UNIVERSITAT POLITÈCNICA DE CATALUNYA

FACULTAT D'INFORMÀTICA DE BARCELONA

BWave

Joint Project - Big Data Management Part 2

Spring 2022

Authors:
ABUSALEH, Ali G. A, *email*: ali.g.a.abusaleh@estudiantat.upc.edu
Lorencio Abril, Jose Antonio, *email*: jose.antonio.lorencio@estudiantat.upc.edu
Mayorga Llano, Mariana, *email*: mariana.mayorga@estudiantat.upc.edu

Professors: **Alberto Abelló, Sergi Nadal**

Contents

1 Introduction 2

2 Formatted Zone 3

3 Exploitation Zone 3

4 Analytical Zone: Recommender System 4

5 Streaming 5
5.1 Kafka 5
5.2 Top 3 elements per Category 6

6 References 7

A Running example 9

1 Introduction

In the rapidly evolving digital landscape, social networks have become indispensable platforms for connecting individuals and facilitating communication. BWave endeavors to elevate user interaction by developing a unique social network that revolves around a conversational chatbot. This chatbot is meticulously designed to comprehend user interests, enabling the system to provide tailored recommendations of other users with shared interests for exploration and engagement.

Our initial prototype involved the creation of a Django application that allowed users to register and log in manually or with their Facebook credentials. Conversations, posts, and their corresponding analyses were stored in MongoDB, serving as a repository for both Facebook posts and chatbot conversations. NLP analysis, powered by openAI, was employed to extract pertinent features from the stored data, which were then formatted and stored as JSON files in a separate MongoDB collection. Although the JSON format exhibited a semi-structured nature and a common context focused on users' interests, it still presented slight variations in data structure across different inputs.

To augment our implementation, we incorporated additional tools to enhance the system's efficiency and responsiveness. Celery, a task queue system, was leveraged to execute tasks asynchronously, ensuring optimal performance and preventing potential bottlenecks in user experience. Redis channels, on the other hand, were utilized to facilitate real-time communication between users and the chatbot, laying the groundwork for future extensions that would enable user-to-user interaction.

For this second phase of implementation, we introduced a Formatted Zone to filter the outputs from the Semi-Formatted Landing, we integrated the Exploitation Zone as a property graph database in Neo4J and developed two distinct approaches for the recommender system, one based on Jaccard similarity coefficient and the second one with an embedding-based technique, to offer users personalized recommendations based on their interests. Finally, streaming was incorporated to identify the trending topics of each category.

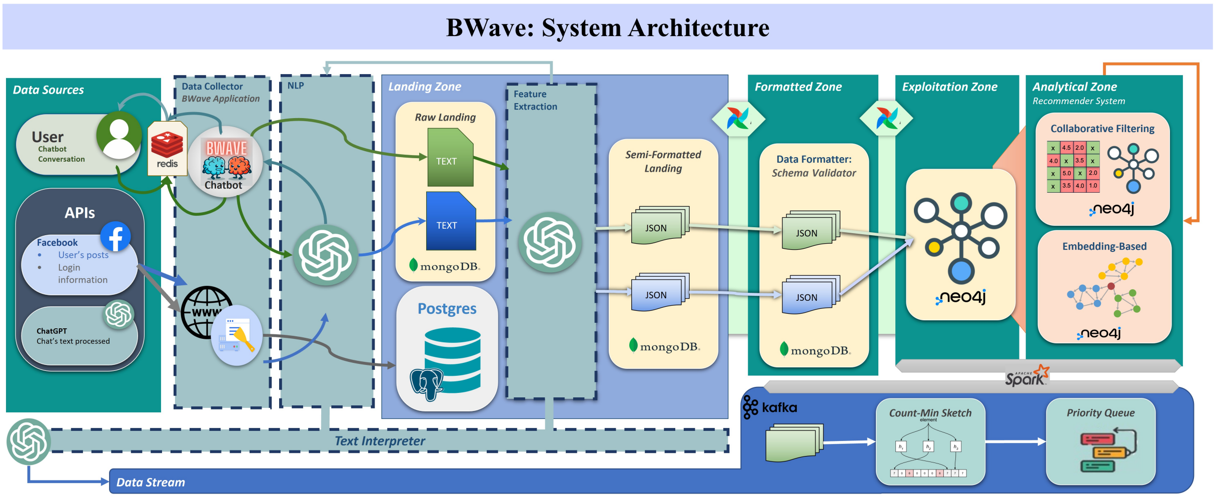


Figure 1: BWave Architecture

Note: Please use the following Github account to access the project if needed Username: "bwave23", Password: "upcjointproject".

2 Formatted Zone

In BWave's architecture, the formatted zone serves as an intermediate stage between the Landing Zone and the Exploitation Zone. In the Semi-Formatted Landing we can find the interpreted interests from our users' conversations stored in a JSON format. However, the structure of the JSON data provided by OpenAI's NLP analysis is not consistent, requiring the implementation of a data formatter to filter and validate only the JSON objects that adhere to the predefined structure with the objective of ensuring data quality and consistency before entering the Exploitation Zone.

The validated JSONS are then stored in collections with the defined schema validation. Schema validation in MongoDB provides a robust mechanism to define and enforce data integrity rules which allowed BWave to specify the expected structure, data types, and required fields for stored data. This validation process ensures that data inserted into the database meets the defined criteria, promoting data quality and consistency and reducing the risk of inconsistencies.

The current implementation focuses on validating specific data focused on Movies, Music and Food stored in a specific structure in the formatted zone to prioritize data quality for the exploitation zone for current recommendations. This implies limited Flexibility to handle less structured or unstructured data. However, future implementations could incorporate data less structured and more varied in order to make recommendations on broader topics and to allow for increased flexibility and accommodate a wider range of data formats.

Schema validation within the database can introduce performance overhead, especially when dealing with complex schemas and large volumes of data. However, in our current implementation, the simplicity of the schemas used outweighs the potential drawbacks. Since our system primarily focuses on graph maintenance rather than real-time or high-throughput processing, the advantages of schema validation are more prominent. Additionally, by validating data directly within MongoDB, we minimize data transfer between the database and external components, leading to improved processing efficiency and reduced latency. This approach ensures that the data is consistent, conforms to the expected structure, and enhances the overall reliability and performance of the system.

Automated data pipelines were implemented with Apache Airflow to extract the data from the Landing Zone and load the validated data into the Formatted Zone incrementally. It was additionally implemented between the Formatted Zone and the Exploitation Zone in Neo4J. This automation eliminates manual intervention and reduces the risk of human errors, ensuring consistent and accurate data preparation. This data transfer is scheduled to be executed daily, ensuring that the graph database stays up to date with the latest validated data. This seamless data transfer ensures the availability of real-time insights and recommendations within our social networking platform.

3 Exploitation Zone

The Exploitation Zone plays a critical role in our social network project, as it enables the utilization of data for running recommender systems and conducting various analyses. Once the data is validated and stored in the Formatted Zone, it is moved to Neo4J through Spark, where it will be incorporated into a graph, which will be further used by the Analytical Zone to run the recommender systems.

Firstly, the choice to adopt a property graph model in Neo4j for the Exploitation Zone is based on the inherent nature of social networks and the relationships present within the data. Property graphs provide a natural and efficient way to capture and model connections between users, locations, and interests, among other elements. By leveraging this model, we can uncover valuable insights about social interactions, communities, and user preferences.

Furthermore, the property graph's ability to facilitate fast and intuitive traversal of relationships offers a powerful foundation for developing recommendation systems in social networks. The graph structure allows us to incorporate multiple factors, including user preferences, social connections, shared interests, and past behaviors, resulting in more accurate and personalized recommendations. Leveraging Neo4j's graph algorithms and querying capabilities, we can implement sophisticated recommendation algorithms that enhance user engagement, enable content discovery, and foster meaningful interactions within our social network. Additionally, Neo4j's graph algorithms could allow us to perform other relationship-centric analyses for further implementations, such as identifying influencers, detecting social communities, or finding common friends, expanding the analytical capabilities of our platform.

Considering the anticipated growth of our social network and the associated increase in data volume, scalability becomes a crucial aspect. Graph databases, like Neo4j, are capable to handle large-scale graphs efficiently. Neo4j's index-free adjacency ensures fast data processing, allowing for real-time analytics and responsiveness, even with a significant influx of users, posts, and relationships. This scalability ensures that our platform can accommodate a growing user base while delivering the performance and user experience expected from a social networking platform.

Moreover, the choice of a property graph design provides the flexibility necessary to adapt to changing data structures without requiring extensive schema modifications. As our project evolves and new types of relationships, interests, or attributes emerge, we can seamlessly incorporate them into our data model. This flexibility ensures that our platform remains relevant and up-to-date, enabling us to keep pace with emerging trends, user interactions, and evolving user preferences. While managing data modeling complexity to ensure data consistency and integrity within a flexible graph structure is a challenge, we have addressed this concern in the earlier stages of data processing in the Formatted Zone. Therefore, the advantages of using a property Graph in Neo4J overweighs the potential drawbacks, making it an optimal choice for our project.

4 Analytical Zone: Recommender System

In the BWave recommender system, we leverage a combination of cosine similarity between embeddings and Jaccard similarity coefficient to provide enhanced friend recommendations, bringing together the strengths of both approaches.

In terms of Accuracy, Jaccard similarity coefficient captures user preferences based on behavior and collective patterns, while cosine similarity between embeddings captures semantic similarities and incorporates additional information, enabling the system to identify connections beyond explicit preferences. This combination can result in more accurate and personalized friend recommendations by considering both explicit and implicit user preferences. It can additionally promote recommendation diversity since Jaccard similarity coefficient focuses on collective behavior ensuring that recommendations cover a wide range of user preferences, while embeddings capture semantic similarities and can identify niche or specific interests.

Data sparsity is a common challenge in recommender systems, especially when the number of interactions is limited and can be a challenge at early stages, however, the combination of Jaccard similarity coefficient and cosine similarity between embeddings can further reduce this impact. The former can leverage the behavior of similar users to fill data gaps and generate relevant recommendations while the later can also handle sparse data by incorporating additional information and capturing implicit preferences. Together, these approaches can improve recommendation quality even when data is sparse.

Additionally, although we are mitigating this issue by retrieving users posts to get more information about their preferences, the integration of cosine similarity between embeddings allows BWave to handle the cold-start problem more effectively. When there is limited interaction data for new users or items, embeddings can leverage available attributes, metadata, or contextual information to generate initial representations. This enables the system to provide meaningful recommendations from the start, even for users or items with sparse data.

5 Streaming

The streaming component of our project plays a critical role in processing and analyzing data in real-time. In this section, we will discuss the key elements of our streaming infrastructure, including the use of Kafka and the implementation of the Count-Min Sketch (CMS) data structure with a priority queue.

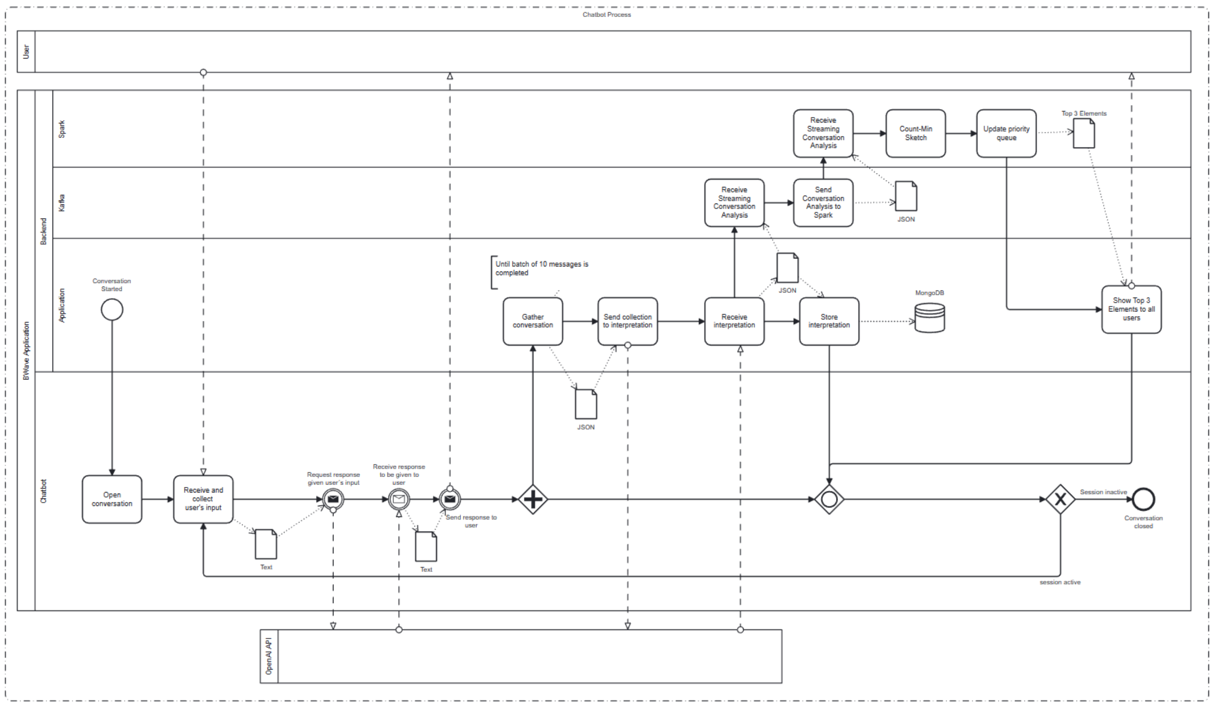


Figure 2: BWave Chatbot Main Process

5.1 Kafka

In our project, the decision to use Kafka as part of our streaming infrastructure is driven by several key advantages and considerations. Firstly, Kafka enables real-time data processing by providing low-latency message delivery, which ensures that data is processed as soon as it arrives in the system, allowing us to perform real-time analytics, monitoring, and decision-making. Secondly, Kafka provides built-in fault tolerance through data replication, ensuring data reliability and preventing data loss, which is essential for maintaining the integrity of our streaming pipeline. Thirdly, Kafka is designed to handle high volumes of data and can scale horizontally by adding more brokers to distribute the load, which will be crucial for our project, as we anticipate processing a large number of data.

5.2 Top 3 elements per Category

As new items entered the data stream, their corresponding counters in the CMS matrix were incremented based on the calculated positions from the hash functions. It is important to acknowledge that CMS is a probabilistic method for frequency counting in data streams, which means there is a possibility of overestimating the true frequency of an item and the accuracy of this approach can be affected by a large number of low-frequency items, leading to hash collisions within the CMS. To address this, we made a deliberate decision to prioritize including potentially non-trending topics in our top-3 list rather than omitting a potentially trending topic.

While the CMS allowed us to estimate item frequencies, it did not inherently provide item rankings. Therefore, we maintained a priority queue that was updated whenever an item's frequency was modified in the CMS. When the priority queue contained fewer than 3 items, the new item was directly added. However, if the queue was already at its maximum capacity, the frequency of the new item was compared with the minimum frequency in the queue. The item with the minimum frequency was removed, ensuring that the priority queue always held the top-3 items based on their frequencies.

This approach facilitated the estimation of item frequencies and real-time tracking of the top-3 items in each category, namely Movies, Food, and Music. Combining both techniques allowed us to strike a balance between space efficiency and accuracy when handling large data streams.

Every time the tops are updated, the results are posted to a different Kafka topic, to which our server listens to, to be able to broadcast them to all active users in real time.

6 References

1. Alberto Abello, Sergi Nadal. (2023) *Big Data Management*. Chapters 1 - 10.
2. Polikoff, I. (2020, August 19). Knowledge Graphs vs. Property Graphs – Part I. Retrieved from https://tdan.com/knowledge-graphs-vs-property-graphs-part-1/27140
3. Polikoff, I. (2020, September 16). Knowledge Graphs vs. Property Graphs – Part II. Retrieved from https://tdan.com/knowledge-graphs-vs-property-graphs-part-ii/27271
4. Foote, K. D. (2022, January 5). Property Graphs vs. Knowledge Graphs. Retrieved from https://tdan.com/knowledge-graphs-vs-property-graphs-part-ii/27271
5. Singh Walia, M. (2022, January 28). A Comprehensive Guide on Neo4j. Retrieved from https://www.analyticsvidhya.com/blog/2022/01/a-comprehensive-guide-on-neo4j-graph-database/
6. Neo4j. (n.d.). Top Ten Reasons for Choosing Neo4j. Retrieved from https://neo4j.com/top-ten-reasons/
7. Neo4j. (n.d.). Why Graph Databases? Retrieved from https://neo4j.com/why-graph-databases/
8. Javed, M. (2020, November 4). Using Cosine Similarity to Build a Movie Recommendation System. Retrieved from https://towardsdatascience.com/using-cosine-similarity-to-build-a-movie-recommendation-system-ae7f20842599
9. Elkhattam, A. (n.d.). Building a Content-based Recommender using a Cosine-Similarity Algorithm. Retrieved from https://a-elkhattam.medium.com/imdb-movie-recommendation-chatbot-942f84dfa0dc
10. Meor Amer. (2022, June 21). Article Recommender with Text Embedding, Classification, and Extraction. Retrieved from https://txt.cohere.com/article-recommender/
11. Maretha, A. (2022, July 31). Item Recommendation using Jaccard Coefficient. Retrieved from https://amaretha.medium.com/item-recommendation-using-jaccard-coefficient-f74df5c255b3
12. Putty, M. (n.d.). Measuring similarity in recommendation systems. Retrieved from https://levelup.gitconnected.com/measuring-similarity-in-recommendation-systems-8f2aa8ad1f44
13. Nandi, M. (2017, July 14). Recommender Systems through Collaborative Filtering. Retrieved from https://www.dominodatalab.com/blog/recommender-systems-collaborative-filtering
14. Ayub, M., Ghazanfar, M. A., Maqsood, M., & Saleem, A. (n.d.). A Jaccard base similarity measure to improve performance of CF based recommender systems. IEEE. Retrieved from https://ieeexplore.ieee.org/document/8343073
15. Bag, S., Kumar, S. K., & Tiwari, M. K. (2019). An efficient recommendation generation using relevant Jaccard similarity. Information Sciences, 475, 228-247. Retrieved from

https://doi.org/10.1016/j.ins.2019.01.023

16. Fkih, F. (2022, October). Similarity measures for Collaborative Filtering-based Recommender Systems: Review and experimental comparison. Retrieved from https://doi.org/10.1016/j.jksuci.2021.09.014

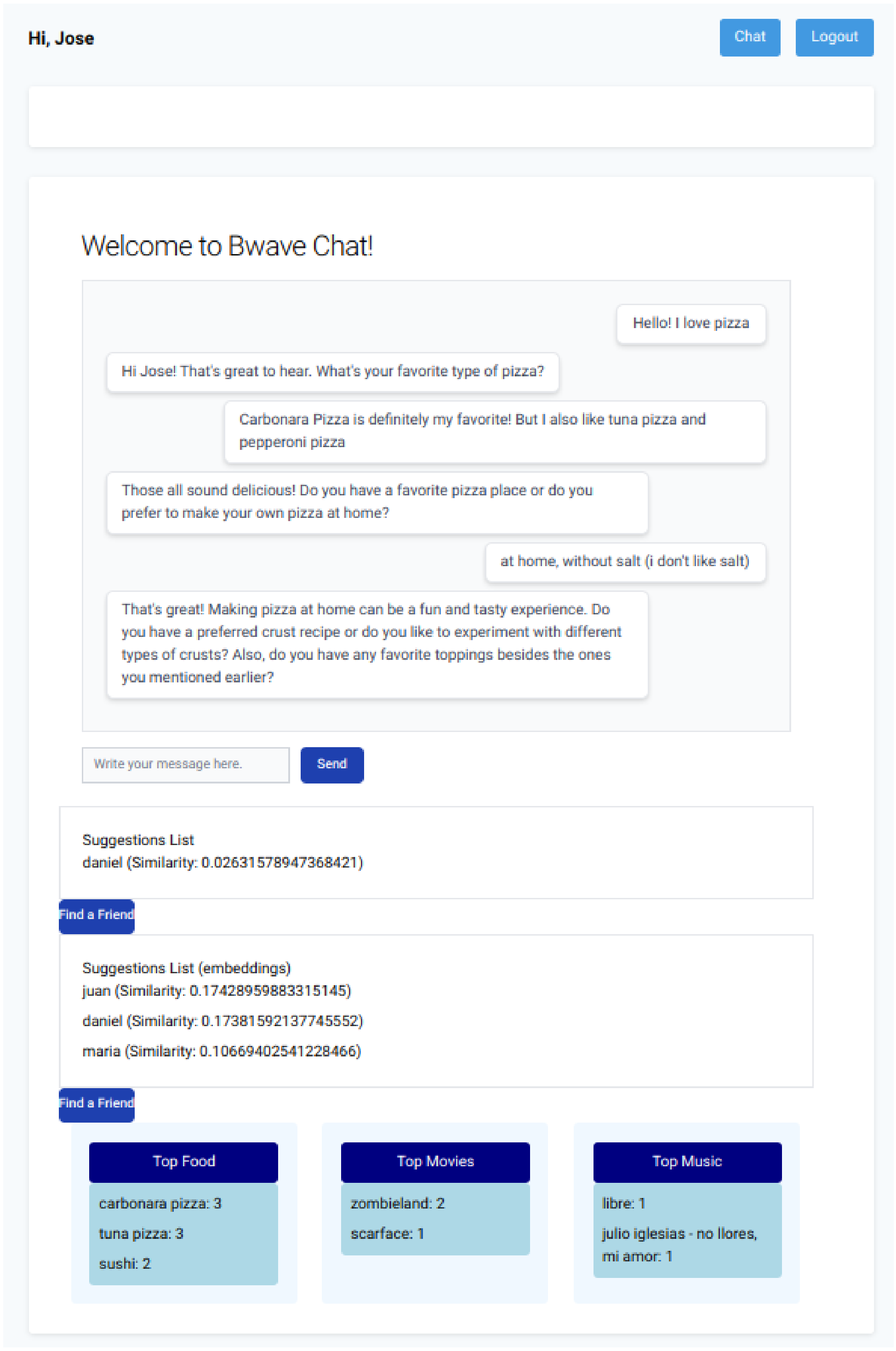
17. Wang, Y., Deng, J., Gao, J.,
& Zhang, P. (2017, August). A Hybrid User Similarity Model for Collaborative Filtering. Retrieved from https://doi.org/10.1016/j.ins.2017.08.008

18. Apache Kafka. (n.d.). Kafka Streams. Retrieved from https://kafka.apache.org/documentation/streams/

19. Babatunde, A. (2022, November 30). Why Spark Structured Streaming Could Be The Best Choice. Retrieved from https://www.netguru.com/blog/spark-streaming#:~:text=Apache%20Spark%20Structured%20Streaming%20is,with%20virtually%20no%20code%20changes.

Appendix A Running example

In Figure 3, one can observe the final outcome of our proof of concept. In this image, we can see how an user named Jose interacts with the bot, and is able to see the topics that people are liking the most, regarding food, music and movies. Whenever he wants, he can press the buttons to search for friends, getting recommendations based on what the system has learnt about him.



Welcome to Bwave Chat!

Hello! I love pizza

Hi Jose! That's great to hear. What's your favorite type of pizza?

Carbonara Pizza is definitely my favorite! But I also like tuna pizza and pepperoni pizza

Those all sound delicious! Do you have a favorite pizza place or do you prefer to make your own pizza at home?

at home, without salt (i don't like salt)

That's great! Making pizza at home can be a fun and tasty experience. Do you have a preferred crust recipe or do you like to experiment with different types of crusts? Also, do you have any favorite toppings besides the ones you mentioned earlier?

Write your message here.

Suggestions List
daniel (Similarity: 0.02631578947368421)

Suggestions List (embeddings)
juan (Similarity: 0.17428959883315145)
daniel (Similarity: 0.17381592137745552)
maria (Similarity: 0.10669402541228466)

Top Food
carbonara pizza: 3
tuna pizza: 3
sushi: 2

Top Movies
zombieland: 2
scarface: 1

Top Music
libre: 1
julio iglesias - no llores, mi amor: 1

Figure 3: Full running example