Optimized Provenance Reconstruction for Machine-generated data

Samridhi Agrawal

A report

submitted in partial fulfillment of the

requirements for the degree of

Master of Science in Computer Science & Software Engineering

University of Washington Bothell

2022

Reading Committee:

Prof. Hazeline Asuncion, Committee Chair

Prof. Brent Lagesse, Committee Member

Prof. Johnny Lin, Committee Member

Program Authorized to Offer Degree:

Computing & Software Systems

University of Washington Bothell

Abstract

Optimized Provenance Reconstruction for Machine-generated data

Samridhi Agrawal

Chair of the Supervisory Committee:
Associate Professor Hazeline Asuncion, Ph.D.
Computing & Software Systems

A lot of data is created, deleted, copied and modified easily over the Internet which makes it difficult to identify the authenticity and credibility of the data. It is important to reconstruct the provenance of data which has lost its provenance information. There are techniques which help in recovering the metadata from which provenance can be reconstructed. However, many systems fail to capture provenance due to lack of provenance capture mechanisms such as the Source file repositories or file storage system. The Provenance-Reconstruction approach proposed by the "Provenance and Traceability Research Group" has numerous projects on reconstructing provenance.

The current research (OneSource) captures various reconstruction techniques for machine generated datasets with attributes such as file size, semantic meaning of the content, and word count in the files. OneSource improves provenance reconstruction for git commit history as machine generated datasets. OneSource algorithm uses a multi-funneling approach which includes

techniques such as data cleaning in python, topic modelling and Cosine similarity for clustering, and lineage algorithm with endpoints known to achieve higher accuracy in recovering valid provenance information. OneSource generates ground truth data by extracting commit history and file versions of a git repository. To assess OneSource model performance, the model is evaluated on various datasets with varying data size and count of files. OneSource reconstructs provenance of clusters and relationship of files (cluster derivation) within the cluster. The evaluation results indicate that OneSource can reconstruct provenance of cluster by attaining 90% precision and reconstruct provenance of cluster derivation by attaining 66% precision with Cosine similarity as the clustering method. OneSource yields improvement in accuracy of 60% for cluster derivation than the existing technique. In the future, research studies may use parallelization for larger datasets as well as optimizations in lineage algorithm potentially improving the model performance.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

It is my pleasure to express my sincere gratitude to my committee members for being so generous with their expertise and time, as well as providing the resources needed for the successful implementation of this research. I am especially grateful to my committee chair, Dr. Hazeline Asuncion, for her help, support, and hard work towards the success of this research. I want to thank Dr. Brent Lagesse and Dr. Johnny Lin for their time and support as well.

Also, I would like to acknowledge and thank former members of my research team. I would like to thank Subha Vasudevan, Andrew Nakaruma (SeekPeek's team) for their valuable insights and, Mathew Hewitt for helping me in my research. My former graduate advisor, Ms. Suzanna Martinez, and international student advisor Jennifer Kim have been a great help throughout the degree program.

I would like to express my sincerest gratitude to the University of Washington Bothell and others who have assisted with this research. Lastly, I would like to thank my family and friends for encouraging me throughout my master's program.

CHAPTER 1 INTRODUCTION

The Internet represents a huge file storage system. We rely on different datasets from the web in our everyday lives, but it is often difficult or impossible to determine where it came from or how it was produced. It is possible that data could have been copied from another source and then mutated. The provenance of information is crucial to making decisions about whether information can be trusted, how to integrate diverse information sources, and how to give credit to originators when reusing information. The speed and convenience at which information is created and distributed on the Internet has made it a challenging task to determine the integrity of our sources and the validity of the information found [1] [2]. This is the main importance of Data Provenance.

Data Provenance is defined as recording metadata whenever data is created, modified, copied, or deleted [3]. This helps us in determining the integrity of the datasets with recorded provenance. The main problem is many of the datasets over the Internet do not have recorded data provenance, which means that their integrity is questionable. If this issue is not investigated, it leads to unreliable results and misleading information. Provenance Reconstruction [4] is an area of study within Data Provenance that attempts to solve this problem by establishing data links between different pieces of data, with the aim of estimating its provenance, therefore reducing the amount of work required by researchers to validate their data. Provenance reconstruction is not only important for academic research, but it has uses in many other areas such as media and fraud detection including metadata annotation, which is when metadata is generated without user input [5].

A technique for provenance reconstruction was developed by "Provenance and Traceability Research Group" [6]. This solution was developed in Java, which uses a multi-level funneling technique that performs Topic Modeling by using the Latent Dirichlet Allocation (LDA) model [7]. A multi-funneling approach is filtering the data through funnels in which they group the files based on semantic content and available metadata information. The approach was applied on small datasets and accuracy was satisfactory. LDA is an unsupervised algorithm that can be used to identify the semantic relationship between words in a set of documents and therefore, divide such documents into topics according to word distribution.

Improvements to this existing approach were as follows randomizing the file names, assuming the endpoints [8] and, chaining algorithm [9]. The major disadvantage of the existing provenance reconstruction approach is the low accuracy in the provenance generated. Accuracy is calculated through precision, recall, and F1 measure. The new approach involves modifications to the existing multi-funneling approach which includes techniques such as data cleaning and pre-processing in python, topic modelling and cosine similarity for clustering, and lineage algorithm with endpoints known to find the source of each file. Each file will then get clustered to a source. This approach is called OneSource. We have designed and implemented OneSource to have higher accuracy than the existing systems. To improve the metrics and derive correct provenance relationships, the solution revised the multi- funneling approach and enhanced the techniques by adding more filters into each part of the execution so that a level produces usable output to the next level.

The report is organized into the following sections: In chapter 2, we investigate related works and provide a comprehensive description of the previous implementations. In chapter 3, we present a multi-funneling approach, and we also discuss the details of the implementation and design. In chapter 4, we discuss the metrics of performance and accuracy obtained by our

implementation and compare the metrics obtained by the previous implementations. At the end of chapter 5, we summarize our achievements, shortcomings and, suggest possible directions for future work in chapter 6.

CHAPTER 2 LITERATURE REVIEW AND BACKGROUND

Before discussing our implementation, it would be beneficial to review the importance and impacts of provenance reconstruction, the existing approaches to provenance reconstruction, and the efforts to optimize our performance.

2.1 DATA PROVENANCE AND RECONSTRUCTION

Data Provenance Reconstruction refers to the process of identifying the relationship between data files that have partial or no provenance recorded. Over many years, research has been conducted to reconstruct provenance through various techniques.

In data supply chains, customer trust builds based on the origin of the product, and how it was produced. This research helps us provide provenance techniques to reconstruct provenance on supply chain data such as shipping, tracking, and delivery [10]. A supply chain tracking software, such as Flexport helps to bring transparency and reliability to the end-products in the market. This is a good example to understand the importance of data provenance.

Data provenance plays an increasingly important role in making responsible Artificial Intelligence Systems [11] in guiding human decisions. If an AI system is biased, it leads to bias decision making. Powerful AI systems are based on our important characteristics: fairness, accountability, transparency, and explainability to avoid biases in the system. This study outlines existing biases and discusses possible implementations of data provenance to mitigate them.

To acquire provenance information on the news articles, existing methods were not universally applicable. This research group extracts information automatically through clustering algorithms, linking similar data and semantic similarity [12]. The result derivations are structured with existing tool called Provenance- Data Model (PROV-DM). The implementation was based on one use case – detecting the sources of news articles. Some of the methodology used in this paper are TFIDF, Cosine Similarity, cluster provenance, and cluster derivation provenance. Similarly, social media has become pervasive for information dissemination such as online journalism, transaction links, and product marketing. It is important for customers to trust the information shared on social media exhibited by this group who reconstruct the provenance of messages on social media over multiple levels mapping to PROV [13].

In another use case, Magliacane tries to address the scenario of multiple collaborators working on a common document [14]. The problem surfaced in her paper is that it becomes hard to track the contribution of each collaboration in the document. Magliacane attempts to resolve it in number of ways starting with calculating the edit distance on multiple versions of a document. Magliacane was able to construct provenance of similar documents using this technique. OneSource calculates similarity using a similar concept to help define a cluster.

A file system requires provenance of source files at various levels of granularity. Thus, a file provenance system is designed to capture automatic collection and management of provenance for all files in a hierarchy [15]. The hierarchy provides design objectives with respect to portability, security, efficiency, and flexibility deriving complete history of data objects. This results in a directed acyclic graph (provenance graph) to increase the visualization of data

provenance in a file system. Similarly, provenance data is represented as a labeled directed acyclic graph in another approach [16]. The challenge was designing the graph with effective interfaces to define provenance of key objects. The problem was solved with the clustering approach which provides better visualization of provenance graphs through user interface.

On another topic, trusted execution environment forms the basis for provenance reliability. If we conduct provenance without a trusted environment, the results can be manipulated. Stamatogiannakis et al. developed a "Data Tracker" to capture provenance at multiple levels of granularity by converting results into W3 PROV format [17]. This increased the data trustworthiness with a common execution environment making the provenance reliable.

RDF data presents another tool to reconstruct provenance of any dataset. The author here uses RDF as a tool running on top of decentralized source code management system Git [18]. The author presents a way to query history graphs and arbitrary revisions of Git versioned stores. They also annotate individual statements with provenance information based out of minimal granularity. OneSouce also uses Git data to construct provenance on machine generated data.

Current computer systems are not equipped with the tools to capture provenance of machine log files. Tan designed a solution that extracts and model's information from log files to derive provenance relations [19]. It helps to resolve the logging errors using provenance information. Log files generated in the system are type of Machine-generated data. OneSource is based on Machine-generated datasets which can further be extended to log files. However, the format and methodology behind the log files provenance is completely different from the current research.

In computer networking, source nodes contain the details of the originating node moving through a network path passing information to the central location. The challenge is to receive complete information from source node to the destination node without information loss. One approach evaluates information received, node references, and information sequence to establish information trust [20]. OneSource also tracks relationships between datasets to establish cluster derivation provenance.

Storing provenance information from scientific experiments has been a major concern for scientists lately. With the introduction of Scientific Workflow Management System, it was resolved as provenance capturing and storing is automatically managed. A survey to analyze the large amounts of provenance data generated by workflow executions that helped to verify mistakes and improve workflow quality [21]. Similarly, provenance information captured in workflows system are used in execution of scientific experiments. Workflow executions can manipulate sensitive datasets that contains information about individuals. To address this problem, another research proposes anonymization of provenance of workflows. This guarantees confidentiality without compromising lineage information [22]. Lineage provides transparency to the relationships between the data records, generated by workflow modules.

Data Provenance literature review was conducted on three dimensions: a) techniques, b) datasets, and c) tools. This segregation helped us compare how different research papers address these fundamental decisions of conducting provenance and find a whitespace to conduct our research.

2.1.1 Techniques

Table 2. 1: Comparison Table based on Techniques

Techniques	Blei et al. [7]	Magliacane [11]	Tan [18]	Walle et al. [10]	Oliveira et al. [22]
TF-IDF vectorization			X	X	X
Topic Modeling	X				
Cosine Similarity			X	X	
Edit distance		X			
Clustering		X		X	

2.1.1.1 TF-IDF Vectorization

Term Frequency - Inverse Document Frequency (TF-IDF) is an algorithm used to transform a text document into number representation of the document. The number representation of the document is used as an input to the machine learning algorithm which generates a prediction. TF-IDF [23] is a measure of originality of a word by comparing the number of times a word appears in a document with the number of documents the word appears in. To calculate document similarly using TF-IDF approach, each document is transformed into a Vector Space Model also known as “bag of words” [12]. OneSource has used TF-IDF method to find the frequency of words with respect to variety of documents.

2.1.1.2 Topic Modeling

Topic Modeling is a branch of natural language processing under machine learning. Topic modeling is a statistical model that identifies abstract topics or keywords from a group of

documents or text articles. Topic modeling is used to predict main themes out of unstructured data [7]. Topic modeling plays a key role in software traceability [24], software defects detection [25], and tag recommendation [26].

The topic modeling technique used in 'OneSource' is Latent Dirichlet Association (LDA). The input to LDA is a collection of documents which results in generation of topics and associated weights. LDA processes the input data for a certain number of iterations and generates a document topic matrix. This matrix is further processed to determine the provenance of these documents. As a result, a document belongs to a cluster of topics that has dominant weight associated to it. Recent applications of LDA include sentiments analysis [27], web page tagging [28], spam filtering [29], and email classification [30]. While LDA increase our ability to detect topics, precision and recall values recorded in OneSource were relatively low. OneSource was further improved using additional techniques.

The MALLET topic modeling tool is a Java-based package for statistical natural language processing [31][32]. MALLET performs machine learning applications such as document classification, clustering, topic modeling, and information extraction. OneSource uses the java-package by enumerating the MALLET command line steps into programming code.

2.1.1.3 Cosine Similarity

Cosine similarity measures similarity between two more vectors [12] [19]. Mathematically, cosine similarity is the cosine of the angle between the vectors where documents are represented as vectors. Using vector representation, two documents can be compared for similarity. OneSource implements cosine similarity to find clusters of related documents, which helps to find the

provenance of documents. Document similarity is also used to predict system error in device networking using cosine similarity [33] (cosine similarity formula taken from [34]), Jaccard similarity, and Euclidean distance.

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}},$$

Figure 2. 1: Cosine Similarity

2.1.1.4 Edit distance

Edit distance is a metric that quantifies how dissimilar two texts are to each other. The common operations include insertion and removal of characters in a string known as longest common subsequence (LCS). Magliacane uses edit distance algorithms to reconstruct provenance automatically in a shared folder [14].

2.1.1.5 Clustering

All the above methods including Term Frequency - Inverse Document Frequency (TF-IDF), Topic Modeling, Cosine Similarity, and Edit Distance can be used to perform clustering, which is a process to group similar set of data points together. Data points are referred to as documents in this research. A document in a cluster will be more similar to documents within its cluster than documents in other clusters. Clustering has played an important role in the field of provenance reconstruction [12][14][16]. Macko uses metadata of data files to form local clustering for provenance graphs [35]. OneSource uses a combination of techniques for clustering identical data files to produce clusters and derive provenance of documents in the cluster.

2.1.2 Datasets

Table 2. 2: Comparison Table based on Datasets

Datasets	Magliacane [11]	Tan [18]	Belhajjame [20]
Human-generated dataset	X		
Machine-generated dataset		X	
Workflows systems			X

Provenance reconstruction challenge was issued in 2014 to recover provenance of machine generated data (Git history) and human generated data (wiki news articles and sources) [6]. Ground truth file was also provided with these data sets to verify the efficiency of the developed technique. Multi-level funneling approach was created to identify the provenance of the provided dataset. The multi-level funneling uses topic modeling as one of the funneling techniques. It has proved to be the most effective step in determining the provenance of machine generated data sets, as well as human generated data sets. A recent study shows how to anonymize workflow systems that contain provenance information without losing the data lineage [22].

2.1.2.1 Human-generated dataset

The human-generated dataset is made up of news articles extracted from the Wikinews website. The document files are articles in HTML format with text-based URLs in the articles' sources. The files are divided into two groups of articles and sources, which needed to be linked to each other. Hence, the goal of the reconstruction process for this dataset was to restore the articles to their sources. This relationship between the article and the source files is classified as "had Primary

Source" relationship. Multiple research work has been done on this dataset for provenance reconstruction, which serves as an impetus to our research [1] [2] [6].

2.1.2.2 Machine-generated dataset

Machine-generated datasets are version-controlled documents, collected from GitHub repositories. GitHub repositories store commit objects referencing the commits immediately before it. This relationship is classified in the reconstruction process as "was Derived From" relationship. The goal of the reconstruction process for this dataset is to restore the source file to its source endpoint. A source endpoint is the latest version of the file which cannot be derived from any other file. OneSource is an optimization on the previous work [8][9] for this dataset.

2.1.2.3 Workflows systems

A workflow is a graphical representation of a dataset. It is an acyclic graph whose nodes are program module and edges are dependencies between program modules. In the scientific field, such as the biomedical field, it is important to prevent sharing of sensitive information about other people. Provenance of such workflow systems is anonymized to conceal individual information without compromising transparency by preserving lineage relationships within workflow provenance. Hence, workflow systems can carry provenance information.

2.1.3 Provenance models

Table 2. 3: Comparison Table based on Provenance Models

Tools	Stamatogiannakis et al. [14]	Sultana et al. [13]	Magliacane [11]	Nies et al. [15]	Tan [18]	Walle et al. [10]	Missier et al. [16]
W3C PROV	X	X		X			X
PROV-DM		X	X			X	
RDF Graphs					X		

2.1.3.1 W3C PROV

W3 PROV refers to W3-Provenance working group [8] [36]. This group has PROV family of documents within the group. W3 PROV defines provenance as any information about an entity, its activity and how it was derived by the people. W3 PROV uses provenance information to assess quality, reliability, and trustworthiness of the entity. W3-PROV provides models to enable the interchange of provenance information in heterogeneous environments, such as the Web.

2.1.3.2 PROV-DM

PROV-DM is provenance data model for the W3-PROV family [37]. It is a core structure that forms the essence of provenance information. Its components are entities or activities (time they were created), entity derivations, links, and mechanisms to support provenance.

2.1.3.3 RDF Graphs

Resource Description Framework (RDF) is a graph data model that stores semantic facts. RDF statements comprise a directed graph that maps the relationship between entities. This graph database stores data for three entities i.e., subject, predicate, and object. An example of RDF

statement is ‘Document2’, ‘is Derived from’, ‘Document1’. Shady Elbassuoni [38], shares a retrieval model that helps users to execute SPARQL(structured queries) on RDF graphs. Pierre Maillot[39] measures the similarity between RDF graphs according to the patterns they share. OneSource uses RDF graphs to build the ground truth for the entities and their relationships.

2.2 THE LDA-GA + SR APPROACH FOR PROVENANCE RECONSTRUCTION

To reconstruct data provenance, our research group introduced a novel approach called Latent Dirichlet Association – Genetic Algorithm + Statistical Re-clustering. This approach was performed on human generated dataset such as Wikipedia articles. The input was a collection of disparate datasets including articles and sources. The goal was to trace articles back to the source files. The output was clusters of articles and sources. As discussed above, number of topics and iterations are inputs to the execution of LDA. Genetic algorithm is usually used for computationally intensive tasks with diversity in datasets in terms of text, structure, and source, different combination of inputs was used to create nine transforms of LDA in Latent Dirichlet Association – Genetic Algorithm + Statistical Re-clustering approach. GA helps to achieve maximum fitness by iterating over all input configurations up to the threshold value provided. In the last stage, statistical re-clustering is performed to join or split the cluster to get mapping between documents and sources. For human-generated dataset, a set of tests was conducted with a range of data sizes, from 10 to 500 articles, followed by a pool of over 5K articles by Vasudevan[2].

The cluster fitness level indicates how well defined a cluster is, including the distance between clusters and the proximity. To calculate the cluster fitness, the Silhouette coefficient is used. Silhouette coefficient is based on distance between the documents [40]. The distance between two documents is inversely proportional to the similarity between them. Silhouette Coefficient values range from -1 to +1. Silhouette Coefficient is positive when a document is close to other documents in its cluster than the center of other clusters. Silhouette Coefficient is negative, if the document is closer to the center of another cluster than to documents in its own cluster. The clusters were created using LDA+GA technique. A cluster includes mapping of an articles to one or more sources.

2.3 LDA + CHAINING ALGORITHM

A related work on provenance reconstruction was done by team on machine-generated dataset [9]. The team executed Latent Dirichlet Association for each document type to generate topic distribution of each document and calculated similarity score between the document based on topic distribution array. The similarity score is then used by the chaining algorithm to derive the relationship between the documents.

LDA + chaining algorithm applied the cosine similarity formula on LDA topic distribution output, but it led to low accuracy score. OneSource uses LDA and cosine similarity directly on the text document which has significantly improved the accuracy score (see results in section 4.2.1). The accuracy of this algorithm was low because no endpoint was defined to start the chaining process. Hence, to retrieve endpoints, there were multiple assumptions made such as documents with small size will be assumed as endpoints, after every 10 iterations, the 11^{th} file is an endpoint. OneSource starts the provenance reconstruction process from the endpoint (see section 3.3.2) which has

resulted in higher accuracy of clusters as showcased in results section in chapter 4. One further investigation, it was also identified that the ground truth generation process (GitScraper) was not generating accurate git file structure. This is resolved in OneSource, which is able to generate the git file structure for 25+ repositories. The chaining algorithm is also modified in OneSource, referred to as a lineage algorithm. For machine-generated dataset, the evaluation on five different datasets based on varying size was presented by [8].

CHAPTER 3 METHODOLOGY

The main goal of this research is to improve the provenance reconstruction of machine-generated datasets compared to the existing system. To achieve this goal, we revised the existing system from generating the ground truth of input GitHub repositories, pre-processing, topic modeling, and clustering, to find the source of each file. The new approach is called OneSource. In this section, we describe our multi funneling approach and enhanced techniques by adding more filters into each part of the execution so that a level produces usable output for the next level.

3.1 PROBLEM STATEMENT

Following are the research questions for this project:

- RQ1: Can we perform provenance reconstruction of machine-generated dataset?
- RQ2: Is it possible to cluster all endpoints and their associated files to represent a group of similar documents in machine-generated dataset?
- RQ3: Is it possible to infer data lineage starting from any file and back to the source file in machine-generated datasets*?*
- RQ4: How do we increase the accuracy of the existing provenance reconstruction algorithms of machine-generated datasets?

3.2 LDA-COSINE SIMILARITY + LINEAGE ALGORITHM

In my research, the provenance reconstruction approach is called Latent Dirichlet Association - Clustering through a technique such as cosine similarity + lineage algorithm for provenance derivation. This approach is called the ‘OneSource’ model and is focused on machine generated dataset such as GitHub repositories. The input is a collection of disparate datasets including

endpoints and unsorted files. The goal of the approach is to reconstruct provenance of each endpoint to its sources.

GitHub repositories store commit objects referencing the commits immediately before it. In the first phase, a "Git Scraper" tool is used to convert git commit objects into code repository files. The output of "Git Scraper" is used to generate ground truth file as well as the experiment file. Next phase, includes a suite of data pre-processing to prepare, normalize, and store the data in a common format to maximize the ability of the algorithm to produce high quality results. After the pre-processing of input files, individual clustering takes place based on file extension.

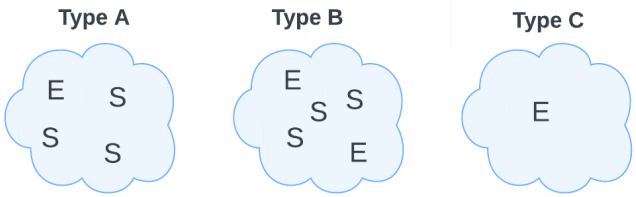

Figure 3. 1: Cluster types

There are three types of clusters [2] which get created – 'type A' clusters with an endpoint and its sources, 'type B' cluster with multiple endpoints and source files, and 'type C' cluster with only an endpoint as shown in Figure 3.1.

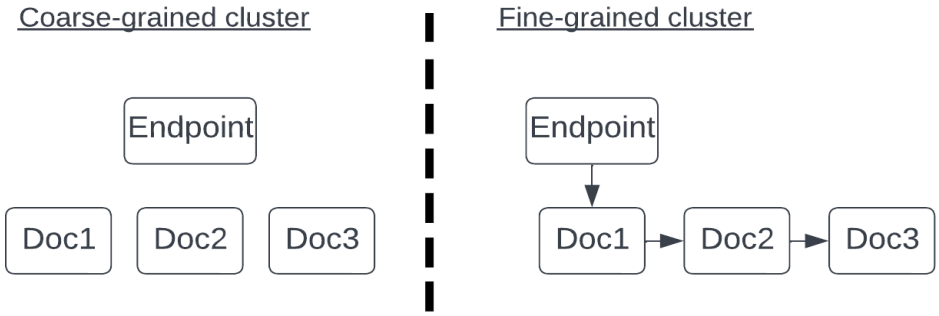

Figure 3. 2: Coarse-grained and Fine-grained cluster

There are two types of clusters that can be created: Coarse-grained cluster and fine-grained cluster. Coarse-grained cluster provide details of endpoint with number of sources related to it whereas fine-grained cluster include relationships between the endpoints and its sources as shown in Figure 3.2. In a cluster, there are multiple files which would have been created at different times. Provenance derivation in a cluster predicts versioning between the files. In our research, we plan to refer to coarse-grained cluster for a specific endpoint as “cluster provenance” and fine-grained cluster for a specific endpoint as “cluster derivation provenance”.

In the fourth phase, LDA performs topic modeling on the pre-processed data to convert text-based topics into machine understandable topic distributions (array of numbers). The topic distribution is used to calculate clusters of similar files using a pre-defined mathematical formula. This was improved in terms of accuracy by adding cosine similarly, which compares the files based on the semantic content. Thus, clustering uses topic distribution and cosine similarity to cluster similar files. The numerical value of similarity gets recorded in every input file.

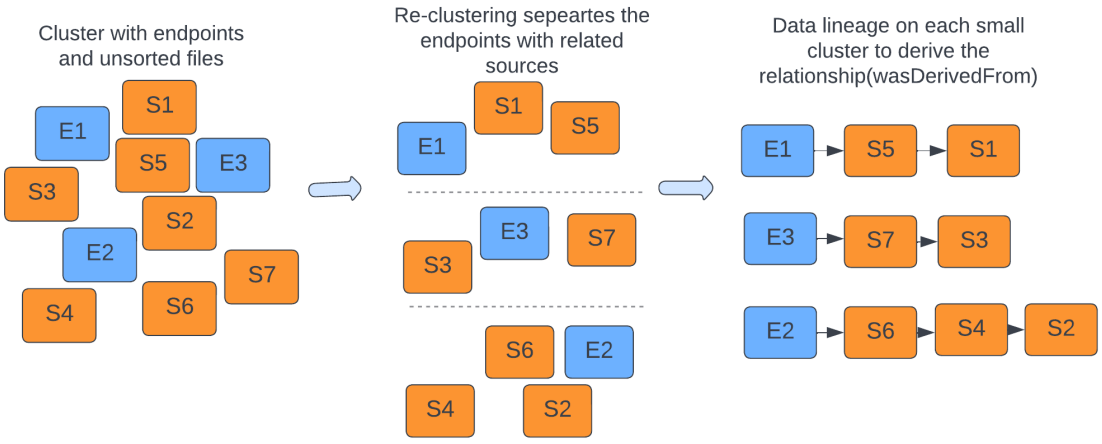

Figure 3. 3: Phases of LDA + Cosine similarity + Lineage algorithm

In the final stage, the lineage algorithm [9] uses similarly to link series of documents in the form of chain. Figure 3.3 shows the phases of cluster provenance and cluster derivation provenance by using OneSource.

3.2.1 Cluster provenance and Cluster derivation provenance

Clusters are created at multiple stages. OneSource is focused on deriving the provenance for ‘type A’ and ‘type B’ clusters because it follows the definition of provenance reconstruction. We are deriving the relationship ‘was Derived from’ in provenance reconstruction. ‘Type A’ and ‘type B’ clusters meet the requirement. LDA + Clustering helps in mapping the endpoint to its sources helping a user to identify related files within a cluster. The lineage algorithm helps in the deriving the versioning inside the cluster, which is important to get complete information about the provenance reconstruction. The various quality measurement processes include F1- score, precision and recall values of a cluster and their relationships. In this research, cluster fitness has been calculated by comparing cluster prediction with the ground truth. In this way, we plan to reconstruct both cluster provenance and cluster derivation provenance.

3.3 ARCHITECTURE

The goal of this project is to reconstruct provenance of machine-generated dataset such that any file can be traced back to the source file using text comparison algorithms of the datasets. Figure 3.4 provides a high-level design diagram for OneSource follows hybrid of Main program and sub-routines and Object oriented design(OOD) architecture; Figure 3.5 provides a detailed flow diagram follows UML activity architectural style.

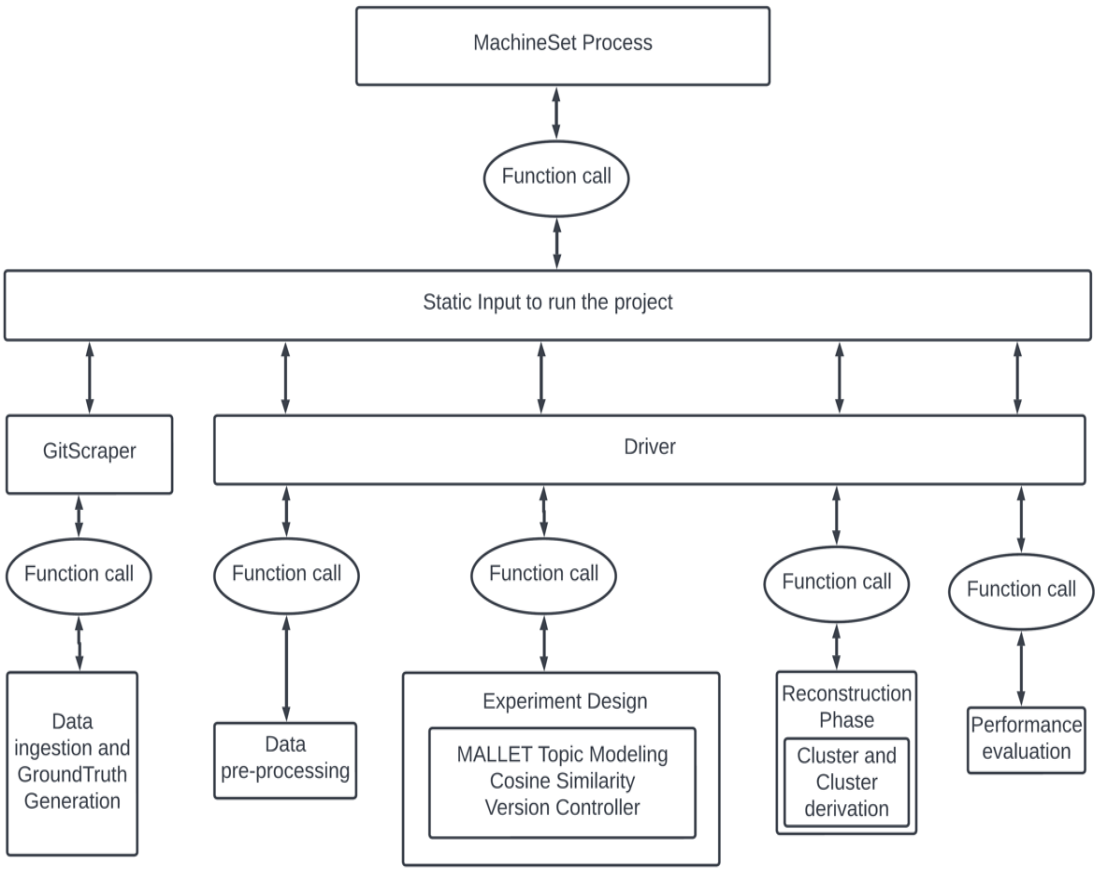

Figure 3.4: High-level design of proposed data provenance reconstruction workflow

We will now look at each stage of the project in Figure 3.5.

3.3.1 Static Input to Run the Project

OneSource is divided into two fundamental workflows: a) Option 1 and b) Option 2. In Option 1, we execute the provenance reconstruction algorithm, whereas in Option 2, we perform data ingestion and ground truth generation from a machine generated data source.

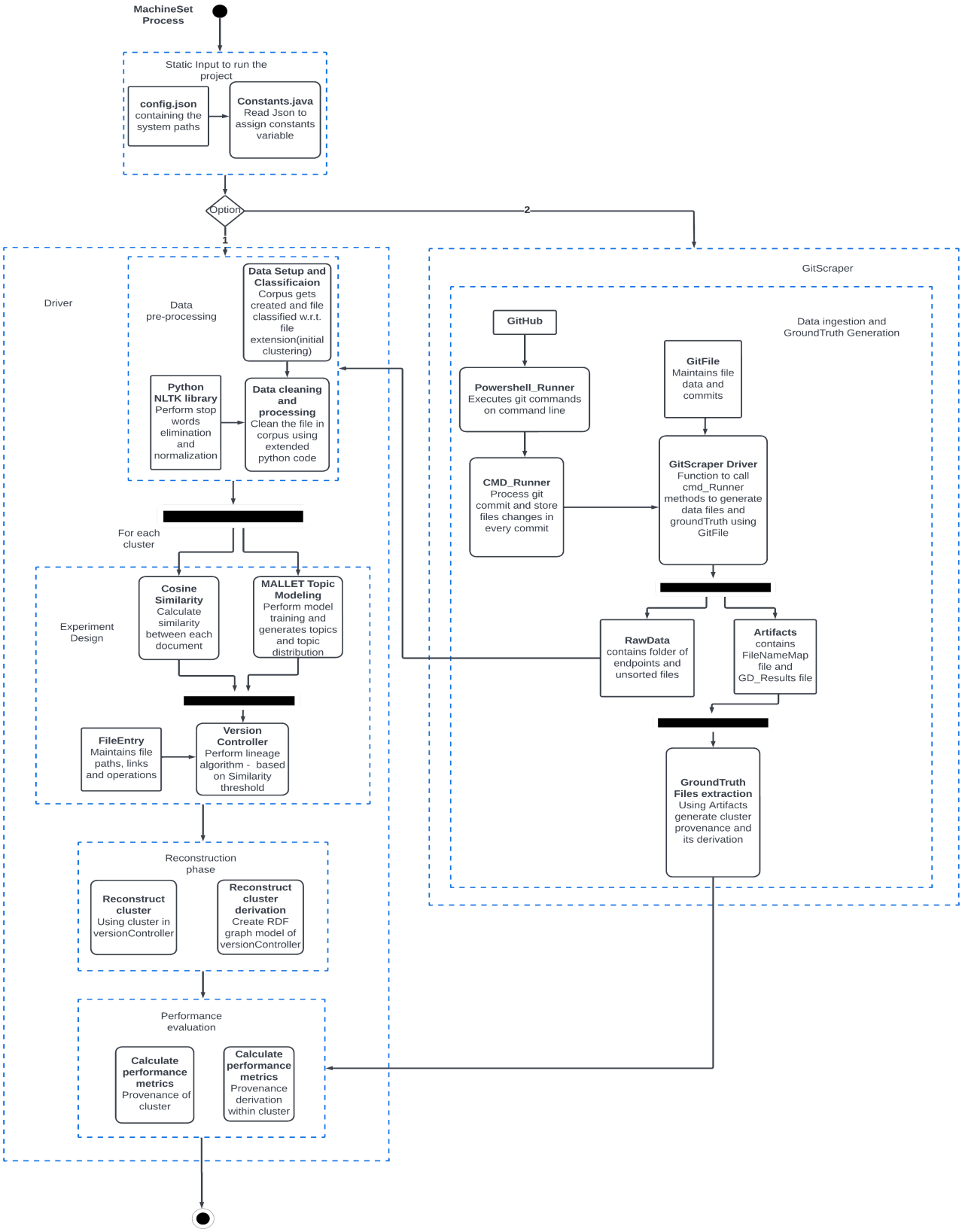

Figure 3.5: OneSource workflow diagram

Config.json

We update the static paths in "config.json" so that the user can change according to the requirements. The json file contains the data in key-value format.

Constants.java

Constants.java reads the configuration from the json file and create variables to be used for the entire process.

3.3.2 Data Ingestion and Ground Truth Generation

The datasets, which were used by previous teams, were not enough to improve the approach for machine-generated data because the ground truth files were not accurate and needed manual intervention. We automated the process of generating ground truth files and raw data needed for the model input.

GitHub

In this research, we use GitHub [41] as the machine-generated data source. We start the project by cloning GitHub repositories on our system.

Cmd_Runner

Cmd_Runner processes the git commit and enumerates file changes in every git commit. Git commits generated by using 'git rev-list—all' command. This command gives all the commit IDs associated to the repository. Git Enumerate function results in the file names which were previously changed in every commit ID. We use the command 'git reset – hard' + [the commit ID]

and followed by the command ‘git show –name-only –oneline’. This generates the files that were updated in the current branch from the previous branch.

Powershell_runner

The commands discussed above are executed on the command line using Powershell_runner.

GitFile

We maintain a git file, file path, its commits, random name generated using GitFile.

Git Scraper driver

Git Scraper driver uses these two functions to create input and generates artifacts, raw data, and the ground truth file.

Artifacts

An artifact folder consists of FileNameMap.txt and GD_results.txt files. FileNameMap is created by original file name, randomized file name, and file path. GD_ results is created by appending all variations of a files to a given file with random name generated.

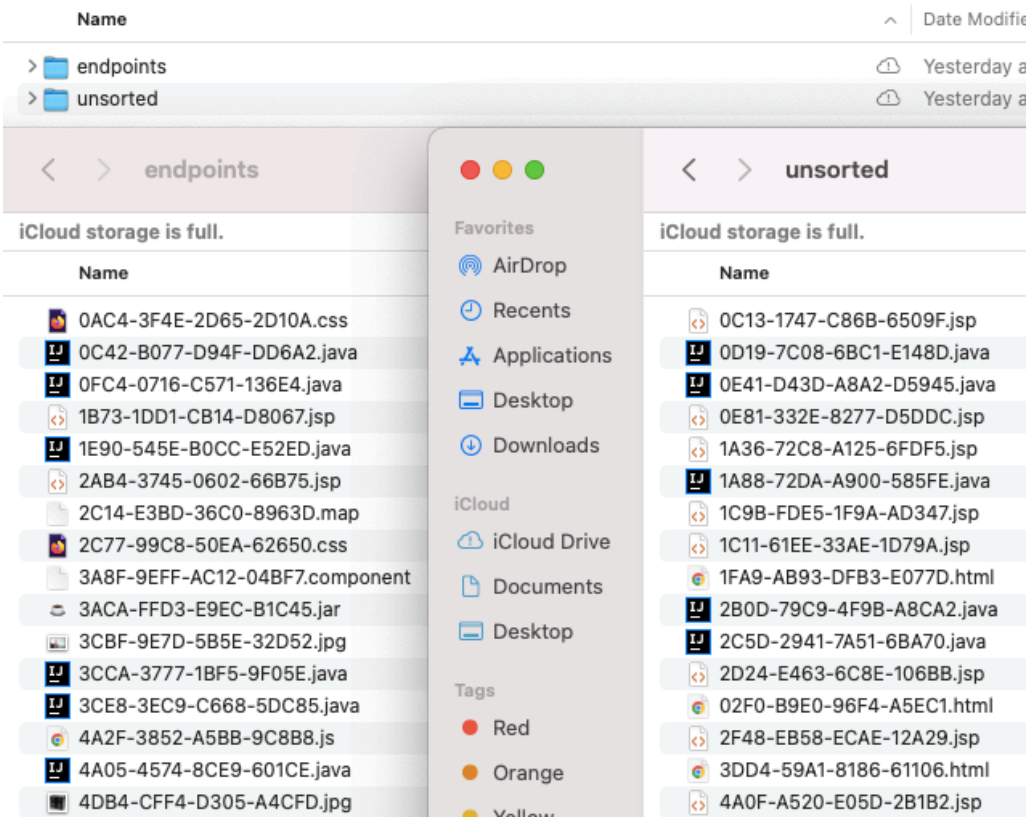

Figure 3.6: Rawdata folder - Input dataset

RawData

Figure 3.6 shows the raw data folder consists of two subfolders: endpoints and unsorted. The endpoint folder contains all the latest versions of different file extensions present in a Git repository. The unsorted folder contains all remaining files in the Git repository.

Figure 3.7: GroundTruth_1.txt cluster file

GroundTruth

There are two ground Truth files generated in this project.

GroundTruth_1 consists of clusters created for each file with its versioning files as shown in Figure 3.7. GroundTruth_1 is generated using file map generated in the process.

```
<FB80-7A61-44CB-41D64.java>
        <prov:wasDerivedFrom>  <A6C9-E25D-259A-6F144.java> .

<40CA-B5D5-4457-589F0.java>
        <prov:wasDerivedFrom>  <31FA-9CC0-B43C-F53DC.java> .

<DCD7-FA36-003F-C50D5.html>
        <prov:wasDerivedFrom>  <3DD4-59A1-8186-61106.html> .

<E53B-BED0-46BA-57067.jsp>
        <prov:wasDerivedFrom>  <CB5C-71CE-90AA-80857.jsp> .

<3DD4-59A1-8186-61106.html>
        <prov:wasDerivedFrom>  <05C5-A1E9-447A-0E30C.html> .

<6F70-151C-6D70-7EB72.java>
        <prov:wasDerivedFrom>  <FB80-7A61-44CB-41D64.java> .

<936B-6046-6552-484A8.java>
        <prov:wasDerivedFrom>  <B5B0-E9C3-F815-405CC.java> .

<4A05-4574-8CE9-601CE.java>
        <prov:wasDerivedFrom>  <0D19-7C08-6BC1-E148D.java> .

<AAC9-0B18-D505-8DEAD.html>
        <prov:wasDerivedFrom>  <6E1D-35D0-9FAC-FFBBC.html> .

<44BE-EC00-A67E-C23B5.css>
        <prov:wasDerivedFrom>  <F2B2-B8C6-C80A-A6EB0.css> .
```

Figure 3.8: GroundTruth_2.txt cluster derivation file

GroundTruth_2 consists of file derivation relationship from another file as shown in Figure 3.8. GroundTruth_2 file is generated using a GD_results.txt. The GroundTruth_2 file contains relationships between endpoints and unsorted files for sample datasets using the RDF graph model. These ground truth files are also used for performance evaluation of the model.

3.3.3 Data Pre-processing

Data setup and classification

In this, we create a data corpus folder which contains all the files from raw data folder. Corpus is a generic term which is used in natural-language processing for containing list of documents. In a

corpus folder, we classify the raw data files into different folders with files of the same extension going to a single folder. Each file in the folder is converted into a ".txt" file for pre-processing, as shown in Figure 3.9.

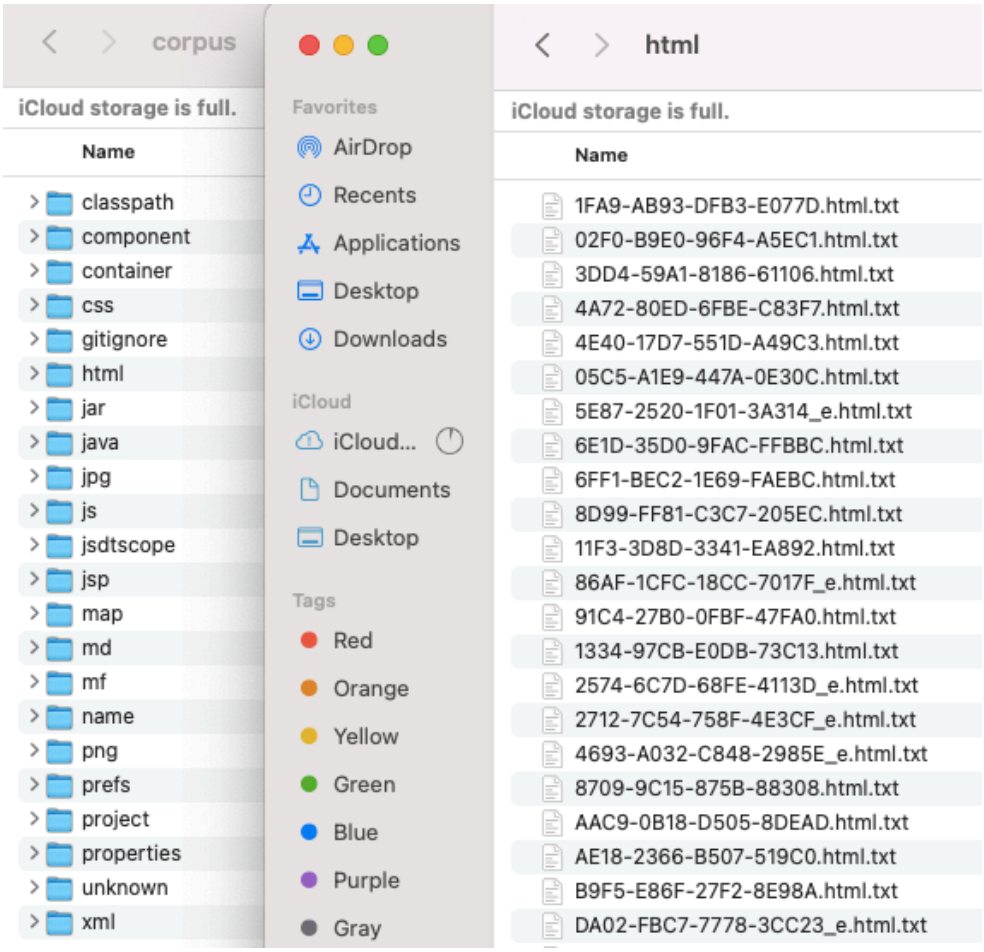

Figure 3.9: Corpus folder and html files

Data cleaning and processing

This section pre-process the files in the corpus folder using python NLTK [42] library in the corpus folder. These are following steps executed to pre-process the text data.

1. Tokenization - Splitting the sentences of words into individual tokens.

2. Removal of unwanted characters, numbers, and symbols - using regular expressions to remove any unwanted characters, numbers and symbols from the text.
3. Changing to lowercase - making the entire text into lower case.
4. Removal of punctuation - Using the NLTK library ‘punkt’ consist of the also punctuations. Removing the punctuation from the text.
5. Removal of stop words - Using the NLTK library, removing the stop words in English like the, not, have, also etc. from the text, so that more focus can be given to those words which constitute the decision making.
6. Removal of hashtags, URLs, Hypertext Markup Language (HTML) tags

The cleaning process ensures that all the data is purged from punctuations, extra spaces, numeric characters, and stop words, while also ensuring all the data is normalized in the same casing. We use process builder API [43] to call Python script from java source code.

3.3.4 Experiment Design

Mallet Topic Modeling

In 3.2.3, we converted the datasets into folders specific to each file extension type. These datasets are input to the MALLET tool. We performed MALLET topic modeling on the pre-processed datasets for each file extension folder. We provided the number of topics as the input square root of a list of words that are found in more than 10% of the documents. The default value of number of topics is taken as 10. The MALLET outputs in creation of topics and topic distribution. The topic distribution is used to calculate similarity between two documents.

Executing MALLET topic modeling on the dataset using ProcessBuilder API has the following steps.

1. Storing the corpus path

2. Importing documents

   The *import_data* function imports the training data into MALLET formatted data that can be used for training. Below is the command to import the data and outputs ".mallet" file with the preprocessing.

   Import command:

   "./bin/mallet import-dir --input corpuspath --output machine.mallet --keep-sequence --remove-stopwords"

3. Building topic model

   The *train_topic_model* function trains an LDA topic model using MALLET. It outputs into topics generated (see in Figure 3.10); topic distribution (see in Figure 3.11). A topic is a probability distribution over the n topics for each document.

   Build command:

   ".bin/mallet train-topics --input machine.mallet --num-topics noOfTopics --num-iterations noOfIterations --output-doc-topics corpusTopics.txt"

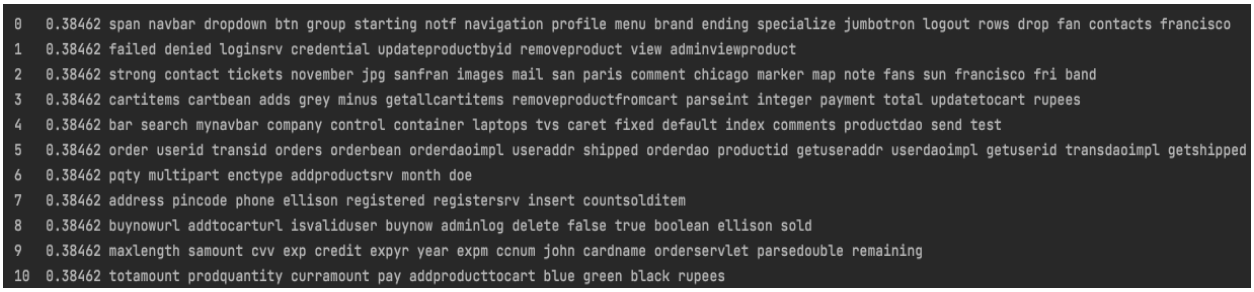

Figure 3.10: Topics keys

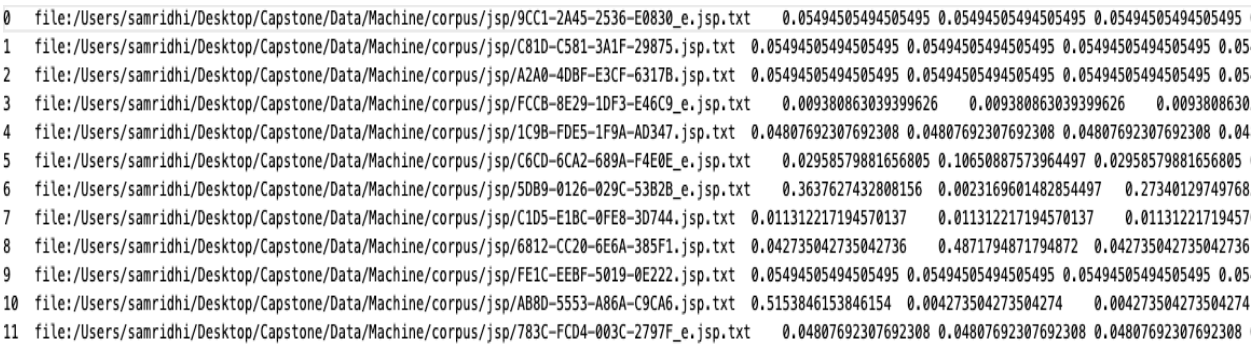

Figure 3.11: Topics distribution

Cosine Similarity

Cosine similarity is calculated for each document in the corpus with other documents. We calculate the similarity between two documents using word frequency vector and distinct keywords. Word frequency vector is the frequency of every word in a document. Distinct keywords is a list of words which are distinct in both the documents. We calculate three vectors: 1) Vector A of document 1, 2) Vector B of document 2, and 3) Vector AB of document 1 and document 2. Vector A is the summation of squared frequency of words in document 1, Vector B is the summation of the squared frequency of words in document 2, and Vector C is the summation of product of frequency in document 1 and document 2. After calculating the vectors, we use Cosine similarity formula to get a similarity score using Vector A, Vector B, and Vector AB as inputs.

Version Controller

Every time Mallet topic modeling and Cosine similarity is performed, Version controller executes three functions: 1) Gather Relationship, 2) Build Relationship, and 3) Update Clusters.

Gather Relationship

Gather relationship uses both the inputs from Mallet topic modeling and Cosine similarity to gather relationships for any file. A file entry is used to store file information. A file entry consists of file name, file extension, file path, related files, links to forward and backward file. A file entry is used to keep track of the updated information of a file. After storing the files as file entry, each file is compared to every other file based on the similarity score calculated in 3.2.4. If the similarity score is greater than the similarity threshold, the current file entry and other file entry are updated to its related documents with the score in the descending order. The current file entry also updates its forward and backward link.

Build relationship

To build the relationship between the files, a lineage algorithm is applied. We compare and sort all the endpoints based on highest similarity score. We start from the endpoints to the source file using the backward line of the file entry. An endpoint cannot be a source to any file. If any file is already associated to a file, it checks for the next possible file. When the backward link is null, the lineage algorithm stops for that endpoint.

Update clusters

This is executed for each endpoint. We update the endpoints with their derivate relationship and outputs the versionController file. Each line of this file represents a cluster. In a cluster, endpoint may have a source or not have a source.

3.3.5 Reconstruction Phase

A reconstruction is defined as transformation of the model output compatible with ground Truth file format. The reconstruction is done in two parts: cluster and cluster derivation reconstruction.

Cluster reconstruction

A mapping of clusters is reconstructed from the output of the model (see in Figure 3.12). A cluster is a group of related files without maintaining the order of its sources. It is then compared with the groundTruth_1 file in the evaluation phase known as cluster provenance.

```
570A-C6B8-093C-D8507_e.java.txt
936B-6046-6552-484A8_e.java.txt
7E34-3176-3189-420A9_e.java.txt
8B50-5C48-0CA5-717AF_e.java.txt,855E-3721-408F-4B932.java.txt,6F70-151C-6D70-7EB72.java.txt,FB80-7A61-44CB-41D64.java.txt,A6C9-E25D-259A-6F144.java.txt
6BA6-0FBB-2B98-4F8B5_e.java.txt
347D-11D5-F3AD-A7B1D_e.java.txt
3920-1827-7CA9-F9D41_e.java.txt,CC97-5F88-0C02-D043E.java.txt
1E90-545E-B0CC-E52ED_e.java.txt
E1C8-9B43-5B2D-BEDA6_e.java.txt
0FC4-0716-C571-136E4_e.java.txt,560F-2B08-DDB1-B7AEA.java.txt,C25A-17FB-DDD8-BC71D.java.txt,B7EA-DFC4-4484-95728.java.txt,B5B0-E9C3-F815-405CC.java.txt,
8C5C-2517-BA08-72482_e.java.txt,152E-FD2F-4A12-7266A.java.txt,347E-E0E6-C96C-A6A92.java.txt,9937-6E02-1D1E-11B83.java.txt,D770-004A-A239-3106D.java.txt,
3CE8-3EC9-C668-5DC85_e.java.txt,587C-5F50-E9B9-8F516.java.txt,FAB1-4B86-3B28-FB859.java.txt,B5D2-9F00-596C-C84D1.java.txt
070C-02FB-ED34-05E2A_e.java.txt
6F9C-9C94-9B85-AE2CC_e.java.txt
5E87-2520-1F01-3A314_e.html.txt
DD88-946E-4C4D-FFA7D_e.html.txt
4693-A032-C848-2985E_e.html.txt
DA02-FBC7-7778-3CC23_e.html.txt
2712-7C54-758F-4E3CF_e.html.txt,DCD7-FA36-003F-C50D5.html.txt,91C4-27B0-0FBF-47FA0.html.txt,1334-97CB-E0DB-73C13.html.txt,B9F5-E86F-27F2-8E98A.html.txt,
86AF-1CFC-18CC-7017F_e.html.txt,AAC9-0B18-D505-8DEAD.html.txt,6FF1-BEC2-1E69-FAEBC.html.txt,6E1D-35D0-9FAC-FFBBC.html.txt,AE18-2366-B507-519C0.html.txt,
2574-6C7D-68FE-4113D_e.html.txt
E2C8-2C18-789C-2FD68_e.png.txt
CBE4-4F21-597C-DD187_e.png.txt
B736-BD9D-7928-86A20_e.jpg.txt
B7EB-FE41-E998-9D9B7_e.jpg.txt
```

Figure 3.12: Sample output of cluster file

Cluster derivation reconstruction

A mapping of the output is transformed in the RDF graph format, as shown in Figure 3.13. An RDF graph model is easier to retrieve using a query string. An entity which does not have derivation is stored as ‘prov: Entity’ as shown in Figure 3.14.

```
"SELECT ?target ?source " +
      "WHERE {" +
      "       ?target <prov:wasDerivedFrom> ?source . " +
      "       }"
```

Figure 3.13: An example of query string

Figure 3.13 shows that a source is mapped to a target file where the target file has a provenance relationship of ‘wasDerivedFrom’ with the source file. This is known as cluster derivation

provenance where each file relationship is derived within a cluster. This mapping will be compared with the groundTruth_2 file in the performance evaluation section.

```
<9B71-F8F5-93FD-C25E1.jsp.txt>
        <http://www.w3.org/2000/01/rdf-schema#type>
                <prov:Entity> .

<6E1D-35D0-9FAC-FFBBC.html>
        <prov:wasDerivedFrom>  <AE18-2366-B507-519C0.html> .

<877E-2F8E-4D3F-42190.java.txt>
        <http://www.w3.org/2000/01/rdf-schema#type>
                <prov:Entity> .

<34A6-2D00-C837-6BFCA.xml.txt>
        <http://www.w3.org/2000/01/rdf-schema#type>
                <prov:Entity> .

<1E90-545E-B0CC-E52ED.java.txt>
        <http://www.w3.org/2000/01/rdf-schema#type>
                <prov:Entity> .

<91C4-27B0-0FBF-47FA0.html>
        <prov:wasDerivedFrom>  <1334-97CB-E0DB-73C13.html> .

<CB5C-71CE-90AA-80857.jsp>
        <prov:wasDerivedFrom>  <54B0-D85E-D179-D1DE7.jsp> .

<3801-7256-0A7D-3D571.jpg.txt>
        <http://www.w3.org/2000/01/rdf-schema#type>
                <prov:Entity> .

<E2C8-2C18-789C-2FD68.png.txt>
        <http://www.w3.org/2000/01/rdf-schema#type>
                <prov:Entity> .
```

Figure 3.14: Sample output of cluster derivation file

3.3.6 Performance Evaluation

In this phase, the output of the reconstruction phase is compared with the ground Truth files to generate model quality metrics and demonstrate the proof points of the approach, using the following criteria: precision, recall and, F1-score. This section will be discussed in detail in chapter 4.

CHAPTER 4 RESULTS

In this section, we discuss the test results of our solution. We first present the metrics used for evaluation and the test scenarios. We then present the results and compare them based on accuracy and performance among the different provenance reconstruction techniques, and against the dataset provided.

4.1 METRICS

We have developed a model which predicts cluster provenance and cluster derivation provenance on the same datasets. Cluster provenance groups similar files while cluster derivation provenance establishes a relationship between files within a cluster. Thus, our model generates two output results and respective ground truth files (demonstrated in figure 3.7 and 3.8). To determine the accuracy of our model, we validate our results for both outputs by generating respective precision and recall values when comparing them with their ground truth.

4.1.1 Metrics for cluster provenance

We validate cluster provenance by comparing it with the ground_Truth1.txt file. The formula to calculate precision and recall used in [2] for a cluster is derived from Wikipedia [44].

Precision = | {relevant documents} ∩ {retrieved documents} / | {retrieved documents} |

i.e., *Precision = True Positive/ (True Positive + False Positive)*

Recall = | {relevant documents} ∩ {retrieved documents} / | {relevant documents} |

i.e., Recall = True Positive/ (True Positive + False Negative)

Equation 4.2: Precision and Recall cluster provenance

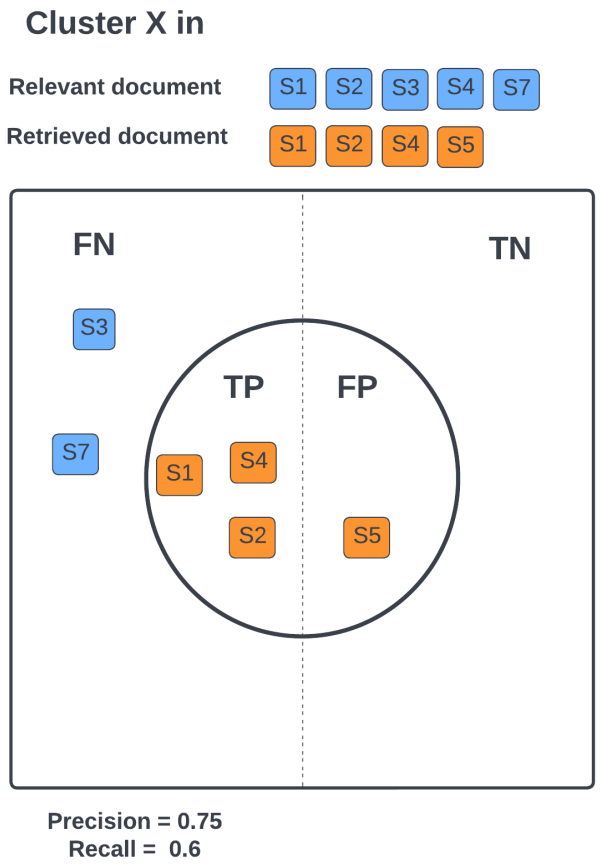

Figure 4. 1: Sample example of P&R calculation for a cluster

Precision is defined as the number of relevant documents out of all the documents retrieved. Recall is the number of relevant retrieved documents out of all the relevant documents. Figure 4.1 shows an example of the calculation of precision and recall for a cluster. As in Figure 3.12, each row in the file is a cluster with endpoints and sources in comma-separated format. The precision and recall value are calculated by comparing each cluster in the output with the corresponding cluster in the ground truth file as shown in Figure 3.7. The overall precision and recall of the output are found by calculating the average precision and average recall of all the clusters [2].

To represent both precision and recall in a single value, we calculate the F1 score. F-Score is the weighted mean of precision and recall. F1 score is a type of F measure where the weights of precision and recall are equal.

*F1-Score = 2 * (Precision * Recall) / (Precision + Recall)*

Equation 4.2: F1 score for cluster provenance

4.1.2 Metrics for cluster derivation provenance

We validate cluster derivation provenance by comparing with the ground_Truth2.txt file. We use precision and recall values of a cluster as

Precision = True Positive Derivation/ (True Positive Derivation + False Positive Derivation)

Recall = True Positive Derivation/ (True Positive Derivation + False Negative Derivation)

Equation 4.3: Precision and recall for cluster derivation provenance

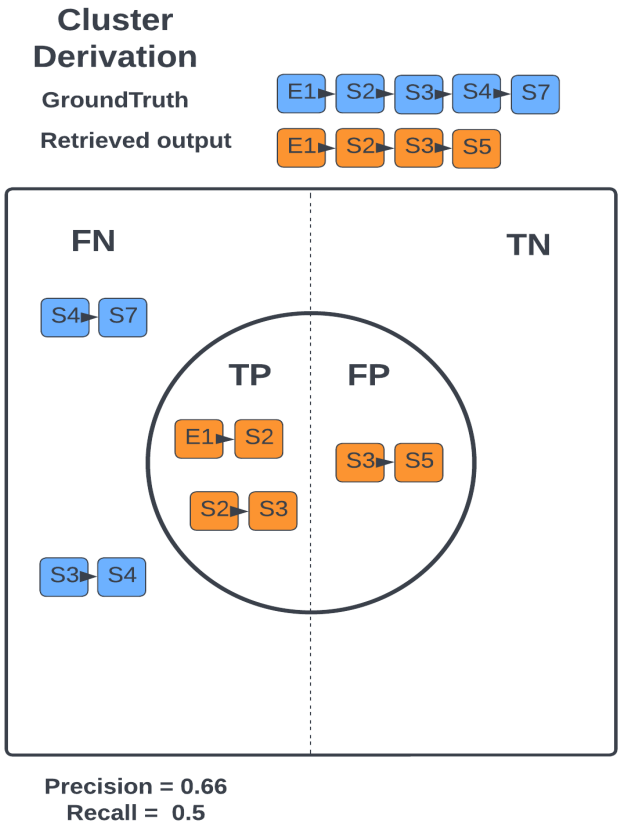

Figure 4. 2: Sample example of P&R calculation for a cluster derivation

Similar to cluster provenance, we need to have a single metric to represent the ratio of precision and recall values which is their harmonic mean as follows

*F1-Score = 2 * (Precision * Recall) / (Precision + Recall)*

Equation 4.4: F1 score for cluster derivation

4.2 EVALUATION

Both LDA and Cosine similarity attempt to extract semantics from documents. Their precision can depend on specific characteristics of documents such as text length and word frequency. Hence, we test our model's robustness with different combinations of input parameters to provide a more balanced comparison using six different dataset samples. We have conducted our experiment on nine different machine generated datasets ranging from 33 to 5894 (6K) files extracted from git repositories to test OneSource's model accuracy. Each dataset sample was executed at least three times for different input parameters and the average of the three was calculated for the results. The datasets for our research are software projects from GitHub. The selection of datasets is based on number of files, size, and number of commits in software project.

Table 4. 1: Input parameters

Input parameters	Type	Value
Clustering method	Internal configuration	(LDA+ Cosine), Cosine
Similarity threshold	Internal configuration	0.8, 0.85 and 0.9
Number of files	External configuration	33 to 5894
Size of dataset	External configuration	0.04 MB to 200 MB

Number of commits	External configuration	16 to 971
No of topics for LDA	Internal configuration	Depends on file data (Default is 10)

Table 4.1 shows the number of input parameters to the model, type of parameters such as internal or external, and values of parameters. We evaluated results below 0.8 ST and could not find high accuracy. Hence, our minimum threshold for similarity was defined as 0.8. We created two additional thresholds at an interval of 0.05 for exhaustive testing scenario. For LDA, the number of topics is the input square root of list of words that are found in more than 10% of the documents. The default value of number of topics is taken as 10. Therefore, each time, LDA generates different number of topics for each file type. The number of topics depends on the file and is automatically generated by the model for all the test scenarios.

The evaluation section is divided into three parts: 1) OneSource accuracy, 2) OneSource accuracy Vs SeekPeek accuracy, and 3) Execution time. For each sub-section, the results are shown in two parts: cluster provenance and cluster derivation provenance. We have used the actual name of the git repositories to represent the dataset. Table 4.2 shows the dataset used in OneSource with number of files, size of dataset and number of commits.

Table 4. 2: Dataset table with attributes

Name of dataset	No of commits	Size (MB)	Number of files
QtNotepad	16	0.04	33
Bluetooth-Home-Automation	28	4.8	76
Shopping cart	18	7.7	223
bachelor_degree_thesis	157	17.8	662

android-open-project	954	95.8	817
Answer	560	137.6	1870
hash-checker	321	103.9	3404
front-end-interview-handbook	608	201.1	5872
tech-interview-handbook	971	185.2	5894

4.2.1 OneSource Accuracy

We have evaluated OneSource accuracy on the input parameters discussed in section 4.2.

4.2.1.1 Clustering method: LDA vs LDA-Cosine vs Cosine

We decided to experiment clustering methods comparing (LDA), (LDA-Cosine) and (Cosine). In (LDA-Cosine) method, LDA clusters files based on the topic distribution and Cosine clusters files based on text similarity. In combining (LDA-Cosine), we used the minimum value of similarity score between them to compare with the similarity threshold. The reason to use LDA with Cosine is low performance of using LDA standalone. On the other hand, we also experimented clustering with Cosine similarity and LDA alone. To make a balanced comparison between the methods, we have used similarity threshold value as 0.85 for both cluster provenance and cluster derivation provenance. The datasets are arranged in the increasing order of the number of files in it. In this section, we are separately comparing precision and recall for both clustering methods because precision and recall trend for clustering methods are different. We then compare F1-scores for both clustering to obtain a performance conclusion on the clustering method.

Comparison between Precision of LDA vs LDA-Cosine vs Cosine

Cluster Provenance

In Figure 4.3, we have plotted the precision for different datasets used in OneSource for (LDA), (LDA-Cosine) and (Cosine) methods for cluster provenance.

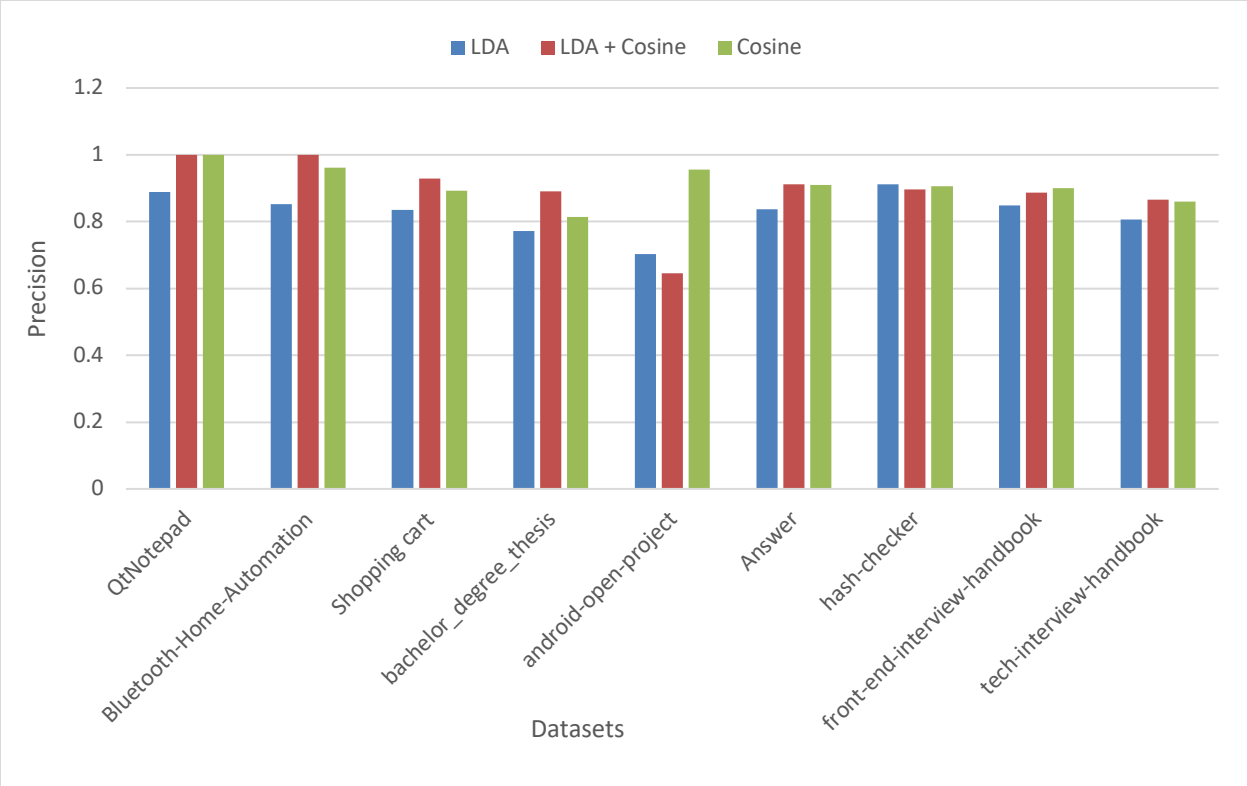

Figure 4. 3: Cluster Provenance: Comparing Precision of (LDA), (LDA+ Cosine), (Cosine) for different dataset samples

Discussion: From the graph, the precision for each dataset shows a similar value in (LDA-Cosine) and (Cosine) as compared to (LDA) for cluster provenance except for android-open-project dataset. In case of (LDA) and (LDA-Cosine) for android-open-project dataset, the low precision means the model also captures unrelated files to the endpoints in a cluster due to the similarity score produced by (LDA) and (LDA-Cosine).

Cluster Derivation Provenance

In Figure 4.4, we have plotted the precision for different datasets used in OneSource for (LDA), (LDA-Cosine) and (Cosine) methods for cluster derivation provenance.

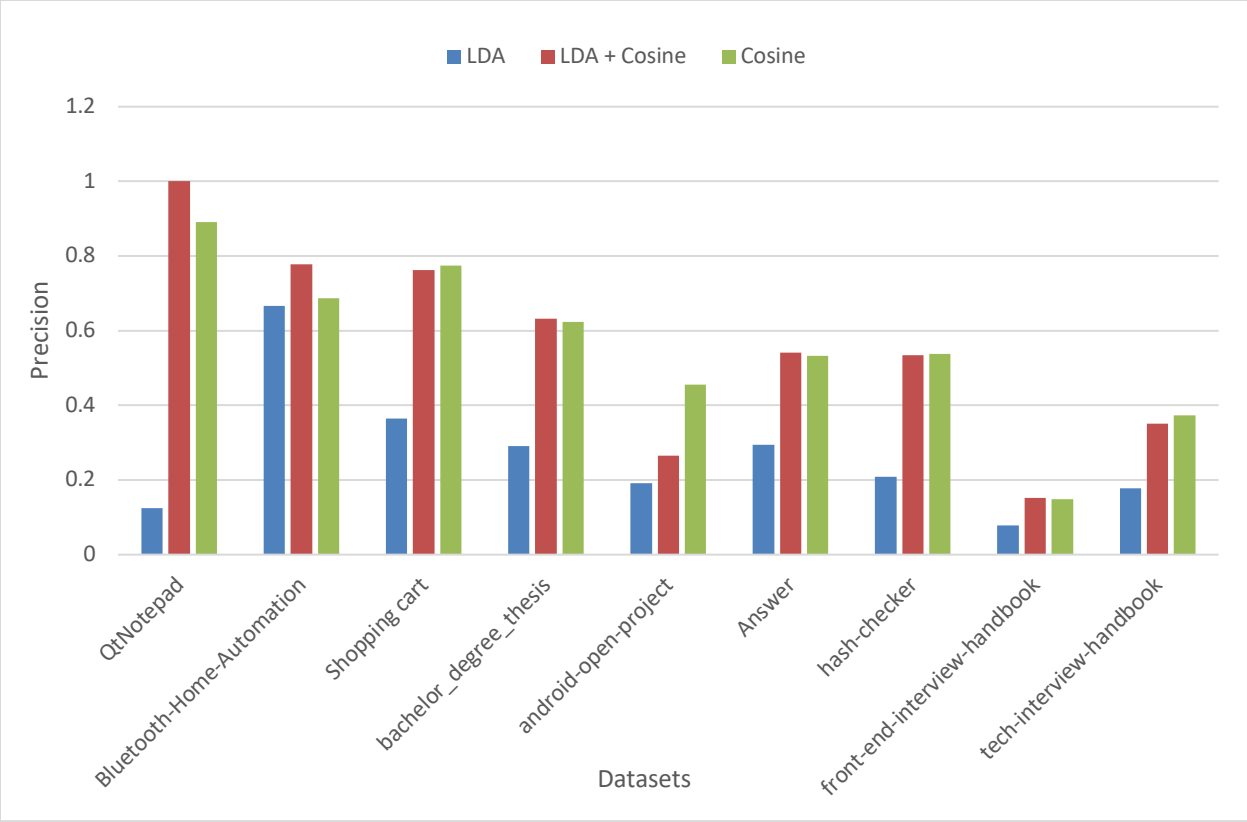

Figure 4. 4: Cluster Derivation Provenance: Comparing Precision of (LDA), (LDA+ Cosine), (Cosine) for different dataset samples

Discussion: From the graph, the precision for each dataset shows a similar value in (LDA-Cosine) and (Cosine) as compared to (LDA) for cluster derivation provenance except for android open project dataset. The performance of (LDA) is relatively low in all the datasets for cluster derivation, which means that the relationship derived between files is not correct even if the cluster provenance is correct. This is because the similarity score produced by (LDA) based on topic distribution is not able to derive correct relationships. In case of (LDA-Cosine) for android-open-project dataset, the low precision means the model derives incorrect relationships between files. The cluster derivation provenance depends on how well the clusters are formed i.e., cluster provenance. It is also seen from figure 4.3 and 4.4 that the precision value has decreased for cluster derivation provenance as compared to the cluster provenance across datasets because model

capturing correct relationships between the files within a cluster is low even if the cluster is well formed. To capture correct relationships, the lineage algorithm needs to be improved.

Comparison between Recall of LDA vs LDA-Cosine vs Cosine

Cluster Provenance

In Figure 4.5, we have plotted the recall for different datasets used in OneSource for (LDA), (LDA-Cosine) and (Cosine) methods for cluster provenance.

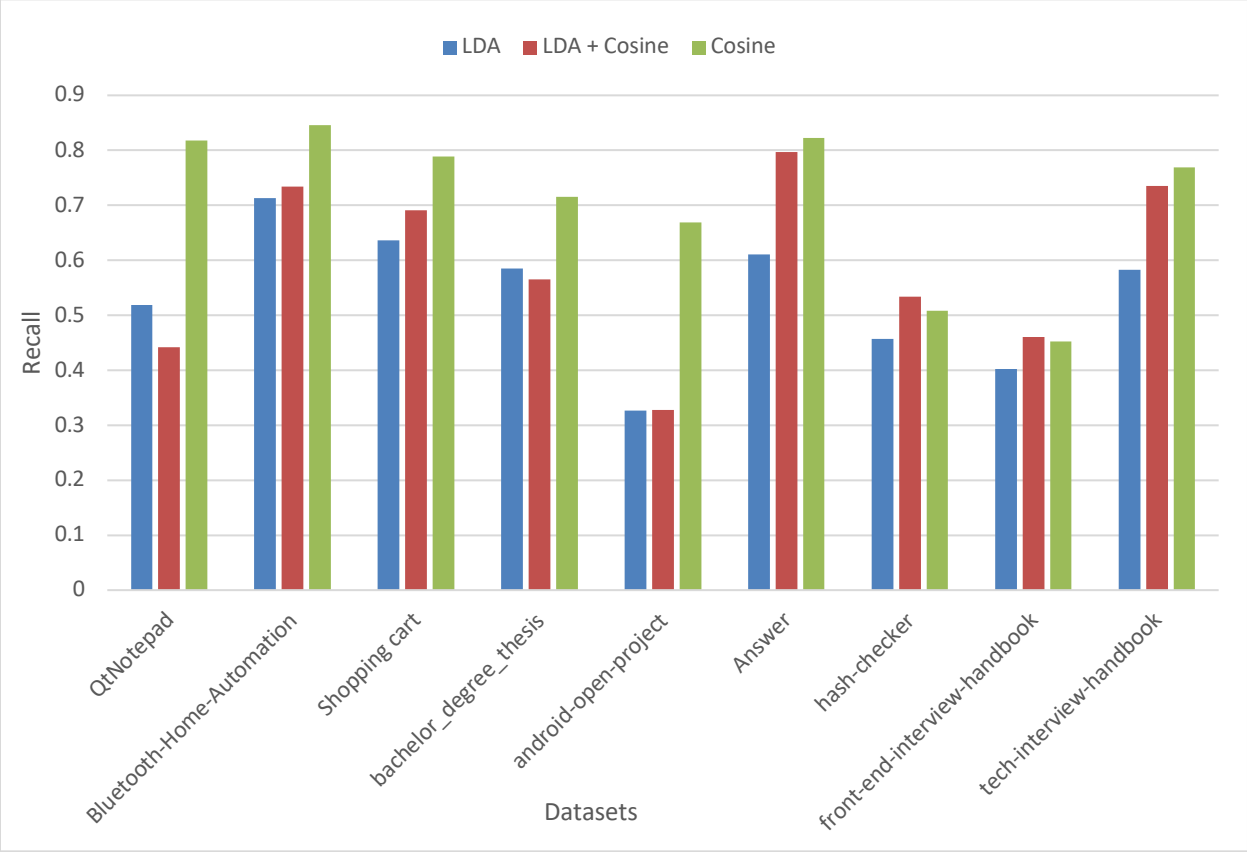

Figure 4. 5: Cluster Provenance: Comparing Recall of (LDA), (LDA+ Cosine), (Cosine) for different dataset samples

Discussion: From the graph, the recall for (Cosine) method performs better than (LDA) and (LDA-Cosine) methods for each dataset for cluster provenance. The performance of (LDA) is even lower than (LDA-Cosine). In case of cluster provenance, the recall value is low for (LDA) and (LDA-

Cosine) method, because false negative is high. This is because the model misses to capture the files with similarity score produced by (LDA) and (LDA-Cosine) method as compared to (Cosine) method. For example, (LDA) and (LDA-Cosine) method produces similarity of 0.6 between two files and (Cosine) methods produces similarity of 0.86 between the same two files, and similarity threshold is 0.85. The relationship between the two files will be captured in case of (Cosine) method but gets missed in case of (LDA) and (LDA-Cosine) method. This will impact the F1-scores discussed in the next section.

Cluster Derivation Provenance

In Figure 4.6, we have plotted the recall for different datasets used in OneSource for (LDA), (LDA-Cosine) and (Cosine) methods for cluster derivation provenance.

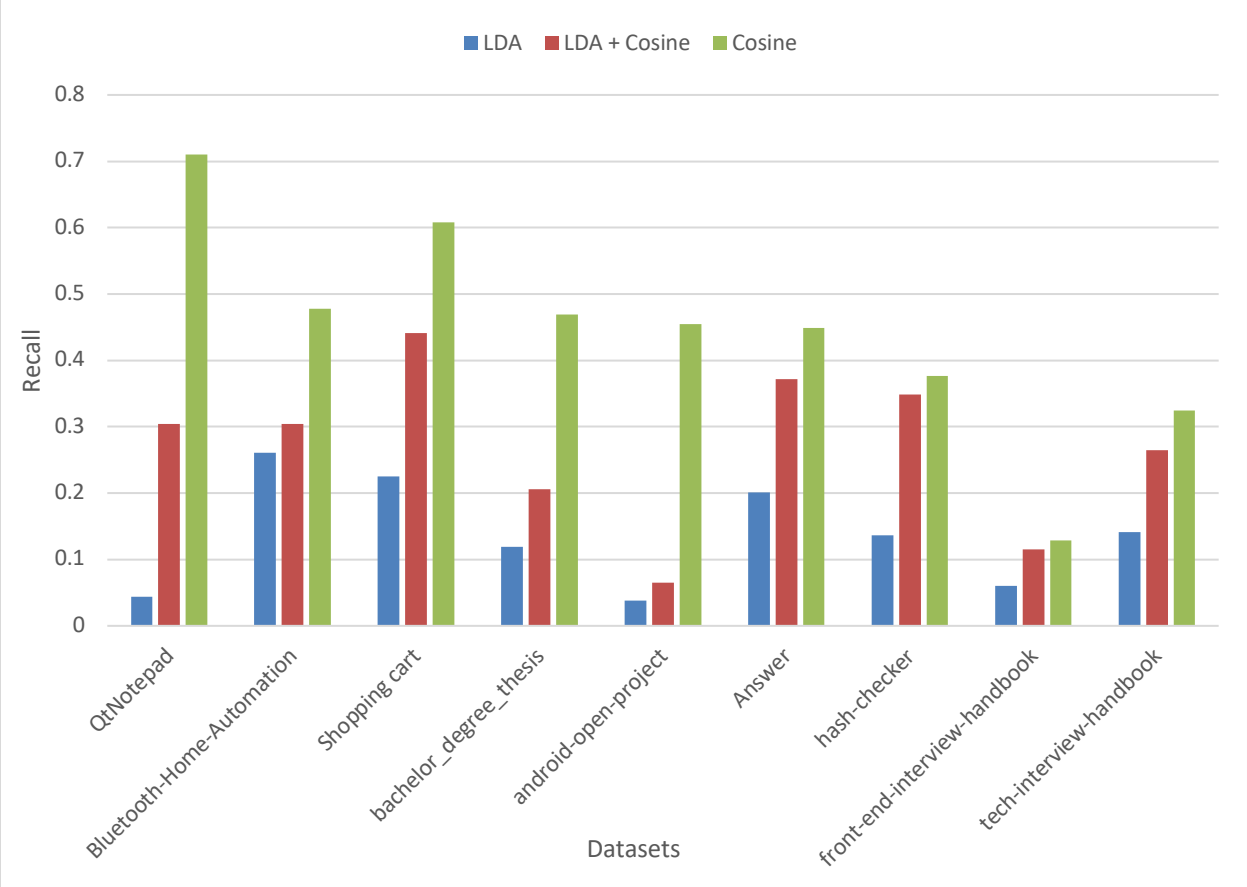

Figure 4. 6: Cluster Derivation Provenance: Comparing Recall of (LDA), (LDA+ Cosine), (Cosine) for different dataset samples

Discussion: In case of cluster derivation provenance, the recall value is much lower for (LDA) and (LDA-Cosine) method than (Cosine) method, because false negative is high as the model misses to capture the files with similarity score produced by (LDA) and (LDA-Cosine) method compared to (Cosine) method. For example, (LDA) and (LDA-Cosine) method produces similarity of 0.6 between two files and (Cosine) methods produces similarity of 0.86 between the same two files, and similarity threshold is 0.85. The relationship between the two files will be captured in case of (Cosine) method but gets missed in case of (LDA) and (LDA-Cosine) method. This will impact the F1-scores discussed in the next section. It is also seen from figure 4.5 and 4.6 that the recall value has decreased for cluster derivation provenance as compared to the cluster provenance across datasets because model capturing correct relationships between the files within a cluster is low even if the cluster is well formed. To capture correct relationships, the lineage algorithm needs to be improved. The performance of (LDA) is relatively low in all the datasets for cluster derivation compared to (LDA-Cosine) because similarity score produced by (LDA) based on topic distribution is not able to capture all the relevant relationships.

Comparison between F1-scores of LDA vs LDA-Cosine vs Cosine

Cluster Provenance

In Figure 4.7, we have plotted the F1-scores for different datasets used in OneSource for (LDA), (LDA-Cosine) and (Cosine) methods for cluster provenance.

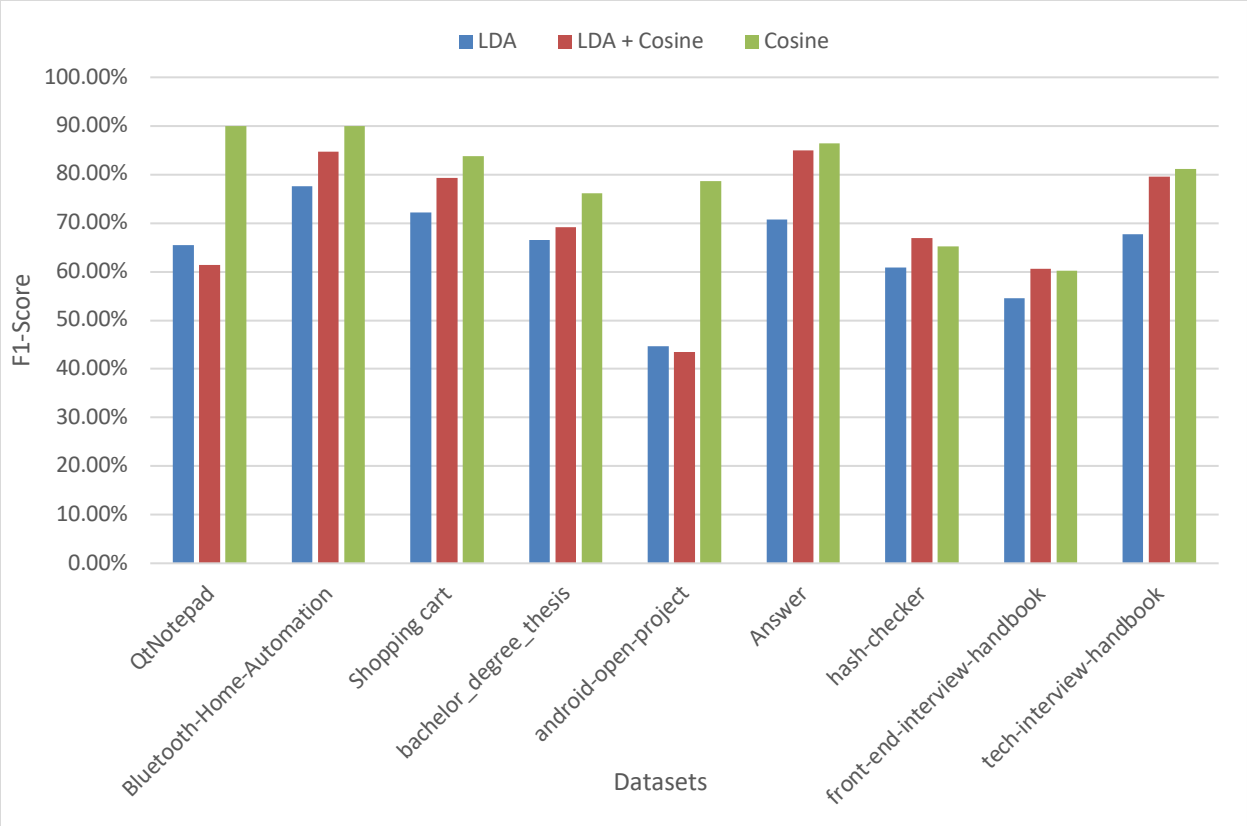

Figure 4. 7: Cluster Provenance: Comparing F1-scores of (LDA), (LDA+ Cosine), (Cosine) for different dataset samples

Discussion: From the graph, the F1-scores for (Cosine) method performs better than (LDA) and (LDA-Cosine) method for each dataset for cluster provenance. The F1-score is based on the precision and recall value discussed above.

Cluster Derivation Provenance

In Figure 4.8, we have plotted the F1-scores for different datasets used in OneSource for (LDA), (LDA-Cosine) and (Cosine) methods for cluster derivation provenance.

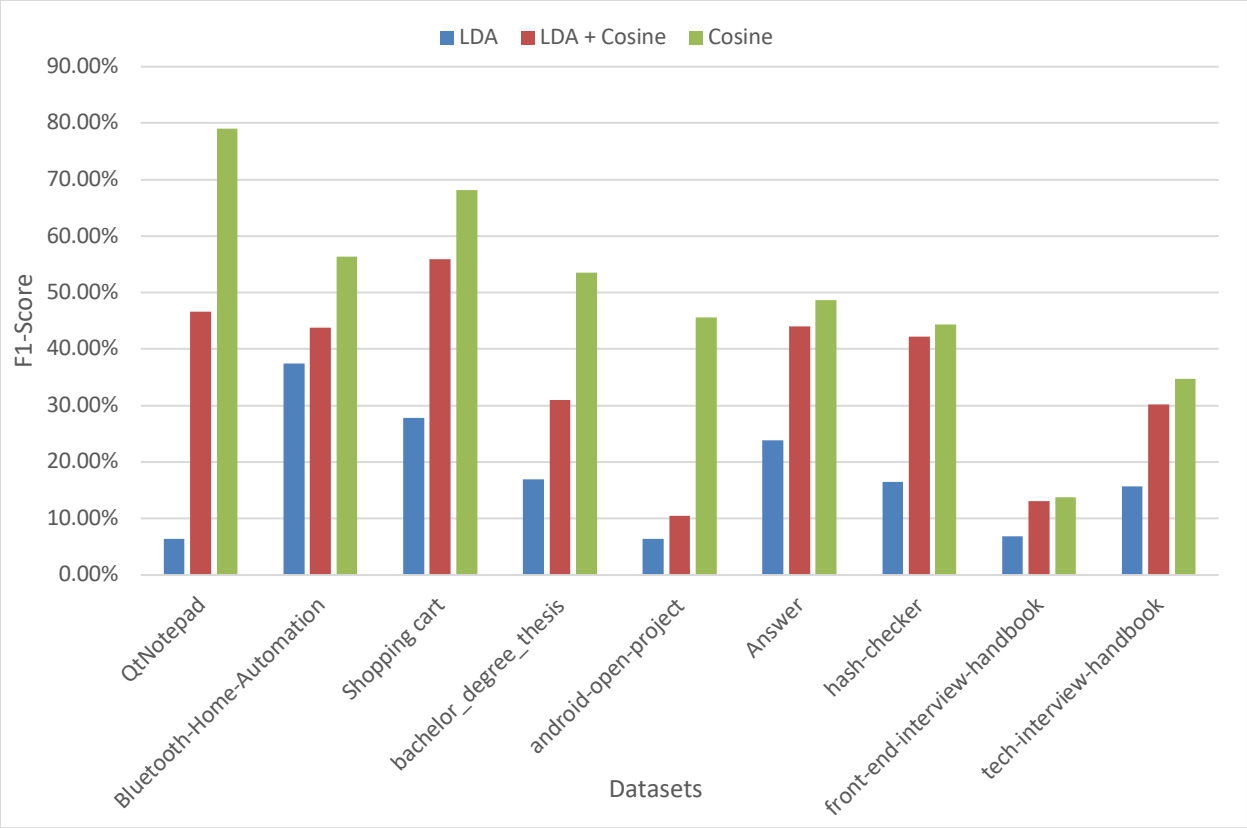

Figure 4. 8: Cluster Derivation Provenance: Comparing F1-scores of (LDA), (LDA+ Cosine), (Cosine) for different dataset samples

Discussion: From the graph, the F1-scores for (LDA) and (LDA-Cosine) method performs bad as compared to (Cosine) method for each dataset for cluster derivation provenance. The F1-score is based on the precision and recall value discussed above.

From the above section on clustering methods, we conclude that clustering method Cosine accuracy is better as compared to both the methods (LDA) and (LDA-Cosine). In next two sections, we will discuss other parameters on Cosine alone.

4.2.1.2 Similarity threshold

We decided to test the accuracy of our model based on similarity threshold for clustering method as Cosine. Each similarity threshold (ST) applied on the datasets is depicted with a different color:

0.8 (blue), 0.85 (red), and 0.90 (green). The datasets are arranged in the increasing order of the number of files in it.

Cluster Provenance

In Figure 4.9, we have plotted the F1 scores for different datasets used in OneSource under different levels of similarity threshold for cluster provenance and data shown in Table 4.2.

Table 4. 3: Cluster Provenance: Comparing F1 Scores Vs Similarity Scores across datasets

Datasets	**ST- 0.8**	**ST- 0.85**	**ST- 0.9**
QtNotepad	90.67%	89.95%	89.95%
Bluetooth-Home-Automation	90.75%	90.01%	85.24%
Shopping cart	80.08%	83.79%	85.44%
bachelor_degree_thesis	75.85%	76.13%	76.46%
android-open-project	49.61%	78.70%	78.70%
Answer	84.35%	86.45%	86.67%
hash-checker	63.63%	65.15%	66.87%
front-end-interview-handbook	59.40%	60.18%	60.08%
tech-interview-handbook	79.96%	81.13%	81.58%

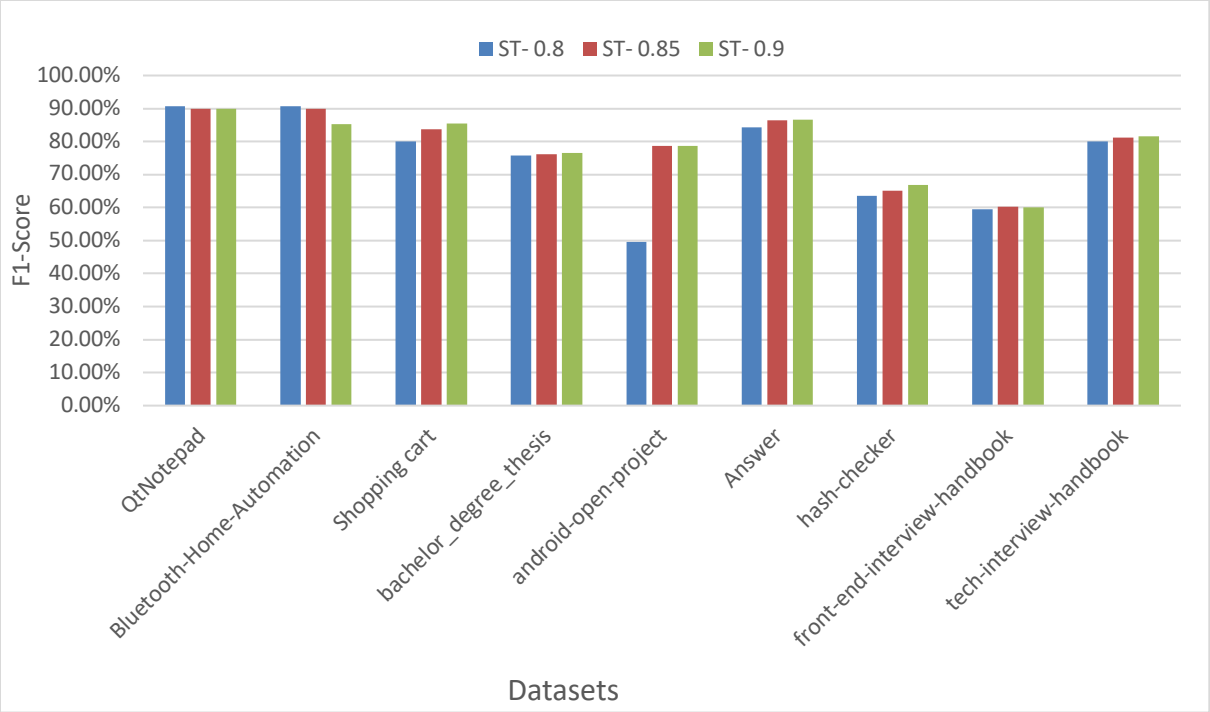

Figure 4. 9: Cluster Provenance: Comparing F1 Scores Vs Similarity Scores across datasets

Discussion: For each dataset, the F1-score increases slightly with the increase in the similarity threshold because at higher similarity threshold, the precision and recall values increase, thereby increasing the F1-score which is the harmonic mean of precision and recall values. In contrast, the Bluetooth-Home-Automation dataset shows a decline in the F1-score as similarity threshold increases. This is an outlier in our experimentation results as we observed a strong correlation between similarity threshold and F1-score across all other datasets. The reason for this outlier is that the recall value decrease from 86% to 76% for similarity scores 0.85 and 0.90, respectively, as false negatives increase (because the model is optimizing for only highly similar clusters and misses less similar files) thereby reducing overall F1 scores. It is also seen from figure 4.9 that android-open-project has lowest F1-score for similarity threshold 0.8 compared to 0.85 and 0.9, because files in the dataset are highly similar to each other which leads to model capturing unwanted files in the cluster.

Cluster Derivation Provenance

Figure 4.10 shows the F1 scores for different datasets used in OneSource under different levels of similarity threshold for cluster derivation provenance and data shown in Table 4.3.

Table 4. 4: Cluster Derivation Provenance: Comparing F1 Scores Vs Similarity Scores across datasets

Datasets	**ST- 0.8**	**ST- 0.85**	**ST- 0.9**
QtNotepad	80.95%	79.01%	79.01%
Bluetooth-Home-Automation	59.99%	56.41%	58.82%
Shopping cart	63.58%	68.13%	74.31%
bachelor_degree_thesis	53.14%	53.56%	48.52%
android-open-project	45.21%	45.55%	45.55%
Answer	47.75%	48.70%	48.36%
hash-checker	44.72%	44.32%	45.06%
front-end-interview-handbook	14.02%	13.80%	13.68%
tech-interview-handbook	34.91%	34.72%	34.19%

Discussion: The F1-scores across similarity thresholds show a similar value for most of the datasets. The reason for this observation is that the cluster derivation precision and recall values peak at 0.8 similarity score and plateau as the similarity increases. Hence, a relationship within a cluster derivation is already created at 0.8 similarity score and cannot be further improved at higher scores.

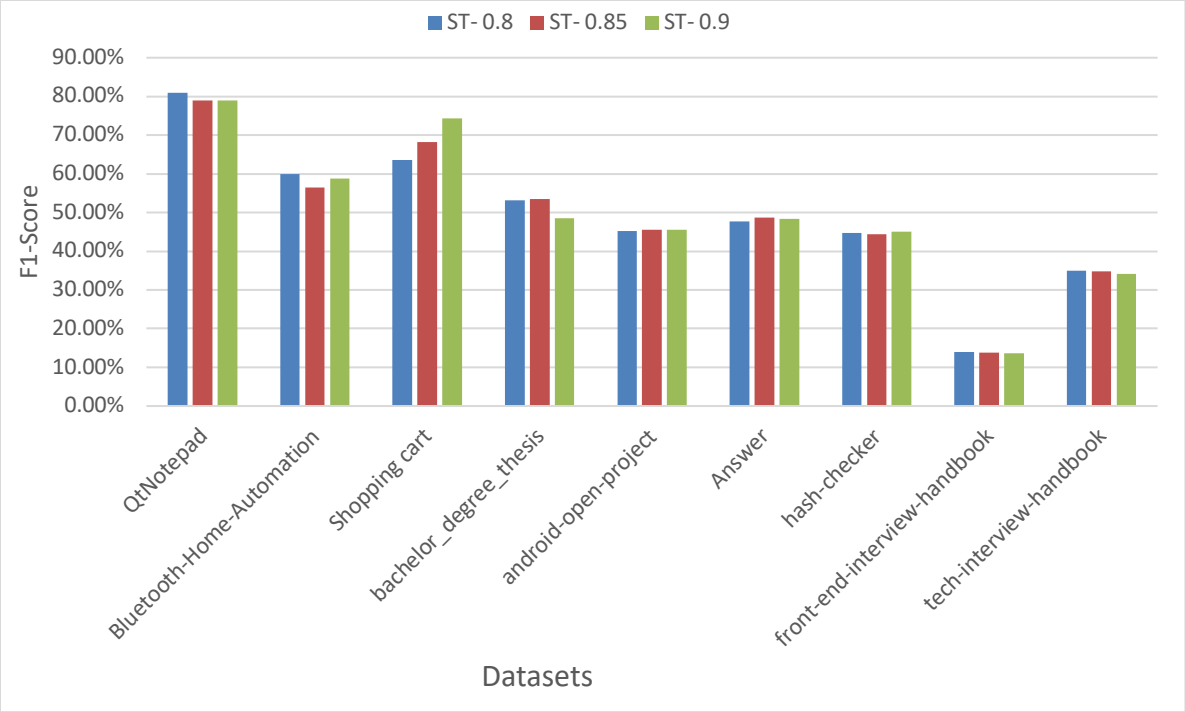

Figure 4. 10: Cluster Derivation Provenance: Comparing F1 Scores Vs Similarity Scores across datasets

Correlation coefficient

The correlation coefficient (CC) is defined as the covariance divided by the product of the two variables standard deviations [45]. Correlation coefficient is used to find how strong a relationship is between two datasets. CC returns a value between -1 and 1, where 1 indicates a strong positive relationship, -1 indicates a strong negative relationship, and 0 indicates no relationship at all. In order to understand the relationship between number of files and F1-Scores, we calculated CC between them. The results of the coefficient were in the range of 0 to -0.8 showing no correlation to negative correlation thereby establishing that the F1 scores are not dependent on the number of files in the datasets.

4.2.1.3 P&R Trends with number of files, size, and number of commits in datasets

To conduct deeper analysis of how the F1-score changes with changes in parameters, we analyze the precision and recall values for different datasets used in OneSource under different levels of similarity threshold. The clustering method is Cosine. The cluster provenance is depicted with number of files and size of dataset. The cluster derivation provenance is depicted with number of files and number of commits in the dataset.

Cluster Provenance

Figure 4.11 shows the maximum precision for cluster provenance is 100% for QtNotepad dataset of 33 files for all three-similarity threshold and minimum precision for cluster provenance is 67.75% for Android-open-project dataset of 817 files for similarity threshold of 0.80. We can see that the model outputs similar results for similarity thresholds of 0.85 and 0.90 as compared to 0.80.

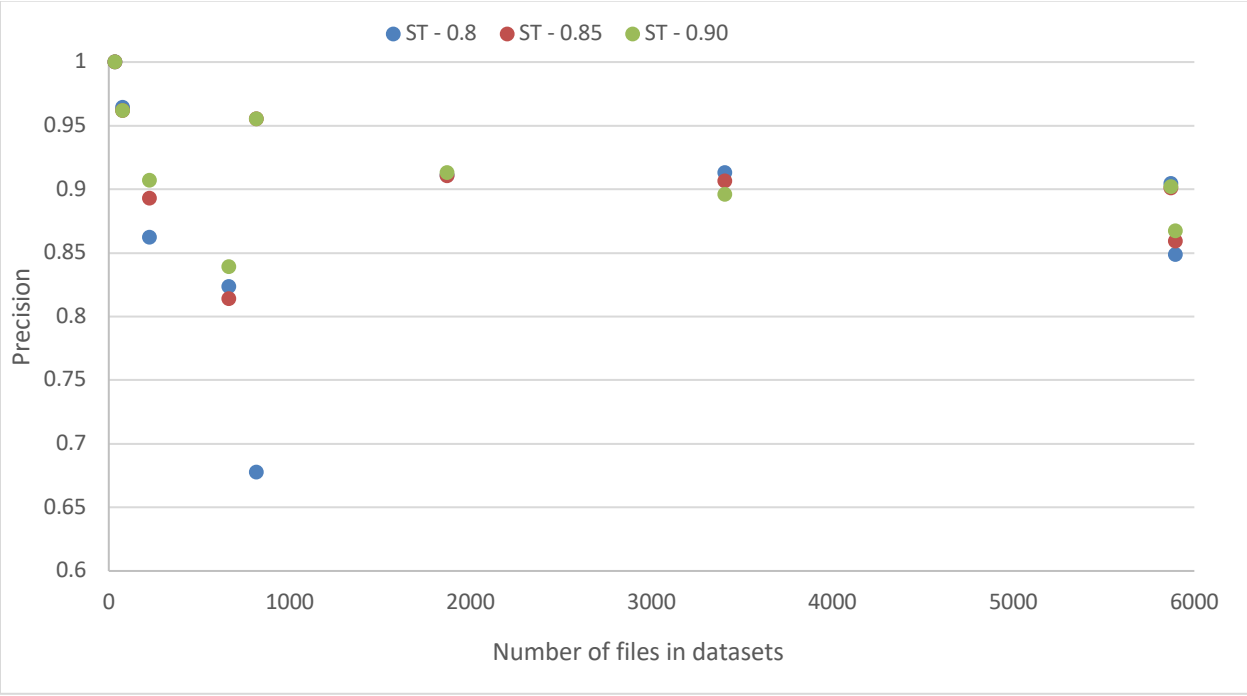

Figure 4. 11: Cluster Provenance: Precision with different Similarity threshold for different dataset number of files samples

Discussion: Overall precision value decreases with the increase in the number of files in the dataset for all similarity threshold values. The model cluster prediction precision decreases between 0 to 1000 number of files followed by an increase for higher number of files. The precision decreases for dataset post 6000 files.

The precision value also depends on the size of the file in the dataset shown in Figure 4.12, as well as the number of files in the dataset. The x-axis in Figure 4.11 and Figure 4.12 represents datasets with number of files and size respectively. The size of the dataset does not depend on the number of files in a dataset. For an example, Answer dataset has size 135MB and 600 files has higher precision than tech interview handbook dataset with size 185MB and has 5894 number of files. From both the graphs, the android-open-project dataset shows a decline in precision for similarity threshold 0.8 with respect to 0.85 and 0.9, because specific to this dataset model also captures files which are not related to the cluster in case of cluster provenance. Overall, as the size increases, the precision of clusters decreases.

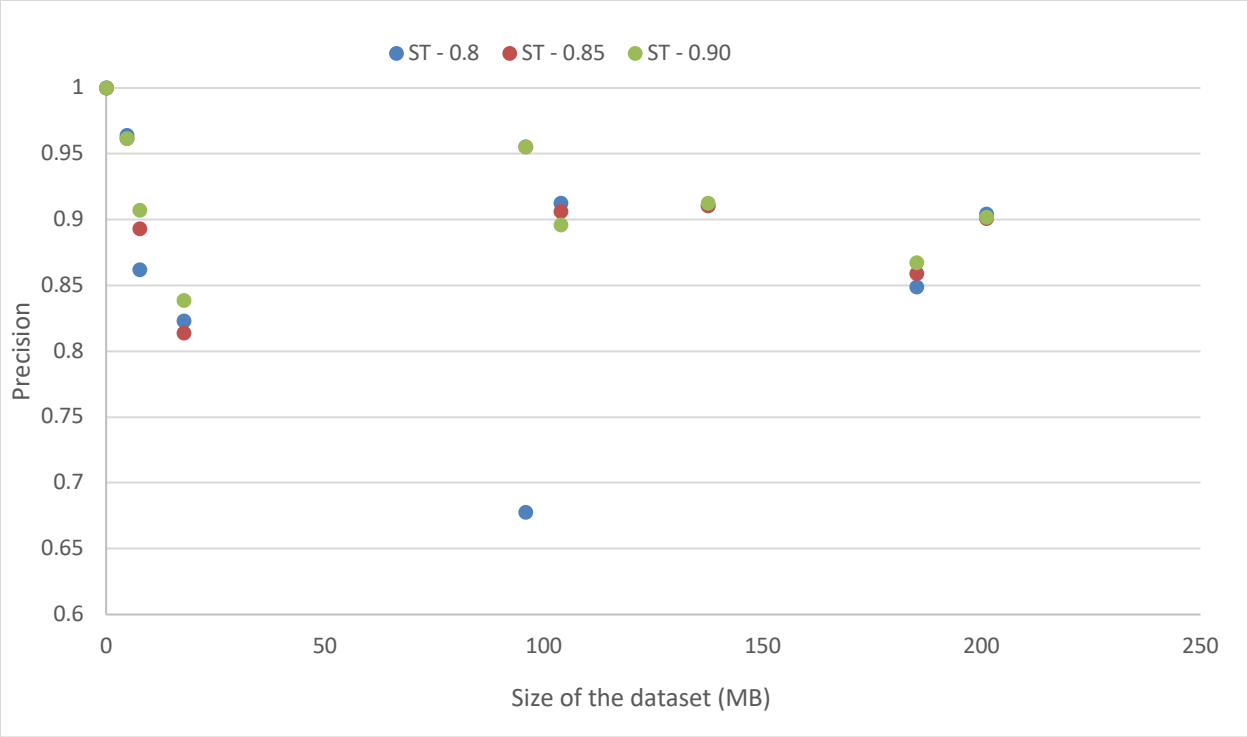

Figure 4. 12: Cluster Provenance: Precision with different Similarity threshold for different dataset size samples

Figure 4.13 shows the maximum recall for cluster provenance is 85.71% for the Bluetooth home automation dataset of 76 files at similarity threshold of 0.80 and minimum recall for cluster provenance is 67.75% for Android-open-project dataset of 817 files for similarity threshold of 0.80. We see that the model outputs approximately similar trend for all the similarity threshold across datasets.

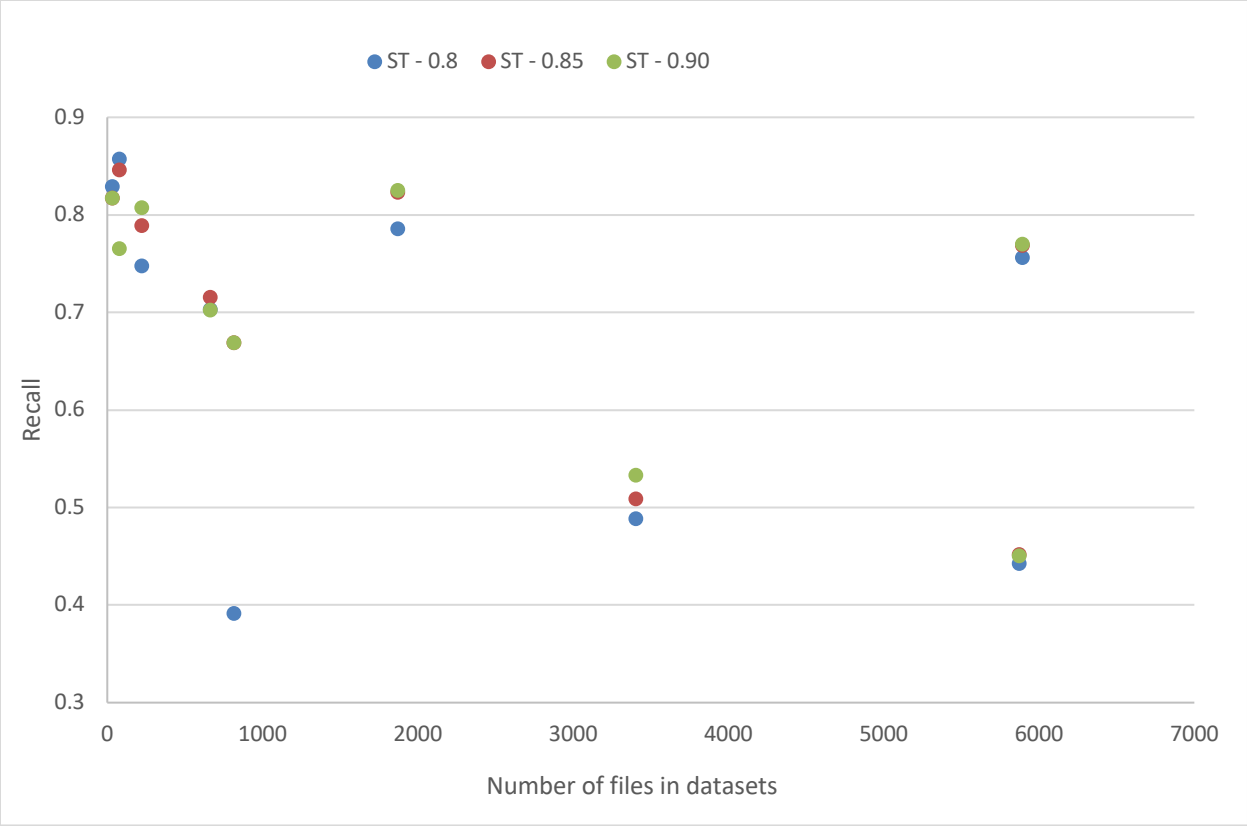

Figure 4. 13: Cluster Provenance: Recall with different Similarity threshold for different dataset number of files samples

Discussion: Recall is the measure of how many retrieved documents are relevant with respect to the number of relevant documents. A high recall value shows that model was able to capture all the relevant clusters. A low recall value shows few clusters were missed by the model. The recall value decreases between 0 to 1000 and then increases as the number of files increase. Further, it slightly decreases at the dataset with 6000 files. The recall trend is very similar to precision. The recall value also shows variation with respect to size of the dataset (see in Figure 4.14). As the size increases, the possibility of missing clusters also increases thereby decreasing recall value. From both the graphs, the android-open-project dataset shows a decline in recall for similarity threshold

0.8 with respect to 0.85 and 0.9, because files in the dataset are highly like each other which leads to model capturing unwanted files in the cluster.

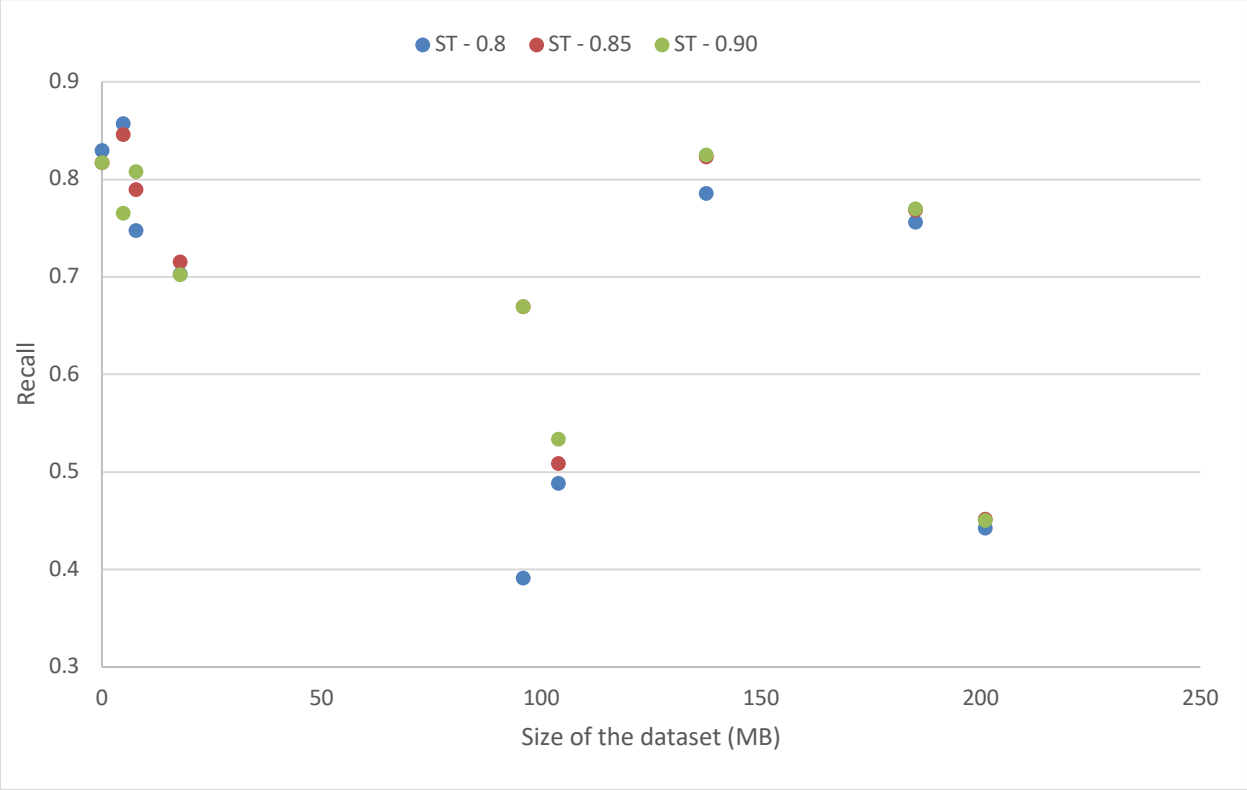

Figure 4. 14: Cluster Provenance: Recall with different Similarity threshold for different dataset size samples

Cluster Derivation Provenance

Figure 4.15 shows the maximum precision for cluster derivation provenance is 90.9% for Bluetooth Home automation dataset of 76 files for similarity threshold of 0.90. The minimum precision for cluster derivation provenance is 14.92% for Front-end-interview-handbook dataset of 5872 number of files for similarity threshold of 0.85. We see that the model outputs similar results for all three similarity thresholds of 0.80, 0.85 and 0.90.

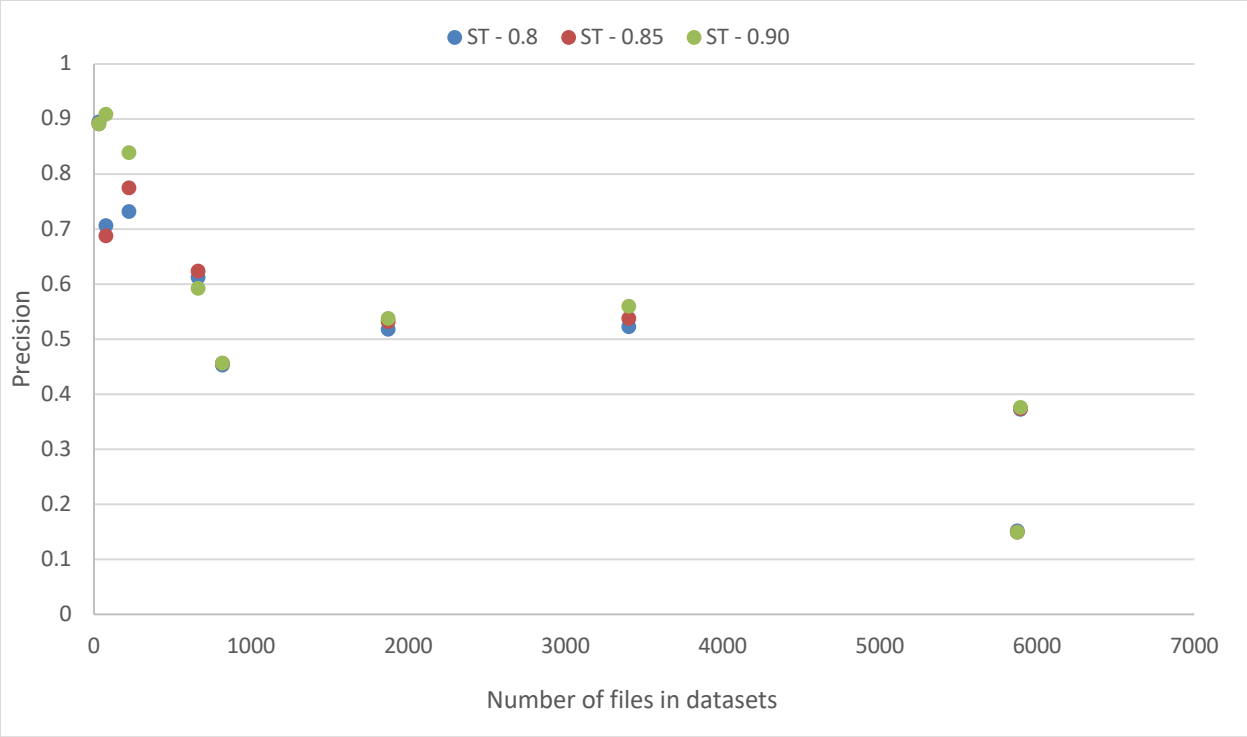

Figure 4. 15: Cluster Derivation Provenance: Precision with different Similarity threshold for different dataset number of files samples

Discussion: Overall precision value decreases with the increase in the number of files in the dataset for all similarity threshold values. The model is not able to capture the right versions of a file in a cluster for large number of files in dataset.

The precision value of cluster derivation also depends on the number of commits to the file in the dataset shown in Figure 4.16, as well as the number of files in the dataset. We have plotted another graph showing precision for all similarity threshold for number of commits to the datasets. The x-axis in Figure 4.15 and Figure 4.16 represents datasets with number of files and number of commits respectively. The number of commits is directly proportional to the number of derivations of a file in a dataset. When the number of commits to the dataset increases, the model precision decreases. Front-end-interview-handbook dataset of 5872 files with number of commits 608 has lower

precision at 14.92% as compared to Android-open-project dataset of 817 files with number of commits 954 with precision of 45.61%. From the graph, we can conclude that number of commits to the dataset and the number of files in a dataset are independent with respect to precision.

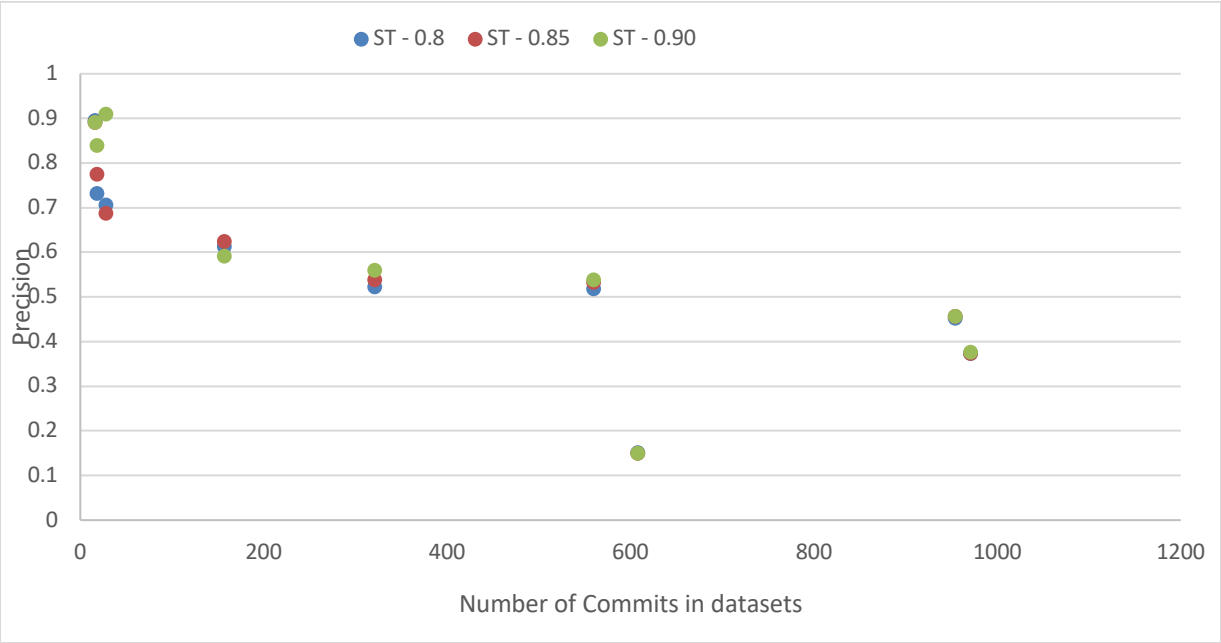

Figure 4. 16: Cluster Derivation Provenance: Precision with different Similarity threshold for different dataset number of commits samples

Figure 4.17 shows the maximum recall for cluster derivation provenance is 73.91% for QtNotepad dataset of 33 files for similarity threshold at 0.8. The minimum recall for cluster derivation provenance is 12.61% for Front-end-interview-handbook dataset of 5872 number of files for similarity threshold of 0.90. We can see that the model outputs approximately similar trend for all the similarity threshold across datasets.

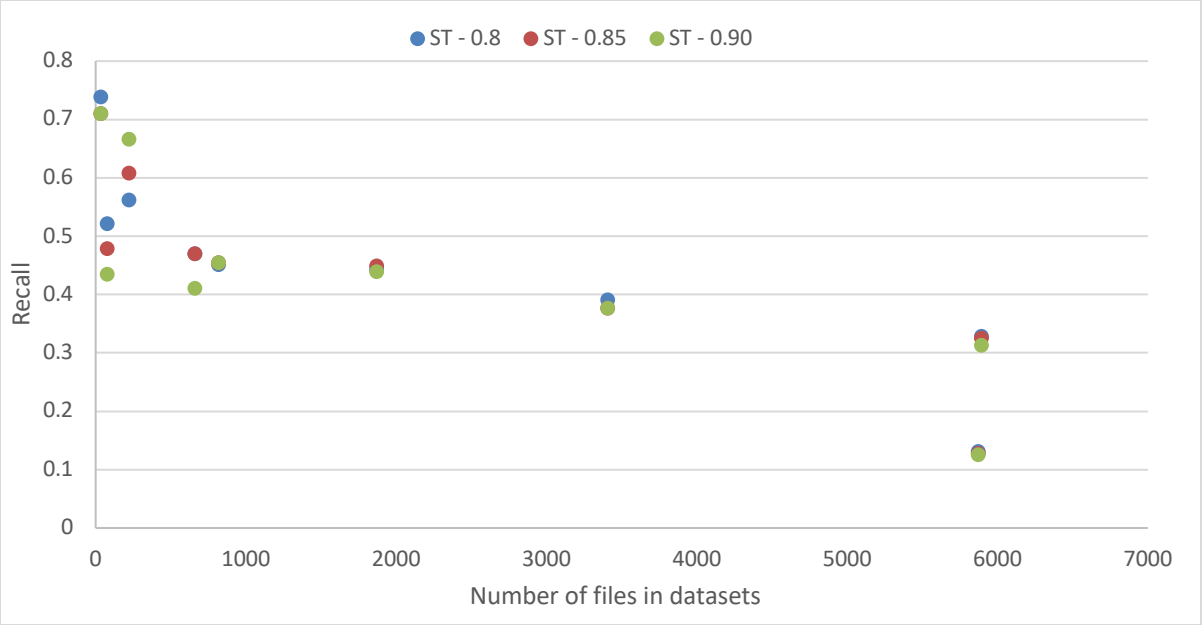

Figure 4. 17: Cluster Derivation Provenance: Recall with different Similarity threshold for different dataset number of files samples

Discussion: Recall is the measure of the predicted derivations over the total number of derivations in the ground truth. A high recall value shows that model was able to capture all the derivations for the dataset and a low recall value shows the file derivations that were missed by the model. As seen in Figure 4.17, the recall value decreases at a lower gradient between 0 to 1000 and then decreases with higher gradient as the number of files increases. The recall trend is very similar to precision. The recall value also shows variation with respect to the number of commits to the dataset shown in Figure 4.18. Front-end-interview-handbook dataset with 5872 number of files and 608 number of commits has the lowest recall value of 12.61% as compared with Tech-interview-handbook dataset highest number of commits is 971 with a recall value of 31.31%. Overall, as the number of commits increases, the possibility of missing file derivation increases.

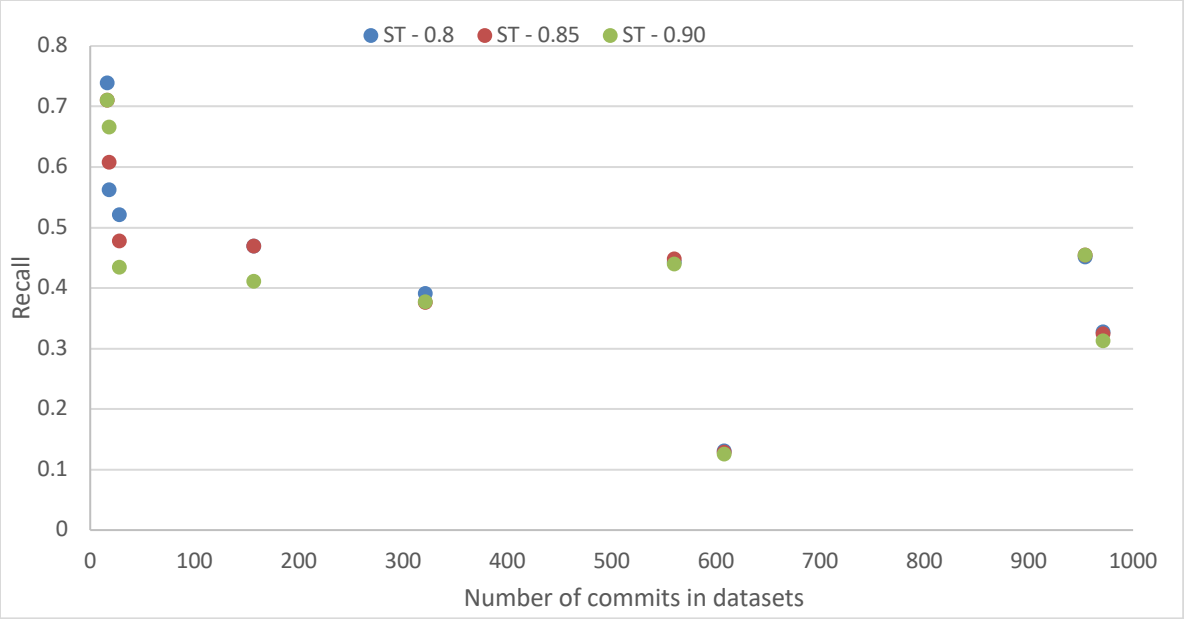

Figure 4. 18: Cluster Derivation Provenance: Recall with different Similarity threshold for different dataset number of commits samples

4.2.2 OneSource Vs SeekPeek Accuracy

We evaluate our model on the same dataset used by SeekPeek [8] for cluster derivation provenance. The similarity threshold was set to 0.85. The output for precision and recall values is populated in Table 4.4 for comparison. The team SeekPeek used LDA as clustering method. The results shown for OneSource uses Cosine as clustering method. We see that SeekPeek has higher precision value than OneSource and lower recall value than OneSource.

Table 4. 5: SeekPeek vs OneSource – Precision and Recall

Datasets	Size	Number of files	SeekPeek	OneSource
ftShare	3.04 MB	274	Precision: 100% Recall: 60%	Precision: 98% Recall: 75%
Hashie	5.64 MB	971	Precision: 100% Recall: 5%	Precision: 57% Recall: 47%
our_git_prov	178 MB	567	Precision: 100% Recall: 15%	Precision: 15% Recall: 13%

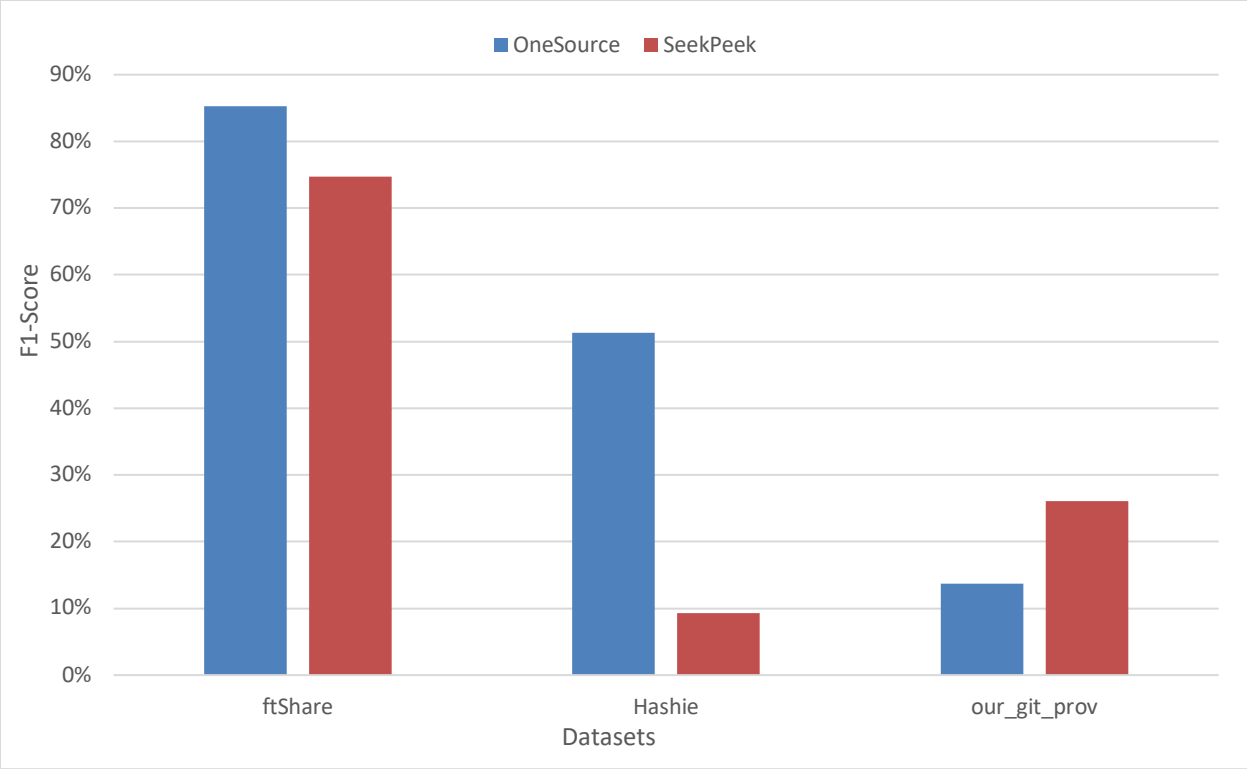

Figure 4. 19: Cluster Derivation Provenance: Recall with different Similarity threshold for different dataset number of commits samples

The F1-score for OneSource and SeekPeek is plotted in Figure 4.19 such that F1-score for OneSource is better than SeekPeek. Thus, OneSource consistently performs better than SeekPeek for datasets of all sizes except in our_git_prov dataset. This is due to a large size of the dataset. The graph also shows a decrease in the F1-score with an increase in the size of the dataset and number of files.

4.2.3 OneSource Execution Time

OneSource execution time changes with the number of files and size of the dataset. The execution time is recorded for similarity threshold 0.85 and clustering method as Cosine. The testing machine specifications are below.

Testing Machine:

- macOS Big Sur - version 11.6.7
- Memory - 4 GB 1600 MHz DDR3
- Processor - 1.4 GHz Dual-Core Intel Core i5

Execution time vs Number of files

We plotted the execution duration values with number of files for different datasets used in OneSource. The Execution time ranges from 16 to 2631 seconds, and datasets range from 33 files to 5894 files.

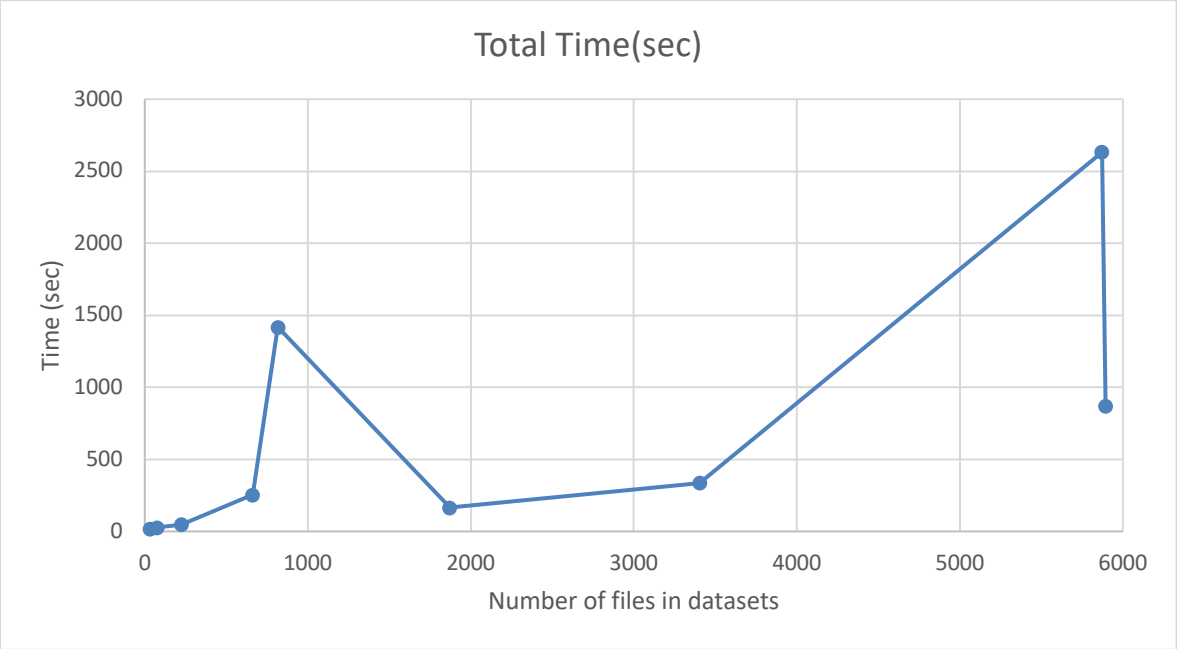

Figure 4. 20: Execution time with number of files different dataset samples

Discussion: Figure 4.20 shows that the overall execution time increases with the number of files in the dataset. The execution time for 5894 number of files dataset is less than 5872 number of files dataset is since the size of the dataset is higher for the 5872 number of files.

Execution time vs Size of Dataset

We plotted the execution duration values with size of datasets for different datasets used in OneSource. The Execution time ranges from 16 to 2631 seconds, and datasets range from 0.04 MB files to 200 MB.

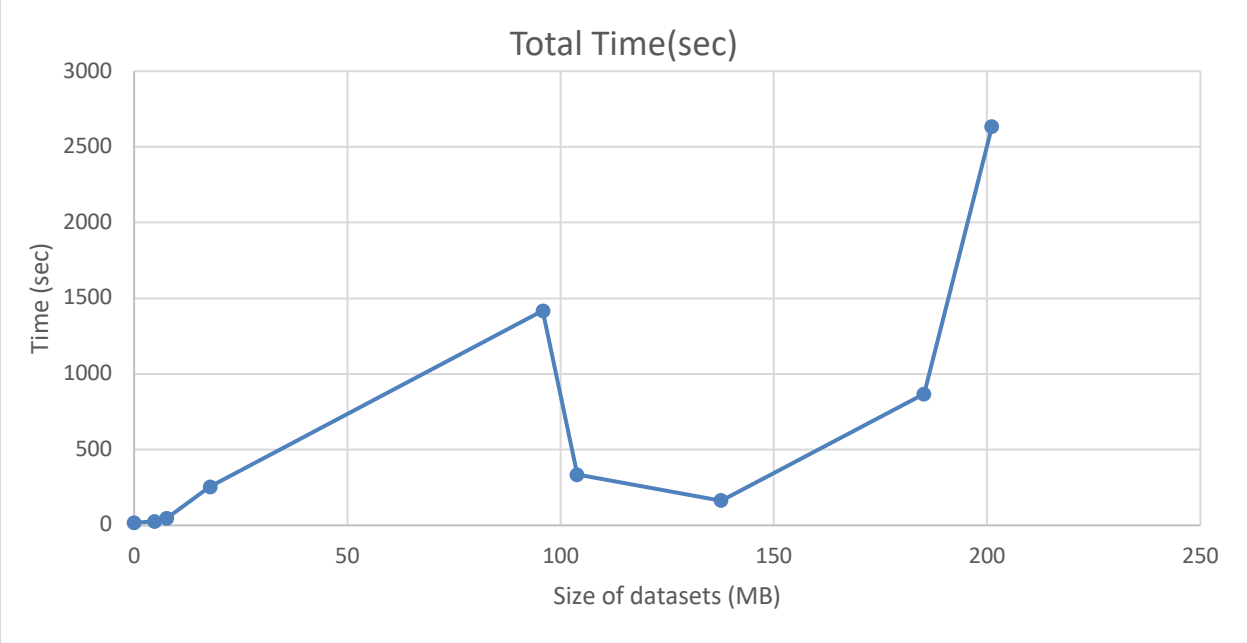

Figure 4. 21: Execution time with size of different dataset samples

Discussion: Figure 4.21 shows that the execution time increases with the size in the dataset, but it also depends on the number of files seen in the above section. The graph shows a decrease in execution time for 103MB size of dataset as compared to 95MB of dataset. This is because the number of commits in the 95MB dataset is higher than the number of commits in the dataset of 103MB.

To summarize, size of the dataset causes more slowdown to the execution time of OneSource with respect to number of commits and number of files.

CHAPTER 5 DISCUSSIONS

We now revisit the research questions we discussed earlier in the paper.

RQ1: Can we perform provenance reconstruction of machine-generated dataset?

Yes, OneSource establishes a multi-funneling approach to reconstruct provenance for machine-generated dataset. The multi-funneling approach includes data pre-processing, topic modeling, clustering, and lineage algorithm. Machine generated data was represented in our research using GitHub repositories.

RQ2: Is it possible to cluster all endpoints and its associated files to represent a group of similar documents in a machine-generated dataset?

Yes, OneSource implements Latent Dirichlet Allocation (LDA) and Cosine similarity to cluster all endpoints and its associated files to represent a group of similar documents in machine-generated dataset. This is known as cluster provenance in our research. The clustering with LDA is calculating the similarity score based on topic distribution generated for each file. The clustering with Cosine similarity is calculating similarity score between two documents using word frequency vector and distinct keywords. We have evaluated the comparison between clustering methods as LDA, LDA-Cosine, and Cosine in section 4.2.1.1. The evaluation results indicate that OneSource can reconstruct provenance of clusters by attaining 90% precision with cosine similarity as the clustering method.

RQ3: Is it possible to infer data lineage starting from any file back to the source file in machine-generated datasets*?*

Yes, OneSource implements lineage algorithm to trace data lineage starting from any file back to the source file in machine-generated datasets. This is known as cluster derivation provenance in our research. The lineage algorithm maps a backward and forward link for any file to trace the

endpoint backwards till it reaches the first source file. The evaluation results indicate that OneSource can reconstruct cluster derivation provenance with 66% precision using Cosine similarity clustering method and lineage algorithm. Please refer to section 4.2.1.3 for more information.

RQ4: How do we increase the accuracy of existing provenance reconstruction algorithms of machine-generated datasets?

OneSource yields an improvement in accuracy of 60% for cluster derivation than the existing technique. OneSource uses Cosine similarity as the clustering method which has improved the accuracy of capturing related files as compared to the LDA used in the existing technique. OneSource also improves the lineage algorithm by defining the endpoints to increase derivation accuracy back to the source file. We do not have similar comparison for cluster provenance as previous groups have not executed it. Please refer to section 4.2.2 for more information.

5.1 CHALLENGES

The selection of the dataset from the GitHub website was challenging as the data is not catalogued as per the size of the repository and one needs to rely on number of commits to create a normalized distribution of datasets as per size and number of files in the repository. To mitigate this challenge, we created twenty test datasets from GitHub repositories to find the optimal distribution of nine datasets.

Another challenge was to execute the model for larger datasets of size 200 MB and more with six thousand files, which took 40 minutes to complete execution. To mitigate this challenge, we need to explore techniques such as parallelization techniques [46] [47] [48]. Researchers at UWB [2] who have done this on human-generated datasets suggest a parallelized model distributed across multiple computing nodes can execute larger datasets.

5.2 LIMITATIONS

OneSource performance drops in F1-Score for datasets with larger memory sized datasets such as datasets of 185 MB size, six thousand files, and thousand commits. Furthermore, our results also shows that OneSource slows down for datasets for similar size datasets. It takes time for the model to process a large number of files which can be remediated using parallelization techniques.

In addition, OneSource drops in performance with datasets of one type of file extension in its folder such as android-open-project dataset used in our experimentation. This dataset has only ".md" files with 817 number of files and had a low accuracy.

CHAPTER 6 CONCLUSION

In this project, we presented a multi-funneling approach to reconstruct provenance in machine-generated datasets using Latent Dirichlet Allocation - Cosine similarity + Lineage algorithm. We presented our results in two ways: cluster provenance and cluster derivation provenance. The goal of the project was to map the source files to the endpoints (cluster provenance) and provide file relationship within a cluster (cluster derivation provenance). The evaluation results indicate that OneSource can reconstruct provenance of cluster by attaining 90% precision and reconstruct provenance of cluster derivation by attaining 66% precision with Cosine similarity as the clustering method.

LDA and Cosine similarity are semantic analysis tools used for clustering based on topic distribution and similarity score. The Cosine similarity performs better than both the methods LDA and LDA-Cosine as shown in section 4.2.1.1 Lineage algorithm is a customized algorithm created to solve the specific use case of mapping forward and backward file links to provide relationships within the cluster. The approach might not be the best solution for all types of machine-generated datasets, but it offers an optimal solution (precision of the model) for semantic based datasets.

OneSource yields improvement in accuracy of 60% for cluster derivation than the existing technique discussed in section 4.2.2. The optimizations to clustering methods and lineage algorithm has boosted the accuracy in the existing system. On the other hand, as discussed in Section 5, as a future work, an investigation is needed to improve the results for larger datasets.

6.1 LESSONS LEARNED

There were many opportunities to learn about different aspects of engineering and applied science:

- In engineering, we learned how to work with machine-generated datasets to kickstart its pre-processing and provide a scalable system architecture for different dataset sizes for model training and model inference. The project involved machine-learning development in java and python using a variety of tools MALLET, GitHub Commands, Zotero, IntelliJ, Lucid IO, Jupiter Notebook, and Docker.
- In applied science, it was essential to understand the model performance measurement in different scenarios. One important decision was to explore the clustering methods with LDA, LDA-Cosine, and Cosine. It was very important to understand model inferences on a dataset for two different use cases (cluster and cluster derivation) to recommend the best solution for the research topic.

6.2 FUTURE WORK

There are many improvements and experiments that can be performed on the approach to make it more accurate and efficient.

- Lineage algorithm works on the condition, if similarity score obtained from Latent Dirichlet Allocation (LDA) - Cosine is greater than the similarity threshold, it updates the related files for that file. We have noticed recall is low for our model as the false negatives are high. False negatives are obtained when the model misses to capture the similar files.
- We implemented mathematical comparisons such as Kullback–Leibler Divergence and Bhattacharyya distance to evaluate if similarity thresholds can be defined with an alternate

approach. We could also evaluate other mathematical comparisons such as JSD, Cramer's, and Neural Networks approaches [49] [50] [51]. These similarity measures could also be implemented in python using these resources [52][53][54]. But the chaining algorithm will need to be modified as per the sample size of similarity scores obtained in each of these mathematical comparisons to come up with results. The implementation issues include inability to define a similarity threshold as the values of Bhattacharyya distance and Kullback–Leibler Divergence broadly vary and needs several rounds of experimentation to define it. Also, the threshold was defined as (-0.5 to +0.5) for Bhattacharyya distance as similarity should be closer to zero, but it did not yield statistically significant results.

- To scale the OneSource model for a higher number of datasets, one can use the parallelization techniques such as Message Passing Interface, MapReduce, and Apache Spark discussed in [47] [46] [48] to reduce execution duration of the model.
- OneSource uses GitHub repositories as the data source because our approach is based on the semantic content of the data. Future researchers can expand this research with more data types such as system logs to further test OneSource's reconstruction techniques.
- Future researchers could make use of meta data from datasets such as author name and timestamp of data generation to increase prediction accuracy.
- Future researchers could increase the data preparation quality by developing another version of Git scrapper to further refine the F1 score and reduce false negatives.

s

BIBLIOGRAPHY

[1] S. Vasudevan, W. Pfeffer, D. Davis, and H. Asuncion, "Improving data provenance reconstruction via a multi-level funneling approach," in *2016 IEEE 12th International Conference on e-Science (e-Science)*, Oct. 2016, pp. 175–184. doi: 10.1109/eScience.2016.7870898.

[2] S. Vasudevan, "Optimized provenance reconstruction using genetic algorithm," 2016.

[3] A. Imran and R. Agrawal, "Data Provenance," 2017. doi: 10.1007/978-3-319-32001-4_58-1.

[4] "Provenance and Traceability Research Group at University of Washington Bothell Provenance Reconstruction." [Online]. Available: http://blogs.uw.edu/ptrg/publications/

[5] P. Groth, Y. Gil, and S. Magliacane, "Automatic Metadata Annotation through Reconstructing Provenance," in *CEUR Workshop Proceedings*, Jun. 2012, vol. 856.

[6] A. Aierken, D. B. Davis, Q. Zhang, K. Gupta, A. Wong, and H. U. Asuncion, "A Multi-level Funneling Approach to Data Provenance Reconstruction," *2014 IEEE 10th Int. Conf. E-Sci.*, vol. 2, pp. 71–74, 2014.

[7] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet Allocation," *J Mach Learn Res*, vol. 3, no. null, pp. 993–1022, Mar. 2003.

[8] A. Nakamura and P. Banik, *SeekPeek's Provenance Reconstruction of Machine Generated Datasets*. University of Washington Bothell. CSS 590 Class Report., 2020.

[9] A. Carle, C. Gebhart, M. Watson, and T. V. Roley, *Provenance Reconstruction*. University of Washington Bothell. CSS 490 Class Report., 2016.

[10] P. Groth, "Transparency and Reliability in the Data Supply Chain," *IEEE Internet Comput.*, vol. 17, no. 2, pp. 69–71, Mar. 2013, doi: 10.1109/MIC.2013.41.

[11] B. Biggio and F. Roli, "Wild Patterns: Ten Years After the Rise of Adversarial Machine Learning," Dec. 2017, doi: 10.1016/j.patcog.2018.07.023.

[12] T. De Nies, S. Coppens, D. Van Deursen, E. Mannens, and R. de Walle, "Automatic Discovery of High-Level Provenance Using Semantic Similarity," in *Proceedings of the 4th International Conference on Provenance and Annotation of Data and Processes*, Berlin, Heidelberg, 2012, pp. 97–110. doi: 10.1007/978-3-642-34222-6_8.

[13] T. De Nies *et al.*, "Towards multi-level provenance reconstruction of information diffusion on social media," in *International Conference on Information and Knowledge Management, Proceedings*, Oct. 2015, vol. 19-23-Oct-2015, pp. 1823–1826. doi: 10.1145/2806416.2806642.

[14] S. Magliacane, "Reconstructing Provenance," in *The Semantic Web – ISWC 2012*, Berlin, Heidelberg, 2012, pp. 399–406.

[15] S. Sultana and E. Bertino, "A File Provenance System," in *Proceedings of the Third ACM Conference on Data and Application Security and Privacy*, New York, NY, USA, 2013, pp. 153–156. doi: 10.1145/2435349.2435368.

[16] L. Karsai, A. Fekete, J. Kay, and P. Missier, "Clustering provenance: Facilitating provenance exploration through data abstraction," in *HILDA 2016 - Proceedings of the Workshop on Human-In-the-Loop Data Analytics*, Jun. 2016. doi: 10.1145/2939502.2939508.

[17] P. and B. H. Stamatogiannakis Manolisand Groth, "Facilitating Trust on Data through Provenance," in *Trust and Trustworthy Computing*, Cham, 2014, pp. 220–221.

[18] N. Arndt, P. Naumann, and E. Marx, "Exploring the Evolution and Provenance of Git Versioned RDF Data," Jun. 2017.

[19] Y. S. Tan, “Reconstructing Data Provenance from Log Files,” 2017.
[20] K. Govindan *et al.*, “PRONET: Network trust assessment based on incomplete provenance,” pp. 1213–1218, Jun. 2011, doi: 10.1109/MILCOM.2011.6127466.
[21] W. Oliveira, D. De Oliveira, and V. Braganholo, “Provenance analytics for workflow-based computational experiments: A survey,” *ACM Comput. Surv.*, vol. 51, no. 3, Apr. 2018, doi: 10.1145/3184900.
[22] K. Belhajjame, “On the Anonymization of Workflow Provenance without Compromising the Transparency of Lineage,” *J. Data Inf. Qual.*, vol. 14, no. 1, Mar. 2022, doi: 10.1145/3460207.
[23] T. Roelleke and J. Wang, “TF-IDF Uncovered: A Study of Theories and Probabilities,” in *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, New York, NY, USA, 2008, pp. 435–442. doi: 10.1145/1390334.1390409.
[24] H. U. Asuncion, A. U. Asuncion, and R. N. Taylor, “Software Traceability with Topic Modeling,” in *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 1*, New York, NY, USA, 2010, pp. 95–104. doi: 10.1145/1806799.1806817.
[25] T.-H. Chen, S. W. Thomas, M. Nagappan, and A. E. Hassan, “Explaining Software Defects Using Topic Models,” in *Proceedings of the 9th IEEE Working Conference on Mining Software Repositories*, Zurich, Switzerland, 2012, pp. 189–198.
[26] B. Hong, Y. Kim, and S. H. Lee, “An Efficient Tag Recommendation Method Using Topic Modeling Approaches,” in *Proceedings of the International Conference on Research in Adaptive and Convergent Systems*, New York, NY, USA, 2017, pp. 56–61. doi: 10.1145/3129676.3129709.
[27] P. C. Kaur, T. Ghorpade, and V. Mane, “Topic Extraction and Sentiment Classification by Using Latent Dirichlet Markov Allocation and SentiWordNet,” in *Proceedings of the International Conference on Advances in Information Communication Technology & Computing*, New York, NY, USA, 2016. doi: 10.1145/2979779.2979865.
[28] R. Krestel, P. Fankhauser, and W. Nejdl, “Latent Dirichlet Allocation for Tag Recommendation,” in *Proceedings of the Third ACM Conference on Recommender Systems*, New York, NY, USA, 2009, pp. 61–68. doi: 10.1145/1639714.1639726.
[29] I. Bíró, J. Szabó, and A. A. Benczúr, “Latent Dirichlet Allocation in Web Spam Filtering,” in *Proceedings of the 4th International Workshop on Adversarial Information Retrieval on the Web*, New York, NY, USA, 2008, pp. 29–32. doi: 10.1145/1451983.1451991.
[30] C. Ozcaglar, “Classification of email messages into topics using latent dirichlet allocation,” Master’s Thesis, 2008.
[31] S. Graham S. Weingart and I. Milligan, “Getting Started with Topic Modeling and MALLET,” 2012. [Online]. Available: https://programminghistorian.org/en/lessons/topic-modeling-and-mallet
[32] A. K. McCallum, “MALLET: A Machine Learning for Language Toolkit,” 2002. [Online]. Available: http://www.cs.umass.edu/~mccallum/mallet
[33] P. Marjai, P. Lehotay-Kéry, and A. Kiss, “Document similarity for error prediction,” *J. Inf. Telecommun.*, vol. 5, pp. 1–14, Mar. 2021, doi: 10.1080/24751839.2021.1893496.
[34] “Cosine Similarity - Wikipedia.” [Online]. Available: https://en.wikipedia.org/wiki/Cosine_similarity

[35] P. Macko, D. Margo, and M. Seltzer, “Local Clustering in Provenance Graphs,” in *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management*, New York, NY, USA, 2013, pp. 835–840. doi: 10.1145/2505515.2505624.
[36] “W3 PROV - PROV-Overview.” [Online]. Available: https://www.w3.org/TR/prov-overview/
[37] “PROV-DM.” [Online]. Available: https://www.w3.org/TR/2013/REC-prov-dm-20130430/
[38] S. Elbassuoni and R. Blanco, “Keyword Search over RDF Graphs,” in *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, New York, NY, USA, 2011, pp. 237–242. doi: 10.1145/2063576.2063615.
[39] P. Maillot and C. Bobed, “Measuring Structural Similarity between RDF Graphs,” in *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, New York, NY, USA, 2018, pp. 1960–1967. doi: 10.1145/3167132.3167342.
[40] M. A. Mercioni and S. Holban, “A Survey of Distance Metrics in Clustering Data Mining Techniques,” in *Proceedings of the 2019 3rd International Conference on Graphics and Signal Processing*, New York, NY, USA, 2019, pp. 44–47. doi: 10.1145/3338472.3338490.
[41] *GitHub*. [Online]. Available: https://github.com/
[42] *"NLTK :: Natural Language Toolkit.”*. [Online]. Available: https://www.nltk.org/
[43] *ProcessBuilder API*. [Online]. Available: https://docs.oracle.com/javase/7/docs/api/java/lang/ProcessBuilder.html
[44] *Wikipedia.org*. [Online]. Available: https://en.wikipedia.org/wiki/Precision_and_recall
[45] “Correlation Coefficient formula.” https://www.statisticshowto.com/probability-and-statistics/correlation-coefficient-formula/
[46] Y. Jayashankar, “Parallelizing LDA-GA (Latent Dirichlet Allocation - Genetic Algorithm) for Data Provenance Reconstruction,” 2022.
[47] X. Ma, “Optimizing the performance of lda-ga funnel using a parallel computing technique,” Master’s Thesis, 2017.
[48] D. Liu, “Data-provenance project performance optimization,” 2018.
[49] B. Lagesse, G. Nguyen, U. Goswami, and K. Wu, “You Had to Be There: Private Video Sharing for Mobile Phones using Fully Homomorphic Encryption,” in *2021 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*, 2021, pp. 730–735. doi: 10.1109/PerComWorkshops51409.2021.9431029.
[50] K. Wu and B. Lagesse, “Do You See What I See?<Subtitle>Detecting Hidden Streaming Cameras Through Similarity of Simultaneous Observation,” in *2019 IEEE International Conference on Pervasive Computing and Communications (PerCom*, 2019, pp. 1–10. doi: 10.1109/PERCOM.2019.8767411.
[51] K. Wu and B. Lagesse, “Detecting Hidden Webcams with Delay-Tolerant Similarity of Simultaneous Observation.” arXiv, 2019. doi: 10.48550/ARXIV.1901.02818.
[52] “Measuring the statistical similarity.” https://medium.com/datalab-log/measuring-the-statistical-similarity-between-two-samples-using-jensen-shannon-and-kullback-leibler-8d05af514b15
[53] “Kullback leibler divergence.” https://www.geeksforgeeks.org/kullback-leibler-divergence/
[54] “Divergence between probability distributions.” https://machinelearningmastery.com/divergence-between-probability-distributions/