Influence of distance measures and data characteristics on time performance in content-based and collaborative filtering datasets

Oleksandr Marchenko[1,†] and Maksym Shevchenko[1,*,†]

[1] *Taras Shevchenko National University of Kyiv, 60 Volodymyrska Street, Kyiv, 01033, Ukraine*

Abstract

This paper presents a comparative study on the time performance of various distance measures within the vector-space model, applied to content-based and collaborative filtering datasets. Euclidean distance, inner product, and cosine similarity were evaluated. A custom experimental framework was developed to assess these measures. The impact of dataset size, dimensionality, and the number of closest vectors returned by queries were analyzed.

Keywords

Chroma DB, collaborative filtering, content-based filtering, distance measures, recommender systems

1. Introduction

Recommender systems have become an integral part of our daily lives, influencing decisions in various domains. From suggesting movies and TV shows on platforms like Netflix, to recommending products on e-commerce sites such as Amazon, and even curating personalized music playlists on Spotify, these systems enhance user experience by providing tailored content based on individual preferences. Unlike traditional recommender systems that operate behind the scenes, dialog-based systems engage users in interactive conversations to gather preferences and provide recommendations through natural language interfaces. This approach offers a more human-centric and engaging way to receive personalized suggestions, potentially leading to improved user satisfaction and increased adoption of recommendation-driven applications [6].

Recommender systems typically rely on two main approaches: content-based filtering and collaborative filtering. Content-based filtering relies on the attributes of items (or their content itself) and user profiles to suggest similar items [15]. For instance, if a user has shown interest in science fiction movies, the system will recommend other science fiction movies based on the features such as genre, director, or actors. Collaborative filtering leverages the preferences and behaviors of multiple users to generate recommendations. It assumes that if users A and B have similar tastes, items liked by user A are likely to be appreciated by user B as well [17]. Both methods aim to discover new content users might enjoy, but they differ in their underlying assumptions and data requirements.

One common technique to implement both filtering types is representing user profiles and items as vectors. This method leverages distance or similarity measures between vectors to identify recommendations [2]. In content-based filtering, the system compares the vectorized user profile with vectorized item profiles to find similar items, ensuring recommendations align closely with the

Information Technology and Implementation (IT&I-2024), November 20-21, 2024, Kyiv, Ukraine

* Corresponding author.

† These authors contributed equally.

omarchenko@univ.kiev.ua (O. Marchenko); maksym_shevchenko@knu.ua (M. Shevchenko)

0000-0002-5408-5279 (O. Marchenko); 0009-0008-5104-9767 (M. Shevchenko)

user's preferences (Fig. 1, left). In collaborative filtering, the system compares vectorized user profiles with each other, identifying users with similar tastes to suggest items they have enjoyed (Fig. 1, right).

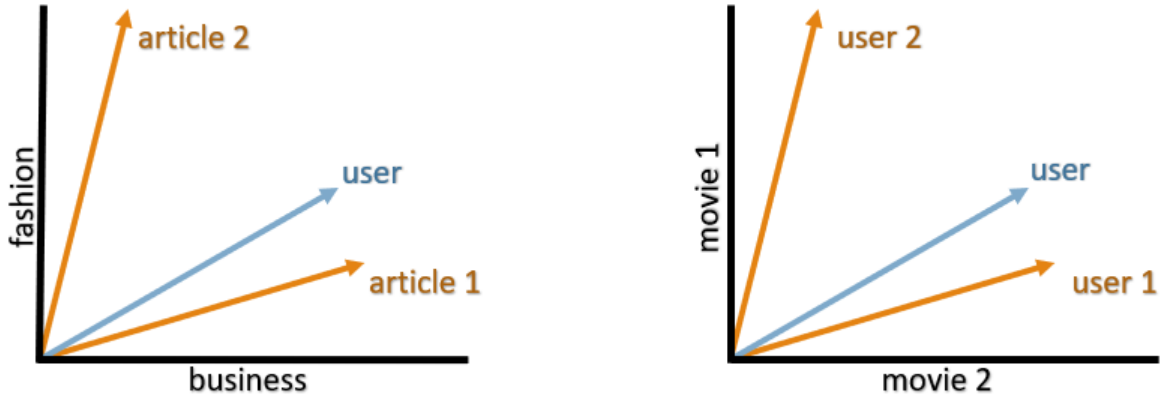

Figure 1: Representation of user profiles in the content-based (left) and collaborative filtering algorithms.

Generative models, such as GPT-like models, are widely used to develop chat-based applications due to their ability to generate human-like text. However, they are not always the ideal solution for recommender systems [19]. These models are trained on extensive corpora that include a vast amount of data unrelated to recommendations, which can dilute their effectiveness in this specific context. Additionally, they may lack relevant data that the system aims to recommend, and their training data may not be updated frequently, limiting their ability to provide recent items. To address these limitations, it is possible to fine-tune the model on relevant, domain-specific data or to incorporate Retrieval-Augmented Generation techniques.

Retrieval-Augmented Generation (RAG) enhances the capabilities of generative models by allowing them to access external knowledge sources [16], making them more effective for recommendation tasks. In a RAG-based recommender system, the model doesn't rely solely on pre-trained knowledge but instead retrieves relevant data dynamically from a connected database or vector store [1]. For example, when a user engages with a dialog-based system, the model generates a query based on the conversation. This query is then used to search the vector store, which contains pre-encoded user and item profiles. By retrieving relevant vectors – such as items similar to the user's preferences – the system can provide up-to-date and accurate recommendations. This approach combines the strengths of generative models with real-time, domain-specific data, ensuring the recommendations are both contextually relevant and current.

While numerous investigations have focused on estimating the impact of distance measures on recommendation accuracy [9], only few of them have considered the time performance of these measures [12]. Time performance is a critical factor in production systems, which must efficiently handle large volumes of data and serve numerous users simultaneously. Slow response times can degrade the user experience and strain system resources, which is unacceptable in real-time applications like recommender systems. Optimizing the time performance is essential to ensure that recommender systems can deliver timely and relevant recommendations, maintaining a seamless user experience even under heavy load. The goal of this study is to assess the time performance of different distance measures within vector stores. The results could provide valuable insights not only for improving recommender systems but also for any other domain that relies on vector stores, from information retrieval to natural language processing and beyond.

2. The experiment methodology

2.1. Hardware and software setup

The experiment was performed on a Windows 10-powered machine equipped with an AMD Ryzen 1500x 3.7 GHz processor, 16 GB of DDR4 3200 MHz RAM, and a 512 GB M2 SSD. While the time performance results may vary slightly based on the hardware configuration, the patterns and trends observed in the experiment should remain consistent across different setups, ensuring the generalizability of the results. All experiments were implemented using Python 3 (version 3.10.5), a widely adopted programming language in machine learning research due to its simplicity and extensive library support.

Chroma DB [4] was used as the vector store in this experiment for several reasons. First, it provides a Python library, making it straightforward to integrate into the experimental pipeline. Second, Chroma allows local data storage without the need for an external API, which helps eliminate delays related to internet connectivity. Finally, it integrates with LangChain [11], a popular Python framework for building controllable agentic workflows, including RAG applications, which enhances the flexibility and functionality of the experiment. Chroma natively supports three distance measures [4]: Squared L2 (Euclidean distance), inner product (IP) and cosine similarity (CS). In Chroma, these distances are defined as follows [4]:

$$D_{L2} = \sum_{i=1}^{n} (A_i - B_i)^2 ; \tag{1}$$

$$D_{IP} = 1.0 - \sum_{i=1}^{n} (A_i \times B_i) ; \tag{2}$$

$$D_{CS} = \frac{\sum_{i=1}^{n} (A_i \times B_i)}{\sqrt{\sum_{i=1}^{n} (A_i^2)} \cdot \sqrt{\sum_{i=1}^{n} (B_i^2)}}, \tag{3}$$

where n is the dimension of the vectors, A_i and B_i are the coordinates of vectors A and B, respectively. All these distances were estimated and compared.

2.2. Content-based filtering dataset

For the content-based filtering part of the experiment, the "All News 2.1" [5] dataset was employed. This dataset contains nearly 2.7 million news articles and essays from 27 American publications, spanning from 2016 to 2020 years. Each record in the dataset includes various attributes such as publication date, author, title, article text, URL, and more (Fig. 2). However, for content-based filtering, only the article (as the text to embed), title (as a metadata field), and row index (as the record ID) were used to populate the vector store.

Chroma's default embedder (Sentence Transformers all-MiniLM-L6-v2 model) [3] was utilized to convert each article into a 384-dimensional vector. To ensure manageable computational time, the dataset was divided into three subsets containing 25,000 (25K), 75,000 (75K), and 125,000 (125K) articles. Larger subsets were not created due to the extended time required to embed large datasets.

Queries to the vector store were executed using 100 pre-embedded texts, which were not part of the created datasets, simulating user profiles. Additionally, experiments were also conducted on generated random data.

row	ind	date	year	month	day	author	title	article	url	section	publication
1	1	09.12.2016	2016	12	9	Lee Drutman	We should take concerns about the health of liberal	This post is part of Polyarchy, an	https://w		Vox
2	2	26.01.2018	2018	1	26	Caroline Williams	You Can Trick Your Brain Into Being More Focused	If only every day could be like this.	https://w	Health	Vice
3	3	27.06.2019	2019	6	27	Mark Bergen	How to watch the Google I/O keynote live	Google I/O, the company's big	https://w		Vox
4	4	27.01.2016	2016	1	27	Tim Hume	China is dismissing unfavorable media reports	China is dismissing unfavorable media	https://w		
6	6	23.06.2019	2019	6	23	Emily Stewart	Elizabeth Warren called me! is turning into a	Elizabeth Warren is giving people a	https://w		Vox
7	7	02.05.2018	2018	5	2	Jessica DiNapoli,	Hudson's Bay's chairman's buyout bid pits retail	(Reuters) - The success of	https://w	Business	Reuters

Figure 2: Data sample of the News dataset.

2.3. Collaborative filtering dataset

For the collaborative filtering experiment, three sizes of the MovieLens (ML) [8] dataset were used: with 100,000 (100K), 1 million (1M), and 10 million (10M) ratings. The larger datasets were not utilized due to limitations in compute resources.

The dataset includes user-item interactions, where each interaction is represented in the format: user ID, movie ID, provided rating and timestamp (Fig. 3). Ratings are given on a 5-star scale with half-star increments, and the timestamp was omitted.

1	userId	movieId	rating	timestamp
2	1	1	4.0	964982703
3	1	3	4.0	964981247
233	1	5060	5.0	964984002
234	2	318	3.0	1445714835
262	2	131724	5.0	1445714851
263	3	31	0.5	1306463578
300	3	70946	5.0	1306463815
301	3	72378	0.5	1306464164
302	4	21	3.0	986935199

Figure 3: Data sample of the MovieLens 100K dataset.

For collaborative filtering, the user-item interactions should be represented as a utility matrix, where each row corresponds to a user, and the columns represent their interactions with all items (movies) from the dataset. Thus, an additional transformation was done to convert the entire data into utility matrix form (Fig. 4).

User ID	Movie ID						
	1	2	3	4	5	6	7
1	4.0		4.0			4.0	
...							
5	4.0						
6		4.0	5.0	3.0	5.0	4.0	4.0
7	4.5						
8		4.0					

Figure 4: The MovieLens dataset in a utility matrix form.

The width of the utility matrix – and, accordingly, the dimensionality of the vectors – varies depending on the number of items (movies) in the dataset. For the 100K dataset, the matrix included 610 users and 9,724 items; for the 1M dataset, it included 6,040 users and 3,706 items; and for the 10M dataset, there were 69,878 users and 10,677 items. This variation allowed for an estimation of how vector dimensionality impacts retrieval time. To query the created vector stores, 10 records were taken from the utility matrix itself, and 90 records were generated to simulate additional users and their interactions.

2.4. Methodology

For each dataset and distance function, 10 queries were conducted using 10 input vectors. The number of closest vectors returned by Chroma DB was parametrized, starting from 10 and increasing in increments up to 3,010 vectors. A smaller number of returned vectors is practical when the vector store directly powers a recommender system, providing a concise set of recommendations. On the other hand, a larger number of returned vectors is beneficial when the query results serve as input for subsequent stages in a recommendation pipeline, such as in hybrid recommender systems [14], where additional post-processing or combination with other algorithms may be required (Fig. 5).

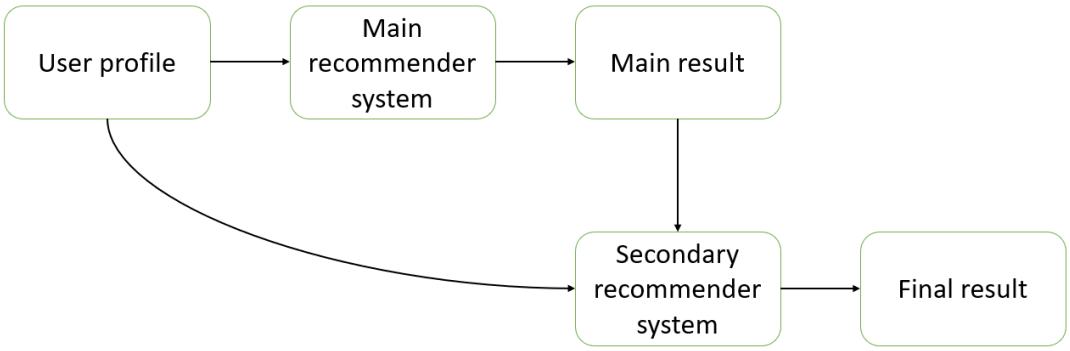

Figure 5: Diagram of a cascaded hybrid recommender system, where the output of the main recommender system is processed by the secondary recommender system.

3. Results

Due to the extensive volume of data generated during these experiments, only key results and trends are presented in this paper. Therefore, readers interested in the raw data and detailed results can access them through the provided link: https://github.com/Gurdel/distances_comparison_public/blob/main/results_raw.xlsx.

3.1. Results on the News dataset

As mentioned in the previous chapter, the content-based filtering experiments were divided into two groups: one using real data and the other using generated data. The time performance across both groups was nearly identical, indicating that the nature of the data did not significantly impact the computational efficiency. Additionally, all evaluated distance measures demonstrated similar response times. The minor variations observed in the results can likely be attributed to experimental error rather than any inherent difference in the algorithms.

Figure 6 illustrates the increase in average response time across all experiments conducted on the News datasets. Although the dataset size increased by 200% (from 25K to 75K) and 400% (from 25K to 125K), the average response time for all distance measures rose by only 36.5% and 58.5%, respectively (Table 1). This indicates that the system scales efficiently, maintaining relatively low increases in response time even as the dataset size grows significantly.

Table 1
An average response time in content-based filtering datasets (in seconds)

	News 25K	News 75K	News 125K
Cosine, real	3.95	5.73	6.65
IP, real	3.92	5.65	6.47
L2, real	4.11	5.63	6.49

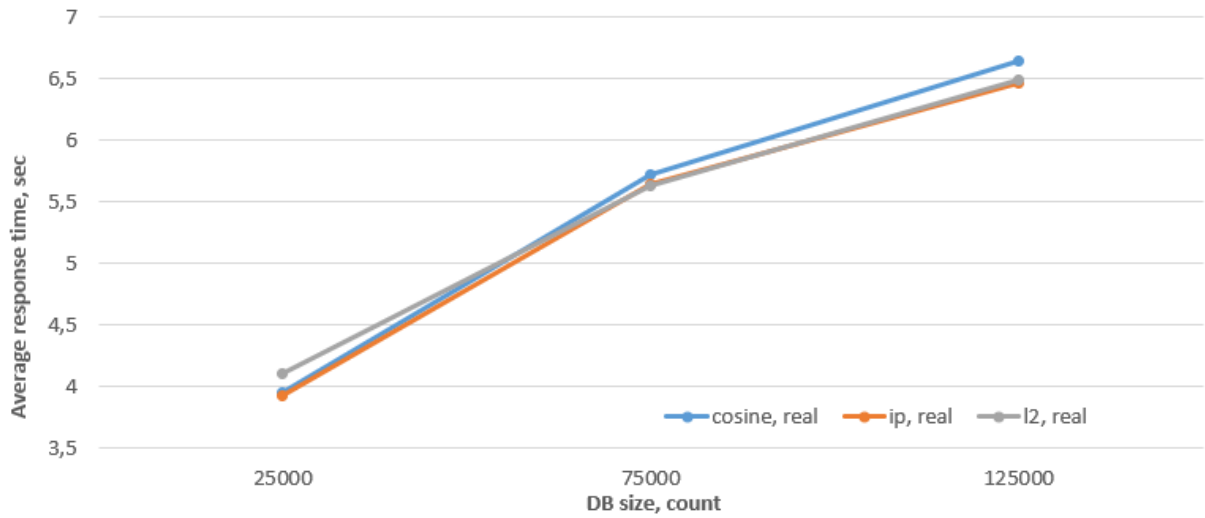

Figure 6: Response time increase in content-based filtering datasets.

Figure 7 shows the increase in response time as the number of closest vectors returned by each query increases. The results are shown for the IP measure on 25K, 75K, and 125K News datasets using real data for queries and 75K News dataset using generated data for queries. The increased response time at the beginning of the experiment with real data can be attributed to the internal processes of Chroma DB, such as initial indexing and caching mechanisms. When the experiment was repeated using generated query data, these spikes were absent, indicating that the system was already optimized after the initial runs. Across all database sizes and distance measures, the trends for response time remained almost linear, suggesting that the system scales predictably and efficiently as the number of retrieved vectors grows.

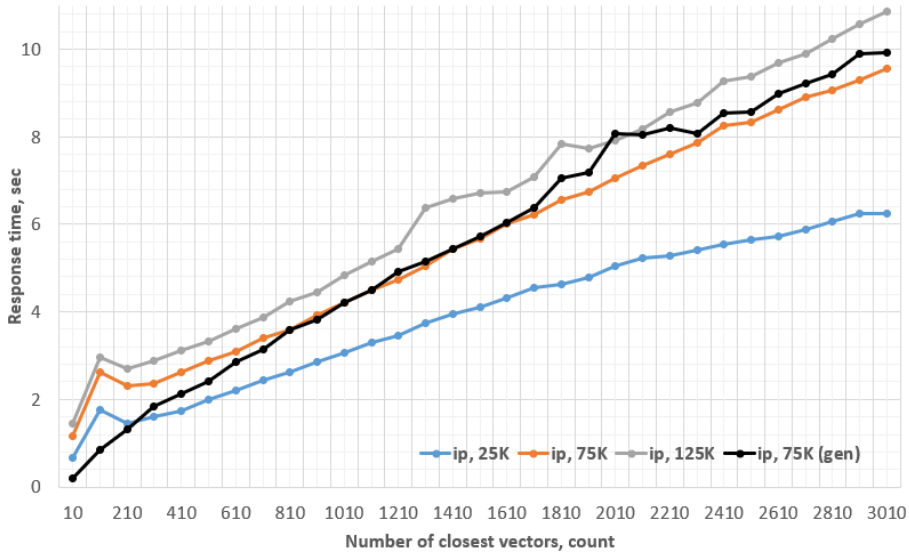

Figure 7: Response time trends for IP measure in the News datasets.

3.2. Results on the MovieLens dataset

Since the smallest collaborative filtering dataset, ML 100K, contains only 600 records (users), querying the vector store for more than 600 results would simply return all available data. During queries requesting between 10 to 510 results, the response time remained consistent across all distance measures and did not increase. Table 2 contains the average response time for all MovieLens datasets when querying up to 510 results. As expected, larger datasets resulted in longer response times. However, even in this small sample, a minor difference in performance was observed among the distance measures (Fig. 8). CS exhibited slightly worse response times compared to IP and L2, suggesting that CS may introduce additional computational overhead in the context of these datasets.

Table 2
An average response time in collaborative filtering datasets (in seconds)

	ML 100K	ML 1M	ML 10M
Cosine	0.39	0.67	2.49
IP	0.38	0.49	2.00
L2	0.39	0.48	1.63

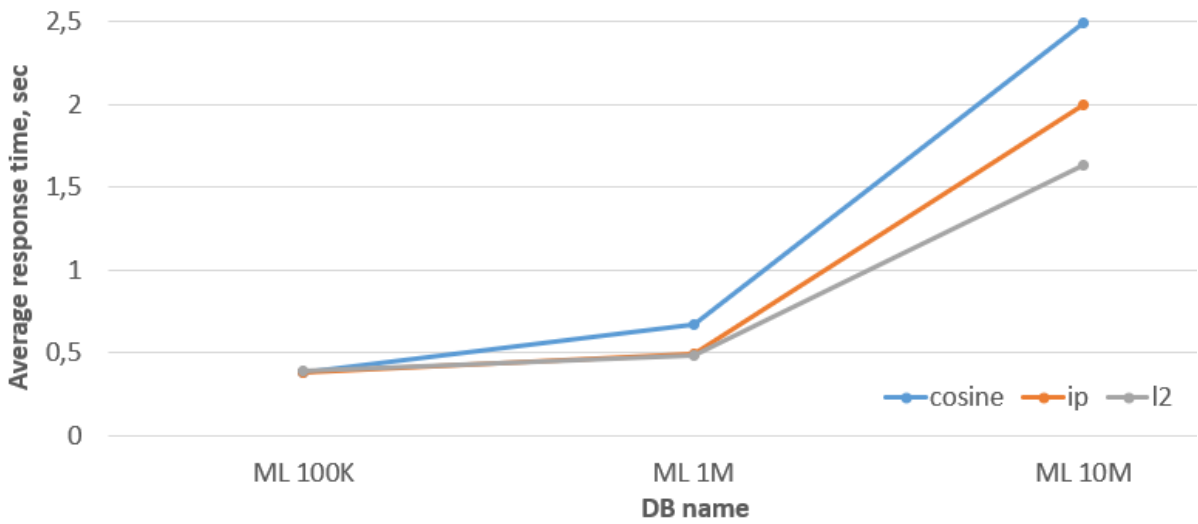

Figure 8: Response time increase in collaborative filtering datasets.

During experiments on the ML 1M dataset, all distance measures exhibited linear trends in time increases (Fig. 9). CS demonstrated a slightly worse average response time (1.47 seconds) compared to IP and L2 measures, which averaged 1.36 and 1.3 seconds, respectively. However, the differences between the measures are marginal, particularly when a higher number of vectors are returned.

In contrast to previous datasets, cosine similarity exhibited significantly worse performance on the ML 10M dataset compared to the other distance measures. As shown in Figure 10, the response time for CS increases more rapidly than for IP and L2, and both of them have almost linear trends. While L2 had an average response time of 2.84 seconds and IP averaged 3.27 seconds, CS took 4.4 seconds on average. This can be explained by the fact that, despite all these measures have similar time complexity O(n), CS requires more elementary operations to be computed according to its formula. This highlights that its performance degradation with high-dimensional data makes IP or L2 preferable for high-efficiency scenarios where response time is critical.

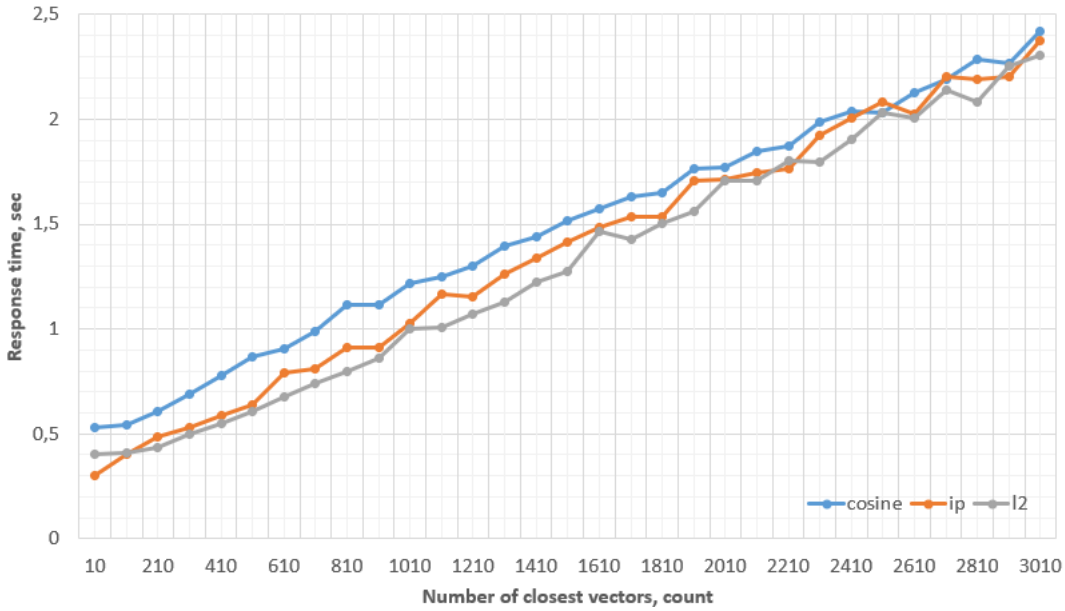

Figure 9: Response time trends in the MovieLens 1M dataset.

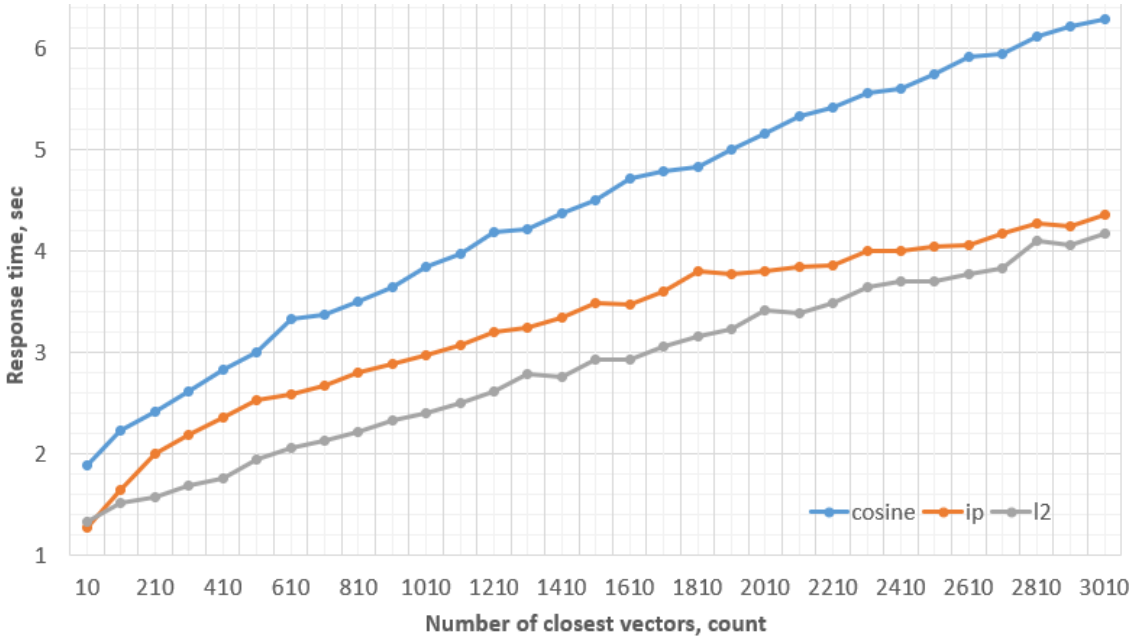

Figure 10: Response time trends in the MovieLens 10M dataset.

An important observation can be drawn from the comparison of response times between the ML 10M and News 75K datasets. Although these datasets have a comparable number of vectors (approximately 70,000 and 75,000, respectively), the dimensionality of the ML dataset is significantly higher (10,677 vs. 384). It was initially assumed that retrieval time on the ML data would be much slower compared to the News dataset. However, while the average response time for all measures in the News 75K dataset is 5.7 seconds, the ML 10M dataset showed only 3.5 seconds across all experiments. This discrepancy can be explained by the sparsity inherent in collaborative filtering data. With far more items than any individual user interacts with, the utility matrix has many empty (or zero) cells, allowing for substantial optimization in distance computations. This reduces the actual number of calculations needed, leading to faster retrieval times despite the higher dimensionality.

4. Conclusion

The experiments conducted on both content-based and collaborative filtering datasets provided valuable insights into the time performance of different distance measures. All compared distance measures showed linear trends in response time as the number of returned closest vectors increased. This consistent behavior across different datasets suggests that the system scales predictably, making it reliable for scenarios where larger sets of vectors need to be retrieved.

Across most datasets, cosine similarity, inner product, and Euclidean distance exhibited similar response times, with only minor differences in computational efficiency. However, as data dimensionality increased, significant variations in performance were observed. CS showed a notably higher response time on the high-dimension datasets, particularly in the ML 10M dataset, where its performance lagged behind IP and L2. This performance gap can be attributed to the higher number of elementary operations required by CS compared to the other measures.

Additionally, the comparison between the ML 10M and News 75K datasets highlighted the impact of data sparsity on performance, with the ML dataset showing faster response times despite its higher dimensionality. This underscores the importance of considering data characteristics when evaluating distance measures.

Moreover, an additional contribution of this study was the development of an experimental methodology and accompanying software to systematically evaluate the performance of various distance measures. This framework enabled consistent testing across datasets and can be extended for future research, providing a valuable tool for benchmarking distance-based retrieval methods in recommender systems.

4.1. Further steps

To further validate our findings and explore the scalability of different distance measures, it is essential to conduct experiments on larger datasets. By incorporating datasets beyond the tested in this study, it will be possible to better understand how different distance measures scale and whether the observed trends hold on even bigger datasets. This would provide more comprehensive insights into the scalability of vector stores and distance measures when applied to real-world, large-scale recommender systems.

To ensure the generalizability of our results, it is crucial to experiment with other vector stores beyond Chroma. Different vector stores may have varying optimizations and internal processes that could affect performance. By comparing multiple vector stores, we can identify the most efficient options for different recommendation scenarios.

Additionally, future research should explore the use of other similarity measures. While our study focused on natively supported by Chroma DB measures, comparing them with other measures will provide a comprehensive understanding of their performance characteristics and help identify the most suitable measures for specific applications.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] A. Alcaraz, Blending Fine-Tuning and RAG for Collaborative Filtering with LLMs, 2023. URL: https://ai.plainenglish.io/blending-fine-tuning-and-rag-for-collaborative-filtering-with-llms-3d71858485e4.

[2] C. Wang, Y. Shen, H. Yang, M. Guo, Improving Rocchio Algorithm for Updating User Profile in Recommender Systems. In: Lin, X., Manolopoulos, Y., Srivastava, D., Huang, G. (eds) Web

Information Systems Engineering – WISE 2013. WISE 2013. Lecture Notes in Computer Science, vol 8180. Springer, Berlin, Heidelberg, 2013. doi: https://doi.org/10.1007/978-3-642-41230-1_14.
[3] Chroma, Embeddings, n.d. URL: https://docs.trychroma.com/guides/embeddings.
[4] Chroma, Usage Guide, n.d. URL: https://docs.trychroma.com/guides.
[5] Components, All the News 2.0 — 2.7 million news articles and essays from 27 American publications, n.d. URL: https://components.one/datasets/all-the-news-2-news-articles-dataset.
[6] D. Pramod, P. Bafna, Conversational recommender systems techniques, tools, acceptance, and adoption: A state of the art review. Expert Systems with Applications 203 (2022) 117539, ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2022.117539.
[7] F. Fkih, Similarity measures for Collaborative Filtering-based Recommender Systems: Review and experimental comparison. Journal of King Saud University - Computer and Information Sciences 34(9) (2022) 7645-7669.
[8] GroupLens, MovieLens, n.d. URL: https://grouplens.org/datasets/movielens/.
[9] J. Joy, R. Renumol, Comparison of Generic Similarity Measures in E-learning Content Recommender System in Cold-Start Condition. In: Proceedings of the IEEE Bombay Section Signature Conference, December 2020. Bombay, India, 2020, 175-179. doi: 10.1109/IBSSC51096.2020.9332162.
[10] K. G. Saranya, G. S. Sadasivam, M. Chandralekha, Performance Comparison of Different Similarity Measures for Collaborative Filtering Technique. Indian Journal of Science and Technology 9(29) (2016) 1-8.
[11] LangChain, Vector stores, n.d. URL: https://python.langchain.com/v0.1/docs/modules/data_connection/vectorstores/.
[12] M. A. Baxla, Comparative study of similarity measures for item based top n recommendation. Unpublished thesis (Bachelor in Computer Science), National Institute of Technology Rourkela, 2014.
[13] M. Wijewickrema, V. Petras, N. Dias, Selecting a text similarity measure for a content-based recommender system: A comparison in two corpora. The Electronic Library 37(3) (2019) 506-527. doi: https://doi.org/10.1108/EL-08-2018-0165.
[14] R. Burke, Hybrid Web Recommender Systems. In: Brusilovsky, P., Kobsa, A., Nejdl, W. (eds), The Adaptive Web. Lecture Notes in Computer Science, vol 4321. Pp. 377-408. Springer, Berlin, Heidelberg, 2007. doi: https://doi.org/10.1007/978-3-540-72079-9_12.
[15] R. van Meteren, M. van Someren, Using Content-Based Filtering for Recommendation, 2000. URL: https://users.ics.forth.gr/~potamias/mlnia/paper_6.pdf.
[16] Restack, Recommendation Systems Using Rag, 2024. URL: https://www.restack.io/p/recommendation-systems-answer-using-rag-cat-ai.
[17] S. B. Belhaouari, A. Fareed, S. Hassan, Z. Halim, A collaborative filtering recommendation framework utilizing social networks. Machine Learning with Applications 14 (2023) 1-20.
[18] S.-B. Sun, Z.-H. Zhang, X.-L. Dong, H.-R. Zhang, T.-J. Li, L. Zhang, F. Min, Integrating Triangle and Jaccard similarities for recommendation. PLoS ONE 12(8) (2017) e0183570. doi: https://doi.org/10.1371/journal.pone.0183570.
[19] Y. Deldjoo, Understanding Biases in ChatGPT-based Recommender Systems: Provider Fairness, Temporal Stability, and Recency. ArXiv, abs/2401.10545 (2024) 27 pages. doi: https://doi.org/10.48550/arXiv.2401.10545.