their strengths in mapper
  Output interest:user combos with their connection strength
  Collect all users for the interest using streaming key values in reducer
    create user pair combos
    attribute interest strength to the user pair
  Output user pairs along with the interests and strengths in reducer
MapReduce8
  Read in and output all the users pairs common interest and their strengths in mapper
  Collect all user pairs interests and strength using streaming key values in reducer
  Output user pairs along with the combined strength of all the common interests
MapReduce9
  Read in and output all the user pairs and interests with counts in first mapper
  Read in all the connected interest pairs in pruned graph and their strengths in second mapper
  Output interest and strengths with 'direct' and 'reverse' with counts
  All interests land in same reducer through partitioner
if interest suffix is 'direct' **then**
    add in direct list in reducer
else
  if interest suffix is 'reverse' **then**
      add inverse list in reducrer
  else
    if interest suffix is 'user' **then**
      for each $interest_i$ in 'direct' **do**
        Output $interest_k : interest_i :' direct', user_i$:score
      end for
      for each $interest_j$ in 'reverse' **do**
        Output $interest_j : interest_k :' reverse', user_i$:score from reducer
      end for
    end if
  end if
end if
MapReduce10
  Read and output $interest_k : interest_i$:'direct',$user_i$:score and $interest_j : interest_k$:'reverse',$user_i$:score in mapper
  Send same interest pairs to same reducer in partitioner
  Combine reverse and direct interest pairs users in reducer
  {direct has user for first interest in value reverse has user for second interest in pair in value}
  Output user pairs along with the interest pairs and strengths
MapReduce11
  Read in and output all the user pairs and common interests strength in first mapper
  Read in all the connected interest pairs with the user pairs in second mapper
  Combine and output connected interests and common interests and their strengths in reducer

with different suffixes in the keys to be routed to the same reducer. The join operation generates the user pairs in a fully distributed manner without having to aggregate the interests for a user in the value. It first emits the user as the key, with the interests and the connection strength as the value. It then collects the different users that share an interest by routing them to the same reducer via the partitioner and generates pairs for all users with edges to that interest. The join on connected users is performed by emitting a user to interest edge with the interest as the key and the user as the value. The pruned interest-to-interest graph is emitted with the interest as the key but emitting each edge with both nodes as keys one at a time and then applying the suffix 'direct' and 'reverse' to the key. These suffixes allow the keys to be sorted which then allows the interests and users for a given key to be aggregated and used to build user pairs. The pairs associate the first interest with the first user and the second with the second user, where the user pair is on the edge of the pruned interest graph. The common and connected interests are then combined to generate the final join results.

6 Experiments

This section presents a series of experiments that were performed to test the effectiveness and scalability of the proposed DIGDUG user and interest discovery framework. They were implemented in Apache Hadoop MapReduce in Java and run on AWS clusters. Section 6.1 provides details of the datasets used and the domains evaluated. Sections 6.2 and Section 6.3 detail the qualitative and quantitative performance, respectively, of the system in summarizing the content of a large number of documents and Section 6.4 describes the results and subsequent analysis for the various domains tested. Section 6.5 examines the use cases and the performance results.

6.1 Experiment Design

The experiments compared DIGDUG with comparable techniques for performing both pruning and user connection operations. The experiment design flow presented in Figure 4 shows how raw data, users and interests are extracted from initial set of documents. Users are then mapped to interests from individual documents and interests to users. The two mappings are then joined to generate the set of users connected to other users based on their shared interests and the links between their interests and other interests based on the most high frequency interests.

6.1.1 Test Datasets

Experiments were performed on three distinct datasets consisting of the full text of patents and papers from academic journals and conferences. These sources capture emerging interests in the

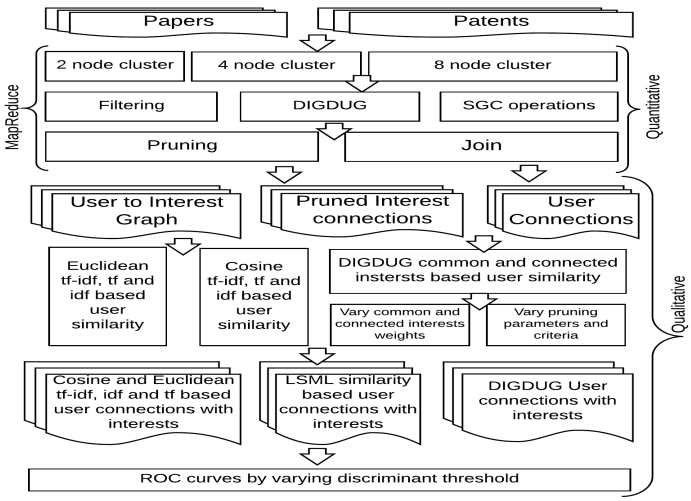


Fig. 4: DIGDUG experiment design.

scientific literature and seek to identify cutting edge, commercially viable findings of technical research. Over 2300 patents and journal papers were collected on topics related to diabetes. Our proposed techniques were applied to conduct a full extraction of all the keywords from the entire text of the papers and patents, along with the authors. The number of nodes in the user-to-interest graph was 243,519 and number of edges was 1,505,424. In the interest-to-interest graph, however, although the number of nodes was 241,596, the number of edges rose to 1,056,528,125 resulting in a dense graph.

6.1.2 Methods

For **pruning**, we compared the DIGDUG graph pruning approach with MapReduce filtering based pruning and $\mathcal{SGC}$ based pruning. Filtering-based pruning curtails the interests for users from patents and journal papers to the least frequent k interests. $\mathcal{SGC}$ based approach uses a combination of $\mathcal{NE}$ and $\mathcal{NE}$ joins to model pruning of most frequent nodes.

$\mathcal{SGC}$ pruning: The first technique utilized $\mathcal{SGC}$ $\mathcal{EN}$ and $\mathcal{NE}$ based MapReduce method to prune the interest-to-interest graph by transforming graph G(V,E) to G'(V',E'), where V' and E' are subsets of V and E, interests are nodes in V, and the interest to interest connections are edges in E. $\mathcal{SGC}$ based pruning combines $\mathcal{EN}$ and $\mathcal{NE}$ joins to create the interest-to-interest graph, $\mathcal{EN}$ joins emit nodes that capture the edge strength of each node and $\mathcal{NE}$ joins output or emit the edges during pruning. That involves incorporating all the connected interests in the value and, if above a certain threshold, inserting a 'removed' value in the emission from the mapper. In the reducer, if a 'removed' flag is found the entire $<key, value>$ emission for the interest is dropped and additional values are emitted for the derived edges. An $\mathcal{EN}$ join adds the derived edges in the set of edges for the nodes.

Filtering pruning: The second technique was a filtering based method on $\mathcal{MRC}$ [Lattanzi et al., 2011] in MapReduce. This technique applies filtering to distribute a key and its associated values on a particular node. Here, in order to perform a graph traversal an iteration was performed over all the keys representing an interest-to-interest graph node and all its connected nodes to determine how many other nodes it is connected to. After finding the strength of its connections and determining if the node needs pruning, the next MapReduce removes all the pruned nodes and from the value of every node to which it has an incoming or outgoing connection. Derived edges were made between nodes connected to pruned node but not connected to each other in following MapReduce.

The **user connection** operation in DIGDUG was compared against the task of finding user connections with $\mathcal{SGC}$ and Filtering based techniques as follows:

$\mathcal{SGC}$ user connections: The first technique utilized here was the $\mathcal{SGC}$ algorithm, which is used to perform graph based operations on large datasets. Here, the $\mathcal{EN}$ and $\mathcal{NE}$ join operations propagate information from nodes onto edges and edges onto nodes. They are modeled as keys consisting of nodes and edges of a graph G(V,E), where edges connect pruned interests with users and interests, and values are characteristics of the node. In this method keys in MapReduce are both users and interests. The values are related interests and edge characteristics such as connection strengths. The joins of user pairs occur by streaming users with the same prefix to the same reducer in MapReduce, where user pairs are created and then joined based on their shared and connected interests.

Filtering user connections: The second technique which is based on Filtering in MapReduce, belongs to $\mathcal{MRC}$. It performs graph based operation on large datasets where the join edges are modeled as keys and the characteristics of the edge are modeled in the values. The user pairs are calculated and modeled as nodes in Graph G(V,E), with edges between the user-to-interest and interest-to-interest connections, along with the characteristics of edges, treated as values. The user1:user2 node keys are suffixed by interest and routed to the same reducer with the interest1:interest2 edges. They are then combined to create user pairs connected by common and connected interests to perform connected user discovery operations.

6.2 Qualitative Effectiveness

The qualitative effectiveness of pruning and user connections was measured. To validate the **accuracy** of the user-to-user and interest-to-interest connections, different metrics were adopted: the True Positive Ratio (TPR) designates the percentage of connection designations that successfully matched the connections specified by an analyst as being relevant, while the False Positive Ratio (FPR) denotes the percentage of connection designations that were not actually relevant subsequent to pruning. An ROC curve was also utilized to evaluate the pruning and user connection performance as the discrimination threshold was varied. Multiple combinations of pruning parameters for interest graph were evaluated along with accuracy of pruning with and without derived edges and unpruned interest graph. The user connections from DIGDUG were compared with other techniques that can be used to connect users. We also evaluated the qualitative effectiveness of the join based on tuning of common and connected interest weights between two users and compared with equal importance of the two weights.

6.2.1 Pruning

The pruning criteria were evaluated by performing sensitivity analysis on the effectiveness of all combinations of standard deviation of edge weights of nodes, average edge weights of nodes and the degree of nodes in pruning interest graphs. The ROC curves for the combinations are shown in Figure 5. The results clearly show

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2020.2983650, IEEE Transactions on Big Data

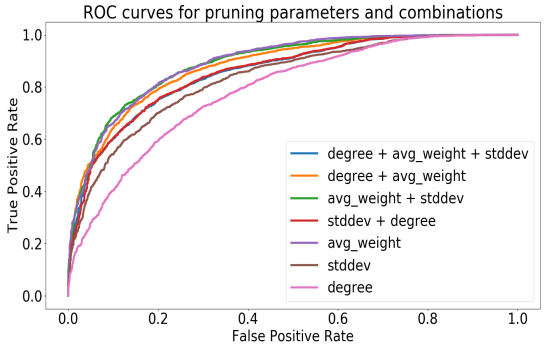


Fig. 5: Qualitative performance achieved by the different combinations of pruning parameters.

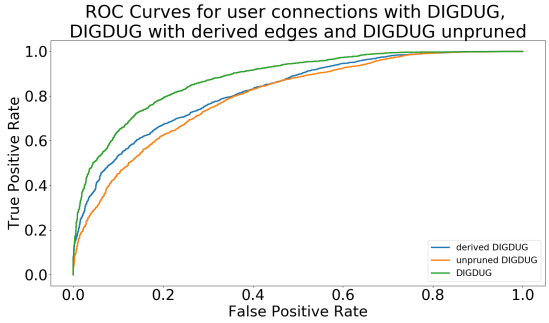


Fig. 6: Qualitative performance of DIGDUG pruned compared to DIGDUG pruned with derived edges and unpruned.

that with constraints on average edge weights, standard deviation of edge weights and degree the pruning becomes strong and the accuracy decreases. If only one of degree, average weight on edges or standard deviation of edge weights is used the pruning is weak and accuracy drops significantly. Degree of nodes by itself is a particularly poor predictor of accuracy. A better predictor is a combination of the average edge weights and standard deviation of edge weights. The combination is a good predictor of the interest not being useful in connecting users and hence a good candidate for pruning.

Pruning without connecting derived edges performed significantly better in predicting user connections compared to pruning with derived edges and DIGDUG without pruning as shown in Figure 6. The criteria average weight and standard deviation of weights for connecting users were utilized in pruning with or without derived edges. The creation of interests graph and using it to determine connected interests of users is still useful but pruning increases accuracy significantly. This illustrates the importance of pruning in removing frequently occurring terms and subsequently determining more relevant user connections.

6.2.2 User Connections

We evaluate the qualitative effectiveness of DIGDUG in generating user connections by comparing it with the Euclidean and Cosine distance based document similarity techniques. We also compared DIGDUG with a locally supervised metric learning (LSML) similarity measure [Ng et al., 2015] used to determine similarity in patients from medical records. An ROC curve was utilized to evaluate the user connection performance as its discrimination threshold for each connection was varied. The ROC

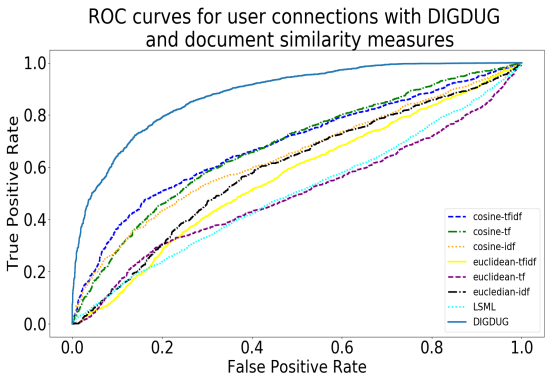


Fig. 7: Qualitative performance achieved by the DIGDUG joins compared to the other document similarity techniques tested.

curves for the usefulness of the results obtained by an analyst with DIGDUG against document similarity measures is shown in Figure 7. We compared DIGDUG's interest graph pruning based user connections and widely used document similarity measures such as Cosine Similarity and Euclidean distance based similarity by examining the terms for each user across all the patents and papers authored by the user. In order to asses the weight of each term, we calculated the term frequency, inverse document frequency and tf-idf. LSML uses a distance measure which relies on a learnt transformation matrix from training data to calculate distance between two patients feature vectors. The training data is generated by labeling users and their interests into classes that is used to calculate the covariance matrix used to learn the distance matrix [Wang et al., 2009]. DIGDUG performed better than any of the other measures in terms of finding more true positive connections and fewer false positives, as determined by the expert on diabetes research. DIGDUG also performed better against LSML that is comparable in complexity.

The user connection accuracy is impacted by the relative importance given to connected and common interests weights between two users. Figure 8 shows the sensitivity analysis with the tuned weights determined by linear curve fitting of the common and connected interest weights to achieve maximum area under ROC curve [Jin and Lu, 2009] compared to weighing them equally. The tuned weight of common interests was 0.0016 and connected interests 0.008 in the ROC curve in figure when derived edges are not created. When derived edges are created the connected interests tuned weight is lower than that of common interests. These tuned weights clearly demonstrate that even though in increasingly strongly connected user pairs both the common and connected interests weights increase, the importance of connected interests weight is significantly higher than common interests weight. Hence connected interests are better indicators of connection between users than common interests. Tuned weights in user connections were used for DIGDUG sensitivity analysis in all experiments.

6.2.3 Cosine and Euclidean distance based user similarity, LSML and DIGDUG similarity

Cosine similarity is often employed by search engines that link terms through TF-IDF. TF-IDF weighs terms higher that have high frequency in a document and low count in the number of documents in which they occur in the corpus. This means that Cosine Similarity between documents is skewed heavily towards

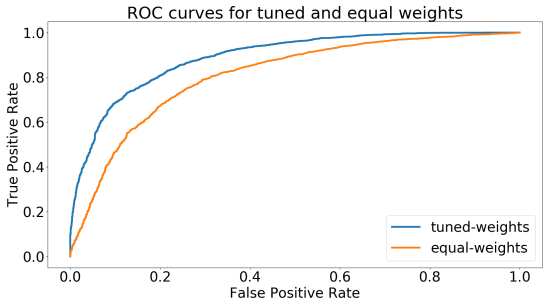


Fig. 8: Qualitative performance of different combinations of common and connected weights in connecting users.

terms that are frequent in a user's profile but not frequent in other user profiles. Unfortunately this does not help identify connections between users based on their similar or connected interests. The same problem occurs with term frequency and inverse document frequency weight based cosine and euclidean similarity measures.

The example in Table 1 illustrates how DIGDUG scores vary dramatically for two user pairs that have close cosine similarity scores. Two author pairs with similar Cosine scores but drastically different DIGDUG scores are Bruce Neal and Lawrence A. Leiter, who have a DIGDUG score of 672 and cosine similarity score of 0.6831 and Carl R. Illig and Zhihua Sui with DIGDUG score of 96879 and cosine similarity score of 0.6519. Examining the terms that connect them, the cosine similarity yields common term interests that are not unique, such as:

- Change
- Study
- Week
- Pfizer
- Inadequate controls

Table 1 shows how cosine similarity equally weighs connections that DIGDUG finds to be materially different in strength. The first column of DIGDUG direct interests are similar to the interests identified by cosine similarity in that they both consist of generic terms. However, where DIGDUG differentiates itself as a technique is in the second column of interests, the connected interests. This column enumerates the most useful, non-generic terms linking the two users. In the example of users with a high DIGDUG score, the connection is strong because of the connected interests and their relationship to the research topic of interest, namely diabetes. In the second row, where DIGDUG found a weaker connection but cosine similarity found an equal strength connection, the difference again lies with the connected interests. In this case, the connected interests are again more significant than the common interests. However, there are fewer interests, and therefore the connection is weaker. Upon expert interrogation, it was found that the overlap in topics of interest between the two users was indeed smaller than the two users in the first row.

At an initial glance, the connected interests in both rows of Table 1, although more specific than the common interests or cosine similarity terms, can still appear moderately generic. Validation based upon manual interrogation of the relationship between the two cited users in each row of Table 1 yielded a significant difference in the strength of the two sets of connections. The first two users with a high DIGDUG connection score, Carl R. Illig and Zhihua Sui, shared common places of employment, common co-inventors on patents, and common coauthors on papers. However, they have not jointly published or invented anything together. In the second example, with a low DIGDUG connection score, the two users (Bruce Neal and Lawrence A. Leiter) both share common research interests, namely cardiovascular disease and diabetes. However, they share no common places of employment, co-inventors on patents, or common coauthors on papers. As per the examples in Table 1, a higher DIGDUG score corresponds to a more meaningful connection between users in an area of study than a lower DIGDUG score. For the same pairs of users, DIGDUG selects more meaningful connections that describe the actual relationship between the research conducted by the two pairs. The terms shown in Table 1 are all largely related to the realm of diabetes research, and are specifically related to individual subsegments of research, in this case clinical studies examining the hemodynamics and metabolic consequences of cerebrovascular activity. Similarly LSML also performs poorly compared to DIGDUG, primarily due to its sensitivity to the training data labels used to generate the learning matrix.

6.3 Quantitative Effectiveness

The **computational** performance of the techniques used in creating user and interest maps and performing pruning and join operations on them at different levels of distribution is evaluated in this subsection. The results for running the techniques on various sized clusters and dataset sizes are also presented. The cluster experiments utilized Amazon EC2 instances of type m4.4xlarge with 16 vCPUs and 64GB RAM as the master and slaves nodes.

6.3.1 Pruning performance comparisons

One of the crucial elements of these experiments is how many interests were chosen from a document. Only DIGDUG was able to extract and process all the relevant terms from a document; the other comparative techniques all suffered from errors when working on such large graphs. In some of the competing techniques, pruning the number of edges that an interest or user node could be connected to is in one value in the key-value pair and hence is is limited by the memory of the node. Consequently, experiments were performed either with the maximum number of connected interests extracted for an interest before pruning or with *max_interest* progressively increasing from 200 to 500 to 1000. For the case of a higher number of maximum interests connected to an interest, some graph algorithms did not complete on smaller clusters and processes ran out of memory. The density of the graph and number of nodes in the graph increases dramatically as the maximum number of interests extracted from a document increases. When the maximum number of interests increase from 500 to 1000, the number of nodes increases from 96201 to 185055 for the entire set of 2300 documents and the number of edges rises from 88,793,762 to 186,776,725. During the experiments, the pruning iterations were stopped for average degree change, average edge weight change and average standard deviation of edge weight change for the graph at 10% threshold which happened after 7 iterations with the number of remaining nodes in the pruned graph at 7149. The distribution of these 3 attributes for the graph nodes also stabilized at that point. This strongly indicates that similar thresholds will work for any other technical corpus but were still made configurable for tuning to other corpuses. After pruning about 30% of the nodes dropped into the final pruned list due to the edges to other nodes pruned away causing the final user join results to be significantly better than other methods with connections from pruned interests.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2020.2983650, IEEE Transactions on Big Data

	Users	DIGDUG connection strength	Common interests (interest,weight)	Connected interests (interest1::interest2,weight)	Cosine Similarity	Cosine Similarity terms (term,weight)
1	Carl R. Illig :: Zhihua Sui	96921	solution,21; group,19; mixture,17; compound,17; disorder,15; water,13; treatment,12; product;12; disease,12; receptor,12; residue,12; methods,11; enantiomer,11; substituents,11; compositions,11; art,10; solvent,10; condition,10; temperature,10; solid,10; reaction,9; concentration,9; term,8; study animals,8; cells,8	patient conventional procedures::pharmaceutically acceptable excipient,24; sterile water::pharmaceutically acceptable diluent,24; free base compounds::composition injectable suspensions,12; carbodiimide esi electrospray ionization etoac::free base compounds,12; mesylation::target both liver,12;...triethanolamine::insulin resistance(derived),12; triphenylphosphine::powders (derived),10	0.6519	trifluoromethyl,0.1134; ethyl,0.0657; example,0.0431; dimethylphenyl,0.0385; difluo-rophenyl,0.0222; solution,0.0218; the procedure,0.0211; chlorophenyl,0.0198; methyl,0.0197...
2	Bruce Neal :: Lawrence A. Leiter	681	change,6; study,6; week,5; effects;5; treatment,4; analysis,4; sodium,4; european association,4; double;4; conduct,4; efficacy,4; sulphonylurea,2; other antihyperglycemic agents,2; fung;2; devineni d,2; consciousness,2;...	rapid hypertonic glucose infusion::rohwedder k,1; world diabetes congress::june 5'th annual meeting,1; plough::oslo university hospital,1; other baseline characteristics::stable thereafter,1; rapid hypertonic glucose infusion::urine output,1;...hypoglycemia::clinicaltrials (derived),8; rosiglitazone::data (derived),5...	0.6831	canagliflozin,0.4968; weeks,0.0224; patients,0.0214; safety,0.0097; canagliflozin mg,0.0071; baseline,0.0061; glimepiride,0.0051

TABLE 1: Examples of connected users and their interests identified by DIGDUG and Cosine Similarity with tf-idf weights.

Pruning for different cluster sizes: An example of the calculation of interest to interest mappings by removing highly connected nodes from interest graphs through pruning when a maximum of 200 interests is extracted from each document is shown in Figure 9a for pruning various sized interest graphs on a 2 data node cluster; Figures 9b and 9c show the same operation on 4 and 8 data node clusters, respectively. The results indicate that for small data sizes, $\mathcal{SGC}$ based pruning by performing all graph operations as a mix of $\mathcal{EN}$ and $\mathcal{NE}$ joins performs similarly or even slightly better than DIGDUG. The performance for interest graph pruning when a maximum of 500 interests are extracted from each document are shown in Figures 10a, 10b and 10c for pruning various sized interest graphs on 2, 4 and 8 data node clusters, respectively. The results show that as the size of graph increases, the performance of DIGDUG improves significantly compared to filtering or $\mathcal{SGC}$ based pruning. A similar set of pruning performances for graphs when 1000 maximum interests are extracted from each document is shown in Figures 11a, 11b and 11c.

Scaling issues: Some experiments on small clusters did not complete due to the large number of value elements associated with some of the keys, which led to memory errors. The results clearly show that not only do filtering based pruning approaches limit the accuracy of the results by limiting the number of connected interests to an interest being evaluated, but also may not finish at all due to the large number of value elements associated with an interest key in reducer. The only reasonable way to perform pruning with an $\mathcal{SGC}$ method is to use an $\mathcal{EN}$ join, which limits performance significantly. Given this limitation, DIGDUG is extremely capable compared to other techniques. The key reason DIGDUG performs better than either $\mathcal{SGC}$ operator based pruning or Filtering pruning is because it combines operations along the edges into nodes when counting the total connection strengths of the nodes at each pruning step. It conducts edge operations along the nodes by deleting all edges with either node a pruned node in a single MapReduce job using streaming, sub-key identification and the primary sort available within the MapReduce framework. This, along with avoiding the need to aggregate the edges for each key as an interest in the value, prevents memory errors for any key and allows horizontal scaling.

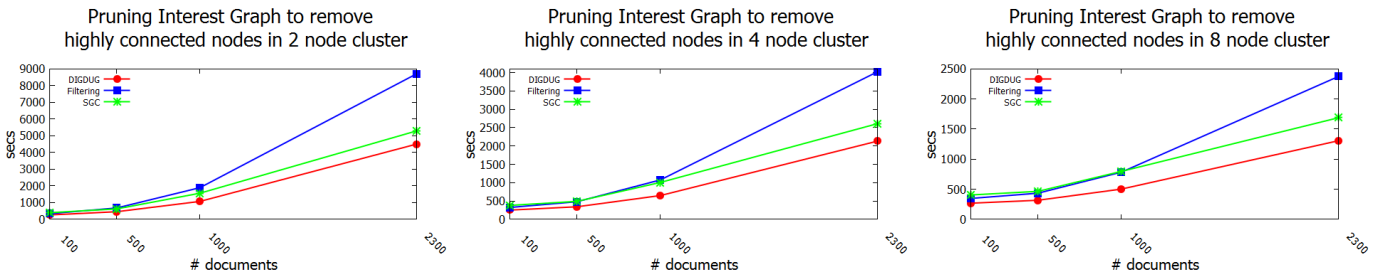
6.3.2 User connections performance comparisons

The calculation of user connections via user-to-interest and pruned interest-to-interest graph joins is extremely efficient in DIGDUG and scales well with an increasing number of interests and users; Figures 12a, 12b and 12c show the efficient nature of DIGDUG's join calculation on a 2, 4 and 8 slave node clusters, respectively, compared to those achieved by either of the $\mathcal{SGC}$ or Filtering techniques for a maximum of 200 interests. The $\mathcal{SGC}$ $\mathcal{EN}$ and $\mathcal{NE}$ joins based technique scales in the beginning but its inability to utilize the sort mechanism available in MapReduce and hence its need to use the smaller number of users related to an interest in the value degrades its performance in the user join step. The performance improvement achieved by DIGDUG becomes even more apparent on larger datasets, when the density of the graph increases due to the association of interests to users from a larger number of documents. This further highlights the performance difference between $\mathcal{SGC}$ and DIGDUG in $\mathcal{SSDGC}$.

Scaling on increasing interests extracted from each document: The filtering based approach fails for larger number of interests; even its user pair calculation is performed outside of the technique as it does not scale to all users in single or split keys in MapReduce. When max_interest increases to 500 and 1000, the interest-to-interest graph increases significantly in size and density even after pruning. Joining the user-to-interest graph with the pruned interest graph thus becomes significantly more computationally and memory intensive. The impact of this on join performance is shown in Figures 13a, 13c and 13c for a maximum of 500 interests. When max_interest rises to 1000, the impact on the join performance is shown in Figures 14a, 14b and 14c. The key reason the DIGDUG join operation performs better than $\mathcal{SGC}$ is the separability of the two graphs, which allows two pairs of edges and nodes to be processed at the same time. By combining the user pair creation propagated on the nodes along the edges with the attributes of the edge creation into a single step, the resulting combination of both connected and common interests boosts performance.

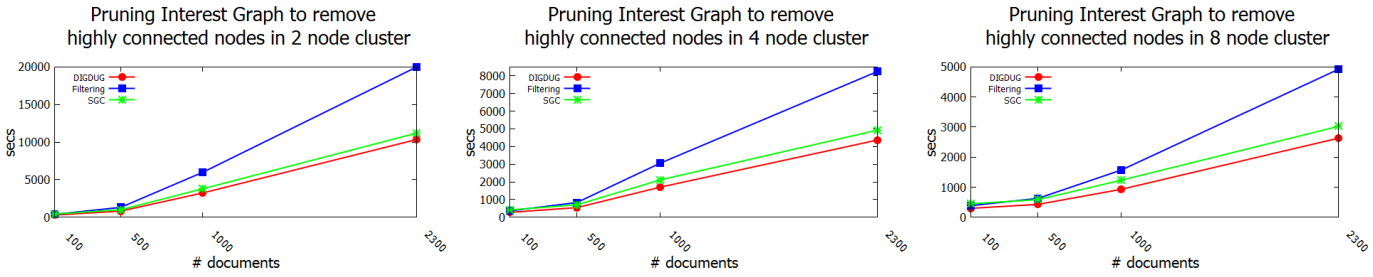
Order of joins between $\mathcal{SGC}$ and $\mathcal{SSDGC}$: MapReduce Graph Join techniques primarily function by joining attributes across

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2020.2983650, IEEE Transactions on Big Data



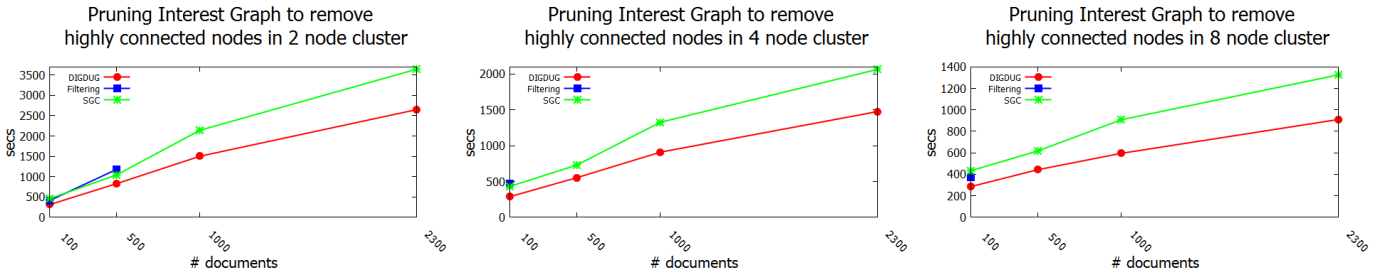
(a) 2 nodes (b) 4 nodes (c) 8 nodes

Fig. 9: Graphs for the pruning performance of DIGDUG, filtering pruning and $\mathcal{SGC}$ pruning for 2,4 and 8 data node clusters for max_interest 200 from a document.



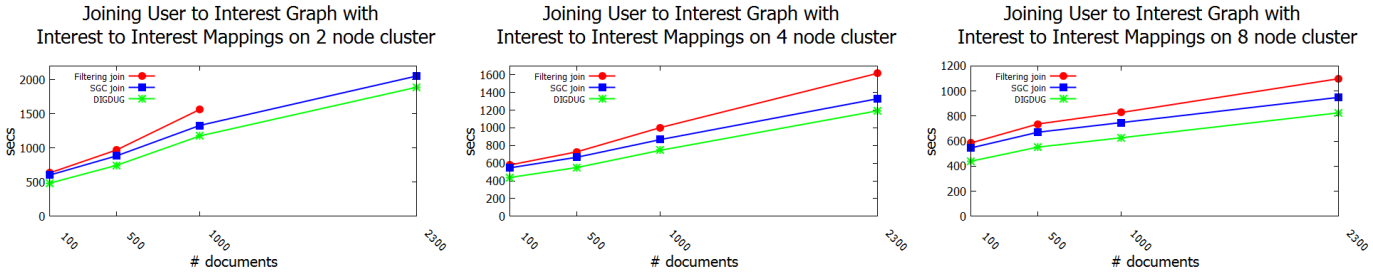
(a) 2 nodes (b) 4 nodes (c) 8 nodes

Fig. 10: Graphs for the pruning performance of DIGDUG, filtering pruning and $\mathcal{SGC}$ pruning for 2,4 and 8 data node clusters for max_interest 500 from a document.



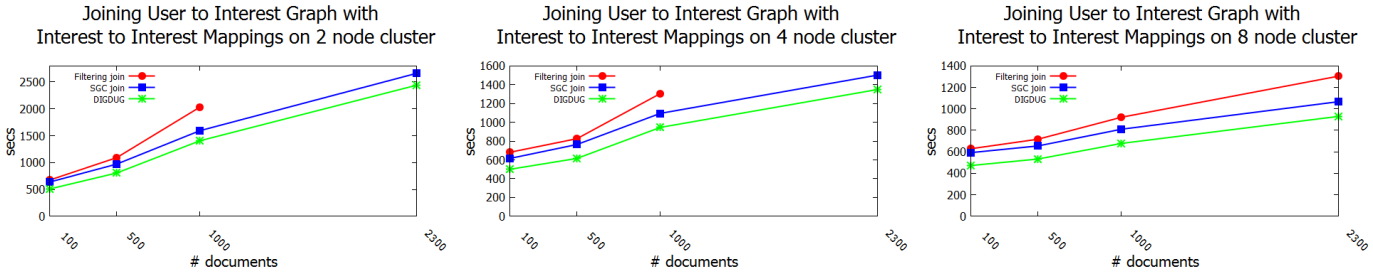
(a) 2 nodes (b) 4 nodes (c) 8 nodes

Fig. 11: Graphs for the pruning performance of DIGDUG, filtering pruning and $\mathcal{SGC}$ pruning for 2,4 and 8 data node clusters for max_interest 1000 from a document.



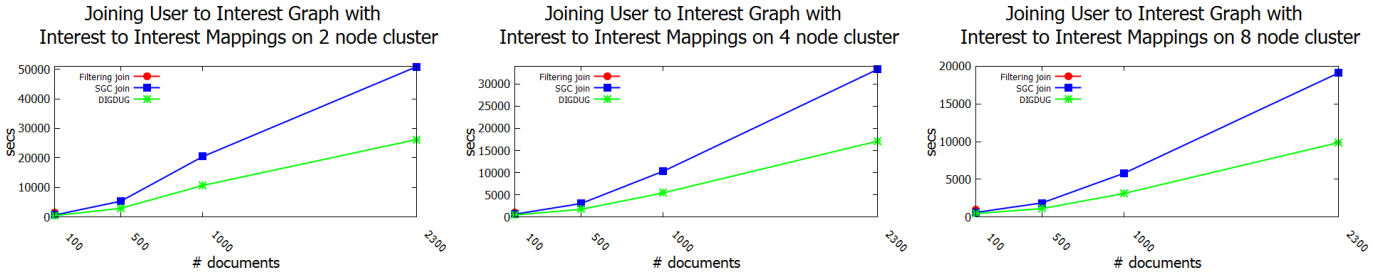
(a) 2 nodes (b) 4 nodes (c) 8 nodes

Fig. 12: Comparison of the performance of DIGDUG and other techniques for user connections with user-to-interest and pruned interest-to-interest graphs for 2,4 and 8 data node clusters for max_interest 200



(a) 2 nodes (b) 4 nodes (c) 8 nodes

Fig. 13: Comparison of the performance of DIGDUG and other techniques for user connections with user-to-interest and pruned interest-to-interest graphs for 2,4 and 8 data node clusters for max_interest 500



(a) 2 nodes (b) 4 nodes (c) 8 nodes

Fig. 14: Comparison of the performance of DIGDUG and other techniques for user connections with user-to-interest and pruned interest-to-interest graphs for 2,4 and 8 data node clusters for max_interest 1000

relations. However, for topics and influencer connections the join requires pairs of influences that are connected to each other through shared common and connected interests to be identified. We assume that n = $|U|+|I|$ where $|U| >> |I|$, U is the user graph and I is the interest graph, m is the number of nodes in the pruned interest-to-interest graph, p is the number of edges in the user-to-interest graph and q is the number of edges in the pruned interest-to-interest graph. For this notation we use r = $|U|$ and s = $|I|$. Due to the need to evaluate the interest pairs for each set of users to match connected interests, the exercise order of r^2 for the $\mathcal{SGC}$ technique will not fit within the value of a key-value pair. However, as the join on interests is driven by users for an interest in the value and using key-value streaming for user pairs that actually do have common and connected interests can be measured as r', $\mathcal{SSDGC}$ minimizes the memory footprint of each reducer and also pays lower costs in communication to those incurred by $\mathcal{SGC}$. The assumption here is that r'<<r. This also decreases the number of interests from s to s', where s' is the number of interests of r', s' <<s. This inversion of value for the key and the use of sorting in streaming allows joins in these graphs to finish even when a smaller node cluster is used. A MapReduce operation to join influencers based on topics in $\mathcal{SSDGC}$ therefore has the following constraints:

- *Disk*: The disk usage explodes to O(r^2+rs+m+p+q) for the number of influencer pairs that must be examined in $\mathcal{SGC}$ which is curtailed to O(r'+r's'+m+p+q) in $\mathcal{SSDGC}$.
- *Memory*: Each machine needs to process influencer pairs and their interests and connected interests, represented as O($\frac{r^2+m+rs}{t}$) which is reduced to O($\frac{r'+m+r's'}{t}$) in $\mathcal{SSDGC}$.
- *Communication*: In each Shuffle round, the machines transmit O(r^2+rs+m) key-value pairs data for $\mathcal{SGC}$ while it transmits O($r'+r's'+m$) for $\mathcal{SSDGC}$.
- *CPU*: In each round, this represents CPU consumption on each machine, which is of the order of O(r^2) in $\mathcal{SGC}$ and O(r') in $\mathcal{SSDGC}$.
- *Number of rounds*: The number of rounds needed to finish the pruning and subsequent joins.

Property	$\mathcal{SGC}$	$\mathcal{SSDGC}$
Disk/machine	$O(\frac{r^2+rs+m+p+q}{t})$	$O(\frac{r'+r's'+m+p+q}{t})$
Disk/total	$O(r^2+s+m+p+q)$	$O(r'+s'+m+p+q)$
Memory/machine	$O(\frac{r^2+rs+m}{t})$	$O(\frac{r'+r's'+m}{t})$
Memory/total	$O(r^2+rs+m)$	$O(r'+r's'+m)$
Communication/machine	$O(\frac{r^2+rs+m+p+q}{t})$	$O(\frac{r'+r's'+m}{t})$
Communication/total	$O(r^2+rs+m)$	$O(r'+r's'+m)$
CPU/machine	$O(\frac{r^2}{t})$	$O(\frac{r'}{t})$
CPU/total	$O(r^2)$	$O(r')$
Number of rounds	$O(1)$	$O(1)$

TABLE 2: Scalable Graph Algorithm classes in MapReduce

The differences between $\mathcal{SGC}$ and $\mathcal{SSDGC}$ for each these criteria are defined in Table 2. These differences consolidate the differences in joining the two graphs for the two classes.

6.3.3 Scaling efficiency

The algorithms scale well with larger datasets. The scaling of both pruning and joins is efficient and we further validate that with strong and weak scaling [Moreland and Oldfield, 2015] results as shown in Figures 15 for pruning algorithm and Figure 16 for $\mathcal{SSDGC}$ join. In strong scaling the problem size stays the same while we measure run time by gradually increasing the number of nodes. In weak scaling we keep the workload on each processor the same while gradually increasing the size of problem along with number of processors. The scaling charts show that with increasing

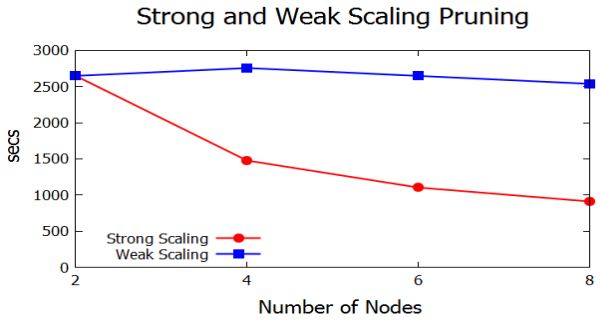


Fig. 15: Strong and weak scaling of interest graph pruning.

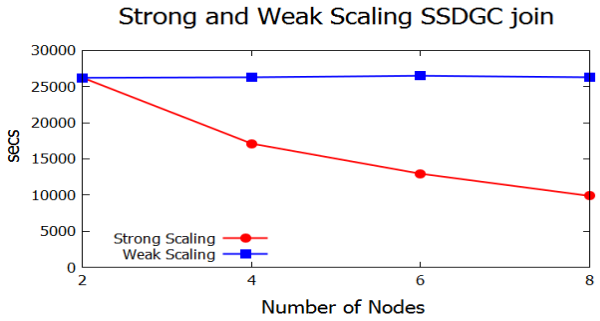


Fig. 16: Strong and weak scaling of $\mathcal{SSDGC}$ join.

number of processors the performance keeps improving, and stays stable for same amount of data per node for increasing data sizes. Value of max_interest was set to 1000 for experiments. The scaling charts demonstrate that the algorithms can be scaled indefinitely with increasing number of documents, interests and authors in a horizontal manner by simply adding nodes to the cluster. During the execution of pruning and join MapReduce jobs the memory of the nodes increases rapidly as more mappers and reducers in MapReduce are started. It stabilizes after a while and processes proceed to release it upon completion.

6.4 Case Study

The user to user maps and interest mappings for the Diabetes use case once the user and interest mappings and interest graph creation and pruning had been performed revealed some interesting results. For this case study, we looked at a wide range of data sources, including medical and academic journals, patents and additional technical sources.

6.4.1 Interest Connections

The top three emerging interests and a sample of their related interests are shown in Table 3.

Connection between butyllithium and other infrequent interests: Interests found through DIGDUG are relevant because although they are non-obvious and uncommon, they are meaningful for the exploration of emerging topics in medical research. One such interest pair surfaced through the aforementioned methodology is the linkage between butyllithium and almorexant. While they initially appear to be loosely related to each other but completely unrelated to the topic of diabetes, further inspection uncovers a very clear relationship between the two terms as well as a direct impact on emerging treatment methods for diabetes. For example, exploring the linkage between butyllithium and almorexant, and their impact on the treatment of diabetes reveals that butyllithium is commonly used in organic chemistry as a lithiating and reducing reagent that has been used to improve the pharmacokinects of GPR40 full agonists.

GPR40 is an important component in the fatty acid augmentation of insulin secretion, which has made it a therapeutic target for the treatment of Type II Diabetes. GPR40, which is also known as free fatty acid receptor 1 (FFA1), is a cellular membrane protein that helps maintain energy homeostasis in the body by binding free fatty acids, thus indicating the presence of nutrients. GPR40 activation triggers increased insulin secretion, boosting the increased absorption of glucose at the cellular level and thus serving as a key management strategy for glucose regulation in diabetic patients. In addition to its effects on GPR40, butyllithium is used in the synthesis of cyclopropane derivatives that are used in pharmaceutical compounds as orexin receptor inhibitors. Orexin receptor inhibitors are used to prevent or treat sleep disorders. One such drug, almorexant is currently being codeveloped by two pharmaceutical companies Actelion and GSK, who are seeking to take advantage of its primary function as a dual orexin receptor antagonist. A common side effect is a decrease in appetite, which results in decreased blood glucose levels. Orexin receptor inhibitors are currently being investigated as potential therapeutic treatments for the control of diabetes.

The connection between butyllithium and almorexant is non-obvious to all but the most highly involved diabetes research specialists. In addition to linking butyllithium to almorexant, the algorithm revealed supporting linked terms such as:

- Impaired insulin secretion
- Hyperinsular obesity
- Insulin analog
- Diet therapy
- Cyclopropane
- Diabetes mellitis
- Adult obesity patient
- hyperinsulinar obesity
- hypothyroid obesity

These connected terms support the story explaining the relevance of butyllithium in the treatment of diabetes through appetite regulation and the modulation of insulin secretion by highlighting different applications for the treatment of diabetes. We tested the hypothesis that pruning interests that connect interests across documents may lead to the loss of meaningful information. In dropping the pruned interests and maintaining the derived connection between two interests across documents resulted only in spurious connections. Table 3 shows examples of derived connections for butyllithium. Every sample of connected indirect interests that were connected by pruned interests resulted from a generic linkage related to organic synthesis that is not unique to the topic of interest, diabetes. Instead, the generic organic synthesis terms are linked across all fields of study that include organic synthesis, and do not represent any meaningful information for the study of diabetes. We did find one example of a pruned term that may have been significant. This term was 'sildenafil', which is a vasodilator that can be used in treating symptoms related to disease progression in diabetes. However, this term was recovered through synonyms elsewhere in the analysis, leading to no meaningful loss of information through interest pruning. Acknowledging that this may be a positive outcome for this particular case study, but not desirable in all instances, we decided to include a switch to provide ability for the user to decide whether to connect interests of pruned terms as derived links or drop them.

Connection between Methyl Bromide and other infrequent terms. Butyllithium Methyl Bromide is a common soil fumigant that

can cause neuropathy in humans (it is a poison), which appears to be the connection to diabetes, which also causes neuropathy. Due to the shared neuropathic results, methyl bromide poisoning can be misdiagnosed as diabetes. The connection here is through Cafestol, a molecule found in coffee beans and is thought to be responsible for the biological effects. It has been patented for the treatment, prevention, and amelioration of Type 2 diabetes. Dibenzylethylenediamine, an amine used in the pharmaceutical version of cafestol, is also used in the treatment of diabetes, syndrome x, and several other conditions listed in connection to methyl bromide. The other terms found relate to either chemical compounds, their synthesis, or conditions associated with diabetes. For example sodium glucose cotransporter is an SGLT2 inhibitor, a new class of diabetes medication. The interests associated with this refer to the way it operates and the effect it has on various functions that impact diabetes. This shows the effectiveness of DIGDUG pruning, which highlights infrequent terms in technical documents and their connections with other infrequent terms.

6.4.2 User Connections

Table 4 displays the top connected users, some of their common and connected interests and their connections strengths. In addition to the top common terms found in the documents of both the authors, it also displays the top related term combinations the first interest of which comes from the first author, the second from the second author; the two interests are found to be connected to each other after the interest-to-interest graph has been pruned. These results demonstrate the effectiveness of our approach in identifying key emerging terms from the medical research literature and patents in the diabetes domain, as well as the connected community of researchers based on their interests.

Connection between Andrew J Karter and Caroline Gaudy: In the example in Table 4, Andrew J Karter and Caroline Gaudy have been identified by the algorithm as being connected. With a net connection score of 237, their common interests show a list of largely generic terms such as study, work, prevalent, accuracy, etc. However, all of the terms in the common interests are common not just to these two users but also to medical literature in general. This shows that scoring connectedness using only common interests produces many spurious connections. Similarly, filtering user connections using only generic terms would have filtered out this user connection, when in reality they share many meaningful connections. These meaningful connected interests surface through the algorithm's layer of abstracted graph mining. These users have linked interests related to their experience in clinical research and their shared focus on oncological effects and epidemiological studies using surveillance data.

Connection between Shizuo and Zenichi Ikeda: Shizuo and Zenichi Ikeda are connected with high connection strength, as shown in Table 4, as both are listed as inventors for various compounds used in the treatment of diabetes that are patented by the same pharmaceutical company (Takeda Pharmaceutical). However, their connected interests are more indicative of the terms used to describe the actual compounds patented and their research rather than their common interests which are all fairly generic. Both have worked with Tohru Yamashita and 3 others and appear on patents with them, but neither appear on the same patent. All worked at the same pharmaceutical company. Similarly Sanath Meegalla and Zhihua Sui both worked with Nalin Subasinghe and 2 other inventors at different pharmaceutical companies but never appear on the same patents. Zhihua Sui works for Janssen Pharmaceutical developing compounds for the treatment of diabetes (listed on patents) while Sanath Meegalla worked for 3-Dimensional Pharmaceuticals on compounds for chronic disorders mediated by certain pathways.

6.5 Analysis of Results

The cases described above show how useful it is to be able to identify relationships between users and interests for a highly technical domain. DIGDUG unearths both connected interests and shared interests. Connected interests are those interests explicitly shared by two people (user1→interest1→user2), while shared interests employ a level of abstraction to find interests that connect users. Shared interests build on a user's list of interests by connecting them to other interests and then identifying which other users are connected to that second layer of interests (user1→interest1→interest2→user2). This makes it possible to identify users who are connected deeply through their research interests by examining the common interests that appear in their respective technical publications while also linking connected interests with the connections between interests being determined by frequently cooccurring terms across the corpus.

These results demonstrate that a combination of connected and common interests offer a better way to find relationships between users. The relationship strength can then be further refined by weighting common terms lower than connected terms. The exploration of the differences between our proposed new approach, DIGDUG, and cosine similarity method for the top 3000 strongly connected user pairs showed that the scoring functions for each of the common user pairs differed significantly (correlation score R=0.18). This is illustrated in the example in Table 1. In addition to the 66% higher find rate, DIGDUG was able to create a more meaningful classification of user connections than cosine similarity.

7 Conclusions

Combining users and interests across documents is key to finding relationships between top influencers and topics in a technical domain. In highly technical documents, it is crucial to remove common terms and instead emphasize specific terms in order to identify emerging topics and their associated influencers in a new and innovative domain. Combining influencers through common terms and infrequent terms connected across documents is critical for finding meaningful connections between experts. DIGDUG greatly extends scalable graph processing by introducing operations that boost the performance of distributed processing in dense separable graphs. The case studies presented here demonstrate the usefulness of the techniques in DIGDUG and their ability to identify key topics and connections between influencers more effectively than existing document similarity based methods in the highly technical domain of diabetes research. They also confirm the excellent performance of the distributed algorithms employed in identifying them.

References

[Baldwin and Dayanidhi, 2014] Baldwin, B. and Dayanidhi, K. (2014). *Natural language processing with Java and LingPipe Cookbook*. Packt Publishing Ltd.

[Baraglia et al., 2010] Baraglia, R., Morales, G. D. F., and Lucchese, C. (2010). Document similarity self-join with mapreduce. In *2010 IEEE International Conference on Data Mining*, pages 731–736.

	Interest	Connected interests and Weights
1	butyllithium	acrylate mixture,1; al somatostatin regulates glp secretion,1; azepanine,1; cyclobutane,1; dihydrobenzimidazole,1; dihydrothiopyran,1; disintegrant,1; hyperplasmic obesity,1; invention diabetes mellitus,1; arrhythmia,1; dihydropyridine,1; gastrointestinal tract,1; hepatitis,1; impaired insulin secretion,1; hypoplasmic obesity,1; alimentary obesity,1; hypophyseal adiposity,1; hypothyroid obesity,1; adult obesity patient,1; cyclopentene,1; diacetic acid,1; dihydrobenzofuran,1;...; ethyl bromoacetate,3; ethylbutyl,3; fragment ion peak,3; hydroflumethiazide,3; ikk inhibitors,3... crude title compound(derived),2; carboxylate(derived),3,...
2	methylbromide	dibenzylethylenediamine,1; dioxaborolane,1; ethyl phenyl,1; glycogen phosphorylase,1; detection reagents,1; tetrahydrofuranyl,1; normal levels,1; cell lines,1; glucosamine,1; rotary evaporation,1; carboxylic acid ethyl,1; density lipoprotein cholesterol,1; dyslipidemias,1; amphetamine,1; metabolic disorders,1; suitable binders lubricants,3; dimethoxybiphenyl,3; dosage level,3; chain alkyl groups,3; drops,3; inert pharmaceutical excipients,3; name,3; other reaction parameters,3; otherwise,3; pharmaceutically acceptable salts thereof present invention,3; reaction temperatures,3; sterile suspensions,3; isobutoxy,3; metabolic syndrome x more preferably,3; chloroaniline,3; foam,3; genetic testing,3; pharmaceutical excipients,3; polyvinyl,3;...alogliptin(derived),4;insulin detemir(derived),2...
3	sodium glucose cotransporter	blind study medication,1; hdl cholesterol level,1; hepatic insulin action,1; regional adipose tissue distribution,1; gluconeogenesis,1; metaanalysis,1; streptozotocin,1; selective sglt inhibitor,1; symptomatic hypoglycemic events,1; urosepsis,1; renal glucose reabsorption,1; systolic hypertension,1; al empagliflozin,3; inhibitor characterisation,3; other sglt inhibitors diabetes,3; al empagliflozin,3; other sglt inhibitors diabetes,3,...pancreatic cancer(derived),2; normal range(derived),1;...

TABLE 3: Interests and connected interests.

	Users	Connection strength	Common interests and weights	Connected interests and weights
1	Shizuo Kasai::Zenichi Ikeda	5633	heterocyclic group,16; compound,14; hydrocarbon group,14, c alkyl group,10;...; antithrombotic agent,5	c alkylthio group:hydrocarbon group,750; hydrocarbon group:fragment ion peak,450; triphenylphosphine::phosphoric acid(derived),7; insulin::hydroxy group(derived),4...
2	Sanath Meegalla::Zhihua Sui	4422	group,19; compound,18; mixture,17; disorder,15; treatment,12; enantiomer,11...	configuration:exacerbation,432; ointment:exacerbation,432; configuration:carrier,300; mesylate:configuration,294;... glucose::succinic acid(derived),4; insulin resistance::ethanesulfonic acid(derived),2;...
3	Matthias Eckhardt:Stephanie Venn-Watson	2731	compounds,9; group,9; acetonitrile,5; pharmaceutical compositions,4; fatty acids,4;inhibition,3...	haemoglobin:step,128; mammals:contact,81; location:step,81; carrier:mammals,72; resistance:leptin,64; tautomers:advantage,48; dibenzylethylenediamine:hydrazine,36;... insulin::incretins(derived),4; ...
4	John J. Acton::Olga BABICH	1279	compound,8; subject,7; treatment,6; reduction,6; ethyl acetate,5; room temperature,5; inhibition,4; methyl,4; vacuo,4; dichloromethane,4	schemes:wide variety,50; imidazolidinyl:benzoxazolyl,49; dichloro:benzoxazolyl,49; benzotriazolyl:imidazolidinyl,49; indolyl:imidazolidinyl,49; dichloro:indazolyl,49; bezafibrate:dibenzylethylenediamine,49;... ;concentration::glucose(derived),2;...
5	Andrew J. Karter::Caroline Gaudy	248	study,4; work,4; american diabetes association readers,2; full access,2; manuscript,2; profit,2; integrity,2; all data,2; article,2; type diabetes,2; absence,2; accuracy,2; diabetes,2,..	causes:confusion,9; diabetic complications:indications,4; refractory:causes,4; japanese patients:fujimoto wy,1; ketones:rapidly,1; aware:surveillance data,1; departments:socioeconomic status,1;substantive differences:polydipsia,1; ketoacidosis:modest differences,1; ketones:causes,1,...type diabetes::accuracy(derived),1;...

TABLE 4: Connected Users and their connection strengths through interests.

[Bordes and Gabrilovich, 2014] Bordes, A. and Gabrilovich, E. (2014). Constructing and mining web-scale knowledge graphs: Kdd 2014 tutorial. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 1967–1967, New York, NY, USA. ACM.

[Borgelt and Berthold, 2002] Borgelt, C. and Berthold, M. R. (2002). Mining molecular fragments: Finding relevant substructures of molecules. In *2002 IEEE International Conference on Data Mining, 2002. Proceedings.*, pages 51–58. IEEE.

[Chen et al., 2014] Chen, Y., Zhao, X., Xiao, C., Zhang, W., and Tang, J. (2014). Efficient and scalable graph similarity joins in mapreduce. *The Scientific World Journal*.

[Cook and Holder, 2006] Cook, D. J. and Holder, L. B. (2006). *Mining graph data*. John Wiley & Sons.

[Cuzzocrea et al., 2017] Cuzzocrea, A., Buyya, R., Passanisi, V., and Pilato, G. (2017). Mapreduce-based algorithms for managing big rdf graphs: State-of-the-art analysis, paradigms, and future directions. In *Proceedings of the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, CCGrid '17, pages 898–905, Piscataway, NJ, USA. IEEE Press.

[Das Sarma et al., 2014] Das Sarma, A., He, Y., and Chaudhuri, S. (2014). Clusterjoin: A similarity joins framework using map-reduce. *Proc. VLDB Endow.*, 7(12):1059–1070.

[Dean and Ghemawat, 2008] Dean, J. and Ghemawat, S. (2008). Mapreduce: Simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113.

[Dingding Wang, 2014] Dingding Wang, Tao Li, M. O. (2014). Generating pictorial storylines via minimum-weight connected dominating set approximation in multi-view graphs. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, pages 683–689.

[Fang et al., 2014] Fang, Q., Sang, J., Xu, C., and Rui, Y. (2014). Topic-sensitive influencer mining in interest-based social media networks via hypergraph learning. *IEEE Transactions on Multimedia*, 16(3):796–812.

[Feldman et al., 2010] Feldman, J., Muthukrishnan, S., Sidiropoulos, A., Stein, C., and Svitkina, Z. (2010). On distributing symmetric streaming computations. *ACM Trans. Algorithms*, 6(4):66:1–66:19.

[Fries et al., 2014] Fries, S., Boden, B., Stepien, G., and Seidl, T. (2014). Phidj: Parallel similarity self-join for high-dimensional vector data with mapreduce. In *2014 IEEE 30th International Conference on Data Engineering*, pages 796–807.

[Furukawa et al., 2015] Furukawa, T., Mori, K., Arino, K., Hayashi, K., and Shirakawa, N. (2015). Identifying the evolutionary process of emerging technologies: A chronological network analysis of world wide web conference sessions. *Technological Forecasting and Social Change*, 91(Supplement C):280 – 294.

[Gao et al., 2014] Gao, J., Zhou, J., Zhou, C., and Yu, J. X. (2014). Glog: A high level graph analysis system using mapreduce. In *2014 IEEE 30th International Conference on Data Engineering*, pages 544–555.

[Gundecha and Liu, 2012] Gundecha, P. and Liu, H. (2012). *Mining Social Media: A Brief Introduction*, chapter 2, pages 1–17.

[Guy et al., 2013] Guy, I., Avraham, U., Carmel, D., Ur, S., Jacovi, M., and Ronen, I. (2013). Mining expertise and interests from social media. In *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13, pages 515–526, New York, NY, USA. ACM.

[Hu et al., 2014a] Hu, H., Wen, Y., Chua, T. S., and Li, X. (2014a). Toward scalable systems for big data analytics: A technology tutorial. *IEEE Access*, 2:652–687.

[Hu et al., 2014b] Hu, P., Huang, M.-L., and Zhu, X.-Y. (2014b). Exploring the interactions of storylines from informative news events. *Journal of*

Computer Science and Technology, 29(3):502–518.

[Huan et al., 2004] Huan, J., Wang, W., Bandyopadhyay, D., Snoeyink, J., Prins, J., and Tropsha, A. (2004). Mining protein family specific residue packing patterns from protein structure graphs. In *Proceedings of the eighth annual international conference on Resaerch in computational molecular biology*, pages 308–315. ACM.

[Jin and Lu, 2009] Jin, H. and Lu, Y. (2009). The optimal linear combination of multiple predictors under the generalized linear models. *Statistics & probability letters*, 79(22):2321–2327.

[Joung and Kim, 2017] Joung, J. and Kim, K. (2017). Monitoring emerging technologies for technology planning using technical keyword based analysis from patent data. *Technological Forecasting and Social Change*, 114:281 – 292.

[Karloff et al., 2010] Karloff, H., Suri, S., and Vassilvitskii, S. (2010). A model of computation for mapreduce. In *Proceedings of the Twenty-first Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '10, pages 938–948, Philadelphia, PA, USA. Society for Industrial and Applied Mathematics.

[Kiveris et al., 2014] Kiveris, R., Lattanzi, S., Mirrokni, V., Rastogi, V., and Vassilvitskii, S. (2014). Connected components in mapreduce and beyond. In *Proceedings of the ACM Symposium on Cloud Computing*, SOCC '14, pages 18:1–18:13, New York, NY, USA. ACM.

[Kondreddi et al., 2014] Kondreddi, S., Triantafillou, P., and Weikum, G. (2014). Combining information extraction and human computing for crowdsourced knowledge acquisition. In *Data Engineering (ICDE), 2014 IEEE 30th International Conference on*, pages 988–999.

[Kumar et al., 1999] Kumar, R., Raghavan, P., Rajagopalan, S., and Tomkins, A. (1999). Trawling the web for emerging cyber-communities. *Computer networks*, 31(11-16):1481–1493.

[Lattanzi et al., 2011] Lattanzi, S., Moseley, B., Suri, S., and Vassilvitskii, S. (2011). Filtering: A method for solving graph problems in mapreduce. In *Proceedings of the Twenty-third Annual ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '11, pages 85–94, New York, NY, USA. ACM.

[Moreland and Oldfield, 2015] Moreland, K. and Oldfield, R. (2015). Formal metrics for large-scale parallel performance. In Kunkel, J. M. and Ludwig, T., editors, *High Performance Computing*, pages 488–496, Cham. Springer International Publishing.

[Moy, 2005] Moy, M. (2005). Using tmods to run best friends group detection algorithm. *21st Century Technologies, Austin, TX*.

[Mukherjee and Tirthapura, 2017] Mukherjee, A. P. and Tirthapura, S. (2017). Enumerating maximal bicliques from a large graph using mapreduce. *IEEE Transactions on Services Computing*, 10(5):771–784.

[Newman et al., 2014] Newman, N. C., Porter, A. L., Newman, D., Trumbach, C. C., and Bolan, S. D. (2014). Comparing methods to extract technical content for technological intelligence. *Journal of Engineering and Technology Management*, 32(Supplement C):97 – 109. Special Issue on Emergence of Technologies: Methods and Tools for Management.

[Ng et al., 2015] Ng, K., Sun, J., Hu, J., and Wang, F. (2015). Personalized predictive modeling and risk factor identification using patient similarity. *AMIA Summits on Translational Science Proceedings*, 2015:132.

[Okcan and Riedewald, 2011] Okcan, A. and Riedewald, M. (2011). Processing theta-joins using mapreduce. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*, SIGMOD '11, pages 949–960, New York, NY, USA. ACM.

[Qin et al., 2014] Qin, L., Yu, J. X., Chang, L., Cheng, H., Zhang, C., and Lin, X. (2014). Scalable big graph processing in mapreduce. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, SIGMOD '14, pages 827–838, New York, NY, USA. ACM.

[Rossetti et al., 2017] Rossetti, M., Vargas, S., Magatti, D., Pettit, B., Kershaw, D., Hristakeva, M., and Jack, K. (2017). Effectively identifying users' research interests for scholarly reference management and discovery. In *Proceedings of the 1st Workshop on Scholarly Web Mining*, SWM '17, pages 17–24, New York, NY, USA. ACM.

[Santos et al., 2016] Santos, R. F. D., Shah, S., Boedihardjo, A., Chen, F., Lu, C.-T., Butler, P., and Ramakrishnan, N. (2016). A framework for intelligence analysis using spatio-temporal storytelling. *GeoInformatica*, 20:285–326.

[Shukla et al., 2016a] Shukla, M., Dos Santos, R., Fong, A., and Lu, C.-T. (2016a). DISTL: Distributed in-memory spatio-temporal event-based storyline categorization platform in social media. In *Proceedings of the Second International Conference, GISTAM 2016, Rome, Italy, April 26-27, 2016*, pages 39–50.

[Shukla et al., 2017a] Shukla, M., Fong, A., Dos Santos, R., and Lu, C.-T. (2017a). *Event Categorization and Key Prospect Identification from Storylines*, pages 62–88. Springer International Publishing, Cham.

[Shukla et al., 2017b] Shukla, M., Santos, R. D., Chen, F., and Lu, C.-T. (2017b). DISCRN: A distributed storytelling framework for intelligence analysis. *Big Data 5:3*.

[Shukla et al., 2016b] Shukla, M., Santos, R. D., Fong, A., and Lu, C.-T. (2016b). DERIV: distributed in-memory brand perception tracking framework. In *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA), Anaheim, CA*, pages 387–393.

[Shukla et al., 2017c] Shukla, M., Santos, R. D., Fong, A., and Lu, C.-T. (2017c). DERIV: distributed brand perception tracking framework. *Journal of Big Data, 4(1), 17*.

[Tang et al., 2015] Tang, S., Wu, F., Li, S., Lu, W., Zhang, Z., and Zhuang, Y. (2015). Sketch the storyline with charcoal: a non-parametric approach. In *Proceedings of the 24th International Conference on Artificial Intelligence*, pages 3841–3848. AAAI Press.

[Vernica et al., 2010] Vernica, R., Carey, M. J., and Li, C. (2010). Efficient parallel set-similarity joins using mapreduce. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, SIGMOD '10, pages 495–506, New York, NY, USA. ACM.

[Wang et al., 2009] Wang, F., Sun, J., Li, T., and Anerousis, N. (2009). Two heads better than one: Metric+ active learning and its applications for it service classification. In *2009 Ninth IEEE International Conference on Data Mining*, pages 1022–1027. IEEE.

[Young et al., 2019] Young, S. L., Goldowsky-Dill, N. W., Muhammad, J., and Epstein, M. M. (2019). Connecting experts in the agricultural and meteorological sciences to advance knowledge of pest management in a changing climate. *Science of The Total Environment*, 673:694–698.

[Zaharia et al., 2012] Zaharia, M., Chowdhury, M., Das, T., Dave, A., Ma, J., McCauly, M., Franklin, M. J., Shenker, S., and Stoica, I. (2012). Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *Presented as part of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)*, pages 15–28, San Jose, CA. USENIX.

[Zhang et al., 2015] Zhang, L., Li, L., and Li, T. (2015). Patent mining: A survey. *SIGKDD Explor. Newsl.*, 16(2):1–19.

[Zhang et al., 2012] Zhang, X., Chen, L., and Wang, M. (2012). Efficient multi-way theta-join processing using mapreduce. *Proc. VLDB Endow.*, 5(11):1184–1195.

PLACE PHOTO HERE

Manu Shukla is the co-founder of Omniscience Corporation based in Palo Alto California.

Dinesh Dharme is a Software Engineer at Omniscience Corporation. He has a ME and BE in Electrical Engineering from the Indian Institute of Technology in Mumbai.

Pallavi Ramnarain is the Director of Applied Machine Learning at Omniscience Corporation in Palo Alto California. She has a PhD in Biomedical Engineering from Virginia Commonwealth University.

Ray Dos Santos is a researcher at the U.S. Army Labs, ERDC, GRL in Alexandria Virginia. He holds a PhD in Computer Science from Virginia Tech.

Chang-Tien Lu is a Professor of Computer Science at Virginia Tech, Falls Church, Virginia.