CS817: Big Data Tools and Techniques

University of Strathclyde

Table of Contents

Question 1 – Big Data Applications

Find two possible big data applications and describe their big data challenges.

Smart Cities

A smart city is a place where traditional networks and services are made more efficient with the use of digital services and solutions for the benefit of its inhabitants and businesses. One of its features includes the use traffic management systems to improve traffic flow, reduce congestion and enhance transportation systems. (European Commission, 2024).

A prime example is the traffic management systems implemented in London including SCOOT (Split Cycle Offset Optimisation Technique), which provides a 12% reduction in delay and an 8% reduction in stops for traffic where installed. (Mayor of London, 2010) Primarily, the system collects data from IoT (Internet of Things) devices embedded within the city's road infrastructure. Over the decades, it has been expanded to provide live data streams to analytics dashboards. These devices include traffic sensors, such as inductive loops placed at intersection and pedestrian crossing, which collect real-time data on traffic flow and congestion levels.

However, the implementation of such sophisticated systems presents several challenges. One primary challenge is managing the vast volumes of real-time data generated by the extensive network of IoT devices and sensors. Scalability and elastic infrastructure are essential, and many data streams contain mission-critical workloads for the system to function. Solutions like Apache Kafka have been proposed to address these needs, offering a robust platform for processing and managing vast data streams.

Another challenge is the integration of diverse data sources including the many different types of sensors. Smart cities are increasing turning to Data Lakes, such as those offered by Azure as well as implementing standardisation and interoperability protocols.

Rolls Royce Trent Engine Digital Twin

Rolls-Royce utilises digital twin technology for real-time monitoring on its modern Trent engines used on commercial aircraft around the globe. The digital twins process huge quantities of data from in-service engines in real-time. They also support the optimisation of maintenance schedules according to the conditions of individual engines. Problems can be detected earlier, preventing malfunctions, minimising disruption and maximising system availability. (Rolls-Royce, 2024)

The core application involves creating virtual replica, it uses real-time data streams from engine sensors that monitor a wide array of parameters, including engine performance, system health, environmental conditions during flights. The data is processed and analysed in the cloud, utilising the cloud platform Azure and tools such as Databricks for machine learning and AI analytics. (Hyscaler, 2024)

One of the primary challenges in this domain is handling the sheer volume of data with rapid real-time analysis. On November 4th, 2010, one of the four engines on Qantas Flight 32 suffered an uncontained failure four minutes after take-off from Singapore. Qantas technicians on the ground were able to access real time information and inform the pilots flying the aircraft the severity of the situation.

Another challenge is extending the lifespan of aircraft engines is crucial within the aviation industry. Rolls-Royce employs advanced machine learning techniques to dynamically update the lifespan predictions of each engine. These predictive models analyse a comprehensive array of data, including metrics like take-off thrust, external environmental factors such as air temperature, alongside engine usage patterns. Additionally, the models integrate historical maintenance data from across the fleet to refine and optimise maintenance schedules. (Hyscaler, 2024) This not only extends the operational life of each engine but also ensures maintenance activities are conducted with greater precision and efficiency, thereby enhancing overall engine reliability.

Questions 2 – Search Engine

A company is trying to build a search engine system to be used by researchers to find journals. Discuss how the web-based journal search engine can be created and used.

Creating a search engine system tailored for researchers to access and discover journals involves several complex processes and decisions. This section will outline the steps and considerations for developing such a system, focusing on index formation, algorithm selection, database management, search query processing, scalability solutions, factors influencing search results and user behaviour analysis.

The foundational task in developing a search engine is constructing an index, a process that significantly impacts data accessibility and searchability. The index forms the principal data structure used by the search engine for searching and relevance ranking. (Clarke, Cormack, & Buttcher, 2010) The creation of an index involves several steps, starting with the collection of documents, in this case would be the journal articles. Within document collection, we would assume that each document (journal) has a unique serial number known as the document identifier. (Drakos & Moore, 2009) During index construction, for each new document encountered we can simply assign successive integers. Once unique identifiers are assigned, each document is processed to break down the text into manageable pieces called tokens. The tokenisation step involves removing punctuation, converting all letters to lower or uppercase for uniformity, and discarding common stop words that are unlikely to be used in searches such as “and”, “the”, “of”. (Text preprocessing with Natural Language Processing (NLP), 2020) During this step, linguistic pre-processing would be conducted to stem words from their root form. This improves the search flexibility and recall, reduces the index size, normalises the vocabulary and enhances the relevance ranking.

Selecting the right algorithm for forming the index is pivotal. Inverted index algorithms are widely recognised for their efficiency in mapping terms to their document locations and facilitating quick searches. At its simplest, an inverted index provides a mapping between terms and their locations of occurrence in a text collection. (Clarke, Cormack, & Buttcher, 2010) For each term, it creates an entry in the index and associates it with a list of document identifiers where the term occurs. In this case where there is a large document collection, indexing can be distributed across multiple machines to enhance performance and scalability. This approach involves partitioning the document collection and running parallel instances of the indexing algorithm, each responsible for a subset of the documents. (Gormley & Tong, 2015)

Given the specific needs of the search engine, a non-relational database is the preferred choice as opposed to traditional relational databases due to their optimisation for data retrieval, with technologies such as Elasticsearch specifically designed for search operations, offering built-in indexing and search capabilities that significantly speed up query processing. (Gormley & Tong, 2015) NoSQL databases are designed to scale by distributing data across multiple servers; this is essential when managing the large volumes of data typical of journal article indexes. Unlike relational databases, NoSQL databases allow for a flexible data model and can accommodate varied metadata and content types without complex database scheme changes.

The initial step in processing a search query involves the search engine's analysis of the query to understand its structure and keywords. This critical phase includes identifying the main terms and operators such as 'AND', 'OR', and 'NOT', which are instrumental in refining the search to match user intent more accurately. Beyond these basics, the engine employs additional pre-processing techniques to ensure the query is optimally positioned for accurate results retrieval. This encompasses spell checking to correct misspelled words, and language detection to tailor the search process to the query's language, enhancing the relevancy of search results. As discussed previously, forming an index involves several steps which include tokenisation, where text is broken down into meaningful elements, and normalisation, which standardises these elements to reduce variability. Together, these processes ensure that the search engine can efficiently map user queries to the most relevant journal articles in the database. (Clarke, Cormack, & Buttcher, 2010) After initial query processing, which includes identifying keywords and applying spell checks and language detection, the engine leverages the TF-IDF measure to rank documents. TF-IDF values are calculated by considering both the frequency of a term in each document and its inverse frequency across all documents in the index. This dual approach helps prioritise articles that not only contain the query terms but do so in a context that's uniquely significant compared to the broader corpus.

When many users run concurrent web searches on a search engine it can significantly impact the system's performance. This can lead to slower response times due to an increased load on the server resources such as CPU, memory, disk I/O operations, required to process and return search results. As discussed previously we would be using a NoSQL data, from this we could implement load balancing which involves distributing incoming search requests across multiple servers to help manage the load effectively. Additionally, elastic scalability can be implemented through a service such as Kubernetes, to scale up or down resources based on the demand. Caching in another system design implementation that can reduce the load on the search engine. It works by storing the results of common queries in a cache, the system can serve these results quickly without having to re-process the query against the entire index.

There are several factors beyond the specific words used in the web search query that can influence the results returned by the search engine. One would be the user context and personalisation, this would include the users' past search history, location and profile information which would be used to tailor the search results to prioritise journal or articles that align with their field of study or location. Another factor would be the temporal relevance with more recent publications being ranked higher in the search results. Highly cited papers or those published may be ranked higher as these metrics are considered indicators of quality and relevance in academic research.

There are several ways that the user's behaviour could be analysed, the most notable would be search query analysis. Search query analysis involves the understanding and examining of the specific words or phrases that users enter into search engines when looking for information. (AIContentfy, 2023) This analysis can guide the refinement of the search algorithm and the user interface. Tracking user engagement metrics such as click-through rates, bounce rates and session durations could also provide valuable feedback on the relevance and quality of various journal articles.

Question 3 – Recommender System

The company now extend the service by recommending journals to researchers based on their research area and interest.

Recommender systems are sophisticated software tools designed to offer suggestions tailored to the user's preferences and needs, often presented as ranked lists to enhance user decision-making. (Ricci, Lior, Kantor, & Shapira, 2011) For academic purposes, such a system can significantly streamline the process of identifying the most relevant research journals for individual researchers by leveraging a combination of content-based and collaborative filtering techniques. This hybrid approach enables the system to analyse the content of journals (e.g., topics, keywords, citation impact) and incorporate user interactions (e.g., previous journal submissions, citation networks) to deliver personalised journal recommendations. This methodology is grounded in the principles outlined by Ricci et al. (2011), who emphasise the versatility and effectiveness of recommender systems in diverse fields, suggesting their potential for impactful applications in academic research dissemination and discovery. (Ricci, Lior, Kantor, & Shapira, 2011)

Collecting user preferences and interactions is one of the most important parts for a recommender system. This process can be challenging, as users often resist the manual entry of their preferences or ratings, seeking to minimize their time and effort commitment. (Zhang & Im, 2002). However, websites can avoid this reluctance by utilising cookies and account information to capture data like user order history, duration spent reading articles, user recommendations and ratings, browsing history. Comprehensive data collections enable the development of rich user profiles, which are essential for personalised recommendations. (Tuzhilin & Adomavicius, 2005) For journals, capturing information is similarly nuanced, focusing on metadata like publication frequency and date, impact factor, subject areas and the citation network of published articles.

The information captured from user interactions and journal metadata can be utilised to construct recommendations through methods like item-based and user-profile comparisons, both employing the vector space model to analyse similarities among journals or user preferences. Item-based comparisons focus on recommending journals like those a user has previously engaged with, while user-profile comparisons suggest journals favoured by similar users, enhancing personalisation through collaborative filtering. For handling more complex and nonlinear data relationships which may be required in the journal example, deep learning and neural network models emerge as a solution, capable of identifying patterns in large datasets to deliver highly accurate recommendations. These approaches collectively enable the creation of personalised recommendation systems that matches users with journals aligned with their research interests and behaviours.

Matrix factorisation allows user-item rating matrix to be decomposed into smaller user-concept matrix and item-concept matrix, it maps both "users and items to a joint latent factor space of dimensionality" (Ricci, Lior, Kantor, & Shapira, 2011). It is a core algorithm in collaborative filtering techniques, which are pivotal in recommender systems for predicting user preferences based on past performances. In matrix factorisation method decomposes a large user journal matrix – into two smaller matrices representing latent features of users and journals through techniques such as Singular Value Decomposition (SVD) as shown on the diagram below.

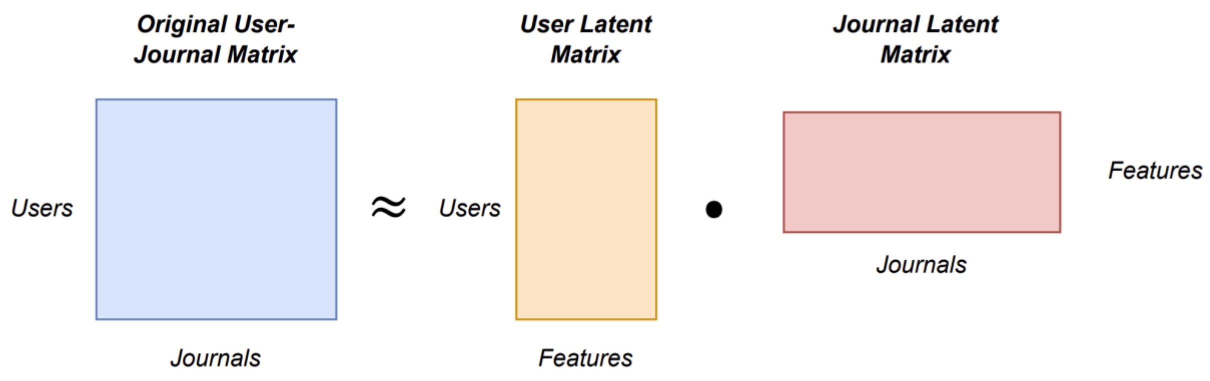

Figure 1 - Matrix Factorisation

To predict the interest a user might have in a specific journal, the dot product of the user's latent feature and the journal's latent feature is computed.

$$\hat{r}_{ui} = p_u^T q_i + b_u + b_i + \mu$$

- $\hat{r}_{ui}$ is the predicted rating of a user μ for journal i.
- p_μ, is the latent feature vector representing user u.
- q_i is the latent feature representing journal i.
- b_u is the user bias term, accounting for the tendency of user u to rate items higher or lower than the average.
- b_i is the item bias term, reflecting the tendency of item i to be rated higher or lower than the average.
- μ is the global average rating across all users and items.
- The subscript T denotes the transpose of p_μ, ensuring the dot product is computed correctly.

To optimise the recommender model, the root mean square error (RMSE) is minimised, with an optimal number of latent factors ensuring a balance between complexity and overfitting.

The results from a user query in a recommender system would be ordered by the predicted relevance score, which represents the likelihood of a user's interest in a journal. As discussed above, the use of matrix factorisation techniques where the dot product of user and journal latent feature vectors indicates preference strength. Journals with the highest scores would be ranked at the top suggesting they are the most relevant to the user's query and interests.

The efficiency of a recommender system can be evaluated by how well it identifies and ranks documents (journals) relevant to the user's query. The primary metrics to measure this are precision and recall. Precision assesses the proportion of recommended journals that are relevant, while recall measures the proportion of relevant journals that have been recommended by the system. (Makaeb, 2017)

The mean average precision represents a standard method for summarising the effectiveness of a recommender system over several recall measures and is particularly useful when there's a set of queries, we want to evaluate (Clarke, Cormack, & Buttcher, 2010). It is a single digit value and is calculated by taking the mean of the average precisions computed for each query, where each average precision of a single query is the average of the precision values obtained for the set of top k journals that have been retrieved. As an example, consider when a ranked list of journals is returned. If a query has 5 relevant journals out of 100 possible journals, and the system returns 2 relevant journals in the top 10 results then the precision is 0.2 at 10. If all 5 relevant journals are returned in the top 20 results, then the precision is 1 at 20. The average precision would be the average of precision values calculated at each point where a relevant journal is retrieved.

Accuracy can also be calculated using metric like Root Mean Square Error (RMSE) and Mean Absolute Error (MAE) if rating predictions are involved. If the system predicts user ratings for journals, then the accuracy of these predictions can be gauged by comparing predicted ratings against actual user-provided ratings. As an example, if a user rated a journal 4.5, but the system predicted 4.0, then the error for this single prediction is 0.5. Squaring this value, summing each squared errors across all predictions, and then taking the quare root of the average gives the RMSE, which indicates the system's prediction accuracy.

References

AIContentfy. (2023, November 6). Retrieved from Understanding the basics of Search Query Analysis: https://aicontentfy.com/en/blog/understanding-basics-of-search-query-analysis#:~:text=Search%20query%20analysis%20is%20vital,%2C%20preferences%2C%20and%20pain%20points.

Clarke, C., Cormack, G., & Buttcher, S. (2010). *Information Retrieval.* Cambridge: MIT Press.

Drakos, N., & Moore, R. (2009). *Introduction to Information Retrieval.* Stanford: Cambridge University Press. Retrieved from Cambridge University Press: https://nlp.stanford.edu/IR-book/html/htmledition/about-this-document--1.html

European Commission. (2024, March 28). *Smart Cities*. Retrieved from European Comission: https://commission.europa.eu/eu-regional-and-urban-development/topics/cities-and-urban-development/city-initiatives/smart-cities_en

Gormley, C., & Tong, Z. (2015). *Elasticsearch The Definitive Guide.* Sebastopol: O'Reilly.

Hyscaler. (2024, March 28). *Digital Twins in Aerospace: Designing the Next Generation*. Retrieved from Hyscalder: https://hyscaler.com/insights/digital-twins-aerospace-powering-next-gen/

Makaeb, M. (2017, August 17). *Recall and Precision at k for Recommender Systems*. Retrieved from Medium: https://medium.com/@m_n_malaeb/recall-and-precision-at-k-for-recommender-systems-618483226c54

Mayor of London. (2010, July 18). *SCOOT traffic control system*. Retrieved from London.Gov: https://www.london.gov.uk/who-we-are/what-london-assembly-does/questions-mayor/find-an-answer/scoot-traffic-control-system

Ricci, F., Lior, R., Kantor, P., & Shapira, B. (2011). *Recommender Systems Handbook.* New York: Springer.

Rolls-Royce. (2024, March 28). *What are Digital Twins*. Retrieved from Rolls Royce Innovation: https://www.rolls-royce.com/innovation/digital/digital-twin.aspx

Text preprocessing with Natural Language Processing (NLP). (2020, September 26). Retrieved from Linkedin: https://www.linkedin.com/pulse/text-preprocessing-natural-language-processing-nlp-germec-phd/

Tuzhilin, A., & Adomavicius, G. (2005). Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 734-739.

Zhang, Y., & Im, I. (2002). *Recommender Systems: A Framework and Research Issues.* Newark: New Jersey Institute of Technology.