

GPTCache: An Open-Source Semantic Cache for LLM Applications Enabling Faster Answers and Cost Savings

Bang Fu, Di Feng
Zilliz Inc.

Abstract

The rise of ChatGPT¹ has led to the development of artificial intelligence (AI) applications, particularly those that rely on large language models (LLMs). However, recalling LLM APIs can be expensive, and the response speed may slow down during LLMs' peak times, causing frustration among developers. Potential solutions to this problem include using better LLM models or investing in more computing resources. However, these options may increase product development costs and decrease development speed. GPTCache² is an open-source semantic cache that stores LLM responses to address this issue. When integrating an AI application with GPTCache, user queries are first sent to GPTCache for a response before being sent to LLMs like ChatGPT. If GPTCache has the answer to a question, it quickly returns the answer to the user without having to query the LLM. This approach saves costs on API recalls and makes response times much faster. For instance, integrating GPTCache with the GPT service offered by OpenAI can increase response speed 2-10 times when the cache is hit. Moreover, network fluctuations will not affect GPTCache's response time, making it highly stable. This paper presents GPTCache and its architecture, how it functions and performs, and the use cases for which it is most advantageous.

1 Introduction

Since OpenAI released ChatGPT, large language models have impressed many people and have been frequently integrated into our daily work and lives. At the same time, more open-source enthusiasts and tech companies have invested time and effort into developing open-source LLMs, such as Meta's Llama (Touvron et al., 2023a,b), Google's PaLM (Chowdhery et al., 2022), Stanford's Alpaca (Wang

et al., 2023; Taori et al., 2023), and Databrick's Dolly (Conover et al., 2023).

There are two ways to use large language models: online services provided by companies like OpenAI, Claude, and Cohere or downloading open-source models and deploying them on your servers. Both methods require payment. Online services charge you based on tokens, while deploying models on your own server requires purchasing specific computing resources. The choice depends on individual needs.

While online services are more expensive, they are more convenient and effective and provide a better user experience than deploying models yourself. Costs and user experience are two critical considerations for building LLM applications. As your LLM application gains popularity and experiences a surge in traffic, the cost of LLM API calls will increase significantly. High response latency will also be frustrating, particularly during peak times for LLMs, directly affecting the user experience.

GPTCache is an open-source semantic cache designed to improve the efficiency and speed of GPT-based applications by storing and retrieving the responses generated by language models. Unlike traditional cache systems such as Redis, GPTCache employs semantic caching, which stores and retrieves data through embeddings. It utilizes embedding algorithms to transform the queries and LLMs' responses into embeddings and conducts similarity searches on these embeddings using a vector store such as Milvus. GPTCache allows users to customize the cache to their specific requirements, offering a range of choices for embedding, similarity assessment, storage location, and eviction policies. Furthermore, GPTCache supports both the OpenAI ChatGPT interface and the Langchain interface, with plans to support more interfaces in the coming months.

Through experiments using the paraphrase-albert-small-v2 model (Reimers

¹<https://openai.com/chatgpt>

²<https://github.com/zilliztech/GPTCache>

and Gurevych, 2019) to embed input in the onnx runtime environment and running it on a local Mac with i7, 4CPU, and 32G memory, the time consumed when hitting the cache is approximately 0.3 seconds. Compared to accessing OpenAI ChatGPT with an average response latency of 3 seconds, the time consumed is only 1/10. Furthermore, no tokens are consumed when hitting the cache. Different embedding models and similarity evaluation algorithms must be selected in real development scenarios based on the tolerance for cache errors. Even so, the entire consumption time is about 3-4 times faster.

2 Related Works

2.1 Accelerating LLM Inference

Accelerating LLM Inference. Large language models (LLMs) typically take seconds to infer answers, prompting researchers to explore ways to reduce inference time and resource consumption. One approach is quantization (Dettmers and Zettlemoyer, 2023), which decreases the number of bits needed to represent each parameter and, therefore reduces the model size. However, this can result in a trade-off between accuracy and memory footprint. Another method is pruning, which can sparsify large-scale generative pre-trained transformer (GPT) models without retraining, as demonstrated by SparseGPT (Frantar and Alistarh, 2023). Additional methods include Compressing (Xu et al., 2020) and Inference with Reference (Yang et al., 2023).

2.2 Widespread application of Caching

Widespread application of Caching. Caching is a commonly used technique to reduce frequent and computationally expensive data accesses, which can improve system query performance. Many different caching schemes have been proposed for various scenarios. For example, semantic knowledge extracted from data can convert cache misses to cache hits, avoiding unnecessary access to web sources (Lee and Chu, 1999). Another example is in querying multiple databases with sensitive information, where a differentially private cache of past responses can answer the current workload at a lower privacy budget while meeting strict accuracy guarantees (Mazmudar et al., 2022). In addition, a cached memory architecture for new changes to embedding tables has been proposed during embedding. In this architecture, most rows in embeddings

are trained at low precision, while the most frequent or recently accessed rows are cached and trained at full precision (Yang et al., 2020). As demonstrated, caching is applied in a variety of real-world development processes.

2.3 Embedding Models

Embedding models (Almeida and Xexéo, 2023) are a type of machine learning model that map discrete symbols or objects (such as text, images, audio, etc.) to continuous vector spaces. These vectors are called embedding vectors and are indispensable in many natural language processing (NLP) and computer vision (CV) tasks.

In NLP tasks, embedding models aim to map text into a low-dimensional continuous vector space. This makes it easier for machine learning models to process the text. The vectors can capture semantic information about the text, such as its meaning in context. In CV tasks, embedding models can map images, videos, or objects into a vector space. This approach allows them to be processed by computer vision algorithms, such as image search and identification. Common text embedding models include BERT (Devlin et al., 2019), GloVe (Pennington et al., 2014), and Word2Vec (Goldberg and Levy, 2014; Mikolov et al., 2013). These models generate embedding vectors by processing large amounts of text data. They can also perform well in many NLP tasks, such as semantic similarity calculation, part-of-speech tagging, named entity recognition, and sentiment analysis.

2.4 Vector Store

A vector database is designed for storing and managing vector data. Vector data consists of sequences of numbers commonly used to represent objects or features in high-dimensional spaces. For example, data types such as images, audio, and natural language text can be represented as vector data.

Vector databases improve the efficiency and accuracy of vector data retrieval by using vector similarity measures to index and query the data. This indexing technique allows the database to quickly find vectors most similar to a query vector, making it useful for various applications such as sentiment analysis, image search, speech recognition, and recommendation systems.

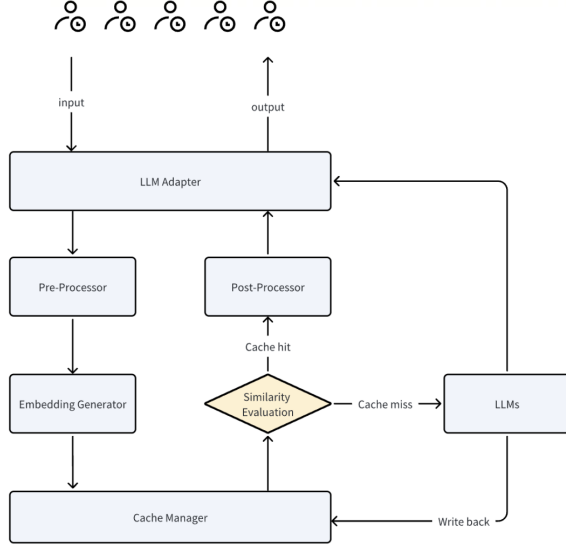


Figure 1: GPTCache: The architecture comprises six core components: adapter, pre-processor, embedding generator, cache manager, similarity evaluator, and post-processor.

3 GPTCache: Semantic Cache for LLMs

The overall workflow of GPTCache follows the general cache pattern - attempting to obtain results from the cache before fetching data or processing requests. If successful, the process terminates immediately. Otherwise, the processing path is the same as if the cache did not exist. However, before returning, the corresponding results are stored in the cache so that repeated actions will retrieve results directly from the cache next time. Using the cache significantly reduces workflow time, which explains why cache designs are ubiquitous in our lives, such as multi-level caches in computers, DNS caches in networks, and Redis/Memcache in management systems.

3.1 Adapter

The adapter serves as the interface for GPTCache to interact with the outside world. It is responsible for converting LLM requests into cache protocols, controlling the entire cache workflow, and transforming cache results into LLM responses. For easy integration of GPTCache into our systems or other ChatGPT-based systems without extra development effort, the adapter should be easy to integrate with all LLMs and flexible enough to integrate more multimodal models in the future.

3.2 Pre-Processor

The pre-processor handles the input of LLM requests primarily by formatting the information as the primary key for the cache data. This includes removing prompt information from inputs, compressing input information, and only retaining the last certain words for long texts or the last round in a multi-round conversation. These operations make the request data more distinguishable from each other and remove redundant and irrelevant information from the requests.

Pre-processing is a critical factor affecting the performance of the cache. For example, suppose both inputs contain a large portion of prompt information, where the key part of the information is only a small portion of the entire input. In that case, the cache cannot obtain the key information without eliminating the prompt. This can result in a high probability that all requests hit the cache. The preprocessed results are passed to the Embedding component for vector conversion.

3.3 Embedding Generator

The embedding generator can convert user queries into embedding vectors for later vector similarity retrieval. There are two methods to achieve this functionality. The first method generates embedding vectors through cloud services (such as OpenAI, Hugging Face, Cohere, etc.). The second method involves generating embedding vectors using local models that can be downloaded from sources such as HuggingFace or GitHub.

3.4 Cache Manager

The cache manager is the core component of GPTCache and has three functions:

- **Cache storage:** stores user requests and corresponding LLM responses.
- **Vector storage:** stores embedding vectors and retrieves similar results.
- **Eviction management:** controls cache capacity and clears expired data according to LRU or FIFO policy when the cache is full.

Before a piece of data is stored, an id will be generated. The id and scalar data will be stored in cache storage, and the id and vector data will be stored in vector storage. In this way, cache storage and vector storage are associated. Eviction management also records these IDs. When cache data

needs to be cleared, the data corresponding to cache storage and vector storage will be deleted based on the id.

The eviction manager releases the cache space by deleting data that has been unused for a long time or is furthest away from using in the GPT-Cache. If necessary, it removes data from both the cache and vector store. However, frequent deletion operations in the vector store can lead to performance degradation. Therefore, GPTCache only triggers asynchronous operations (e.g., index building, compression, etc.) upon reaching deletion thresholds.

3.5 Similarity Evaluator

GPTCache retrieves the Top-K most similar answers from its cache and uses a similarity evaluation function to determine if the cached answer matches the input query. The similarity evaluation module is also crucial for GPTCache. After research, we eventually adopted the fine-tuned ALBERT model. Of course, there is still room for improvement here, and other language models or LLMs (such as LLaMa-7b) can also be used.

3.6 Post-Processor

The post-processor is responsible for preparing the final response to be returned to the user. It can either return the most similar response or adjust the response's randomness based on the request's temperature parameter. If a similar response is not found in the cache, the LLM will handle the request to generate a response. The generated response will be stored in the cache before being returned to the user.

3.7 Key GPTCache Use Cases

Not all LLM applications are suitable for GPT-Cache, as the cache hit rate is a crucial factor for the cache's effectiveness. If the cache hit rate is too low, the return on investment cannot balance the input, and there is no need to spend effort on this feature. This is similar to traditional caching scenarios, where caching is usually done only on frequently accessed public nodes to maximize resource utilization and system performance and improve user experience.

This paper introduces three critical practical situations where GPTCache is most beneficial:

1. LLM applications designed for specific domains of expertise, such as law, biology,

medicine, finance, and other specialized fields.

2. LLM applications applied to specific use cases, such as internal company ChatBots or personal assistants like chat-pdf and chat-paper. These applications can be enhanced with a cutting-edge AI technology stack called CVP³ (ChatGPT+Vector DB]+prompt engineering). This combination overcomes the limitations of knowledge bases and enables further expansion and innovation.
3. LLM applications with large user groups can benefit from using the same cache for user groups with the same profile if user profiling and classification can be done. This approach yields good returns.

4 Experiments

To evaluate GPTCache, we randomly scrape some information from the webpage, and then let chatgpt produce a corresponding data (similar or exactly opposite). And then we created a dataset consisting of three types of sentence pairs:

- Similar sample pairs: two sentences with identical semantics
- Opposite sample pairs: two sentences with related but not identical semantics
- Unrelated sample pairs: two sentences with completely different semantics

Then we evaluate the effectiveness of cache through five indicators, which are:

1. Cache Hit, which successfully finds similar values based on the input, which consists of Positive Hits and Negative Hits.
2. Cache Miss, no similar value was found based on the input
3. Positive Hits, the obtained cache value is confirmed to be similar to the input value
4. Negative Hits, the obtained cache value is found to be not similar through inspection.

³<https://zilliz.com/blog/ChatGPT-VectorDB-Prompt-as-code>

Cache Hit	Cache Miss	Positive Hits	Negative Hits	Hit Latency
876	124	837	39	0.20 s

Table 1: Results for Caching Hit and Miss Samples, Caching Mixed Positive and Negative Queries, and Hit Latency

5. Hit Latency, it includes pre-processing time, cache data search time, similarity calculation time and post-processing time. The pre-processing and post-processing do not use the model during the test process, and are just simple character or number comparisons.

In addition, we tried different similarity algorithms and found that they had no impact on the results, so we used the common cosine similarity.

First, we cached the keys of all 30,000 positive sample pairs. Next, we randomly selected 1,000 samples and used their peer values as queries. Table 1 presents the results.

We found setting the similarity threshold of GPT-Cache to 0.7 achieves a good balance between hit and positive ratios. So we used this for subsequent tests.

To determine if a cached result is positive or negative to the query, we used the similarity score from ChatGPT with a positive threshold of 0.7. We generated this by prompting:

Please rate the similarity of the following two questions on a scale from 0 to 1, where 0 means not related and 1 means exactly the same meaning. And questions, "Which app lets you watch live football for free?" and "How can I watch a football live match on my phone?" The similarity score is.

We issued 1,160 queries with 50% positive and 50% unrelated negative samples. Table 2 presents the results. The hit ratio was about 50%, and the negative hit ratio was similar to Experiment 1, indicating GPTCache successfully distinguished related and unrelated queries.

Next, we tried to also cache all negative samples and queried with their peers. Surprisingly, despite high ChatGPT similarity scores (over 0.9) for some pairs, none hit the cache. The cause of the cache error could be the similarity evaluator’s fine-tuning on this dataset correctly undervalued the similarity of negative pairs.

Cache Hit	Cache Miss	Positive Hits	Negative Hits	Hit Latency
570	590	549	21	0.17 s

Table 2: Results for Caching Hit and Miss Samples, Caching Mixed Positive and Negative Queries, and Hit Latency

The initial experiments demonstrate that GPT-Cache can effectively utilize semantic similarity to cache LLM query-response pairs and achieve significant speedups. We plan to conduct more rigorous evaluations on larger and more diverse datasets. When tuning the similarity threshold, further investigation is required to balance cache hits versus false positives.

5 Future Challenges

One core factor affecting GPTCache’s caching effectiveness is the choice of embedding model. Compared to other component selections, the choice of embedding model is crucial because subsequent vector database retrieval relies on the embedding vectors. If the vectors cannot adequately capture the features of the input text, the retrieval results will be very noisy or even counterproductive, returning completely irrelevant cached data. Our testing has shown that even the best cache hit rates do not exceed 90% with current embedding models. This means that negative cache hits are noticeable during use. While this may not greatly impact individual users, it would be unacceptable in real production scenarios. Although other methods, like more strict similarity evaluation, could improve positive cache hit rates, this would also decrease the overall hit rate. Most current embedding models are likely optimized for search scenarios but may not work as well for cache matching. For example, results with semantics opposite to the input text are acceptable in search since they have structural similarity, but this is unacceptable in caching scenarios. Naturally, how to obtain embeddings suitable for caching is an open area for exploration.

Even with a suitable embedding model, positive hit rates are unlikely to reach production requirements, such as 99%, without decreasing cache hits. The similarity evaluation module plays a core role in improving positive cache hit rates by filtering incorrect hits. Our current implementations include vector distance, retrieving distance, cohere rerank

API, and sbert cross-encoder. However, testing shows these methods do not sufficiently distinguish between positive and negative cache hits. To address this, we are using large models to judge sentence similarity and distill them into a small model to obtain a specialized model for textual similarity.

As large language models are widely adopted, their supported token counts have increased from 2k initially to 100k. However, if a single input exceeds the LLM's token count limit, it cannot process the request. Similarly, conversations with total tokens exceeding the limit must drop some information. Large token counts from long texts or conversations pose a challenge for caching, making it difficult to identify key information and generate representative vectors. Currently, we utilize summary models to pre-process and shorten long inputs, but this approach increases cache instability, and its effectiveness is not optimistic. Therefore, special cache lookup methods may be needed for long texts.

As mentioned earlier, is there any alternative to retrieving cache data using vector databases? For example, can we use traditional databases like MySQL, PostgreSQL, SQL Server, or Oracle to store cache data, with textual pre-processing to standardize user inputs? For instance, when the inputs are "tell me a joke" and "I want to get a joke", can we convert them to a certain string, like "tell a joke", or a same number? Cache hits could then utilize string matching or numeric ranges instead of vectors.

6 Conclusion

GPTCache is a caching solution tailored for LLM applications. It brings the following benefits to the LLM app developers:

- Less costs: Most LLM services charge fees based on a combination of the number of requests and token count. GPTCache can effectively minimize expenses by caching query results, thereby reducing the number of requests and tokens sent to the LLM service.
- Faster response times: LLMs utilize generative AI to produce responses in real-time, which can be time-consuming. However, when a similar query is cached, the response time greatly improves, as the result is retrieved directly from the cache without interaction

with the LLM service. In most cases, GPTCache can also offer better query throughput than standard LLM services.

- More scalable and available: LLM services often impose rate limits on the number of access requests within a given timeframe. If these limits are exceeded, additional requests are blocked until a cooldown period has elapsed, leading to service outages. GPTCache allows you to easily scale and handle increasing query volumes, ensuring consistent performance as your application's user base expands.

By utilizing semantic similarity search and vector embeddings, GPTCache provides an effective caching solution that enhances performance, reduces costs, and improves scalability for applications that use large language models. Our initial experiments have shown great potential, and we plan to conduct more comprehensive evaluations on diverse real-world datasets and application scenarios.



References

- Felipe Almeida and Geraldo Xexéo. 2023. [Word embeddings: A survey](#).
- Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayanan Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. [Palm: Scaling language modeling with pathways](#).
- Mike Conover, Matt Hayes, Ankit Mathur, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, and Reynold Xin. 2023. [Free dolly: Introducing the world's first truly open instruction-tuned llm](#).

- Tim Dettmers and Luke Zettlemoyer. 2023. [The case for 4-bit precision: k-bit inference scaling laws](#). In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 7750–7774. PMLR.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of deep bidirectional transformers for language understanding](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- Elias Frantar and Dan Alistarh. 2023. SparseGPT: Massive language models can be accurately pruned in one-shot. *arXiv preprint arXiv:2301.00774*.
- Yoav Goldberg and Omer Levy. 2014. [word2vec explained: deriving mikolov et al.’s negative-sampling word-embedding method](#). *CoRR*, abs/1402.3722.
- Dongwon Lee and Wesley W. Chu. 1999. [Semantic caching via query matching for web sources](#). In *Proceedings of the Eighth International Conference on Information and Knowledge Management, CIKM ’99*, page 77–85, New York, NY, USA. Association for Computing Machinery.
- Miti Mazmudar, Thomas Humphries, Jiaxiang Liu, Matthew Rafuse, and Xi He. 2022. [Cache me if you can: Accuracy-aware inference engine for differentially private data exploration](#).
- Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. [Efficient estimation of word representations in vector space](#). In *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*.
- Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. [GloVe: Global vectors for word representation](#). In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.
- Nils Reimers and Iryna Gurevych. 2019. [Sentence-BERT: Sentence embeddings using Siamese BERT-networks](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.
- Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca.
- Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023a. [Llama: Open and efficient foundation language models](#).
- Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023b. [Llama 2: Open foundation and fine-tuned chat models](#).
- Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khoshnab, and Hannaneh Hajishirzi. 2023. [Self-instruct: Aligning language models with self-generated instructions](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13484–13508, Toronto, Canada. Association for Computational Linguistics.
- Canwen Xu, Wangchunshu Zhou, Tao Ge, Furu Wei, and Ming Zhou. 2020. [BERT-of-theseus: Compressing BERT by progressive module replacing](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7859–7869, Online. Association for Computational Linguistics.
- Jie Amy Yang, Jianyu Huang, Jongsoo Park, Ping Tak Peter Tang, and Andrew Tulloch. 2020. [Mixed-precision embedding using a cache](#).
- Nan Yang, Tao Ge, Liang Wang, Binxing Jiao, Daxin Jiang, Linjun Yang, Rangan Majumder, and Furu Wei. 2023. [Inference with reference: Lossless acceleration of large language models](#).



Selecting Open Source Projects for Traceability Case Studies

Michael Rath^(✉) , Mihaela Todorova Tomova, and Patrick Mäder 

Technische Universität Ilmenau, Ilmenau, Germany
{michael.rath,mihaela.todorova-tomova,patrick.maeder}@tu-ilmenau.de

Abstract. [Context & Motivation] Once research questions and initial theories have shaped, empirical research typically requires to select cases to study subsumed ideas. Issue trackers of todays open source systems (OSS) are a gold mine for empirical research, not least to study trace links among the included issue artifacts. [Question / problem] The huge amount of available OSS projects complicates the process of finding suitable cases to support the research goals. Further, simply picking a large number of projects on a random basis does not imply generalizability. Therefore the selection process should be carefully designed. [Principle ideas / results] In this paper we propose a method to choose OSS projects to study trace links found in issue tracking systems. Builds upon purposive sampling and cluster analysis, relevant project characteristics are identified whereas irrelevant information is filtered. Every step of the method is demonstrated on a live example. [Contributions] The proposed strategy selects an information-rich, representative and diverse sample of OSS to perform a traceability case study. Our work may be used as practical guide for other researchers to perform project selection tasks.

1 Introduction

Case study is a commonly conducted research strategy. It has been successfully applied in different domains including software engineering, medicine, political and social science [23]. A case study allows to test an individual hypothesis and can be viewed as a robust research method when performing an in-depth investigation [11, 28]. To fully maximize the benefits of a case study, it is important to select samples in a systematic manner. However, time constraints, and lack of a well developed program theory often constrain grounded-theoretical approaches suggested in the literature. Therefore researchers fall back to basic selection (random) schemes, impressionistic evidence, or guesswork to aid them in making the critical decision of which cases to include in their study [28]. Further, simply increasing the sample size does not imply more generalizability of the performed study [18]. Nevertheless, different kinds of sampling methods and strategies exist to aid researchers in making decisions on how to perform the case sampling.

In this paper, we present a practical example how to select software projects to perform a case study in context of requirements traceability research. The

presented approach is based on a qualitative method called purposive sampling followed by a clustering technique. For the outlined example, we formulate a research topic driving the project selection process and results in small yet information-rich and representative collection of projects. This collection is suitable to actually perform the case study, which itself is out of scope of this work.

The remaining paper is structured as follows. Section 2 presents related work concerning case studies. The next section, provides the background on purposive sampling and clustering algorithms, the main techniques used in our approach. Afterwards, in Sect. 4, we describe the approach starting with formulating the research topic, identifying relevant case characteristics and performing the clustering. Section 5 evaluates and interprets the created project clusters. Further is shows how to chose individual projects from the clusters in order to create the final set of projects for the case study. The paper ends with a conclusion.

2 Related Work

In Flyvbjerg [11], the author investigates five common misunderstandings about case studies. Common wisdom is that a case study is the detailed examination of a single example, which “cannot provide reliable information about the broader class”. The author argues, that a case study *is* reliable and further is more than a pilot method to be used only in preparing the real study’s larger surveys. The discussed misunderstandings center around concerns about the theory, reliability, and validity of a case study as a scientific method. The author shows, that case studies are generalizable. However, a strategic selection of cases is required, whereas a random or stratified sampling may not be sufficient. Therefore different information-related selection methods are presented.

Curtis et al. [8] examine samples for three different case studies and which are later evaluated based on six guidelines as suggested by Miles and Huberman [17]. Two guidelines check whether the sampling strategy can be considered as ethical and if it is relevant to the research question and conceptual framework. A third guideline considers factors such as accessibility, time, and cost. The remaining three guidelines examine if the sample generates rich information about the researched topic, enhances the generalizability of the findings, and produces believable explanations/descriptions. In their work, the authors acknowledge the importance of guidelines, but a “simple blueprint” for qualitative research is very hard to construct since each study depends on a different strategy. Depending on the researched topic misinterpretation of parts of the guidelines can arise. For example, researchers may have different opinions on what is considered believable and ethical.

Nagappan et al. [18] investigate diversity and representativeness and thus, whether phenomena found in few case (e. g. projects) are reflective of others. The paper introduces a measure called *sample coverage* combining the two concepts to access the quality of a sample. A sample is diverse, if it contains members of every subgroup of a population. On the other hand, a sample is representative, if the size of each subgroup is proportional to the size of the subgroup in the

population, i.e. each sample in the population can be chosen with equal probability [12]. A better coverage can be achieved when the similarity between the candidate samples is smaller, and therefore the distance, defined by a similarity function, between projects is larger. Thus, new projects should be added to an existing sample set in order to maximize the sample coverage.

Another important step when performing studies is to find information-rich cases that can help researchers to better understand complex issues. While quantitative research methods, such as random sampling, concentrate on selecting unbiased, representative samples, qualitative research methods give researchers the opportunity to clarify or deepen their understanding about the phenomenon under study [13]. A widely used qualitative research method for identification and selection of information-rich cases is purposive sampling [19, 24, 25]. We discuss purposive sampling in more detail in Sect. 3.

Ryzin [28] uses hierarchical, agglomerative clustering to guide purposive selection of projects. After building the clusters, the author proposes to select projects randomly or by choosing projects most similar to the means of the whole cluster. The author highlights multiple benefits of using cluster analysis in project selection. Clusters can be easily build with different number of samples: small and large. After cluster creation, the assignment of projects to clusters can be integrated with judgmental criteria that provide additional information and insight to guide the selection process. At last, the results of a cluster analysis are a basis to decide about the extent to which the findings can be generalized to the population of projects.

3 Background

This section provides background of important concepts used in the paper.

3.1 Purposive Sampling

Purposive sampling, also known as *judgment* or *purposeful* sampling, is a non-probability sampling method. The samples are selected according to the objective of the study. Contrasting probability sampling, which leads to greater breadth of information from a larger number of units, purposive sampling leads to greater depth of information from a small number of carefully selected cases. Patton [20] examines 16 different purposive sampling methods. However, the precise understanding of what sampling methods are part of purposive sampling often differs in the literature.

Teddlie and Yu [25] categorizes purposive sampling methods, including those found in [20], based on the strategies they describe. The authors identified four categories:

1. Sampling to achieve representativeness or comparability
2. Sampling special or unique cases
3. Sequential sampling
4. Sampling using multiple purposive techniques.

The first category handles samples that represent a broader group of cases or sets up comparisons among different types of cases. Six types of purposive sampling procedures are present this category: typical case sampling, extreme or deviant case sampling, intensity sampling, maximum variation sampling, homogeneous sampling, and reputational sampling. This category concentrates on both representative (e.g. typical case sampling) and contrasting cases (e.g. extreme case sampling).

The second category deals with special and unique cases rather than typical ones. It consists of four types: revelatory case sampling, critical case sampling, sampling politically important cases, and complete collection or criterion sampling.

The third category is based on sequential selection. Here, the main objective is to select units or cases based on their relevance. Often such techniques are used when the goal of the project is to generate theories or the sample evolves during data collection (known as gradual selection). Theoretical sampling, confirming and disconfirming cases, opportunistic sampling and snowball sampling are examples of sequential sampling.

The last category represents combination of two or more purposive sampling techniques. Such an approach might be useful depending on the complexity of the research study.

Table 1 shows purposive sampling technics applied in this paper.

Table 1. Purposive sampling methods used in this paper.

Method	Pro/Con	Example
TYPICAL	<ul style="list-style-type: none"> + Describes illustrative samples + Separates familiar from unfamiliar – Correctly identifying typical case 	R [†] Graduating students S [‡] Schools where nothing unusual is found
MAXIMUM VARIATION	<ul style="list-style-type: none"> + Covers cases that are extreme in one way and average in other ways – Depends on variety of the samples 	R People listening to radio Participants must differ from each other as much as possible (age, gender)
HOMOGENOUS SAMPLING	<ul style="list-style-type: none"> + Recruitment costs an efforts are low – Yields estimates that are not generalizable to the target population 	R Leadership in villages after natural disaster S Cases where leader is present and examine his qualities

[†]Research, [‡]Sample

3.2 Clustering Analysis

Clustering analysis is a multivariate classification technique used in quantitative research. The main idea of clustering is to group data into sets of related observations. Thus, observations in the same cluster are more similar to observations from other clusters. At the end of the analysis, every observation is part of a single cluster. The individual clusters are built by grouping observations based distance (linkage) function, which describes the similarity of the observations. Different linkage methods exist to measure the distance and dissimilarity between two clusters: minimal linkage, maximal linkage, average linking, or *Ward* linkage. The latter is based on sum of squares between all observations in the cluster and the centroid of the cluster. It aims to build homogeneous groups since clusters are joint based on the least variation [28].

The result of clustering is frequently presented as a tree like diagram called dendrogram. Each level of the dendrogram represents a segmentation of the data, and thus a different number of clusters. Finding the number of clusters is a manual step dependent on the specific research topic. Usually the most appropriate way to do this is by analysing the distances between the levels of the dendrogram and the resulting clusters [28]. The final number of clusters is then determined by the placing a *cut-off line*. When the distances between the levels is very high, i. e. a jump [27] exists, the cut-off line needs to be placed at a smaller distance.

4 Project Selection Approach

In this section we describe our project selection approach. Starting with the definition of a research topic, we derive characteristics describing the topic. Next, we calculate the characteristics for a large population of open source software (OSS) projects. By applying the introduced sampling techniques and clustering (see Sect. 3), we select a set of projects suitable to perform the case study defined by the initial research question.

4.1 Example Research Topic: Trace Links in Issue Tracking Systems

To demonstrate our approach, we first need to define a research topic for the case study. Usually, this initial step is not required and the topic is already at hand. As an example, we want to study trace links in open source issue tracking systems (ITS). In particular, we are interested whether existing trace links could be derived from properties of the linked issue artifacts or if the links introduce new information. For example, developers might link two issues because they share similar textual information or they belong to the same software component. In this situation, the links basically introduce no new information and could be potentially automatically created.

Research Topic: Do trace links in open source issue tracking systems introduce new information?

The goal of our approach is to identify a subset of open source systems suitable to perform the case study and provide answers to the research topic.

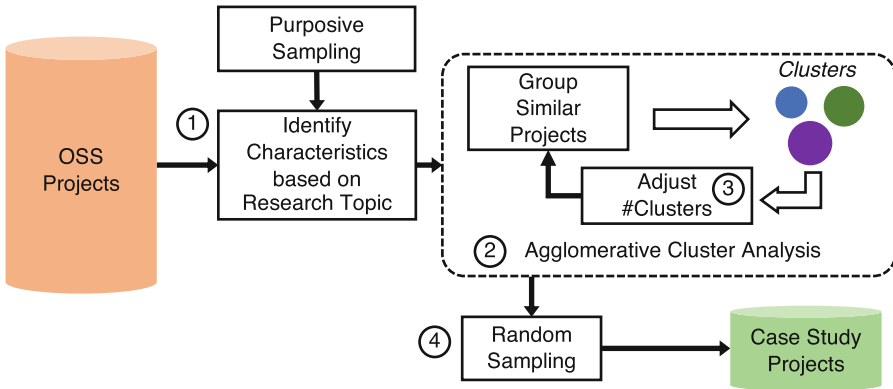


Fig. 1. Schema of the proposed approach to select projects for a case study.

4.2 Schematic Overview

The schema of the proposed selection method is depicted in Fig. 1. At first (①), characteristics that best describe the research topic need to be defined. Here, only relevant data should be included. Most analysts recommend the use of a limited number of clearly interpretable variables. If irrelevant information is not omitted, false interpretations can occur in later stages of the study or it can be much harder to analyse and interpret data. As we show, purposive sampling methods help in defining the characteristics. Next (②), a cluster analysis is performed to group similar projects. At this point, the number of clusters need to be determined. To ensure that the optimal amount of clusters, different placements of the cut-off (③) line as well as the validity of the results must be considered. The resulting clusters build a representative set of projects for the case study. To ensure diversity, projects from every cluster must be selected. Depending on the size of the clusters multiple projects can be chosen. The selection can be performed on a random basis (④).

4.3 Data Source

The defined research topic deals with open source projects using issue tracking systems. This topic defines the population, or *universe* [18], of the research. Based on popularity of ITS [15], we applied *homogenous case sampling* by selecting the Jira Issue Tracker [16]. Focusing on one ITS, further simplifies the data collection process described in the next section.

The Apache Software Foundation (ASF)¹ is the world largest open source foundation hosting 350+ projects [1]. Having Jira as a key tool, the ASF offers an Jira instance for every contained project. The projects have different sizes, use different programming languages, and stem from a variety of domains [1] resulting in an ideal source for searching for projects used in case studies.

Pig / PIG-4059
Pig on Spark

Details

Type: **New Feature**
 Priority: **Major**
 Affects Version/s: **None**
 Component/s: **spark**
 Labels: **spark**
 Hadoop Flags: **Reviewed**

Status: **CLOSED**
 Resolution: **Fixed**
 Fix Version/s: **spark-branch, 0.17.0**

People

Assignee: **Praveen Rachabattuni**
 Reporter: **Rohini Palaniswamy**
 Votes: **23** Vote for this issue
 Watchers: **73** Start watching this issue

Dates

Created: **17/Jul/14 14:54**
 Updated: **21/Jun/17 09:15**
 Resolved: **29/May/17 21:05**

Description

Setting up your development environment:
 0. download spark release package(currently pig on spark only support spark 1.6).
 1. Check out Pig Spark branch.
 2. Build pig by running "ant jar" and "ant -Dhadoopversion=23 jar" for hadoop-2.x versions

Issue Links

is related to **PIG-3446** Umbrella jira for Pig on Tez **RESOLVED**
PIG-4266 Umbrella jira for unit tests for Spark **CLOSED**
 supercedes **PIG-1734** Pig needs a more efficient DAG execution **RESOLVED**

Activity

All Comments Slack Work Log History Activity Transitions

▼ **Jarek Jarcec Cecho** added a comment - 17/Jul/14 16:34
 I would love to see ability to run Pig on top of Spark, hence I've voted for this JIRA 😊

▼ **Gregory Owen** added a comment - 11/Aug/14 19:13
 Added a link to the benchmark test wishlist. If there are any particular types of queries that you would like to see run during the performance comparison, add them to the doc.

Fig. 2. Example feature and its properties as represented in Jira issue tracker.

4.4 Data Representation and Acquisition

Figure 2 shows an example issue PIG-4059² from project PIG. An issue is the fundamental artifact used in Jira ITS. It summarizes a variety of information including *summary/title* (❶), *description* (❷), *meta data* (❸), *issues links* (❹), and *comments* (❺). The meta data *type* specifies the artifact (issue) type. Jira has a predefined list of issue types containing *bug*, *improvement*, *new feature*, and *task*. The link property allows to create trace links between different artifacts within a projects. Similar to issues, trace links are typed and the predefined list of link types includes *relates*, and *clones*. Selecting projects based on the links is the purpose of our case study.

The Jira platform offers a RESTful web service, which allows to interact with the system programmatically. We used the data collection process defined in [21]

¹ <https://www.apache.org>.

² <https://jira.apache.org/jira/browse/PIG-4059>.

to retrieve the issues and trace links of the projects hosted by AFS. The captured data is locally stored within a database for further processing. The project data collection was performed early in May 2018.

4.5 Sampling Strategies

All 350+ AFS projects are potential candidates for the example trace link study. We apply different purposive sampling techniques and apply metrics based on diversity and representativeness to identify a smaller, yet still information-rich subset to be used in the study.

The projects hosted by AFS are at different maturity levels. The AFS defines a rigorous *incubation process* [2] for the projects, which enter the program as *candidates*, then become *podlings*, and ultimately lead to a new Apache *Top-Level-Project (TLP)*. One criteria to become a TLP is to setup a specific project infrastructure including an issue tracker. Aiming for mature projects which follow the AFS guide lines, we use *homogenous case sampling* and only consider TLP projects. At the time of data retrieval, 219 Top-Level-Projects³ using Jira existed.

Next, we need to define characteristics (i. e. dimensions [18]) of the remaining projects. Later, the clustering is performed based on these characteristics.

When studying trace links, the most important characteristic is the amount of existing links in a project. Further characteristics are derived directly from the issues properties (see Fig. 2).

Textual artifact information, available in issue title and description, is often used in trace link analysis. It allows application of a wide range of textual similarity algorithms, such as TF-IDF [6], LSI [9, 22] or LDA [7, 10]. Thus we incorporate the existence of textual information in linked issues in the set of characteristics.

A previous study by Tomova et al. [26] identified the issue type as a relevant property when considering trace links. Especially the issue types *bug* (a problem which impairs or prevents the functions of a product), *feature* (a new feature of the product), and *improvement* (an enhancement to an existing feature) are of major interest.

To identify possible link patterns, i. e. combinations of issue types and link types, characteristics complementing textual analysis might be beneficial. As such, the component information is of special interest. Issues having the same type and component are similar to each other and may be more likely linked as duplicates or cloners. Further comments attached to issues are valuable, too. There, developers record additional details and collaborate with each other [3]. This activity might trigger the creation of trace links to other issues.

4.6 Data Preparation

After identification, the characteristics need to be calculated and prepared in order to apply the cluster analysis.

³ <https://projects.apache.org/projects.html>.

At first, we built the set $I_{BFI} \subset I$ of all issue I representing the bugs, features and improvements and for every project. Projects without all three issue types were discarded. Next, we calculated the set $I_{lnk,BFI}$ containing all $i \in I_{BFI}$ that are linked to another issue $j \in I_{BFI}, j \neq i$. Next, we calculated the following values for every project:

- $n_{BFI} = |I_{BFI}|$: the number of bugs, features and improvements in a project. Its value estimates the size of a project. There is no such thing as *the* size of a (software) project. The standard ISO/IEC 14143/1 [14] defines functional size by quantify the functional user requirements (FUR). Counting the respective issues is one way reflecting the number of requirements what the software or product shall do.
- $n_{lnk,BFI}$: the total number of links between issues of type bug, feature and improvement. This value also reflects the size of a project, but from a linking perspective.
- $mean_{com,BFI}$: the average number of comments for an issue of type bug, feature and improvement. It captures the collaboration activity in the respective project.
- $lnk_{BFI} = \frac{|I_{lnk,BFI}|}{|I_{BFI}|}$ which represents the percentage of linked issues of type bug, feature, and improvements.
- $lnk_{desc,BFI}$: the fraction of linked issues $i \in I_{BFI}$, that have non-empty description.
- $lnk_{comp,BFI}$: the fraction of linked issues $i \in I_{BFI}$, that have an assigned component.

Aiming for many links, we filtered the projects on a link basis using lnk_{BFI} . Being a fractional quantity, the value accounts for different project sizes. We removed those projects having lnk_{BFI} lower than the average of 13% of all TLP. After this filtering step, 93 out of the initial 219 projects remained.

We applied *typical case sampling* by placing constraints on lnk_{BFI} . The remaining characteristics are left untouched, aiming for *maximum variation sampling*.

4.7 Clustering Analysis

The characteristics n_{BFI} , $n_{lnk,BFI}$, and $mean_{com,BFI}$ greatly vary in magnitude. In order to make the data less sensitive to the existing differences, they need to be scaled [28]. We applied a normalization scheme, such that the values have zero mean and unit variance. The other three characteristics represent percentages and thus are already in a closed range. The resulting transformed dataset is used as input for agglomerative clustering. We chose the common *ward* linkage to create homogenous clusters. The outcome of the clustering is shown as dendrogram in Fig. 3.

5 Evaluation

Analysing the dendrogram shown in Fig. 3, an important decision is to place the cut-off line, which ultimately defines the number of clusters. Trying different

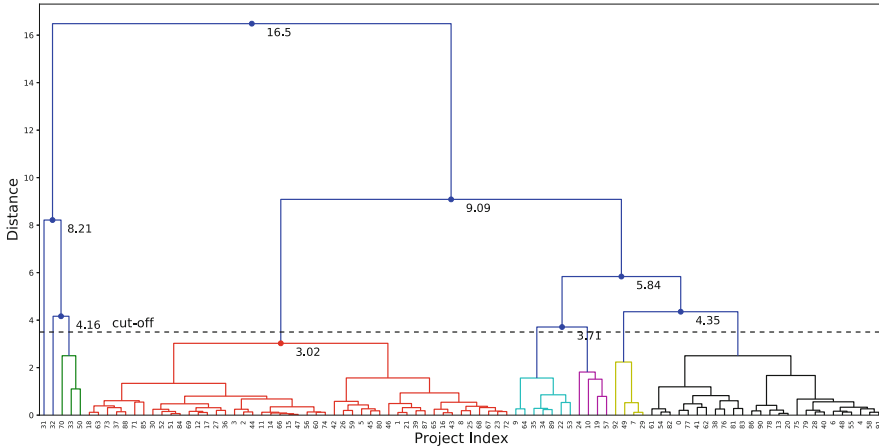


Fig. 3. Dendrogram resulting of hierarchical clustering with cut-off line at distance 3.5.

Table 2. Identified clusters, their size and example projects within the clusters.

Cluster	#Projects	Example projects
C_1	3	HIVE, LUCENE-CORE, SPARK
C_2	1	HBASE
C_3	1	HADOOP
C_4	47	ARCHIVA, APACHE COMMONS-LANG, SHIRO
C_5	7	CAMEL, DRILL, MAVEN, WICKET
C_6	4	CASSANDRA, CORDOVA, FLINK, OFBIZ
C_7	4	BOOKKEEPER, GORA, LOG4NET, ZOOKEEPER
C_8	26	GIRAPH, JENA, PIG, SUBVERSION

configurations, we finally settled to place the cut-off line at a distance of 3.5 and thus constructing 8 clusters. At this distance, no more major distance jumps are present in the tree. Additionally, the resulting clusters are well interpretable. Table 2 shows the number of projects (cluster size) as some example projects for each cluster. A common strategy to interpret the cluster meanings is to examine their differences compared to the grand means (macro averages) calculated from all 93 input projects [28]. The cluster mean values for all six characteristics along with the grand means are shown in Table 3. The interpretation of the clusters is as follows.

C_1 This cluster combines three projects, which have an equal size, i.e. minor variation, in terms of n_{BFI} and $n_{lnk,BFI}$, but are among the largest of all projects. The remaining characteristics are above average, except for the fraction of linked bugs, features and improvements ($lnk_{comp,BFI}$). The value is the lowest among all clusters.

Table 3. Cluster means of all projects. The color coding indicates, whether the respective value is **greater** or less then the grand mean (macro average).

Cluster	Cluster means					
	$\overline{n_{BFI}}$	$\overline{n_{lnk,BFI}}$	$\overline{mean_{com,BFI}}$	$\overline{lnk_{BFI}}$	$\overline{lnk_{desc,BFI}}$	$\overline{lnk_{comp,BFI}}$
C_1	16,536	3,619	11	0.28	0.92	0.21
C_2	15,274	2,547	25	0.22	0.95	0.12
C_3	31,073	10,407	18	0.39	0.96	0.25
C_4	1,225	172	5	0.21	0.92	0.14
C_5	6,781	948	6	0.21	0.94	0.18
C_6	9,626	1,288	12	0.18	0.95	0.13
C_7	1,116	197	17	0.23	0.97	0.18
C_8	2,610	390	10	0.20	0.94	0.14
Grand Mean	3,353	587	8	0.21	0.93	0.15

- C_2 This cluster consists of APACHE HBASE, whose characteristics are above the grand means.
- C_3 The third cluster represents the project APACHE HADOOP, which has by far the most issues and links. Further, HADOOP achieves the highest value in nearly all characteristics (except for $lnk_{comp,BFI}$).
- C_4 Cluster four consists of 47 projects, making it the largest one. However, all other characteristics are below the grand averages.
- C_5 The average characteristics of the seven projects in cluster C_5 are all above average, except for the low fraction of mean number of comments per bug, feature, and improvement.
- C_6 The sixth cluster consists of four quite large (n_{BFI} and $n_{lnk,BFI}$) projects. However, the mean number of comments per issue is low in these projects. The remaining characteristics are above average.
- C_7 This cluster combines four small projects. All projects have a high fraction of linked bugs, features, and improvements and thus $lnk_{desc,BFI}$ is the highest of all clusters. The remaining characteristics are also above the grand means.
- C_8 The second largest cluster has 26 small projects in terms of n_{BFI} and $n_{lnk,BFI}$. The percentage of bugs, features, and improvements that are linked (lnk_{BFI}) and those with an assigned component ($lnk_{comp,BFI}$) is below average. The remaining two characteristics are above average.

The projects in the eight clusters represent the characteristics that describe the topic under study. The number of the samples, software projects in our exemplary study, within clusters can be very large, e. g. like clusters C_4 and C_8 which together represent 80% of all projects. Depending on the situation, not every observation of a cluster must be considered for further analysis. However, to ensure diversity, at least one project from each cluster must be selected. To guarantee unbiased selection random sampling can be used to pick more projects.

No imperative exists, to actually chose from each cluster [28]. This depends on the design of the research. In our example, special care should be taken of

clusters C_2 and C_3 , representing the two projects APACHE HBASE and APACHE HADOOP. Each cluster consists of only one project, and thus no actual grouping occurred. Further, the seven calculated values for each cluster are near the upper and lower bounds compared to grand means. Therefore the two projects could be seen as outliers and may not be included in the final project selection.

6 Threats to Validity

In this section, we discuss threats to the validity of our study and how we mitigated them.

A threat to internal validity exists by choosing only projects of the Apache Software Foundation (ASF). We used this setup, because the project host defines the concept of *project*, i. e. it needs to have an issue tracking system, a web page, documentation and thus a certain level of maturity defined by ASF. However, this does not limit the scope of our approach researching trace links among artifacts in the projects.

We only consider Jira as issue tracking system. In our approach, we calculate seven project metrics based on issue types, links between issues, and textual description/comments of issues. These properties are not specific for Jira. Basically every issue tracking system, whether open-source like Bugzilla, Github Issues, MantisBT, Redmine, and Trac, or proprietary including HP Quality Center, IBM Rational, Microsoft Team Foundation Server, and JetBrains YouTrack provides these basic features [5]. Therefore the presented metrics can be calculated for these issue tracking systems as well. We settled on Jira based on its popularity [15] and ease of accessing the stored artifacts via the RESTful web service.

Our approach focused solely on open-source projects. A potential threat to external validity arises when we want to generalize our findings to a wider set of projects, including commercial development. Jira is used by over 60.000 customers according to its owning company Atlassian, including many large and well-know commercial enterprises [4]. We expect similar usage of issue artifacts in these projects in respect to our calculated metrics, i. e. creating, typing and linking of issue artifacts.

Another threat exists when generalizing our approach outside of traceability analysis. As depicted in Fig. 1, performing steps ① and ③ is inherently tied to a specific research question by defining case characteristics and placing the cut-off line. Other research topics may require different characteristics and cluster adjustments. However, the overall idea of purposive sampling, clustering and random selection is generalizable. Depending on the initial size of projects to choose from, the clustering step is optional.

7 Conclusion

When conducting a case study it is important to gather information-rich samples, such as software projects, that well represent the researched topic and give

scientists the opportunity not only to test hypothesis but also to learn from the selected samples. In this paper, we first investigated different strategies and important metrics when selecting samples for a study. Based on related work, we proposed an approach to select projects in a systematic way while taking into account important characteristics of the researched topic and metrics such as representativeness and diversity. As running example, we formulated a practical research question on the trace links in open source projects. We applied several purposive sampling techniques to describe trace link of the projects hosted by the apache software foundation. In this process, unnecessary information was filtered and characteristics characteristics strategically chosen. A hierarchical clustering algorithm was applied to identify patterns and group similar projects. Choosing projects out of the clusters via random sampling guarantees a variety of representative and diverse samples for later stages of the case study. Hopefully our work may be used as practical guideline to support other researchers for setting up their case studies.

In future work, we want to answer the stated research question, which only served as an interesting placeholder to drive the project selection method.

References

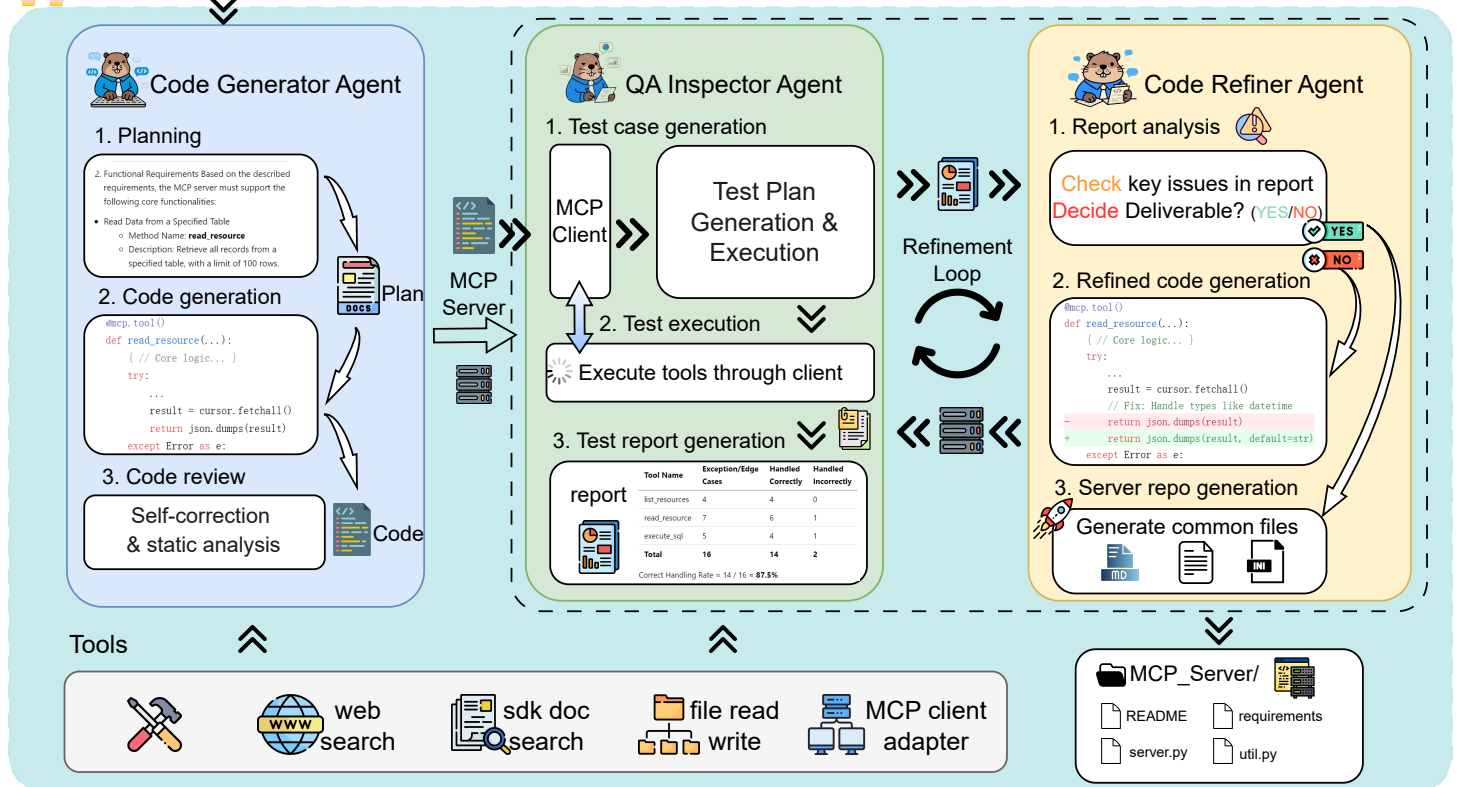
1. Apache Annual Report FY2018. <https://s.apache.org/FY2018AnnualReport> (2018). Accessed 29 Sept 2018
2. Apache Incubation Process. <https://incubator.apache.org/policy/process.html> (2018). Accessed 29 Sept 2018
3. Commenting on an Issue. <https://confluence.atlassian.com/jira064/commenting-on-an-issue-720416302.html> (2018). Accessed 29 Sept 2018
4. <https://www.atlassian.com/customers> (2019). Accessed 03 Jan 2019
5. Comparison of issue-tracking systems. https://en.wikipedia.org/wiki/Comparison_of_issue-tracking_systems (2019). Accessed 03 Jan 2019
6. Antoniol, G., Canfora, G., Casazza, G., De Lucia, A., Merlo, E.: Recovering traceability links between code and documentation. *IEEE Trans. Softw. Eng.* 28(10) 970–983 (2002)
7. Asuncion, H.U., Asuncion, A.U., Taylor, R.N.: Software traceability with topic modeling. In: *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering, ICSE 2010, Cape Town, South Africa, 1–8 May 2010*. vol. 1, ACM (2010)
8. Curtis, S., Gesler, W., Smith, G., Washburn, S.: Approaches to sampling and case selection in qualitative research: examples in the geography of health. *Soc. Sci. Med.* 50(7–8), 1001–1014 (2000)
9. De Lucia, A., Fasano, F., Oliveto, R., Tortora, G.: 20th IEEE International Conference on Software Maintenance (ICSM) (2004)
10. Dekhtyar, A., Hayes, J.H., Sundaram, S.K., Holbrook, E.A., Dekhtyar, O.: Technique integration for requirements assessment. In: *15th IEEE International Requirements Engineering Conference, RE 2007, 15–19th October 2007, New Delhi, India*. IEEE Computer Society (2007)
11. Flyvbjerg, B.: Five misunderstandings about case-study research. *Qual. Inq.* 12(2), 219–245 (2006)

12. Foucault, M., Palyart, M., Falleri, J., Blanc, X.: Computing contextual metric thresholds, ACM (2014)
13. Ishak, N.M., Bakar, A.Y.A.: Developing sampling frame for case study: challenges and conditions. *World J. Educ.* **4**(3), 29–35 (2014)
14. ISO/IEC 14143/1: Information technology, software measurement, functional size measurement, Part 1: definition of concepts. Standard, International Organization for Standardization, Geneva (2007)
15. Issue management tools - popularity ranking (2017). <https://project-management-zone/ranking/category/issue>
16. Jira Issue Tracking System (2018). <https://www.atlassian.com/software/jira>
17. Miles, M.B., Huberman, A.M., Huberman, M.A., Huberman, M.: *Qualitative Data Analysis: An Expanded Sourcebook*. Sage, Thousand Oaks (1994)
18. Nagappan, M., Zimmermann, T., Bird, C.: Diversity in software engineering research, ACM (2013)
19. Palinkas, L.A., Horwitz, S.M., Green, C.A., Wisdom, J.P., Duan, N., Hoagwood, K.: Purposeful sampling for qualitative data collection and analysis in mixed method implementation research. *Adm. Policy Ment. Health Ment. Health Serv. Res.* **42**(5), 533–544 (2015)
20. Patton, M.Q.: *Qualitative Evaluation and Research Methods*. Sage Publications, Thousand Oaks (1990)
21. Rath, M., Rempel, P., Mäder, P.: The IImSeven dataset. In: 25th IEEE International Requirements Engineering Conference, RE 2017, Lisbon, Portugal, 4–8 September 2017. pp. 516–519. IEEE Computer Society (2017)
22. Rempel, P., Mäder, P., Kuschke, T.: Towards feature-aware retrieval of refinement traces. In: 7th International Workshop on Traceability in Emerging Forms of Software Engineering, TEFSE 2013, 19 May 2013, San Francisco, CA, USA. IEEE Computer Society (2013)
23. Runeson, P., Höst, M., Rainer, A., Regnell, B.: *Case Study Research in Software Engineering - Guidelines and Examples*. Wiley, Hoboken (2012)
24. Suri, H.: Purposeful sampling in qualitative research synthesis. *Qual. Res. J.* **11**(2), 63–75 (2011)
25. Teddlie, C., Yu, F.: Mixed methods sampling: a typology with examples. *J. Mixed Methods Res.* **1**(1), 77–100 (2007)
26. Tomova, M.T., Rath, M., Mäder, P.: Use of trace link types in issue tracking systems. In: Proceedings of the 40th International Conference on Software Engineering: Companion Proceedings, ICSE 2018, Gothenburg, Sweden, 27 May–03 June 2018. ACM (2018)
27. Tryfos, P.: *Methods for Business Analysis and Forecasting: Text and Cases*. Wiley, Hoboken (1998)
28. Van Ryzin, G.G.: Cluster analysis as a basis for purposive sampling of projects in case study evaluations. *Eval. Pract.* **16**(2), 109–119 (1995)



Input: Natural Language Requirement

MCPybarra: Generation Workflow



MCPybarra: Automated Evaluation System

