SAPIENZA
UNIVERSITÀ DI ROMA

Language Models for Information Quality: Methods and Applications

Department of Computer, Control and Management Engineering
Antonio Ruberti

National PhD Program in Artificial Intelligence – XXXVII Cycle

Candidate

Jerin George Mathew
ID number 1996766

Thesis Advisors

Prof. Donatella Firmani
Prof. Massimo Mecella

January 2025

Thesis defended on 23/01/2025
in front of a Board of Examiners composed by:

Prof. Antonio Chella, Università degli Studi di Palermo

Prof. Stefania Costantini, Università degli Studi dell'Aquila

Prof. Gabriella Pasi, Università degli Studi di Milano-Bicocca

External Reviewers:
Prof. Laure Berti-Equille, French Institute of Research for Sustainable Development
Prof. Letizia Tanca, Politecnico di Milano

Language Models for Information Quality: Methods and Applications
Ph.D. thesis. Sapienza – University of Rome

This thesis has been typeset by LaTeX and the Sapthesis class.

Website: http://diag.uniroma1.it/en/users/jerin-george_mathew

Author's email: mathew@diag.uniroma1.it

To my brother

Contents

Extended Abstract

Data and information form the foundation of modern decision-making systems, powering operational processes and shaping strategic insights. While data consists of raw, unprocessed facts, information is the result of contextualizing and processing this data into meaningful, actionable insights. As organizations increasingly rely on diverse data sources—ranging from structured databases to unstructured text—the challenge of ensuring high data and information quality becomes crucial [158].

Data quality (DQ) focuses on the technical integrity of data, which can be assessed with specific measures, such as *accuracy* and *consistency* [185, 158]. Accuracy reflects how closely data values agree with the true or accepted values of the corresponding real-world entities, whereas consistency refers to the degree to which different representations of the same data agree with one another across datasets. However, as the variety and complexity of data increase, traditional data quality measures need to be extended. *Information quality* (IQ) broadens the scope by assessing how effectively processed data supports decision-making, incorporating dimensions such as *relevance*, i.e. the degree to which data meets the needs of its users and *interpretability*, which reflects how easily users can comprehend and apply the data [158]. These IQ dimensions become particularly critical when dealing with unstructured data, such as user-generated content or clinical notes, where meaning and relevance are not as easily quantified as in structured datasets.

To achieve high standards of information quality, several key techniques can be employed. *Data cleaning*—the process of detecting and correcting corrupt or inaccurate records from a dataset—is instrumental in ensuring the consistency and accuracy of the information [189]. Within data cleaning, *Entity Resolution* (ER) is a critical technique for identifying and merging duplicate or inconsistent records. Whether reconciling patient records in healthcare or customer profiles in e-commerce, ER ensures that systems rely on unified and accurate data, improving the quality and reliability of the information generated.

Moreover, as data systems evolve to handle complex inputs, *ethical data quality* [79] emerges as a key concern, particularly in ensuring fairness, transparency, and accountability. *Fairness* is an ethical dimension that ensures data and algorithms do not exhibit bias, which is especially critical in applications such as recommendation systems and hiring platforms [213]. In systems reliant on subjective data, such as reviews or user feedback, fairness ensures that information quality is not compromised by overrepresented or biased data sources. By integrating fairness into both data collection and processing, we strengthen the trustworthiness of the insights derived from these systems [79].

Uncertainty Estimation plays a key role in enhancing the reliability of information,

particularly in scenarios involving incomplete and noisy data. This is especially relevant in domains such as open-domain question answering and real-time fact verification, where responses must adhere to high standards of factual accuracy and contextual relevance. Quantifying uncertainty associated with retrieved or generated information enables systems to identify instances where data may lack reliability or contain conflicting details. This in turn can allow users to prioritize data verification, ultimately supporting more informed decision-making [1].

Another essential aspect of information quality is the ability to meet user demands for information in real-time. This concept of *data on demand* refers to the dynamic retrieval of relevant data in response to specific queries or needs. In many applications, users require timely and accurate information to support decision-making processes [30]. Data on demand integrates with systems to ensure that the right data is made available at the right time, further enhancing the relevance of information in contexts like real-time healthcare monitoring, customer support, and dynamic pricing systems [72].

Modern techniques such as *Machine Learning* (ML) and *Large Language Models* (LLMs) are increasingly used to support these efforts. ML algorithms can improve data quality by automating anomaly detection and predictive data cleaning, to ensure more consistent and accurate data inputs. On the other hand, LLMs are particularly effective in extracting meaning from unstructured data, such as user-generated content, improving interpretability and relevance.

Research Contributions

Figure 1 illustrates the key tasks, data types, and domains addressed in this thesis, organized as a graph. At the core of this graph is the central notion of information quality, with various tasks (represented by blue nodes) aimed at addressing specific challenges related to IQ, such as ER, Fairness, and Uncertainty Estimation. These tasks target different aspects of data quality and can be applied across multiple data domains. The yellow nodes represent these data domains, which include healthcare, manufacturing and e-commerce, among others. The data within these domains can be classified into structured and unstructured data (represented by red nodes), which may appear in a variety of formats. In domains like manufacturing and healthcare, data often appears in structured formats such as tables and Knowledge Graphs (KGs), which represent entities as nodes and relationships as edges, providing the ability to capture complex, interconnected data relationships. On the other hand, domains like e-commerce and question answering (QA) deal with both structured data as well as unstructured data, including subjective and factual data.

This thesis is organized around a total of four objectives, dubbed **O1**, **O2**, **O3**, and **O4**, and corresponding research contributions, aimed at advancing information quality across the data management tasks illustrated in Figure 1. Each objective is designed to address a specific task, with contributions structured accordingly.

We begin with the task of data cleaning, which is discussed in Chapters 2 to 4, where we develop methods to detect and manage duplicate records, a challenge pervasive in large datasets, such as those found in e-commerce platforms. Central to data cleaning are Clean-clean and Dirty ER [172], which aim to identify records that

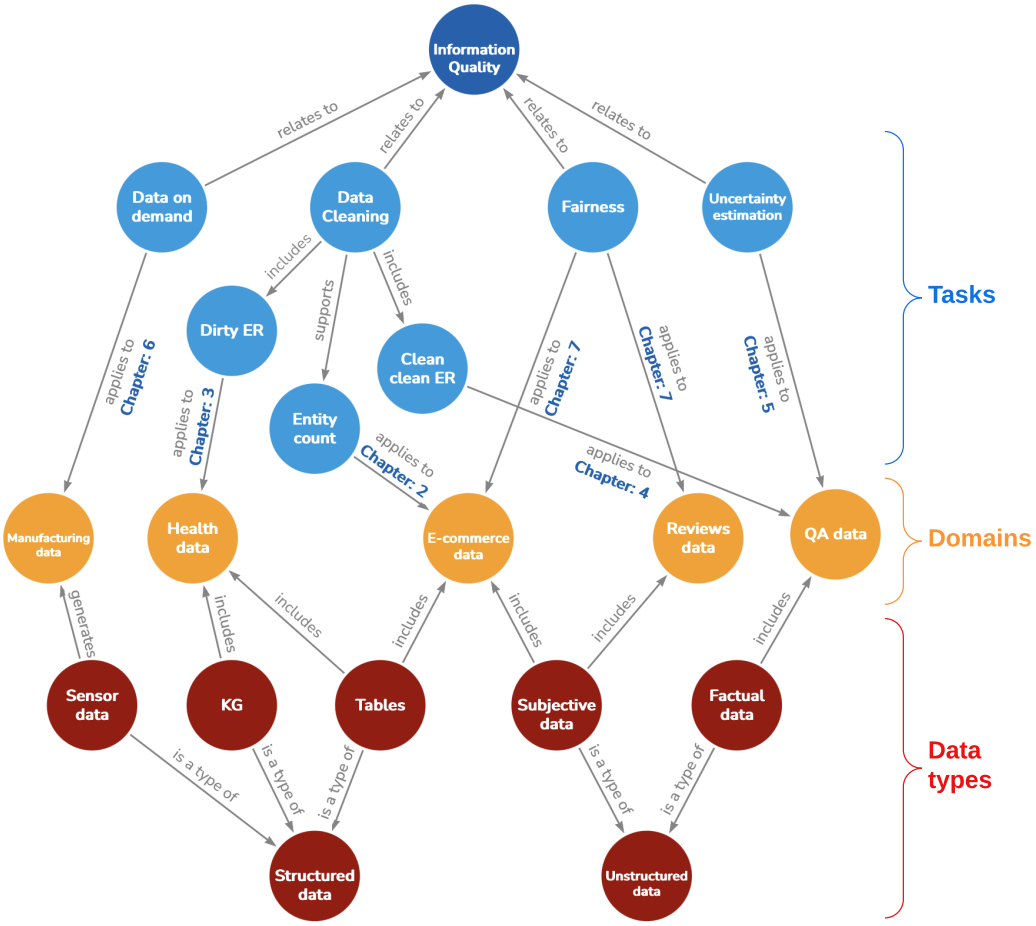


Figure 1. KG of tasks, domains, and data types related to Information Quality covered in this thesis

refer to the same real-world entity. Clean-clean ER, also known as Record Linkage in databases, assumes that each source is free of duplicates internally, meaning duplicates may only exist across sources. On the other hand, Dirty ER, also referred to as Deduplication in databases, addresses scenarios where one or more sources may contain duplicates within them. Additionally, the task of *entity count* is closely linked to Data cleaning and focuses on quantifying the number of unique entities in a dataset. Note that, while related, ER and entity count serve different purposes: ER focuses on matching and merging records, while entity count aims to quantify the unique entities present in the data.

We now outline the objective **O1** for the data cleaning task, followed by the specific research contributions achieved in pursuit of this objective:

O1 The objective is to develop effective and efficient methods for cleaning duplicates and resolving inconsistencies in large datasets. Additionally, the goal is to improve accuracy in duplicate detection by addressing challenges related to ambiguous and context-dependent data in specialized domains

With reference to objective **O1**, the following research contributions have been

achieved:

R1-1 In [77] we introduce a framework and a design space for entity count estimation that integrates and evaluates several methods from machine learning-based ER, clustering and statistical approaches. Through evaluations across different dataset types, we analyze under which conditions these methods perform effectively, with scalability supported by a sampling-based approach that reduces computational requirements through the use of sampling and upscaling techniques

R1-2 In [10] we explore the use of KGs to improve the ER process, particularly in specialized domains. We identify the challenges and opportunities of using KGs to reduce human involvement in domain-specific ER tasks, such as in healthcare, thereby improving the accuracy of the ER process in vertical applications.

R1-3 In [7, 8] we propose a novel NLP-based workflow to manage large repositories of multiple-choice questions (MCQs), focusing on detecting and managing conceptually similar but syntactically different questions. The use of transformer-based NLP techniques ensures the removal of duplicates, improving the quality of MCQ repositories for educational assessments.

Following data cleaning, the next task addressed in this thesis is uncertainty quantification in LLM-generated data, discussed in Chapter 5. In question-answering applications, LLMs generate responses based on training data that can be incomplete or ambiguous, leading to uncertainty in output reliability. This uncertainty is particularly challenging when multiple models provide varied responses, raising questions about data trustworthiness and consistency. Addressing this is critical to enhancing data reliability and interpretability, aligning with the broader goals of information quality. We now outline objective **O2** as follows:

O2 The objective is to develop methods for quantifying and addressing uncertainty in outputs generated by LLMs in question-answering tasks. In contexts where multiple LLMs provide varying outputs, uncertainty arises regarding which information can be trusted. This objective focuses on improving the reliability of such outputs, enabling a more informed decision on how to integrate their responses.

With reference to objective **O2**, the following research contribution has been achieved:

R2-1 In [248] we introduce a Bayesian framework that models each LLM as part of a crowd, treating their outputs as independent annotations. The framework aggregates these outputs and quantifies uncertainty by assigning reliability scores to each model's response.

The next task we address in this thesis is data on demand, which will be discussed in Chapter 6. This capability is particularly valuable in domains such as manufacturing, where operators rely on up-to-date metrics for production efficiency, equipment performance, and supply chain status. We now discuss objective **O3** and the achieved research contribution as follows:

O3 The objective is develop techniques for automated code generation that enable non-technical users to access and integrate real-time data from diverse industrial sources, supporting effective decision-making through reusable data solutions.

With reference to objective **O3**, the following research contribution has been achieved:

R3-1 In [151] we present a system that leverages LLMs to address the challenge of generating data from heterogeneous industrial sources in real time. The system interprets user requests in natural language and, using LLMs, generates code that creates and integrates data from multiple platforms and formats, thus enabling non-technical users to dynamically produce new data on demand.

Finally, the last task covered in this thesis is ethical data quality. More specifically, we focus on one ethical dimension: fairness, which refers to the principle of ensuring that data and algorithms do not systematically favor certain groups over others. While multiple definitions of fairness exist across the literature, it broadly concerns providing equal treatment and preventing bias based on sensitive attributes like age, gender, or ethnicity.

Fairness becomes particularly critical when dealing with subjective data, such as user reviews on e-commerce platforms or in recommender systems. In these cases, each review is not only a reflection of the individual's experience but also carries the potential for bias tied to sensitive attributes of the reviewer, such as age, gender, or other personal characteristics.

O4 The objective is to assess the fairness of ranking systems in collaborative rating platforms that use subjective data, such as reviews. These platforms often present ranked lists of items and associated reviews, where certain groups of reviewers may be overexposed or underexposed based on demographic or opinion-based attributes. The focus is to identify and measure how fairly these systems treat different reviewer groups and understand the implications of biased exposure in these platforms.

With reference to objective **O4**, the following research contribution has been achieved:

R4-1 In [15] we introduce a fairness assessment pipeline that measures how ranking systems in collaborative platforms expose different groups of reviewers. The pipeline quantifies fairness at both the item and query levels: item-level fairness evaluates how evenly exposure is distributed among different groups for individual items, while query-level fairness examines how fairly groups are represented across the ranked lists returned for user queries.

Parts of our work were published in the following papers. Each is labeled with the research challenges addressed.

[77] **Firmani, F., Mathew, J. G., Srivastava, D.**
Evaluating Methods for Efficient Entity Count Estimation. Manuscript under review.
Research Contributions: **R1-1**

[10] **Andreou, A. S., Firmani, D., Mathew, J. G., Mecella, M., Pingos, M.**
Using Knowledge Graphs for Record Linkage: Challenges and Opportunities. 1st Workshop on Knowledge Graphs for Semantics-driven Systems Engineering, held in conjunction with the 35th International Conference on Advanced Information Systems Engineering (CAiSE 2023).
Research Contributions: **R1-2**

[7] **Albano, V., Firmani, D., Laura, L., Mathew, J. G., Paoletti, A. L., Torrente, I.**
NLP-Based Management of Large Multiple-Choice Test Item Repositories). Journal of Learning Analytics (JLA 2023).
Research Contributions: **R1-3**

[8] **Albano, V., Firmani, D., Laura, L., Mathew, J. G., Paoletti, A. L., Torrente, I.**
Resolving duplicates in Large Multiple-Choice Questions Repositories). 20th conference on Information and Research science Connecting to Digital and Library science (IRCDL 2024).
Research Contributions: **R1-3**

[248] **Yao, P., Mathew, J. G., Singh, S., Firmani, D., Barbosa, D.**
A Bayesian Approach Towards Crowdsourcing the Truths from LLMs. Workshop on Bayesian Decision-making and Uncertainty, held in conjunction with the 37th Conference and Workshop on Neural Information Processing Systems (NeurIPS 2024), To Appear.
Research Contributions: **R2-1**

[151] **Monti, F., Mathew, J.G., Firmani, D., Leotta, F., Mecella, M.**
Composing Smart Data Services in Shop Floors through Large Language Models. 22nd International Conference on Service-Oriented Computing (ICSOC 2024), To Appear.
Research Contributions: **R3-1**

[15] **Balzotti, L., Firmani, D., Mathew, J. G., Torlone, R., Amer-Yahia, S.**
R-Fairness: Assessing Fairness of Ranking in Subjective Data. Submitted to the ACL Rolling Review, Under minor revision.
Research Contributions: **R4-1**

Thesis Outline

The thesis is divided into three parts, each covering specific objectives and associated research contributions.

First, in Chapter 1 we provide an overview of key background concepts, including similarity computation, record representation techniques, LLMs, and handling structured and unstructured data.

Then, in Part I we address objective **O1** related to data cleaning. More specifically:

- Chapter 2 addresses **R1-1** and presents an analysis of methods for estimating the number of distinct entities in large datasets. Approaches from ER, clustering, and statistical techniques are evaluated, highlighting their effectiveness in achieving scalable and accurate entity count estimation.

- Chapter 3 addresses **R1-2** by examining the integration of domain-specific knowledge to improve the accuracy of ER tasks, addressing challenges related to ambiguity in detecting and resolving duplicates in specialized datasets. Techniques for incorporating structured knowledge are discussed to overcome limitations of traditional methods.

- Chapter 4 is related to **R1-3** and addresses the challenge of identifying and managing semantic duplicates in large repositories of multiple-choice questions (MCQs). It introduces methods for computing semantic similarities between

questions and discusses strategies to ensure diversity and reduce redundancy in educational assessments.

Part II focuses on methods for data uncertainty and data on demand, addressing both objective **O2** and objective **O3**.

- Chapter 5 addresses **R2-1** and investigates methods for managing uncertainty in LLM-generated outputs within a crowdsourcing setting. It focuses on techniques for aggregating and evaluating responses from multiple LLMs, improving the overall reliability and consistency of the outputs in collaborative data fusion tasks.
- Chapter 6 addresses **R3-1** and discusses methods for transforming natural language queries into structured data, specifically within industrial environments. It examines how LLMs can be used to automate the generation of structured data from multiple, heterogeneous sources, enhancing decision-making on the shop floor.

Part III addresses fairness in data quality, focusing on objective **O4**:

- Chapter 7 addresses **R4-1** and explores fairness in ranking systems that rely on subjective data, such as user reviews in e-commerce platforms. It examines fairness at two levels: the item level, ensuring equitable exposure across items, and the query level, focusing on fair representation of different groups in the ranked lists. Methods for measuring and mitigating biases are discussed, ensuring that ranking algorithms do not disproportionately favor or disadvantage specific groups based on sensitive attributes like age or gender.

Finally, Chapter 8 concludes the discussion and traces some future developments that might arise from the basis of this work.

Visiting Periods

During the Ph.D., the author completed a three-month research stay (from May 2024 to August 2024) at the University of Alberta (Canada). This period was spent at the Computing Science department, under the supervision of Prof. Denilson Barbosa. During the visiting period, the research contribution **R2-1** – [248] was developed. Additionally, in the context of the DESTINI H2020 project, the author conducted a three-week research visit in September 2022 at the Cyprus University of Technology (CUT), Limassol (Cyprus), under the supervision of Prof. Andreas Andreou. This collaboration produced key insights shaping the research contribution **R1-2** – [10].

Additional Work

During the PhD program, the author has been involved in four additional research projects focused on manufacturing data and data generated in the cloud continuum.

Specifically, the author has been a member of the TopKontrol project[1], an industrial project related to Smart Manufacturing (SM) that uses computer vision techniques and Internet of Things (IoT) sensors to monitor the production process of corrugated cardboards. The author has also been a member of Electrospindle 4.0, a MISE-funded industrial project that uses data analytics, IoT and ML to develop smart spindles, monitor their usage and predict anomalies in their operations ahead of time. The author has been a member of the "DESTINI H2020 Twinning Project, Smart Data Processing and Systems of Deep Insight"[2] project, whose aim is to facilitate the transfer of scientific knowledge and expertise, as well as the best research practices from the leading institutions to Cyprus University of Technology. Finally, the author is currently a member of INTEND[3], an Horizon 2020-funded project that aims to develop a toolbox to help users deploy their distributed application to a cloud platform using natural language interfaces powered by LLMs.

We report the contributions to these projects here in terms of published papers for sake of completeness, though the themes covered in these papers are out of scope for this thesis.

[9] **Bernasconi, E., Catarci, T., Leotta, F., Mathew, J.G., Mecella, M., Monti F., et al.**
Electrospindle 4.0: Towards zero defect manufacturing of spindles. Research Projects, 16th International Conference on Research Challenges in Information Science (RCIS 2022).

[127] **Leotta, F., Mathew, J. G., Mecella, M., Monti, F.**
Supporting Zero Defect Manufacturing Through Cloud Computing and Data Analytics: The Case Study of Electrospindle 4.0. 4th International Workshop on Key Enabling Technologies for Digital Factories, held in conjunction with the 34th International Conference on Advanced Information Systems Engineering (CAiSE 2022).

[29] **Calamo, M., De Santis, G., Leotta, F., Mathew, J.G., Mecella, M., Monti, F., et al.**
TopKontrol: a monitoring and quality control system for the packaging production. CAiSE Research Projects Exhibition, held in conjunction with the 35th International Conference on Advanced Information Systems Engineering (CAiSE 2023).

[76] **Firmani, D., Mathew, J. G., Santoro, D., Simonini, G., Zecchini, L.**
Bridging the Gap between Buyers and Sellers in Data Marketplaces with Personalized Datasets. 31st Symposium on Advanced Database Systems (SEBD 2023).

[75] **Firmani, D., Leotta, F., Mathew, J. G., Rossi, J., Balzotti, L., et al.**
INTEND: Intent-Based Data Operation in the Computing Continuum. CAiSE Research Projects Exhibition, held in conjunction with the 36th International Conference on Advanced Information Systems Engineering (CAiSE 2024).

[160] **Monti, F., Mathew, J. G., Leotta, F., Koschmider, A., Mecella, M.**
On the application of process management and process mining to Industry 4.0. Software and Systems Modeling (SoSyM 2024).

[1] https://ikarton.it/topkontrol/
[2] https://www.destini2020.eu
[3] https://intendproject.eu

Chapter 1

Background and Context

1.1 Data Types and Domains

Data can generally be classified into two broad categories: structured and unstructured. Structured data is commonly found in domains such as finance, healthcare, and logistics, where predefined relationships between entities are essential. This type of data is typically organized in a schema that allows for efficient querying, analysis, and processing. For example, in software engineering, code can be structured as trees, providing a hierarchical format that supports tasks such as code analysis or optimization. Another common form of structured data is graph data, where entities are represented as nodes and relationships as edges, forming a network of connected information. Graph data is widely applied in areas such as network analysis, social media, and recommendation systems. In this scenario, knowledge graphs are a specific type of graph data used in applications that require the representation of complex relationships, such as semantic web technologies and knowledge management systems. In these environments, structured data is crucial for ensuring accuracy and consistency, as the explicit relationships between entities support reliable data integration and querying across systems.

Unstructured data, by contrast, lacks a predefined format and is prevalent in domains where information is more varied and comes in diverse forms, such as text, multimedia, or user-generated content. Examples include customer reviews, social media posts, and question-answering datasets. Unstructured data presents challenges for traditional processing methods due to its variability and complexity. However, advancements in natural language processing (NLP) have greatly improved the ability to extract insights from unstructured data. Techniques such as sentiment analysis and topic modeling are now widely employed to analyze large volumes of text in domains like e-commerce, where understanding customer feedback is critical for decision-making.

The interplay between structured and unstructured data is vital for addressing complex data and information quality challenges. In many real-world applications, both data types are processed together. For instance, in e-commerce systems, customer feedback (unstructured) must often be linked to structured product catalogs or user profiles. Integrating structured and unstructured data requires techniques such as embeddings and ML models, which enable the transformation of unstructured

text into structured vector representations. This transformation facilitates similarity computations and other forms of analysis, enabling smoother interactions between data types and supporting tasks like data integration and clustering.

In this thesis, studying and analyzing both structured and unstructured data, as well as the diverse variety of data types within these categories, is fundamental to addressing a wide range of data-related challenges across domains such as healthcare, finance, and e-commerce. Table 1.1 provides a summary of the works discussed in the subsequent chapters, categorizing them based on the data type and the domain from which the data originates.

Chapter	Data Type	Domain	Data Source
Chapter 2	Tables	E-commerce	Open-source benchmarking datasets for ER
Chapter 3	Tables, Graphs	Healthcare	Private healthcare data from a case study in the Republic of Cyprus
Chapter 4	Questions	Education	Private QA repository of the Italian Public Administration
Chapter 5	Questions	General Knowledge	Open-source QA datasets
Chapter 6	Tables, Sensor Data, Code	Manufacturing	Synthetically generated sensor data for a cardboard manufacturing process
Chapter 7	Reviews	E-commerce	Subjective data crawled from Yelp and Amazon

Table 1.1. Overview of data types and domains by chapter

1.2 Similarity Computation

Similarity computation is a key aspect of many data processing and analysis tasks [53], providing a quantitative method to evaluate how closely related two records, entities, or objects are, based on their respective features. These computations are critical in domains such as ML, information retrieval, and data integration, where understanding relationships between items is essential for tasks like clustering, classification, and record matching.

The process of similarity computation generally involves two key components: (i) how records are represented and (ii) the specific similarity functions used to evaluate the closeness of those representations. The representation of records can take various forms, from traditional methods such as bag-of-words models, where text is represented as a sparse vector of word counts, to more advanced approaches like embeddings, which capture semantic meaning in dense vector spaces. Once records are appropriately represented, similarity functions such as cosine similarity, Jaccard similarity, or Euclidean distance are applied to measure the degree of similarity between two records.

1.2.1 Record Representation Types

Record representation methods can be broadly categorized into two types: word-based representations, which rely on sparse vectors of word frequencies, and embedding-based representations, which encode data into dense vector spaces that capture semantic relationships. We discuss both approaches in detail below.

Word-based representations. These methods represent documents or text sequences as high-dimensional sparse vectors, where each dimension corresponds to a unique term from a predefined vocabulary. The most common word-based models are the *bag-of-words* (BoW) model [204] and *Term Frequency-Inverse Document Frequency* (TF-IDF) [204]. In the BoW model, documents are represented as vectors of raw word counts, ignoring word order and treating each word independently. TF-IDF refines this by weighting words based on their importance in the document relative to their occurrence across the entire corpus, reducing the impact of frequently occurring but uninformative words. While these methods are simple to implement and computationally inexpensive for small corpora, they suffer from several limitations, including the inability to capture context or relationships between words and the curse of dimensionality, which makes them inefficient for large vocabularies or corpora. For example, in a TF-IDF vector space, the document *"machine learning is important"* would be represented by counting the occurrences of each word in the document, but this method does not capture the fact that "machine" and "learning" together have a different meaning than the words in isolation. As such, word-based representations are often limited when tasks require deeper understanding of word semantics.

Embedding-based representations. These methods encode words, sentences, or even entire documents as dense, low-dimensional vectors in continuous vector spaces, where the distance between vectors reflects their semantic similarity. These representations are generated through unsupervised learning models that capture contextual and syntactic relationships between words. Embedding-based models have become the standard for many natural language processing (NLP) tasks due to their ability to capture both local and global relationships in text. *Word embeddings* such as Word2Vec [155], GloVe [181], and FastText [23] are among the earliest and most widely used embedding-based models. Word2Vec uses a neural network to learn vector representations by predicting surrounding words (context) for a target word (skip-gram model) or predicting a target word based on surrounding context (CBOW) [155]. Similarly, GloVe learns word embeddings by factoring a global word co-occurrence matrix, ensuring that words frequently appearing together are closer in the vector space. FastText improves on Word2Vec by representing words as combinations of subword components (n-grams), allowing it to handle out-of-vocabulary words by generating embeddings for words that were not seen during training.

For example, in the Word2Vec model, words like "king" and "queen" will be positioned close to each other in the embedding space because of their similar roles and meanings in many contexts, and the vector difference between "king" and "queen" will be similar to that between "man" and "woman", reflecting their gender relationships.

More recently, *transformer-based* embeddings have set a new standard by pro-

ducing *contextualized* representations. Unlike static embeddings, transformer models such as BERT [49] generate dynamic embeddings that reflect the context in which a word appears. These models use self-attention mechanisms to process entire sequences at once, capturing nuanced dependencies between words in different contexts. In BERT, for example, the word "bank" in the sentence "I deposited money in the bank" will have a different vector representation than in the sentence "The river overflowed the bank".

1.2.2 Similarity Functions

Once records are represented as vectors, similarity functions can be applied to quantify the degree of closeness between them. The choice of similarity function is crucial and depends on the type of data representation, whether word-based or embedding-based, as well as the nature of the data. Different similarity measures are suitable for different data types, and the choice of similarity function has a significant impact on the performance of the underlying task. The following are some of the most commonly used similarity measures:

Euclidean distance. Euclidean distance is a straightforward and widely used similarity measure that calculates the straight-line distance between two points in an n-dimensional space. It is primarily used with *embedding-based representations*, where the magnitude of differences between continuous values is meaningful. The Euclidean distance between two vectors $\mathbf{x}$ and $\mathbf{y}$ is given by:

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

Euclidean distance is effective for continuous data but is sensitive to differences in scale, making it necessary to normalize the data before applying the measure in some cases.

Cosine similarity. Cosine similarity measures the cosine of the angle between two vectors, focusing on the direction rather than the magnitude of the vectors. It is well-suited for *both word-based* and *embedding-based representations*, particularly in high-dimensional, sparse vector spaces such as text-based data. Cosine similarity is computed as:

$$\cos(\theta) = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\| \|\mathbf{y}\|}$$

where $\cdot$ denotes the dot product and $\|\mathbf{x}\|$ is the Euclidean norm of vector $\mathbf{x}$. Cosine similarity is widely used in document similarity tasks and word embedding spaces, where the frequency of word occurrences or semantic relationships can vary significantly. Its normalization property makes it robust against variations in document length, which is a common issue in text retrieval tasks.

Jaccard similarity. Jaccard similarity measures the overlap between two sets by dividing the size of their intersection by the size of their union. This measure is particularly useful for *word-based representations*, especially when dealing with binary attributes, categorical data, or bag-of-words models. The Jaccard similarity between two sets A and B is given by:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

Jaccard similarity is widely used in applications that require comparison of sets, such as finding common elements between user profiles, detecting duplicate records, or comparing attributes between different entities. It is often applied in tasks where words or terms are treated as distinct entities, such as in bag-of-words models.

Additional similarity functions. For specific applications, more advanced similarity measures are often used. *Hamming distance* is appropriate for *binary vectors*, measuring the number of positions where two vectors differ, making it relevant for word-based binary data. *Mahalanobis distance* accounts for correlations between variables and is suited for *embedding-based representations*, particularly when multicollinearity is present in the data. *Dynamic Time Warping* (DTW) [200] is frequently employed in *time-series data*, where sequences may have temporal distortions or shifts, making it more suited to specialized domains. These measures provide tailored solutions to complex data types, ensuring that similarity computation is aligned with the characteristics of the data.

1.3 Applications of Similarity Computation

Similarity computation is a cornerstone in many practical applications, from improving the efficiency of ER tasks to ensuring fairness in algorithmic decision-making. Below, we discuss several key applications.

Matching record pairs in ER. Entity Resolution (ER) involves determining whether two records, r_1 and r_2 (e.g. two product descriptions, such as *"Apple iPhone 16, 64GB, Black"* and *"Black 64GB iPhone 16 by Apple"*), refer to the same real-world entity, yielding a binary outcome (match or non-match). To make these decisions with high confidence, ER often relies on embedding-based similarity computations, particularly for complex, unstructured, or textual data. Embedding methods, such as BERT and Sentence Transformers [191], generate dense vector representations of records, capturing both syntactic and semantic information. Once these embeddings are generated, similarity functions such as cosine similarity and dot product can be applied to compare the vector representations of the records [62]. These functions allow for nuanced comparisons by reflecting contextual relationships between words, enabling ER systems to surpass the limitations of basic string matching in terms of accuracy.

Improving efficiency in ER. ER can be computationally expensive, with the cost of performing pairwise comparisons scaling quadratically in the worst-case scenario. As the dataset size increases, the sheer volume of potential comparisons poses significant challenges for scalability. To mitigate this, blocking is employed as a fundamental step in ER, partitioning records into smaller, more manageable subsets known as blocks. . The effectiveness of blocking relies on *similarity functions*, which measure the closeness between specific attributes such as names, addresses, or other identifying information. Lightweight similarity measures, such as Jaccard similarity and cosine similarity, are commonly used to group records that share similar attribute values into the same block [43]. For example, Jaccard similarity effectively

captures overlap in categorical attributes, while cosine similarity is well-suited for comparing high-dimensional text data represented as vectors. This initial filtering through similarity-based blocking dramatically reduces the number of comparisons, improving scalability. Advanced blocking methods further optimize this process by refining the similarity criteria over multiple passes. For instance, techniques like *sorted neighborhood methods* and *multi-pass blocking* rely on iteratively adjusting the similarity thresholds or blocking keys to capture more candidate pairs while minimizing false negatives [171]. These refinements ensure that blocks maintain high precision and recall, even when working with noisy or incomplete data.

Community detection and clustering. In both community detection and clustering, similarity computation is essential for identifying groups of related entities. Algorithms such as spectral clustering and hierarchical clustering can rely on similarity functions such as cosine similarity to determine the relationships between nodes or data points in complex datasets. These measures are especially valuable in high-dimensional spaces, where traditional distance metrics may struggle to capture meaningful relationships due to the curse of dimensionality (e.g. Euclidean distance). Graph-based clustering methods, which use similarity between nodes to define clusters, are also becoming more popular. By leveraging similarity scores between data points, these methods can detect communities in social networks or identify patterns in large datasets with multiple attributes [245].

1.4 From Language Models to Large Language Models

The development of language models (LMs) has transformed NLP, enabling advancements across diverse applications such as ER, information retrieval, and question answering. These models play a crucial role in tasks that require understanding and processing of text, allowing systems to operate more effectively on both unstructured data and structured data.

LMs are core components in NLP, designed to estimate the probability distribution over sequences of words. Early approaches, such as n-gram models, relied on fixed-length word sequences, using statistical methods to predict the next word based on preceding words [24]. While n-gram models and other statistical approaches were widely used, they were constrained by limited context windows and an inability to capture deeper linguistic structures. As datasets grew larger and tasks became more complex, these limitations highlighted the need for models capable of processing longer dependencies and more abstract relationships. Neural language models addressed these challenges by leveraging deep architectures that could learn continuous vector representations of words, capturing both semantic and syntactic information [17]. These models laid the foundation for more advanced approaches, such as recurrent neural networks (RNNs) and long short-term memory (LSTM) networks [92], which extended context beyond fixed windows. However, the sequential nature of these models limited their ability to parallelize computations, which became a bottleneck for large-scale data processing.

The introduction of the Transformer architecture [225] revolutionized language modeling. Built on the Transformer architecture, Large Language Models (LLMs) like LLaMA 3.2 [58], Gemini [190], and GPT-4 [4] represent a significant leap in

capability. These models are pre-trained on extensive corpora and fine-tuned for specific tasks, allowing them to capture long-range dependencies and semantic nuances more effectively than earlier models. LLMs are now widely applied in tasks such as machine translation, text generation, and question answering, marking a paradigm shift in the scalability and versatility of NLP systems.

1.4.1 The Transformer Architecture

Introduced in [225], the Transformers architecture discard the need for recurrent or convolutional structures, relying instead on an *attention* mechanism to process input sequences in parallel. This parallelism is key to the architecture's scalability and efficiency, particularly in handling long-range dependencies in textual data.

At its core, the Transformer consists of two main components: an *encoder* and a *decoder*. The encoder processes the input sequence, generating contextualized representations, while the decoder uses these representations to generate output sequences. Both components are built from stacked layers of multi-head self-attention and feed-forward neural networks. The self-attention mechanism allows the model to dynamically focus on different parts of the input sequence, assigning varying weights to different tokens based on their relevance to the task at hand.

The Transformer architecture has given rise to three key model families:

- **Encoder-only models**: These models, like BERT [49], RoBERTa [142], and DeBERTa [91], use only the encoder component to generate bidirectional representations. This makes them highly effective for understanding the context in tasks such as sentence classification, named entity recognition, and question answering, where the entire input sequence needs to be processed simultaneously.

- **Decoder-only models**: Models such as GPT [25] and more recent models like LLaMA 3.2 [58] and GPT-4 [4], are decoder-only architectures optimized for autoregressive tasks like text generation, where tokens are generated one at a time. These models excel in tasks such as text completion, language generation, and dialogue systems by predicting the next word in a sequence based on previously generated tokens.

- **Encoder-decoder models**: Architectures like T5 [188], BART [128], and mBART [140] use both the encoder and decoder. These models are well-suited for tasks requiring both understanding and generation, such as machine translation, text summarization, and paraphrasing.

1.4.2 Prompting Techniques

Prompt engineering has become an essential technique in leveraging LLMs for a wide range of tasks. A *prompt* is the input or instruction given to a LM to shape its output for a specific task. Prompts provide essential context, guiding the model on what kind of response to generate. By carefully designing input prompts, users can direct the behavior of the LLM without the need for extensive fine-tuning or retraining, making it an adaptable framework for numerous applications. This section discusses

advanced prompting patterns that have been shown to improve model performance, grounded in recent developments in NLP.

One of the core techniques in prompt engineering is *few-shot learning*, which allows LLMs to generalize from a small set of examples included directly in the prompt. Instead of requiring large labeled datasets for training, few-shot learning enables the model to perform tasks such as translation and summarization with minimal supervision. For example, by providing the model with a few input-output pairs, it can extrapolate and generate correct responses for new inputs in the same context. This approach has demonstrated impressive performance across a variety of tasks, reducing the need for task-specific retraining [25].

Another key strategy in prompt engineering is *role prompting*, where the LLM is instructed to assume a particular role, such as that of a teacher, legal advisor, or customer support agent. This strategy provides additional context, guiding the model's output to align with the expectations and tone required for the assigned role. Role prompting is particularly effective in domain-specific tasks where the model's response should reflect certain expertise or formality. In fact, clearly defining the role within the prompt enables the model to generate more accurate and context-aware responses, which makes this technique particularly valuable in dialogue systems, task-oriented bots, and professional applications [120].

More recently, *Chain of Thoughts* (CoT) [236] prompting has emerged as a technique to improve performance in tasks requiring multi-step reasoning. In this approach, LLMs are guided to break down problems into smaller, intermediate steps rather than generating a direct output. This results in improved interpretability and accuracy, particularly in complex tasks such as mathematical problem-solving and logical reasoning. A related variant, called *Tree of Thought* (ToT) [249], extends this concept by allowing the model to explore multiple reasoning paths simultaneously, offering a more comprehensive exploration of potential solutions in decision-making and problem-solving scenarios.

In recent advancements, debating techniques [55, 69, 112] have emerged as a novel method for enhancing reasoning capabilities in LLMs. In this framework, two or more models engage in debate, presenting alternative viewpoints. A third-party evaluator, either a human or another model, can assess the strength of the arguments and determine the most convincing response. This adversarial process has shown to be especially effective in situations where a weaker evaluator, with limited access to information, needs to choose between the competing arguments. The underlying principle is that debates encourage models to produce more thorough and well-reasoned responses, as they are motivated to not only generate but also defend their solutions under scrutiny. For example, in a question-answering scenario, one model may argue in favor of a particular solution while another model offers a counter-argument [112]. The evaluator assesses both positions based on persuasiveness, factual correctness, and logical coherence.

Finally, we mention *LLM agents* as an extension of prompt engineering, leveraging prompts not only to generate text but also to interact with external tools and systems. At the core of LLM agents is the ability to invoke external APIs and services, enabling them to perform complex tasks such as data retrieval and information processing. Architecturally, LLM agents are typically composed of three integral components: tools, planning, and memory. The tools module interfaces

with external systems, allowing agents to execute actions beyond natural language processing. The planning component is responsible for reasoning and task decomposition, facilitating the execution of multi-step procedures that require strategic decision-making. Meanwhile, the memory module maintains state across interactions, ensuring that the agent can recall previous actions or inputs, thereby improving coherence and enabling long-term task execution. Frameworks such as ReAct [250] and Reflexion [206] provide systematic approaches to structuring LLM agents. ReAct integrates reasoning and action, allowing agents to iteratively plan and execute tasks, while Reflexion incorporates memory-based mechanisms for self-evaluation, enabling agents to improve their performance on tasks through reflective feedback loops.

1.4.3 Applications of LMs and LLMs in Data and Information Processing

LLMs have rapidly evolved to become fundamental tools in data and information processing, extending far beyond traditional natural language tasks. We describe below some representative examples of these applications.

Using LLMs for ER. Traditional machine learning (ML) methods for ER required extensive feature engineering to capture essential characteristics of records, often leveraging specific attributes such as names, addresses, or numerical identifiers. The emergence of deep learning (DL) models, such as DeepMatcher [162] and DeepER [152], significantly advanced ER by learning features directly from raw data, reducing the reliance on hand-crafted features. These DL-based approaches use recurrent neural architectures to capture complex patterns and relationships between entities, handling heterogeneous data and accommodating different data sources with minimal manual intervention. The advent of Transformer models brought further advancements to ER. Transformer-based models such as Ditto [137] using self-attention mechanisms to capture contextual dependencies between records in a shared embedding space, enabling precise entity matching. Ditto, for instance, effectively applies data augmentation techniques to enhance the model's generalization capabilities, making it robust across multiple ER tasks [137]. By using pre-trained embeddings, these transformer models require minimal labeled data and can be fine-tuned for domain-specific applications, demonstrating significant improvements over traditional DL models in both accuracy and efficiency. Recently, large language models (LLMs) have been integrated into ER, using prompt-based approaches to simplify the process of entity matching across diverse datasets. In the approach proposed in [180], LLMs like GPT-3 are guided through detailed prompts to compare records based on pre-trained linguistic and contextual knowledge. This method leverages the ability of LLMs to parse nuanced record descriptions, reducing the need for domain-specific fine-tuning and extensive labeled data.

Leveraging LLMs for Question Answering. Traditional question answering (QA) systems heavily relied on rule-based techniques or manually curated knowledge bases, often requiring significant domain-specific engineering to interpret user queries and retrieve accurate responses. The introduction of machine learning (ML) models marked an improvement by using large-scale question-answer pairs to better handle information retrieval tasks [59]. However, these early models still struggled with

capturing the subtleties of natural language, particularly in complex or context-dependent scenarios. Neural network-based models, like BiDAF, introduced attention mechanisms to enhance comprehension by effectively leveraging bidirectional context [205], while DrQA demonstrated the benefits of integrating external knowledge from sources like Wikipedia [36]. The emergence of LLMs has further revolutionized QA by leveraging transformer-based architectures. Models like T5 redefined QA tasks as text-to-text problems, using transfer learning to achieve superior performance across a variety of scenarios [188]. GPT-3, known for its zero-shot and few-shot capabilities, has shown exceptional results in answering diverse questions with minimal additional training [25]. More recent advancements include UnifiedQA, which unifies multiple QA formats into a single model, improving generalization across domains [114]. Additionally, retrieval-augmented generation (RAG) models have combined the strengths of retrieval mechanisms with generative LLMs to provide accurate and up-to-date answers by accessing external knowledge bases dynamically [129]. These innovations have made LLMs highly effective and versatile in addressing complex QA challenges across various contexts.

Part I

Data Cleaning

Chapter 2

Evaluating Methods for Efficient Entity Count Estimation

In this chapter we focus on the data cleaning task, which we study in the context of structured tables in the e-commerce domain, by considering several well-known ER benchmarking datasets. We summarize the concepts covered in this chapter in Figure 2.1.

Figure 2.1. Tasks, domains and data types covered in this chapter

The problem of estimating the size of a query result has a long history in data management. When the query performs entity resolution (aka record linkage or deduplication), the problem is that of estimating the number of distinct entities, referred to as the *entity count*. This problem has received much attention from the statistics community but it has been largely overlooked in the data management literature. In this work, we formally define the entity count problem from a data management perspective and decompose it into a framework of fundamental steps. We explore approaches from both statistics and data management, systematically identifying a design space for different pipelines that address this problem. Finally, we provide extensive experiments to highlight the strengths and weaknesses of these approaches on real-world benchmarks.

2.1 Introduction

The problem of estimating the size of a query has a long history in data management, ranging from early works on selectivity estimation [164] to more recent approaches for join query size estimation [27]. In particular, the literature is rich with methods to estimate the size of a *select* or a *join* query. Many applications need to work with the set of distinct entities contained in a dataset, in which case the query performs entity resolution (aka record linkage or deduplication) on the dataset [239]. In this

ID	Title	Length	Artist	Year
A	Astronomy Domine - The Piper at the Gates of Dawn	4.202	Pink Floyd	'87
B	The Jimi Hendrix Experience - Ain't No Telling	112	NaN	10
C	Astronomy Domine	252093	Pink Floyd	1987
D	Ain't No Telling (Axis: Bold as Love)	01:52	Jimi Hendrix Experience	2010
E	001-Astronomy Domine	4m 12sec	Pink Floyd	NaN
F	Pink Floyd - Astronomy Domine	252	NaN	87

Table 2.1. Exemplary set of records from the *Music Brainz 20k* dataset [198]

case, the problem is that of estimating the number of distinct entities, referred in the following as the *entity count*

Datasets often contain duplicate records that refer to the same entity, making the number of rows a poor estimate of its entity count. Consider the example dataset in Table 2.1: the number of rows is 6 but it only contains 2 distinct songs. The problem of recognizing duplicates, also known as Entity Resolution (ER), has received extensive attention from the the statistics, machine learning, and data management communities, from the seminal work of Fellegi-Sunter to the recent breakthroughs based on language models [137], showing an unprecedented ability to identify how humans represent and misrepresent information. On the other hand, the problem of estimating the entity count has only received attention from the statistics community, with a variety of works such as capture-recapture methods [73] and Bayesian population estimation techniques [217].

Challenges. One natural baseline to solve entity count is pipe-lining ER and cluster counting. In our example, ER would return the set of matching pairs $M = \{(A, C), (A, E), (C, E), (B, D)\}$, which identifies the clustering $\{\{A, C, E\}, \{B, D\}\}$, and cluster counting would return "2" as an answer. Optionally, such a baseline can be enhanced by selecting a sample of s records for the ER step and then applying upscaling techniques [80] in the counting step. Nonetheless, the baseline would require in the worst case $O(s^2)$ matching queries before returning the entity count. Matching queries need to be issued to a machine learning classifier [221], to a human crowd [227] or to a large language model [137], thus incurring significant latency and monetary costs. Even progressive ER methods [174], that are designed to reduce latency for high F-measure, still require to run exhaustively before their results can be used for entity count.

Entity count works from the statistics community take a different route, often employing probabilistic models and Bayesian approaches for ER, such as Fellegi and Sunter's framework [68], Blink [211], and d-Blink [149]. These methods can solve ER by estimating matching probabilities, which can be used directly for entity count estimation. However, these works do not easily incorporate support for

powerful machine-learning matching methods, often leaving the matching decisions to statistical inference. In this landscape, efficient and effective solutions for the entity count problem are still out of reach, leaving much space for further improvements.

Our contribution. In this work we formally define the entity count problem from a data management perspective and decompose it into a framework of fundamental steps. Then, we consider works in both the statistics and data management literature, and systematically identify a design space for pipelines that can solve this problem. Finally, we provide extensive experiments to highlight their strengths and weaknesses on popular real-world benchmarks. Specifically

- We propose a framework that integrates key steps for entity count estimation, combining methods from machine learning-based entity resolution (ER), statistical approaches and clustering;
- We introduce a sampling-based variant of the entity count pipeline, which reduces computational overhead by computing entity counts from a sampled subset and upscaling the result to approximate the full dataset;
- We systematically evaluate pipelines from both data management and statistical literature, testing their effectiveness and efficiency on real-world datasets. Our experiments provide insights into the strengths and weaknesses of each approach, highlighting scenarios where certain techniques outperform others in terms of accuracy and efficiency.

The remainder of this paper is structured as follows. In Section 2.2, we review related work on tasks related to the entity count problem. Section 2.3 defines the entity count problem and Section 2.4 introduces the general framework. Section 2.5 describes a taxonomy of pipelines for entity count. Section 2.6 describes the proposed sampling-based approach and its implementation. We present experimental results in Section 2.7, evaluating the performance of different pipelines across several datasets. Finally, Sections 2.8 and 2.9 summarize the results and conclude the paper, outlining future directions for research.

2.2 Related Work

In this section, we identify methods that are strongly related to the entity count problem and discuss how they can be leveraged to address our problem.

Join size estimation. Represents a key task for query optimization [99], and involves predicting at query time the size of the join between two tables after applying selection predicates to each. Several sampling-based approaches have been developed to address this task. For instance, Correlated Sampling [226] constructs small space synopses for quick join size estimates subject to dynamically specified predicate filter conditions. We mention also [40], which introduces a sampling algorithm called two-level sampling, which combines advantages of previous sampling methods, outperforming them on various join types.

Entity Resolution. Entity resolution (ER) aims at matching records that refer to the same real-world entity. This task has been widely studied for the last 50

years, starting from the seminal probabilistic approach by Fellegi and Sunter [68], followed by more recent approaches based on bayesian inference such as Blink [211] and its scalable variant, d-Blink [149]. More recently, several Machine Learning (ML) and Deep Learning (DL) approaches to ER have been proposed, achieving state-of-the-art results. DeepER [61] combines pretrained word embedding models such as GloVe [181] with an LSTM [92] based DL model for ER. DeepMatcher [221] represent a generalization of the former work and proposes a DL framework for ER that supports several word embedding models. Ditto [137] combines pretrained BERT [49] based language models with external, domain-specific information and data augmentation, which provides more synthetic training instances. ZeroER [242], is an unsupervised approach for ER based on Gaussian Mixture Models that achieves comparable performance to supervised ML approaches.

Clustering. Clustering is strongly related to the task of estimating the number of unique entities in a dataset. The goal of clustering is to create clusters where objects within a cluster are highly similar, while objects in different clusters are dissimilar. Assuming that each cluster groups records referring to the same entity, the entity count task can be seen as counting the number of clusters given an input dataset. Common techniques include correlation clustering [16], BIRCH [256], and DBSCAN [66]. The above mentioned methods work in an offline setting where the original dataset is fully accessible. Streaming clustering algorithms address the challenge of clustering continuously generated data in a single pass with limited memory. These algorithms must adapt to evolving data and form new clusters as needed [252].

Number of connected components estimation. This represents another problem that has strong ties with entity count. This task consists in estimating the number of connected components in a graph, called parent graph, using a sampled subgraph. In this scenario the entity count tasks can be seen as estimating the number of connected components in a parent graph where nodes corresponds to records in the dataset and edges links records referring to the same entity. Note that these edges are not known a priori, so they must be inferred as well. Starting from the seminal work by Frank [80], several studies have explored this problem. The authors in [116] proposed a novel estimator using network motif counts, providing guarantees on mean squared error for graphs with bounded spectral gaps. The authors in [118] characterized the optimal sample complexity for chordal graphs in the sublinear regime, constructing linear-time estimators and proving minimax lower bounds.

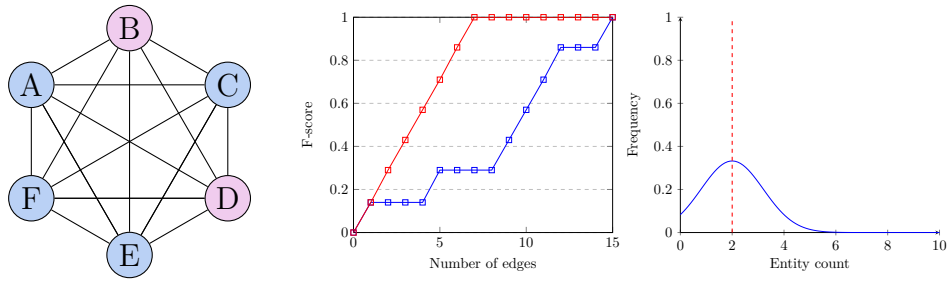
2.2.1 Discussion

The above works can be broadly organized based on two complementary dimensions: duplication type and the use of sampling. The duplication type indicates whether a method expects duplicate records to be exact replicas of their original counterparts, i.e. exact duplicate, or might differ from the latter, i.e. approximate duplicate. The second dimension indicates whether a method makes use of sampling techniques for efficiency. We observe that none of the above line of works focus on approximate duplicates in a sampling setting where subset(s) of data are used to make the final prediction. Join size estimation works with exact duplicates (i.e. the join keys)

and make use of sampling, while ER and clustering work in a noisy setting where approximate duplicates might be present, but do not make use of sampling for efficient computation. For instance, d-Blink [149] makes use of a combination of partitioning and parallelism to make the computation faster compared to the original Blink method. While d-Blink scales horizontally to handle large datasets, we focus on the effect of sampling with fixed computational resources.

In this work we attempt to fill the above gap by (i) outlining a general framework of fundamental steps for entity count in a sampling scenario, and (ii) providing an empirical study of several families of established methods from the ML and statistics literature to estimate the entity count.

2.3 Problem Statement



(a) Graph representation of Table 2.1 **(b)** Evaluation of an ER model **(c)** Evaluation of an ECE method

Figure 2.2. Different type of evaluation for ER and ECE

Let D be a dataset of record entries r. Records can be either structured (e.g. set of key-value pairs describing the technical specs of a mobile phone) or unstructured data entries (e.g. reviews about that same mobile phone model). Each record $r \in D$ is associated to a specific real-world entity $e \in E$. Let $f(r) \in E$ denote the underlying entity $e \in E$ associated to r. The *entity count* of D, denoted with $c(D)$, is the number of unique entities present in D, i.e. $c(D) = |\{f(r) \mid r \in D\}|$.

Example 1. *Table 2.1 contains a set of exemplary records from the Music Brainz 20k dataset [198]. Each record in Table 2.1 represents a specific song, with some noticeable (noisy) duplicated records. For instance, records* `A`*,* `C`*,* `E` *and* `F` *refer to the song "Astronomy Domine" from Pink Floyd, while records* `B` *and* `D` *refer to the song "Ain't No Telling" from Jimi Hendrix Experience. The underlying number of unique entities in Table 2.1 is thus* $c(D) = 2$.

We now introduce the entity count estimation problem.

Problem 1. *Given a dataset D, the entity count estimation (ECE) problem consists in providing an estimate $\hat{c}(D)$ of the true entity count $c(D)$.*

Note that the problem formulation only takes into account the entities featured in the input dataset, thus $1 \leq \hat{c}(D) \leq |D|$, i.e. the estimate cannot be larger that the input dataset.

Given an ECE method $\hat{c}(\cdot)$, its accuracy can be evaluated in terms of *approximation error* δ, which is defined as follows:

$$\delta(D) = \frac{\hat{c}(D) - c(D)}{c(D)} \tag{2.1}$$

where an approximation error of 0 implies a perfect estimate, while large negative or positive values denote substantial underestimation or overestimation of the result respectively.

Discussion. While in principle ER approaches can be used for ECE, ER approaches are typically evaluated using a different metric, namely f-score. The underlying objectives associated with these two metrics do not necessarily overlap, and this can possibly lead to cases where a highly accurate ER method yields poor results when adapted for the ECE task.

Example 2 (ER evaluation)**.** *Figure 2.2a illustrates how records in Table 2.1 can be arranged to form a graph. Each record is associated with a distinct node in the graph and nodes sharing the same color refer to the same song. ER approaches are focused on finding matching edges in the graph, i.e. edges linking records that refer to the same entity. The dataset in Table 2.1 basically consists of two clusters of songs, one with 4 records (Pink Floyd) and the other one with 2 records (Jimi Hendrix Experience). The number of matching edges is thus* $4 \cdot 3/2 + 2 \cdot 1/2 = 7$*. The blue line in Figure 2.2b shows an exemplary f-score trend of an oracle ER model when fed with the edges in Figure 2.2a in a specific order. One desirable outcome for ER approaches is to maximize the progressive f-score [84], i.e. finding an ordering of the edges that leads to maximize the area under the f-score curve (red line).*

Example 3 (ECE evaluation)**.** *Figure 2.2c shows the performance of a hypothetical ECE method. The ECE method in this example is based on a stochastic process and the blue line shows the trend of the entity count estimates provided by the ECE method over several trials. The red line represent the actual number of entities in the dataset. Let us assume that that by averaging the trials, the stochastic process yields an entity count estimate of 2.1. In this case, the approximation error would be* $(2.1 - 2)/2 = 0.05$*, indicating that the ECE method overestimates by 5% the actual entity count.*

Example 4 (Adapting ER for ECE)**.** *A straightforward way to use an ER method for ECE would consist of finding all the matching edges in the graph and remove edges that are non-matching. The resulting graph will consist in a set of connected components, with each connected component containing records that refer to the same entity (according to the ER model). In this way the ECE task would reduce to count the number of connected components. Let us now assume that the ER model in the previous example introduces some errors. Let assume for instance that the ER model mistakenly classifies the edge between records* `B` *and* `D` *as non-matching. In this case, the f-score would slightly drop, from* 1.0 *to* $12/13 \sim 0.92$*. If we were to use such an ER model for ECE, we would end up with three connected components, namely* `{A,C,F,E}`*,* `{B}`*,* `{D}`*. The approximation error would be* $(3 - 2)/2 = 0.5$*, i.e. we would overestimate the actual entity count by* 50%*.*

In this work we study several families of methods for efficiently estimating the entity count of a dataset by using samples $D' \subseteq D$ of the original dataset D. Note that this also includes the case where the original dataset is fully sampled.

We unify several methods under a common framework by assuming that the original dataset is mapped to an underlying (unknown) *parent graph*. The parent graph consists of a union of cliques, where each clique contains records from D that refer to the same entity. Given one or more samples $D' \subseteq D$, our goal is to estimate the total number of connected components in the parent graph based on the observed (or estimated) number of connected components in the subgraph induced by the sampled dataset. Through this framework, we aim to develop scalable techniques that maintain accuracy while reducing computational costs, especially when dealing with large datasets.

Given a subset $D' \subseteq D$ of the input dataset, we will denote with $\hat{c}(D'|D)$ the entity count estimate provided by $\hat{c}$ using D'.

2.4 A General Framework for ECE

We illustrate in Figure 2.3 a general framework to estimate the entity count of a dataset D using samples of records $D' \subseteq D$. Conceptually, the framework consists of three main steps. The first step consists in sampling a set of records $D' \subseteq D$. The following step is estimating the number of unique entities in D', which we frame as estimating the number of connected components induced by the subgraph associated by D'. Finally the last step is estimating the number of unique entities in D, which we frame as upscaling the number of connected components found in the subgraph.

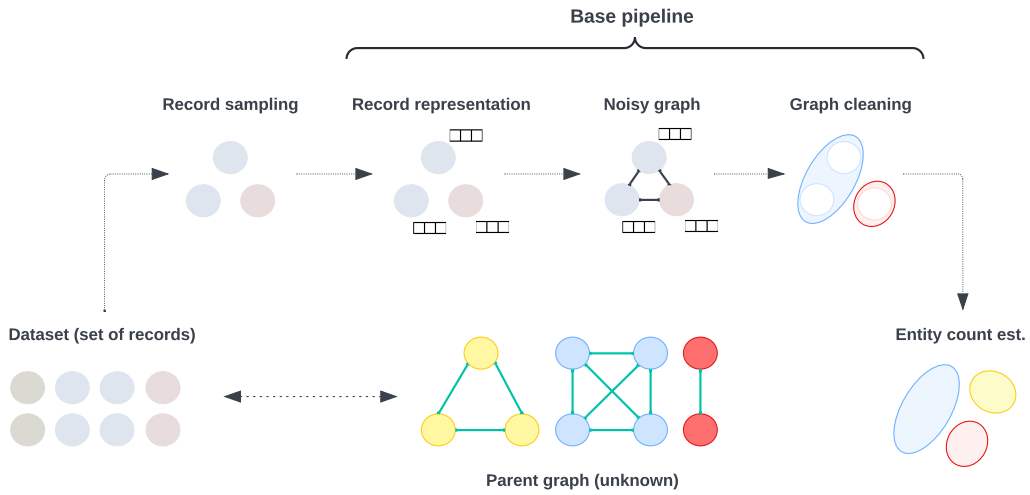


Figure 2.3. Overview of the main stages in our ECE framework

We now describe the individual steps in the devised framework for ECE below:

Record sampling. This step consists of selecting a representative sample $D' \in D$ of the dataset to be used to estimate the entity count of the original dataset D. Note that this also includes the case where the entire dataset is sampled. This step aims at providing a trade-off between accurate and efficient estimation of the entity

count of a dataset. In fact, small sample sizes can provide fast estimation of the entity count but the accuracy might be low. On the other hand, larger samples can lead to more accurate results but can be expensive in terms of running time and might suffer from scalability issues when it comes to large datasets.

Record representation. This step consists of processing the records in D' and modeling them in a meaningful format for the subsequent step of generating a graph from D'. In general, different techniques can be used to model the records, ranging from standard bag-of-words (BoW) model to more recent dense vector (embedding) based approaches. Ideally the selected format should capture the similarity between records, such that the records that are similar to each other have similar representations.

Noisy graph generation. This step uses the sampled records and their representations to generate a weighted graph $G(V, E)$ where each node $v \in V$ corresponds to a record $r \in D'$ and edge weights measure the similarity of pairs of records, based on their representation. Ideally, such graph should result in a union of disjoint cliques, where only records referring to the same underlying entities are linked to each other. However, since records are approximate duplicates instead of exact duplicates, the resulting graph might contain spurious edges that link records associated with different entities, as well as missing links between records that refer to the same entity.

Graph cleaning. This step aims at cleaning the previously generated noisy graph such that it results in a union of cliques. Such union of cliques should ideally represent an induced subgraph of the unknown parent graph associated with D, given the set of sampled records D'. This graph cleaning step can be implemented with a wide range of techniques, ranging from pairwise ML-based ER methods to statistical ER and clustering. For instance, one could use a ML-based ER methods to find matching pairs of records and discard non-matching edges from the noisy graph.

Entity count estimation. In this step, we use the union of cliques generated in the previous phase to estimate the number of entities in the original dataset, D, by scaling the number of cliques. Various scaling techniques have been proposed in the literature, particularly for estimating the number of connected components in a graph based on an induced subgraph, such as [118, 80]. These techniques involve statistical estimators, which can be either biased or unbiased, and aim to estimate the total number of connected components in the original graph by analyzing the number of nodes within each clique in the induced subgraph.

Among the steps in the pipeline illustrated in Figure 2.3, we identify the combination of *Record representation*, *Noisy graph generation* and *Graph cleaning* as being the core components in estimating the entity count of D. In fact, those three steps can be readily run to estimate the entity count of the dataset without the need to sample the original dataset. Moreover, these steps allow space for a variety of techniques, ranging from ER-based ones to possibly new techniques for ECE. We will call the combination of those three steps *base pipeline*. The joint use of *Record sampling* and *Entity count estimation* on top of a base pipeline allow to scale to larger datasets and offer a trade-off between accuracy and efficiency.

In this work, we focus on four main families of base pipelines and study their

performance in terms of accuracy and efficiency for ECE. These pipelines draw from established methods in entity resolution (ER), clustering, and statistics, as well as recent advancements in machine learning and embedding models.

1. **ML ER** This base pipeline represents the adaptation of ER methods for the entity count task. This pipeline uses popular blocking algorithms (e.g. [21, 82]) to generate a noisy graph followed by the use of ML-based ER methods (e.g. [137]) to filter out noisy edges and generate a union of cliques.

2. **Simulation** This base pipeline provides a faster approximation of the ML ER approach by simplifying the ER process for entity count estimation. Instead of exhaustively applying the ML ER model, it approximates the model's output by mapping record pair similarities to calibrated matching probabilities. These probabilities are derived by querying the ML ER model on a subset of records to speed up the process while still grouping similar records into cliques for entity counting.

3. **Statistical ER** This base pipeline allows traditional statistical ER approaches (e.g. [68, 211, 149]) to be repurposed for the ECE task by using the linkage structure provided by these methods to induce a union of cliques that groups records that refer to the same entity.

4. **LLM embeddings** This family frames the entity count task as a clustering problem. It utilizes large language models (LLMs) (e.g., [165]) to embed the records into dense vector representations. These embeddings are then fed into clustering algorithms (e.g., [66]) to form a union of cliques, where each clique corresponds to records representing the same entity.

The family of base pipelines that we consider in our study are not meant to be exhaustive, but represent a comprehensive set of methods that can be used to address the ECE problem. In general, the outlined framework can be extended with new methods for ECE as long as they generate a union of cliques from a (sample of) records. Additionally, in all of these pipelines, sampling and upscaling techniques can be applied to improve efficiency and scalability, allowing for quicker approximations of the entity count without the need for exhaustive computation.

2.5 Base Pipelines

We now describe in more detail the base pipelines by breaking down their inner workings in terms of their implementation of the *Record representation*, *Noisy graph generation* and *Graph cleaning* steps.

2.5.1 ML ER Pipeline

The ML ER pipeline adapts machine learning-based ER techniques to address the entity count estimation task.

Record representation. A common approach in ER is to represent records using a Bag-of-Words (BoW) model, where the semantic similarity between records is

measured based on shared tokens. Techniques like TF-IDF weighting are frequently employed to give more weight to informative terms, helping to distinguish between similar records. Other methods, such as embedding-based representations can also be applied when deeper semantic similarities are required to handle more complex data relationships.

Noisy graph generation. In ER, reducing the number of pairwise comparisons is crucial, particularly for large datasets. Blocking algorithms, such as adaptive blocking [21], are commonly used to group records into blocks and limit the number of record pairs that need to be compared. From the output of blocking, a noisy graph can be constructed by connecting records within the same block, with edges representing potential matches. Alternatively, metablocking techniques [172] can be employed to directly create a *metablocking graph*, where edges between records are weighted based on their co-occurrence across multiple blocks. These approaches allow for a significant reduction in the number of comparisons while ensuring that relevant record pairs are retained for further analysis. Similarity between records can then be measured using metrics like cosine similarity, depending on the chosen blocking approach and assigned as weights to the edges.

Graph cleaning. The final step applies well-established ER principles to group records referring to the same entity. Once a noisy graph is constructed, edges are classified as matches or non-matches using machine learning-based ER models, to identify which edges are matching and which not. To improve efficiency, edges can be sorted based on criteria such as node ordering [78], ensuring that the most promising edges are evaluated first. Before passing these edges to the ER model, transitivity can be employed to infer additional matches (e.g., if record A matches record B, and record B matches record C, then A can be inferred to match C). This step reduces the number of queries made to the ER model, improving efficiency. To ensure robustness, additional queries can be made for edges involving records in the same cluster as the endpoints of a matched edge. This method [81] increases confidence that classified edges are truly matches, thus refining the entity groupings. By applying these ER techniques, the pipeline constructs a union of cliques, with each clique corresponding to a unique entity, representing the resolution of duplicates in the dataset.

2.5.2 Simulation Pipeline

The Simulation pipeline provides an approximation of the ML-based ER process for entity count estimation, focusing on improving computational efficiency by simulating the edge classification process. The *Record Representation* and *Noisy Graph Generation* steps are the same as in the ML ER pipeline, where records are represented and a noisy graph is constructed based on blocking techniques.

Graph cleaning. The core difference in the Simulation pipeline lies in how it handles edge classification. Instead of invoking the ER model for every edge, the Simulation pipeline approximates this step by mapping similarity scores (edge weights) into calibrated matching probabilities. These probabilities are based on the matching outcomes that the ML ER model would typically provide for different ranges of similarity scores. To create this mapping, edges are grouped into bins, where each bin

corresponds to a range of similarity score values. A sample of edges from each bin is queried using the ER model, and the aggregated (e.g. average) matching outcome for each bin is used to assign calibrated matching probabilities to the remaining edges in that bin. This allows the pipeline to simulate the matching process by sampling the matching outcome from these probabilities, rather than explicitly classifying every edge with an ER model. This calibration process offers design flexibility: the width of the bins, the number of edges sampled per bin, and the method of aggregating matching outcomes can all be adjusted to optimize accuracy and efficiency. Once the matching probabilities are established, the pipeline processes edges similarly to the ML ER pipeline, iterating over them in either in a random or a defined order [78], with transitivity applied to reduce the number of necessary queries. To further enhance the robustness of the approximation, additional edges within the same cluster as matched edges can be queried [81] to ensure consistency in the matching decisions.

2.5.3 Statistical ER Pipeline

The Statistical pipeline uses ER techniques from the statistical literature to derive the linkage structure of records and generate a union of cliques.

Record representation. In this pipeline, records are treated as raw textual representations without requiring preprocessing steps, such as transforming them into vectorized formats. Statistical ER methods like Blink [211], d-Blink [149], and other Bayesian models grounded in Fellegi and Sunter's record linkage theory [68] are capable of working with the original records while computing similarities internally. These models can incorporate embedding-based similarities or rely on empirical priors, as described by Steorts [211], to estimate the likelihood of records belonging to the same entity.

Noisy graph generation. Unlike the previous pipelines, the Statistical ER pipeline does not explicitly construct a noisy graph or clean spurious edges. Instead, the statistical ER model directly generates a union of cliques based on probabilistic assignments. Methods like Fellegi and Sunter's [68] model assign a matching status to record pairs based on likelihood ratios, while d-Blink and Bayesian models use empirically motivated priors to probabilistically assign entity identifiers to records. This step bypasses the need for graph construction, as the output of these models immediately provides the necessary entity groupings.

Graph cleaning. The statistical models treat records as input and assign entity identifiers (e.g., integers) probabilistically, based on the likelihood of records representing the same entity. Once records are assigned to entities, the union of cliques is generated by grouping records with the same identifier into the same clique. Statistical methods such as Blink and d-Blink apply Bayesian principles to estimate the linkage structure, making the process highly adaptable to different datasets and priors.

2.5.4 LLM embedding pipeline

The LLM Embedding pipeline uses embeddings generated by LLMs and clustering techniques to estimate entity counts. In this approach, records are transformed into

dense vector representations, which are then clustered to identify distinct entities. Depending on the clustering algorithm used, a similarity graph may be computed explicitly or implicitly.

Record representation. Each record is transformed into a dense vector using LLM-based embedding models, such as BERT [49] or other large language models. These embeddings capture the semantic content of the records in a high-dimensional vector space, enabling more accurate comparisons between records than traditional vector-based methods.

Noisy graph generation. In this pipeline, the generation of a noisy graph is sometimes implicit, depending on the clustering algorithm. While some clustering algorithms may rely on the computation of pairwise similarities (resulting in an explicit similarity graph), others may cluster records directly based on their distances in the embedding space without materializing a graph. For example, algorithms such as DBSCAN [66] or OPTICS [11] do not require a prior graph, but still effectively group records based on their relative proximity in the embedding space. A critical feature of the chosen clustering algorithm is that it should not require the number of clusters to be specified in advance, as the purpose of the pipeline is to estimate this value (the entity count).

Graph cleaning. The result of clustering the records based on their embeddings is treated as a union of cliques, where each cluster corresponds to a set of records representing the same entity. High-dimensional clustering techniques, such as DBSCAN or OPTICS, are particularly suited for this task, as they can handle varying densities and do not require pre-specifying the number of clusters. The clusters produced by these algorithms form the union of cliques, implicitly grouping records that are similar in the embedding space.

2.6 Sampling-based Pipelines for ECE

While the previously mentioned pipelines can estimate the entity count of D, their computational complexity may limit scalability to datasets containing millions of records. For instance, the ML ER pipeline exhibits a time complexity that is quadratic in the number of records in the worst case, making it impractical for large-scale datasets. To address this, the framework in Figure 2.3 introduces two steps: *Record sampling* and *Entity count estimation*. These steps improve the efficiency of the core method by (i) selecting a subset of records $D' \subseteq D$ and (ii) scaling the number of cliques detected in D' to estimate the total entity count in D.

In our experiments, we focus on Bernoulli sampling paired with the *Klusowski Bernoulli* estimator. Bernoulli sampling independently selects each record from D with a fixed probability $p < 1$, yielding an expected sample size of $N \cdot p$, where N is the total number of records in D. After obtaining the sample, the Klusowski Bernoulli estimator is applied to upscale the result.

The *Klusowski Bernoulli estimator* [118] refines the *Frank Bernoulli estimator* [80], focusing on minimizing variance in the estimation process. The Frank Bernoulli estimator provides an unbiased estimate of the number of connected components (or cliques) in the parent graph based on an induced subgraph sampled via Bernoulli sampling. However, it can suffer from high variance and occasionally

produces negative estimates due to its alternating series structure. The Klusowski Bernoulli estimator, while biased, reduces this variance and ensures more stable and reliable results, especially for larger and more complex datasets.

Although this work primarily focuses on Bernoulli sampling and the Klusowski Bernoulli estimator, other methods such as *Simple Random Sampling (SRS)* can also be employed. In SRS, a fixed-size subset D' is selected from D, where every possible subset of size k has an equal probability of being chosen. The *Frank SRS* estimator can upscale results from SRS, but similar to the Frank Bernoulli estimator, it may yield negative estimates due to its reliance on an alternating series.

2.7 Experiments

2.7.1 Datasets

We conducted experiments using a total of 10 datasets, which are listed in Table 2.2. We classify these datasets based on their original size (number of rows) and their actual size (number of unique entities). Given that the number of unique entities can vary significantly across different datasets, we compute a score to measure the duplication level of each dataset, defined as follows:

$$\text{Duplication factor}(D) = \frac{|D| - c(D)}{|D| - 1}$$

This results in four different categories of datasets, which are further described below:

Small dataset with few duplicate. For this category we consider 2 datasets. `DBLP-Scholar` is dataset from the ER benchmark [122] and consists of (duplicate) entries from the DBLP and Google Scholar bibliographic sources. *Music-Brainz-20k* is a dataset of roughly 20.000 records from [122] and contains actual song records from the MusicBrainz database. This dataset spans five sources and includes duplicates for half of the original records. Each duplicate is highly corrupted to rigorously test entity resolution and clustering methods.

Small dataset with multiple duplicate. We consider 3 datasets for this category. `WDC xlarge computers` is a dataset from the WDC benchmark and contain duplicated computer listings from several web sources. `Alaska (monitor)` is a dataset from the Alaska benchmark and consists in duplicated specification of monitors from more than 20 data sources. Since the set of attributes can significantly vary among records from different sources, we attached each record an additional synthetic attribute consisting of the concatenation of its original attributes. Finally, `Cars` is a synthetic dataset generated by scraping textual descriptions of vehicles, such as make and model, from various online sources [81].

Large datasets with few duplicates. For this category we consider two splits of the `Music-Brainz` dataset, containing 200,000 and 2 million records respectively. We also consider the 5 million split of the `North Carolina Voters` dataset, from [122], which is based on real records from the North Carolina voter registry. The datasets comprises records from 5 sources and contains both non-corrupted duplicates and corrupted duplicates across all sources.

Large datasets with multiple duplicates. For this category, we augmented the `Cars` dataset and the `WDC xlarge computers` dataset with synthetic records until they reached at least 1 million records. Each synthetic record was generated by combining different attribute values from records referring to the same entity, while also randomly dropping tokens from both the original and synthetically generated records.

2.7.2 Metrics

We can evaluate different approaches for ECE in terms of *accuracy* and *efficiency*. We measure the accuracy of an ECE method in terms of approximation error, which is defined in Equation 2.1. Note that the approximation error can be also be computed for sampling-based pipelines as follows

$$\delta(D') = \frac{\hat{c}(D|D') - c(D)}{c(D)}$$

We measure the efficiency in terms of *sample size* $|D'|$ and the running time required to compute the $\hat{c}(D|D')$. These two captures different notion of efficiency, namely the *sample efficiency* and the *computational efficiency*.

In fact, on one side we are interested in analyzing the accuracy of different ECE approaches with varying sample sizes. In this context, an efficient ECE approach should provide accurate estimates (i.e. low approximation error) even with a small sample w.r.t. number of records in the original dataset, i.e. $|D'| = o(|D|)$.

On the other hand, we also interested in the running time required to compute $\hat{c}(D|D')$. Ideally, an efficient algorithm should not only require a small sample size but also have a reasonable running time that does not grow exponentially for large datasets.

2.7.3 Implementation Details

We now describe the implementation detail of the four base pipelines we use in the experiments in terms of their implementation of the core steps outlined in Figure 2.3.

ML ER. We implement the ML ER pipeline by following three main steps: record representation, noisy graph generation, and graph cleaning. Records are modeled using a Bag-of-Words (BoW) representation with TF-IDF weighting, where tokens are generated by tokenizing the attributes and TF-IDF assigns higher weights to informative terms. To generate a noisy graph, we apply token blocking to group records sharing at least one token into overlapping blocks, filtering out blocks with more than 10% of the dataset to reduce the potential number of edges. A blocking graph is built by connecting records within the same block, with edge weights assigned using a weighted Jaccard similarity based on TF-IDF scores. Note that due to the block cleaning phase, some matching records might end up into separate connected components of the blocking graph, causing an overestimate of the entity count. We measure how much the matching records are spread across separate connected components in terms of *sparsity*, which is defined as $\sum_i^n c(CC_i)/c(D)$, where n is the number of connected components in the blocking graph and CC_i denotes the i-th connected component. For instance, a sparsity of 1.20 indicate that

Category	Name	Num attributes	Split	Num records	Num matches	Num entities	Duplication score
Small dataset with few duplicates	DBLP-Scholar	2	N/A	7,626	13,760	2352	69.17%
	Music Brainz 20k	5	20k	19,375	16,250	10,000	48.39%
Small dataset with multiple duplicates	WDC xlarge (computers)[1]	4	N/A	4,676	9,990	1,080	76.92%
	Alaska (monitor)	1	N/A	2,273	12,949	231	89.88%
	Cars	6	N/A	16,185	5,954,651	48	99.71%
Large dataset with few duplicates	Music Brainz 200k	5	200k	193,750	162,500	100,000	48.39%
	Music Brainz 2M	5	2M	1,937,500	1,624,503	1,000,000	48.39%
	North Carolina Voters 5M	4	5M	5,000,000	3,331,384	3,500,840	29.98%
Large dataset with multiple duplicates	Cars 1M	6	1M	1,000,024	22,762,570,161	48	99.99%
	WDC xlarge (computers) 1M[2]	4	1M	1,042,500	5,462,382,825	1,080	99.90%

Table 2.2. List of datasets used for the experiments

[1] http://webdatacommons.org/largescaleproductcorpus/v2
[2] http://webdatacommons.org/largescaleproductcorpus/v2

a perfect entity count estimation method would overestimate the entity count by 20%. To further reduce the overall number of edges to process, we filtered out edges with low Jaccard similarity scores. We set the minimum similarity threshold to 0.1. Note that while this edge filtering step can improve the efficiency of the ML ER approach, it can however further contribute to overestimating the final entity count, due to the removal of low-similarity edges linking two matching records within the same connected component. For the graph cleaning step, we make use of a *noisy oracle* to control the rate of errors produced by the ER model and explore how this can affect the performance in terms of entity count estimation. The noisy oracle make use of an edge error model, i.e. for each edge, it draws the matching outcome according to a predetermined distribution. We use the normal distribution in the experiments for both the matching edges and non-matching edges. We report the list of noisy oracle settings used in the experiments and their distribution model in Table 2.3. For instance, in the *Noisy 1* setting, the matching outcome of matching edges is drawn from a normal distribution centered at 0.9 with a standard deviation of 0.1 while the matching outcome of non-matching edges is drawn from a normal distribution with a mean of 0.1 and a standard deviation of 0.1. We explore a total of 4 settings, where we gradually reduce the noise related to the non-matching edges from *Noisy 1* to *Noisy 3* and then in *Noisy 4* we use a nearly ideal oracle. We also added a synthetic delay for each edge probed with the noisy oracle, to simulate the inference time of an ML ER model. We set the delay to 0.1 seconds, which was the average inference time of Ditto on the datasets in Figure 2.2. For all the noisy settings we leverage progressive ER methods to minimize latency and rapidly achieve high F-scores by dynamically adjusting the processing order of edges, using the node ordering strategy [78].

Setting name	**Noisy 1**	**Noisy 2**	**Noisy 3**	**Noisy 4**
Mean (match)	0.9	0.9	0.9	0.999
Std (match)	0.1	0.1	0.01	0.01
Mean (non-match)	0.1	0.01	0.001	0.001
Std (non-match)	0.1	0.01	0.01	0.01

Table 2.3. Setting for the noisy oracle in the experiments with the ML ER pipeline

Simulation. The Simulation pipeline approximates the behavior of the ML ER pipeline by using a calibration process instead of querying the ER model for each edge. We approximate the matching probabilities based on a sample of edges. Specifically, we construct an equiwidth histogram with 10 bins, each representing a range of similarity values from 0 to 1. For each bin, we randomly sample up to 100 record pairs (for small datasets) and up to 1000 pairs (for large datasets). These sampled pairs are then fed into the ER model (i.e. the noisy oracle) to obtain their matching outcomes. Each bin is assigned a calibrated matching probability, computed as the average matching outcome of the sampled edges within the bin. When processing an edge, the pipeline determines its matching probability by identifying the corresponding bin based on the edge's similarity score and sampling a matching outcome from a Bernoulli distribution with that probability. This reduces the need to invoke the ER model for every edge, making the process more computationally efficient. As in

the ML ER pipeline, control queries are used [81] to ensure robustness, querying additional edges from the same cluster as matched edges to verify their classification. The edges are processed randomly with a fixed seed. Transitivity is used to avoid redundant queries by inferring matches based on previously classified edges.

Statistical ER. For this pipeline, we use Blink [211], an unsupervised Bayesian ER method that operates on string and categorical features. In our implementation, all features across the datasets were treated as string attributes to ensure compatibility with Blink. While Blink typically uses a default string similarity function, we modified this by replacing the standard distance metric with an embedding-based similarity function, using an MPNet-based Sentence Transformer model [210, 192]. The model processes these string features and handles both the record representation and noisy graph generation steps. Instead of explicitly constructing a graph, Blink assigns entity identifiers probabilistically during its Gibbs sampling process, which estimates posterior probabilities for linking records to latent entities. We used the same hyperparameters as in the original Blink paper. The output is a union of cliques, where records sharing the same entity identifier are grouped together.

LLM embedding. For this pipeline, we leverage OpenAI's text-embedding-3-large [169] to generate dense vector representations of records. For the clustering step, we use DBSCAN [66], a density-based algorithm that does not require the number of clusters to be pre-specified. DBSCAN clusters records based on their proximity in the embedding space, where each cluster represents a distinct entity. The LLM Embedding pipeline does not explicitly construct a noisy graph. Instead, the clustering process operates directly on the distances between embeddings. As DBSCAN groups records based on their relative distances, it implicitly generates a union of cliques, with each clique corresponding to a unique entity.

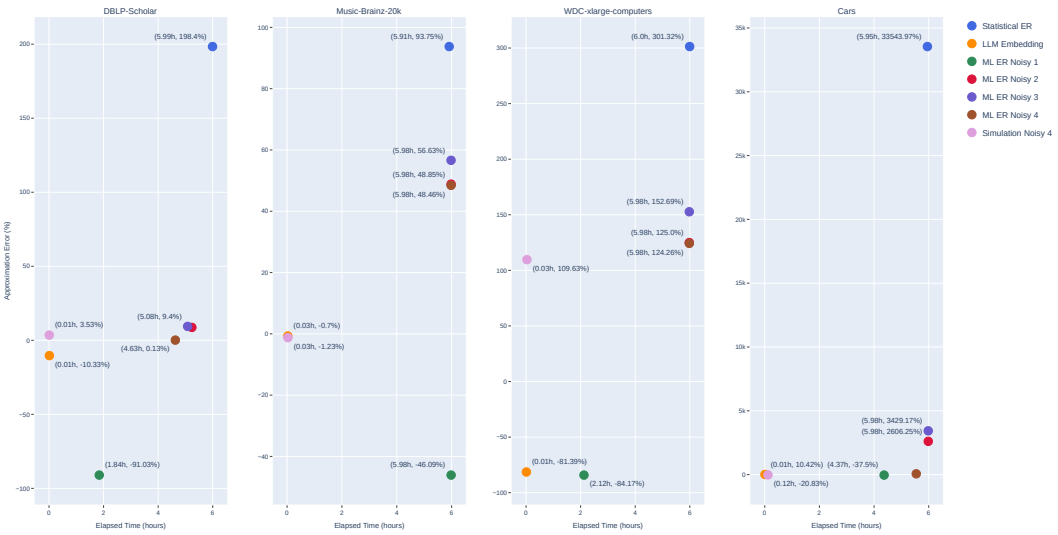


Figure 2.4. Base pipeline performance on small-sized datasets for low and high duplication factor datasets

2.7.4 Methodology

We evaluated the base pipelines on the datasets with a maximum runtime of 6 hours. For the sampling pipelines, we allocated 10% of this time, giving them a 36-minute runtime limit. The sampling pipelines were tested with four sampling ratios: 1%, 5%, 10%, and 20%. For each ratio, we ran the sampling pipelines five times, each time with a different random seed. After each run, we upscaled the entity count and compared with the ground truth entity count to compute the approximation error. We then computed the average approximation error among the five random seeds. For the base LLM embedding pipeline, embedding binarization was applied for datasets with over 1 million records to avoid memory issues. In the statistical sampling pipeline with Blink, Blink requires that records are organized into "files" (tables), where duplicated records are spread across different tables. Each record was assigned a file ID to separate duplicates. Since Blink requires at least two records per file, we resampled when the sample resulted in files with only one record, ensuring all tables had at least two records.

2.7.5 Experimental Results

In this section, we evaluate the performance of the proposed base and sampling pipelines in terms of both accuracy and efficiency across the evaluation datasets. The experiments were performed on a commodity server with an AMD Ryzen 9 5950X 16 core processor, 64 GB RAM, and an NVIDIA GeForce RTX 3090 GPU with 24 GB of VRAM. The main findings reveal that the LLM embedding and Simulation pipelines generally perform well on small datasets, while the Statistical and ML ER pipelines encounter scalability issues on larger datasets. Furthermore, we observe that while sampling-based pipelines offer considerable improvements in computational time, their accuracy is influenced by both the dataset's duplication factor and the upscaling process.

On small datasets simulation and LLM embedding perform better compared to the other base pipelines. Figure 2.4 reports the performance of the base pipelines across several small-sized datasets with varying duplication factors. For the Simulation pipeline we show the results using the noisy setting dubbed *Noisy 4*. Accuracy-wise, the LLM embedding and Simulation pipelines were overall the best performing approaches across the small-sized datasets. The LLM Embedding pipeline resulted the best performing approach on Music-Brainz 20k, a low duplication factor dataset (-0.7%) followed by the Simulation approach (-1.23%). Both approaches also outperformed the others on Cars, a high duplication factor dataset (10.42% error for LLM Embedding and -20.83% for Simulation). The LLM embedding approach also resulted the best performing approach on WDC-xlarge-computers, although with an approximation error of -81.39%. We notice that for this dataset, none of the pipelines were able to provide an accurate estimate of the entity count. For DBLP-Scholar, the ML ER Noisy 4 pipeline, which has the lowest amount of noise ended up being the best performing model, followed by the Simulation pipeline. We notice also that the ML ER Noisy 4 pipeline, despite its nearly perfect accuracy for ER, can still exhibit very high approximation errors, such as in WDC-xlarge-computers (124.26%) and Cars (62.5%). We note that in both cases, the main source of errors was the

underlying noisy graph. For instance, the noisy graph for the Cars dataset had a sparsity of 1.45 (i.e. an ideal oracle would overestimate the entity count by at least 45%). Moreover, even within the same connected component of the noisy graph, the truly matching records records might not be directly linked (due to the edge filtering) and be instead connected through a path that includes a non-matching record, thus further increasing the entity count estimate. We noticed in fact that the noisy graph of Cars contained only 74% of the matching edges. Music-Brainz-20k and WDC-xlarge-computers were also affected with the same issues in terms of sparsity and missing matching edges, resulting in an overestimation for the ML ER Noisy 4 pipeline. The Simulation pipeline showed to be a fair approximation of the ML ER Noisy 4 pipeline on most of the datasets, except Music-Brainz-20k, while being much faster. We also note that the amount of errors present in the noisy oracle can affect the final entity count estimate. For instance, the ML ER Noisy 1 pipeline tends to underestimate the entity count across all datasets, due to its high number of false positives. In fact, even though the matching outcome for non-matching edges is drawn from a normal distribution that leans towards 0 ($0.1\pm$ 0.1), the high number of non-matching edges, particularly for low-duplication factor datasets, can have a significant effect, since each false positive reduces the entity count by 1 unit. On the other hand, the ML ER Noisy 2 and 3 pipelines have a lower amount of false positives, which helps in addressing the underestimation of the Noisy 1 setting. However, the errors on matching edges, combined with the low recall and high sparsity of the blocking graph, can cause significant overestimation, similarly to the ML ER Noisy 4 pipeline. Across all datasets, the Statistical approach consistently overestimated the entity count, producing results only slightly below the total number of records. This outcome is likely due to the limited number of Gibbs iterations the Statistical pipeline could execute before hitting the timeout. As shown in [211], Blink can provide accurate results on small datasets ($\sim$40,000 records), but it might need a very large number of iterations (e.g. 10,000) to achieve that level of precision. For reference, the number of Gibbs iteration recorded for the Music-Brainz-20k dataset was $\sim$ 150. The low number of Gibbs iteration is mainly due to the use of an embedding model, in contrast to the use of standard string-based similarity measures in the original paper [211] which might not capture the semantic nuances of different wordings of the same concept. In terms of runtime, the LLM Embedding and Simulation pipelines are the most efficient, completing within 1 hour across all small-sized datasets. The Statistical pipeline instead takes the whole time cap of 6 hours, since it lacks of a stopping criterion, and this resulted running this pipeline for a certain number of Gibbs iterations until the time cap was reached. The ML ER pipelines suffered from high runtime, due to the simulated inference time of the noisy oracle, leading to runtime close to 6 hours on datasets with more than 10,000 records, such as Music-Brainz-20k and Cars. Moreover, moving from Noisy 1 to Noisy 3 we note an increasing trend in elapsed time, due to transitivity closure. In fact, the higher number of false positives in ML ER Noisy 1 makes much more likely to have cases where the matching outcome of edges can be inferred. On the other hand, the lower number of false positives in Noisy 2 and Noisy 3 reduces the number of inferrable edges, thus increasing the overall elapsed time.

Base pipelines fail to scale for million-scale datasets. In this section, we

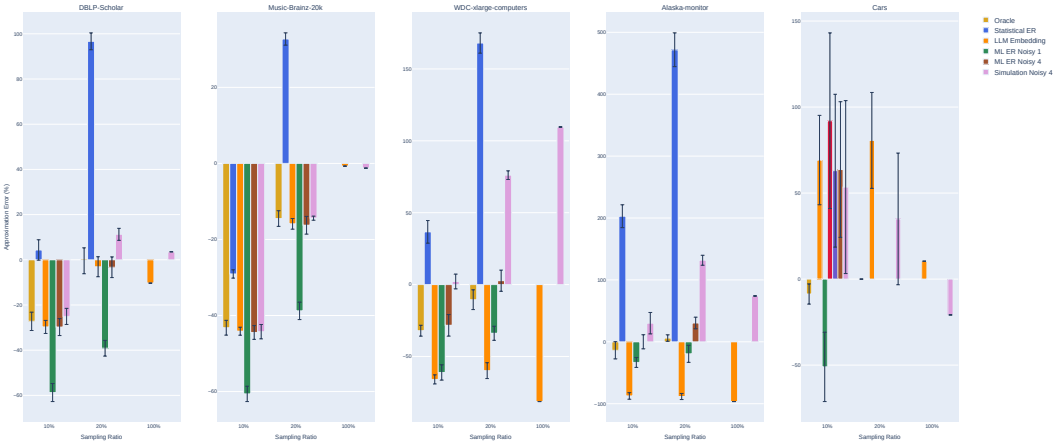


Figure 2.5. Sampling-based pipeline results on a selection of small datasets

discuss the performance of the base pipelines on larger datasets. None of the pipelines were able to run on the million-scale datasets within our evaluation setting, with the only exception being the LLM Embedding pipeline, which was able to run on the Music-Brainz 200k dataset, with an average approximation error of almost -11% after approximately 40 minutes.

For the Music-Brainz 200k dataset, the Statistical pipeline failed due to a memory error. This issue stems from Blink's approach of precomputing a similarity matrix for each attribute, where each cell represents the similarity between pairs of records. While this design is effective for smaller datasets, it quickly leads to memory issues on larger datasets, including those in the million-record range.

The ML ER and the Simulation pipeline had a similar issues on the Music-Brainz 200k dataset, as well as the other large scale datasets. Both approaches rely on a blocking graph, whose size can get significantly large up to the point it cannot fit entirely into the memory. This is particularly true for high duplication factor large datasets, where the edges can be in the order of hundred of millions. The large size of the noisy graph, combined with the latency of Ditto, prevented it from processing all edges within the time limit of 6 hours. As a result, the ML ER pipeline ended up overestimating the entity count. For datasets with high duplication ratios, the number of edges in the noisy graph grows significantly, causing the pipeline to run out of memory at an even faster rate. The LLM embedding pipeline also faced scalability issues on the million-scale datasets, as it required computing an embedding for each record, which takes around 1.7 hours for 1,000,000 records. In the remaining time, the DBSCAN clustering algorithm was unable to cluster the records within the time cap, which resulted in having no results for those datasets.

The upscaling estimator can introduce errors and is sensitive to the duplication factor. Before evaluating the sampling pipelines, we first analyze how the upscaling procedure, specifically the Klusowski Bernoulli estimator, can impact the final entity count. To isolate the effect of upscaling itself, we use an *oracle* sampling pipeline. The oracle pipeline, by design, introduces no errors in the graph cleaning step, as it returns the true clusters from the sampled dataset. Therefore, any observed error in the final entity count must be attributed solely to

the upscaling procedure. We show the results of the oracle pipeline in Figures 2.5 and 2.6 for different datasets and sampling ratios. The results indicate that the upscaling process can indeed introduce errors, especially when using smaller sampling ratios (e.g., 1% and 5%, see Figure 2.5) on large scale datasets. The only exception was the Cars 1M dataset, which consistently shows 0% approximation error, even for very small sample sizes. This behavior is due to the Cars-1M dataset having only 48 entities, which can makes it easier to sample at least one record per entity, even with a 1% sampling ratio (about 10,000 records). As a result, the upscaling process can work well in estimating the total entity count. In contrast, for other datasets, the sampled data may not always include records from all entities, making it harder to estimate the entity count accurately. This challenge is even more pronounced in datasets with low duplication factors, where the duplication factor of the sample is often significantly lower than that of the original dataset, as shown in Table 2.4. This is evident in Figure 2.5, where the approximation error of the oracle is much higher in low duplication datasets (DBLP-Scholar, Music-Brainz-200k) compared to the other datasets, which have much higher duplication factor. We note that these underestimation errors tend to decrease as the sampling ratio increases, and in some cases, such as DBLP-Scholar and WDC-xlarge-computers (Figure 2.5), the error approaches zero with larger sample sizes. Interestingly, for datasets with high duplication factors, like WDC-xlarge-computers-1M and Alaska, the upscaling error can shift from underestimation with small sample ratio to overestimation when using larger samples (e.g., 20%).

Sampling-based pipelines can perform comparably to base pipelines on small datasets. We evaluate the performance of sampling-based pipelines on small datasets, shown in Figure 2.5, and compare them with the Simulation and LLM Embedding base pipelines, which generally performed better in runtime and approximation error. Due to the high error introduced by the upscaling process at low sample sizes, combined with the small dataset sizes, we focus on 10% and 20% sampling, with base pipeline results shown under the 100% label. For the ML ER pipeline, we consider only the noisy oracles with the highest noise level (Noisy 1) and lowest noise level (Noisy 4). Increasing the sampling ratio often improves approximation accuracy, though with notable exceptions. The Statistical sampling pipeline, for example, tends to overestimate the entity count also when taking samples of the data, leading to overestimation at the sample level, as shown in Table 2.5). At 10% sampling, the upscaling process can partially counteract this overestimation, leading to more accurate counts, but at 20%, the correction effect diminishes, making overestimation more visible on datasets like DBLP-Scholar and WDC-xlarge-computers. The Simulation pipeline shows a similar trend, consistently overestimating on high-duplication datasets like WDC-xlarge-computers and Alaska-monitor. This overestimation effect is less pronounced at 10%, where the upscaling process helps counterbalance it. The LLM Embedding pipeline, meanwhile, performs best on datasets with low duplication factors. For datasets like DBLP-Scholar, WDC-xlarge-computers, and Alaska-Monitor, sampling actually reduces approximation errors compared to full dataset processing, while providing much faster runtimes. The ML ER pipeline demonstrates different patterns depending on the oracle's noise level: the Noisy 1 oracle underestimates, as also observed with its base counterpart,

while Noisy 4 provides accurate results on high-duplication datasets like WDC and Alaska-monitor. However, for larger datasets with high duplication factors, such as Cars (over 16,000 records), the ML ER pipeline sometimes fails to complete at 20% sampling within the time cap, due to the very high number of edges to probe combined with the oracle's (simulated) inference time. This consistently resulted in a significant overestimation (over 6,000% for ML ER Noisy 4 on Cars with 20% of the data), and for this reason, results for the ML ER pipeline on Cars are excluded from Figure 2.5. In general, sampling-based pipelines perform well on small datasets, particularly those with low duplication factors, and in some cases outperform base pipelines in both accuracy and efficiency, assuming that the sample size is sufficiently large. However, high-duplication datasets, like Cars, lead to increased approximation error and variance across most sampling pipelines.

Sampling-based pipelines can scale but can provide reasonable estimates only on very highly duplicated datasets. We evaluate the performance of sampling-based pipelines on large datasets in Figure 2.6. To handle large-scale datasets, we focus on the 1% and 5% sampling levels and limit our evaluation to the ML ER (Noisy 1 and Noisy 4), Simulation, and LLM Embedding pipelines, which performed best on smaller datasets. On low-duplication datasets, we see significant underestimation at the 1% sampling level, driven by both the small sample size and errors introduced by the oracle. While the ML ER pipeline completes on Music Brainz 200K at 1%, it struggles at 5% and fails to run on the million-scale datasets due to the large number of edges that need to be checked (e.g., over 500,000% for Cars-1M) combined with the inference time of the noisy oracle. As a result, we omit ML ER results for these larger datasets. The Simulation and LLM Embedding pipelines show similar patterns on low-duplication datasets, where underestimation remains high at the 1% sample size due to the effects of upscaling. For high-duplication datasets, the Simulation pipeline consistently overestimates, a trend we also observed on smaller datasets. This effect is pronounced on Cars-1M, reaching overestimation of over 20,000%, so we leave out these results in Figure 2.6. On Cars-1M, the LLM Embedding pipeline generally performs well at lower sample sizes, with the 1% sample (about 10,000 records) providing better estimates than larger samples on the smaller Cars dataset. When we increase the sample size to 10% and 20%, scalability issues emerge. The ML ER and Simulation pipelines hit memory limits due to the blocking graph's size, while the Statistical pipeline runs into memory limits with its similarity matrices. Clustering is manageable up to 10% sampling, but beyond this, it either times out or exceeds memory as the embeddings remain in memory. In summary, while sampling-based pipelines offer scalability, their reliability is most evident on high-duplication datasets like Cars and Alaska, particularly with the LLM embedding approach. For other datasets, achieving a stable approximation remains challenging, suggesting a need for adaptive techniques to improve accuracy across varying duplication factors.

2.8 Key Takeaways

The experimental findings highlight that sampling-based pipelines offer a scalable approach for entity count estimation, particularly effective in high-duplication

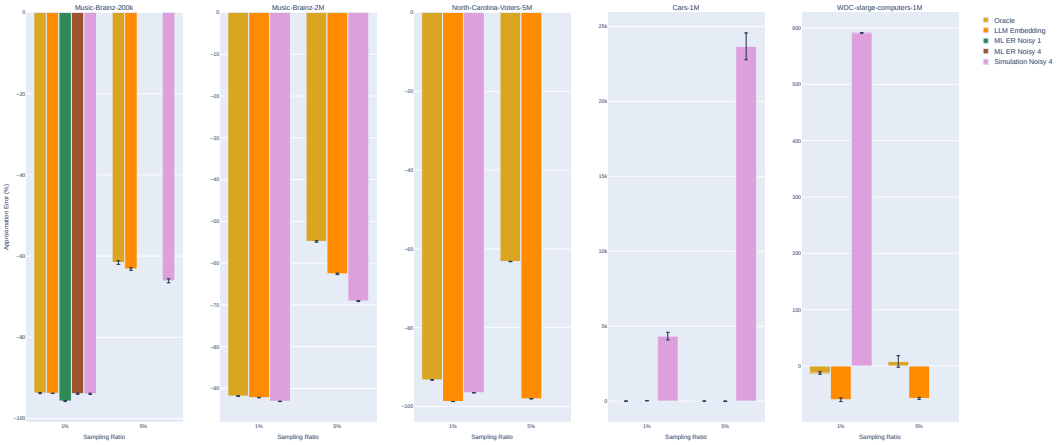


Figure 2.6. Sampling-based pipeline results on large datasets.

Dataset name	Sampling ratio			
	1%	5%	10%	20%
DBLP-Scholar	1.67	9.02	15.43	25.77
Alaska-monitor	4.69	24.85	42.19	60.47
Music-Brainz 2M	0.78	4.03	7.91	14.83
North Carolina Voters 5M	0.66	3.2	6.12	11.21
Cars 1M	99.53	99.91	99.95	99.98
WDC-xlarge-computers	94.22	98.5	99.17	99.54

Table 2.4. Average sample data duplication factor across different sampling ratios for various datasets

datasets such as Cars and Alaska. However, their accuracy is sensitive to factors including the upscaling method, sampling ratio, and dataset characteristics. In low-duplication datasets like Music Brainz 200k, approximation errors are notably higher, indicating a need for refined sampling strategies or adaptive upscaling to improve generalizability across diverse datasets. The LLM Embedding pipeline exhibited strong performance overall, though with some underestimation observed on complex, high-duplication datasets such as WDC-xlarge-computers-1M. The ML ER pipeline, which employs a simplified ER approach, maintained accuracy on smaller datasets but encountered scalability limitations on million-scale datasets due to the high memory requirements associated with large blocking graphs. The Simulation pipeline provided a computationally efficient approximation of the ML-based ER, achieving high accuracy on larger datasets like DBLP-Scholar and Alaska Monitor, though its estimation accuracy diminished with lower sampling ratios. Conversely, the Statistical ER pipeline consistently overestimated entity counts, particularly in larger datasets, suggesting limitations in modeling duplication patterns with statistical methods when sampling does not capture sufficient diversity. Finally, alternative estimators like Frank [80] and Berg [80] were evaluated but showed substantial variance, often resulting in extreme overestimates, underestimates, or negative values, despite being theoretically unbiased, due to their high variance compared to the Klusowski Bernoulli.

Dataset	Sampling Ratio	ML ER Noisy 4	Simulation Noisy 4	Statistical ER	LLM Embedding
DBLP-Scholar	1%	-11.57±1.68	-0.33±1.15	35.00±9.63	-0.59±0.80
	5%	-25.03±1.41	-0.25±0.82	8.51±2.27	-1.07±0.62
	10%	-31.29±0.42	0.22±0.66	16.57±1.15	-1.98±0.62
	20%	-40.21±1.39	-0.19±1.46	32.50±1.44	-3.26±0.55
Alaska	1%	-25.39±4.49	-2.34±10.61	24.31±44.31	9.51±9.51
	5%	-28.68±4.85	2.14±7.66	28.07±9.16	-72.29±5.74
	10%	-32.30±3.42	8.31±3.29	72.09±3.90	-84.78±3.59
	20%	-38.64±6.75	31.73±5.40	151.27±10.71	-91.14±0.73

Table 2.5. Approximation error on the sampled dataset for DBLP-Scholar and Alaska monitor benchmarking datasets

Limitations. Most pipelines in this study relied on in-memory techniques, leading to scalability limitations on datasets with millions of records. For example, the ML ER and Simulation pipelines used standard token-blocking for noisy graph construction, but more advanced metablocking techniques [172, 63, 171] that can scale with large datasets and offload data to disk were not employed. Integrating such methods could enhance scalability, especially for datasets with high edge counts. In the LLM clustering pipeline, both embeddings and clustering operations required in-memory processing, limiting scalability. Using clustering algorithms that are optimized for large datasets would allow handling beyond memory constraints [67]. The Statistical ER pipeline faced similar in-memory restrictions with Blink. For the ML ER sampling pipeline, we had to halt execution on Cars at 20% sampling and all million-scale datasets due to the large number of edges requiring oracle processing with a simulated 0.1-second delay per query—reflecting the average Ditto runtime on commodity hardware. This delay stems from hardware limitations; however, Ditto's original benchmarks [137] show that advanced servers can process millions of pairs within hours, improving feasibility for large datasets. Additionally, the time caps of 6 hours for base pipelines and 36 minutes for sampling pipelines would likely be reduced on high-end systems, yielding runtimes of only a few hours and minutes, respectively. Since the goal is fast entity count estimation, running on high-performance servers in future work would provide more insights into the pipelines' efficiency under optimal conditions.

2.9 Concluding Remarks

This work addressed the challenge of entity count estimation, proposing a general framework that combines machine learning-based entity resolution (ER), statistical record linkage, and clustering techniques. A key contribution was introducing a sampling-based variant, which estimates the number of entities within a sampled subset and upscales the result to the full dataset, to reduce the computational overhead for large datasets with millions of records.

The sampling pipelines showed marked improvements in time efficiency, particularly for datasets like Cars 1M, where smaller sample sizes still produced accurate entity count estimates. However, for datasets with lower duplication factors, the

accuracy diminished, often leading to underestimations. The upscaling process, while efficient, introduced errors, though in some cases these errors compensated for inaccuracies in the sample entity count, highlighting both the strengths and limitations of the approach.

There are several directions for future work. Exploring alternative sampling strategies like Simple Random Sampling (SRS) and integrating advanced ER methods like d-Blink [149] could improve the generalizability of the evaluation by including a wider range of approaches. While this work focused on sampling to improve efficiency, other techniques, such as parallelism and distributed processing, could further enhance scalability and were outside the scope of this study. Finally, memory constraints remain a concern for very large datasets. Future research could explore memory-efficient techniques, such as streaming algorithms, to ensure scalability to even larger datasets.

Chapter 3

Using Knowledge Graphs for Entity Resolution: Challenges and Opportunities

In this chapter we continue our discussion on data cleaning, which we study in the context of structured data, specifically tables and KGs in the healthcare domain. We summarize the concepts covered in this chapter in Figure 3.1.

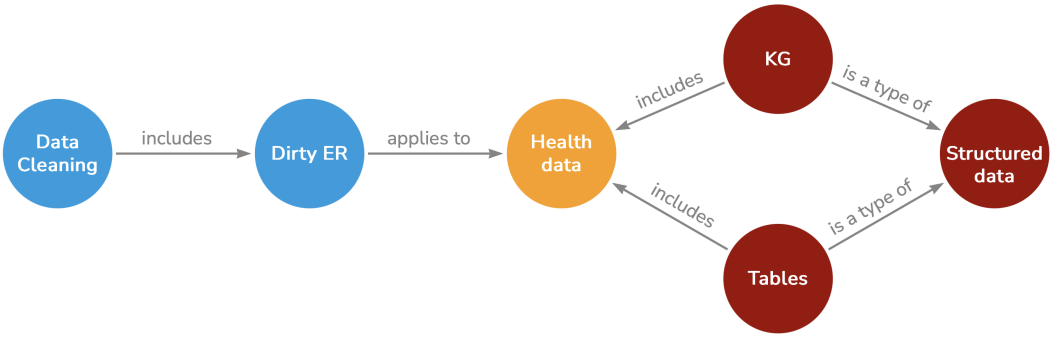


Figure 3.1. Tasks, domains and data types covered in this chapter

More specifically, in this chapter we explore how Knowledge Graphs (KGs) can potentially benefit Entity Resolution, also known as Record Linkage (RL). RL is the process of identifying and resolving duplicate records across different data sources, including structured, semi-structured, and unstructured data (e.g, in data lakes). RL is a critical task for information systems that rely on data to make decisions and is used in a wide variety of fields such as healthcare, finance, government and marketing. Due to recent advances in machine learning, there has been a significant progress in building automated RL methods. However, when dealing with vertical applications, featuring specialized domains such as a particular hospital or industry, human experts are still required to enter domain-specific knowledge, making RL prohibitively expensive. Despite KGs can be powerful tools to represent and derive domain-specific knowledge, their application to RL has been overlooked. Inspired by a healthcare case study in the Republic of Cyprus, we aim at filling this gap by identifying challenges and opportunities of using KGs to reduce the effort of solving

RL in vertical applications.

3.1 Introduction

The increasing value of data to individuals, organizations, and businesses is revolutionizing decision-making, problem-solving, and innovation. However, information systems face challenges in storing and analyzing vast amounts of heterogeneous data coming from multiple sources, such as humans, processes and devices (e.g., IoT devices), and different formats such as text, tables and images. Novel approaches such as data lakes, on the one hand, can mitigate this problem by allowing flexible storage of data from disparate sources such as databases and file systems, but on the other, they can lead to duplicate or conflicting information about real-world entities, such as persons or machines, making it difficult to obtain a complete and accurate understanding of the application domain. Record linkage (RL) is the process of identifying and linking records that correspond to the same real-world entity across multiple data sources to extract valuable insights, in a wide variety of application scenarios, from traditional databases to afore-mentioned data lakes [146]. RL is used in a broad range of domains, including healthcare, finance and government. For example, in healthcare, RL can help to identify patients who have been treated at multiple hospitals or clinics, and to ensure that their medical records are accurate and up-to-date. Over the years, researchers in the data management and the information systems community have developed different RL methods, including probabilistic methods [212], rule-based methods, and machine learning (ML) based methods [137]. Advances in natural language processing and increasing investments in data-driven information systems have fueled interest in RL in the last few years, leading to significant progress when dealing with highly descriptive data such as e-commerce and bibliographic datasets. Unfortunately, when dealing with vertical applications that are specific to a particular domain (e.g., healthcare or a particular industry) RL techniques must be tailored to take into account the unique characteristics of the data. This requires domain-specific knowledge which nowadays is provided by human experts, making RL in those applications significantly more expensive.

Our contribution. We envision how Knowledge Graphs (KGs) could reduce the effort in solving RL in vertical applications and describe specific opportunities and challenges for research in this direction. KGs are a powerful approach to manage domain-specific knowledge and their potential of capturing complex relationships and hierarchies that are often present in vertical applications has been demonstrated in a multitude of applications [102]. Yet, the integration of readily available KGs into RL algorithms to improve their accuracy remains largely unexplored. We utilize examples from a healthcare case study in the Republic of Cyprus to provide a tangible illustration of our vision. We believe that the benefits of KGs extends effortlessly to other RL domains.

3.2 Background and Related Works

We now provide the simplest definition of RL, without loss of generality. Given two data sources, U and V, the goal of RL is to find all pairs of records $u \in U$, $v \in V$

Record 1

Name	Surname	Age	Date	Time	Treatment	Hospital name
John	Doe	37	2023-03-14	12:00AM CET	Rifampin	Duke University Hospital

Record 2

Patient name	Surname	Gender	Date	Time	Drug	Hospital
John B.	Doe	Male	14-03-2023	12:00 AM	Isoniazid	Durham Regional Hospital

Match or no-match?

Figure 3.2. Illustrative example of a RL task on Electronic Health Records

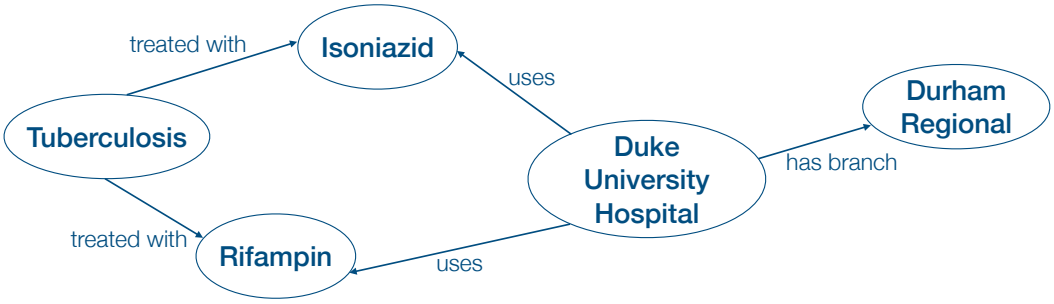


Figure 3.3. A schematic representation of a KG as a directed labelled graph.

that refer to the same real-world entity. Many RL approaches employ at their base a binary classifier: given records u, v the classifier predicts a *match* if u and v refer to the same entity and *non match* otherwise.

Figure 3.2 shows an illustrative example of an RL task in the healthcare domain where we need to identify patients. Differences in the anagraphic data can be handled with available RL techniques while information such as Treatment and Drug require domain-specific knowledge for correct disambiguation. In real-world applications, healthcare dataset can include also patients' medical history and lab results in a variety of formats. In our case study, datasets include lab results in structured format and Bilateral carotid duplex scannings (images).

A Knowledge Graph (KGs) is a graph-based store of real-world information, where the set of nodes represent entities and edges are labelled with semantic relations; two entities h and t linked by an edge with relation r form a *fact*, i.e., a directed labelled edge $\langle h, t, r \rangle$, where h is the head and t is the tail. Figure 3.3 shows a schematic representation of a KG in the healthcare field. Consider again Figure 3.2: by knowing from the KG that (i) Rifampin and Isoniazid are used in combination to treat the same condition and (ii) the hospital in record 1 is a branch of the hospital in record 2, we can automatically predict *match*.

Related works. Several works have explored and proposed solutions in the intersection of RL and KG. We can distinguish between two directions, one focusing on the use of RL to augment an existing KG, and another one that instead leverages the semantics encoded in the KG to design new RL methods. Works related to the first direction include [199], which presents and evaluates methods for incrementally

constructing a KG from multiple data sources using RL and [187], which discusses the different phases of KG construction that require the use of RL, such as when integrating new data into an existing KG or when merging information from two or more KGs, and proposes a RL model that can be applied in these settings. For the second research direction, we include [167] that investigates the usefulness of graph embeddings learned from a KG as input for ML classifiers for RL and [85] that proposes a comprehensive pipeline to extract textual features from images in order to build a KG and the learned KG embeddings for a downstream RL task. All these works focus on the usage of KG Embeddings or KGEs, that are low-dimensional vector representations of nodes and edges in the KG. KGEs can enable ML algorithms to operate on the KGs more efficiently but they have known drawbacks such as lack of interpretability, biased results (especially for rare entities) and limited support to KG updates. We also note that there are in literature RL methods that can use semantics relations, such as those in the KG, these techniques are often general-purpose and their effectiveness in vertical domains is still under investigation. In this chapter, we discuss advantages that KGs may bring to RL in the most general setting, without solely restricting to the use of KGEs. In fact, some of the opportunities outlined in Section 3.3 might rule out the use of KDEs, requiring a transparent, unbiased and continuous access to the evolving KG.

3.3 Using KGs for Solving RL Tasks

We envision that KGs can provide the necessary domain-specific knowledge with minimal or no human intervention. At a high level such system comprises three blocks: a RL model, a KG, and an intermediate block to facilitate communication between the RL model and the KG. More specifically, if the KG is stored in a graph database, the intermediate block would translate the queries from the RL model into graph database queries (e.g. SPARQL). For instance, when processing the pair of records in Figure 3.2, the RL model can submit to the intermediate block values of non-matching attributes, e.g., “Rifampin” and “Isoniazid” and “Duke University Hospital” and “Durham Regional Hospital”. Then the block, by querying the KG can return paths connecting related entities such as $\{\langle \text{Durham University Hospital, Uses, Rifampin} \rangle, \langle \text{Durham University Hospital, Uses, Isoniazid} \rangle\}$
and the singleton path $\{\langle \text{Durham University Hospital, has branch, Durham Regional} \rangle\}$. Finally, paths can be used in the RL model for learning similarities in vertical application. We now discuss opportunities (OPs) and challenges (CHs) in this setting.

OP1: Improving Accuracy. In our example, without domain-specific knowledge, patient records with different date, time, hospital name and treatment could be erroneously classified as non-match. Human experts are currently required to add specific rules to prevents such mismatches. However, a KG such as the one in Figure 3.3 could provide additional information on hospital structure and therapies to identify that both records refer to the same patient. Many domain-specific KG are already available (e.g. the PyKEEN repository `https://github.com/pykeen/pykeen`) and more are coming thanks to recent progresses in the fields of automatic KG construction and completion [215].

CH1: The main challenges are (i) mapping values of records (e.g., an hospital name)

to entities in the KG and (ii) learning which relations are relevant to the linkage process (e.g., analogies of treatments and branching of hospitals)

OP2: Data Augmentation. State-of-the-art techniques for RL are predominantly based on Deep Neural Networks. While these methods have demonstrated remarkable performance in practice, they can be as good as the labelled data fed to them. Unfortunately, manually acquiring enough non-trivial labelled data in vertical applications can be prohibitively expensive. Synthetic data generation and dataset augmentation techniques are often used in such scenarios [61] but they can yield non-realistic entries if not properly guided by domain knowledge. To address this issue, after identifying which relations are relevant to the linkage process, domain-specific KGs can be used to perturb available labelled data while preserving their semantics. For instance, one specific treatment could be replaced with an analogous one, and lab results could be generated by taking into account a coherent set of medical conditions.

CH2: The completeness and quality of the KG is critical, as incomplete information can lead to unrealistic synthetic data. Moreover the generated synthetic data may be too specific and as a result, the RL model may not be able to generalize well to new scenarios.

OP3: Temporal Variability. One important aspect of real-world data is its continuous evolution over time. For example, in the healthcare sector, patient data can change constantly due to various factors such as new treatments, hospitalisations, and the development of new medical conditions. Temporal Record Linkage (TRL) is a variant of the traditional RL problem aimed at linking records over time [132] such as the same patient across decades. Most methods for TRL are based on a time decay function that aims at capturing the effect of time elapsing on value evolution for the records. For instance, let us consider the records in Figure 3.2 and let us assume the existence of a third, older record referring to the same patient but with a lower age value. In this case, a TRL approach would assign a lower importance to the age feature of the third record when attempting to match the entries in 3.2. Domain-specific temporal KGs [102] may provide a more robust solution for the TRL task as these KGs are explicitly designed to incorporate the temporal axis in the graph structure. By using these KGs, the accuracy of the RL algorithms might significantly improve.

CH3: Research challenges include discovering relevant time-sensible relations without human intervention (e.g. the age feature in the previous example) and properly weighting such time-sensible features during the RL task based on their update frequency and their history in the temporal KG.

OP4: Multiple modalities. Most RL methods are focused on a single modality, such as textual data. However there is a significant lack of work on Multimodal RL (MRL), i.e. a variant of the RL problem where records comprise different modalities, which is often crucial in vertical applications, such as our healthcare case study, where tabular, text and image data coexist. The most recent MRL work we are aware of is [203], dealing with pairs of records containing text in one source and images on the other. On the other hand, there are several works on handling different modalities in the KG community, spanning from building a multimodal KG [108] to multimodal KG representation learning [161]. Incorporating these multimodal

techniques from the KG community might foster the development of novel MRL approaches.

CH4: In this context a major barrier is designing methods at the intermediate block level that conveniently translates the multimodal values in the RL records for querying the (possibly) multimodal KG.

OP5: Explainability. The growing trend of devising and implementing large ML models for RL tasks has brought significant improvement in accuracy at the cost of transparency, which can prevent their application in healthcare and other critical decision-making domains, such as finance and government. Existing methods to explain RL results, such as [220], aims at mitigating RL opacity. Unfortunately their results are limited to either the record pair at hand or an additional subset of the dataset used to train the RL model. For instance, let us consider the toy example in Figure 3.2 and assume we have an accurate model for RL that predicts these two records to be a matching pair. An explainability framework might unveil that those two records are matching due to the name, surname, treatment and hospital name features. However, it might be unclear why the hospital name is such a relevant feature for the matching outcome. In this context, using a KG from the same domain as the RL task (e.g. the KG in Figure 3.3) can potentially give access to a broader set of contextual data that can be used to design more effective and intuitive explanations.

CH5: Challenges include identifying relevant information in the KG to help explaining the RL prediction as well as developing algorithms to automatically extract relevant information from the KG and integrate it with the RL model.

3.4 Concluding Remarks

In this chapter we discussed how exploiting Knowledge Graphs for Record Linkage in vertical applications can bring several advantages for specific, vertical domains, such as healthcare, and explored the main benefits of such integration. We believe that KGs may empower RL task by improving their accuracy while reducing effort of human experts, providing high-quality training data for RL algorithms, improving the handling of temporally-changing features and multimodal data and finally potentially provide better explanation for the predictions made by ML based RL models. We also identified the main challenges when pursuing application of domain-specific KGs to RL. We leave the investigation of such challenges to a future work.

Chapter 4

NLP-based management of Large Multiple-Choice Test Item Repositories

In this chapter we further continue our discussion on data cleaning, which we study in the context of unstructured data, specifically QA from large scale repositories of multiple-choice questions. We summarize the concepts covered in this chapter in Figure 4.1.

Figure 4.1. Tasks, domains and data types covered in this chapter

Multiple-choice questions (MCQs) are widely used in educational assessments and professional certification exams. Managing large repositories of MCQs, however, poses several challenges due to the high volume of questions and the need to maintain their quality and relevance over time. One of such challenges is the presence of conceptually duplicate questions but formulated differently. Such questions can indeed elude syntactic controls but provide no added value to the repository.

In this chapter, we focus on this specific challenge and propose a workflow for the discovery and management of potential duplicate questions in large MCQs repositories. Overall, the workflow comprises three main steps: MCQs preprocessing, similarity computation and finally a graph-based exploration and analysis of the obtained similarity values. For the preprocessing phase we consider three main strategies: (i) removing the list of candidate answers from each question, (ii) augmenting each question with the correct answer or (iii) augmenting each question with all candidate answers. Then, we use deep learning-based natural language processing (NLP) techniques, based on Transformers architecture, to compute similarities between MCQs based on semantics. Finally, we propose a new approach to graph exploration based on graph communities to analyze the similarities and relationships between MCQs in the graph. We illustrate the approach with a case

study of the "Competenze Digitali" program, a large-scale assessment project by the Italian government.

4.1 Introduction

Multiple-choice questions (MCQs) are a popular choice for knowledge assessment in several contexts, ranging from university admission to candidates evaluation for a job position, from self assessment to game shows like Who Wants to Be a Millionaire? and several successful apps for mobile gaming, such as QuizDuello.

Many large-scale standardized tests use multiple-choice questions (items) that typically contain four response options, where one option is the correct response (item key) and the other three are the incorrect responses (distractors).

The academic research about MCQs focused naturally on their effectivess as a tool for evaluation; from a learning analytics point of view, we refer to the work of Azevedo, Oliveria and Damas Beites [14] where the authors focused on the problem of finding appropriate forms of analysis of multiple-choice questions (MCQ) to obtain an assessment method, as fair as possible, for the students. We recall that a commonly accepted definition of Learning Analytics (LA) is "*the measurement, collection, analysis and reporting of data about learners and their contexts, for purposes of understanding and optimising learning and the environments in which it occurs*" [144].

Thus, in the context of learning analytics, we also cite a research area focused on the automatic generation of multiple-choice questions, since, as observed for example in [157], their manual creation is a laborious and time-consuming task. The approaches, based on Natural Language Processing (NLP), focused on several different techniques such as the use of WordNet, shallow parsing, corpora, ontologies, and, more recently, deep neural networks.

In this chapter we propose a novel approach for a related task: the *maintanance* of a large repository of multiple-choice questions, and present a prototypal tool, used in a real large scale assessment project by Italian Government. The tool is based on NLP and graph visualization to support the administrator of the learning system in its dynamic use. Our tool can be effective for:

- Finding questions that are *too similar*.
- Automatically check the coherence between a question and the area of the syllabus it belongs to.
- Support the *refactoring* of a syllabus, i.e., when the syllabus changes and thus the questions need to be rearranged properly.
- Support the merging of two repositories, by matching the imported questions against the new Syllabus they are moved to.

The results suggest that NLP-based techniques can be beneficial in the identification of conceptually duplicate MCQs compared to traditional frequency-based methods, particularly when it comes to questions having the same semantics despite having different wordings. Furthermore, our proposed workflow integrates graph

exploration based on graph communities, which presents a novel avenue for comprehending the intricate similarities among MCQs. Our graph exploration step proved helpful in understanding the complex relationships between duplicate questions in our case study. Our proposed approach augments the existing theoretical framework by introducing a practical means to visualize and analyze the complex interconnections within large-scale MCQ repositories.

This chapter is organized as follows: next section provides the necessary background, whilst Section 4.3 presents our approach, that is then detailed, in Section 4.4 against a concrete example: the *Competenze Digitali* program, i.e., the before mentioned large scale assessment project made by the Italian Government. In this section we present the results we obtained using our prototypal tool, developed in Python, using Scikit-learn , the Transformers library [240] and the Sentence Transformers library [192] for the NLP part and Networkx library for graph management.

4.2 Related Work

The use of Multiple Choice Questions (MCQs) as an assessment tool is garnering increasing interest within the modern educational landscape [244, 126] MCQs have found extensive application in various contexts, including college admissions and job placements, serving as a guiding factor in decision-making processes. Despite their unquestionable advantages in knowledge evaluation, MCQs present certain challenges, such as designing questions that align with course objectives and outcomes [218], as well as the potential for answer memorization through the repetition of MCQ stems (i.e. questions), thereby undermining the validity of the examination [241]. As mentioned in the introduction, so far learning analytics field of research focused, with respect to multiple-choice tests, mainly on their effectivess as a tool for evaluation; in particular, as observed by Azevedo, Oliveria and Damas Beites [14], there are two prominent theories regarding the analysis of questions in assessment tests: the Classical Test Theory (CTT) and the Item Response Theory (IRT); in the CTT the unit of analysis is the overall test, whilst in IRT it is the single item.

There are several techniques that have been used in NLP, including Vector Space Model [100], word2vec [88] and the related word embedding [23], semantic features [201], Topic Modellng [101], and Deep Neural Networks, used to generate embeddings [202] and to mimic several of the other approaches (see, e.g, the survey of Kanwal et al. [109]).

More related to our line of research, we mention the approaches aimed to automatically generate multiple-choice questions using some of the NLP techniques, mentioned above, including ontologies [173] Machine Learning (ML) based approaches [32] and hybrid approaches that combines ontologies and ML [126]. Additionally, there is the related area of automatic answering of multiple-choice questions, which has been investigated through word-occurrence based approaches [150] as well as more recent ML and deep learning techniques [34, 115].

The problem of identifying duplicate entries (that are, in this case, questions) in a database is generally referred to as Entity Resolution [87]. In traditional data management applications, the most popular approaches include supervised methods bases on learning a vector representation of database records and their text

attributes [26, 62, 162]. In this chapter, we focus instead on similarity computation and leave the final decision to a human expert, in the spirit of recent oracle-based approaches such as [74, 83].

4.3 Managing Large MCQs Repositories: the workflow

In this section we detail our proposed approach for the management of large MCQs repositories. The key of the whole approach is the computation of the similarities between the questions; after this computation we (i.e., the experts) need to do some (simple) exploratory data analysis in order to find the threshold value σ of the similarity, i.e. the value that "separates" similar questions from non similar questions. Deciding the threshold value σ is a choice that will affect all the subsequent steps: a small value of σ will results in several pairs of similar questions, whilst a high value might conduce to few pairs of them.

In order to decide a *reasonable* value of σ we do have two tools: on the one side we can plot the distribution of the similarity values, in order to get an idea of the cardinality; on the other side we can inspect, for a given pair of questions, the actual text of the questions, in order to decide whether or not we do believe them to be similar.

Once we select a value of σ, we can build the graph G_σ, i.e. the graph whose nodes are the questions and there is an edge between two nodes n_i and n_j (corresponding to the questions q_i and q_j) if and only if the similarity of q_i and q_j is greater than σ.

Finally, we can visually explore the graph in order to understand the potential problem between the questions. In this case, it is particularly relevant to focus on the connected components of the graph, i.e. clusters of nodes that are connected and, thus, are similar between each other. In the following section we do provide some examples.

Summing up, the workflow proposed is the following:

1. Similarity Computation: compute the similarity values between all the pairs of question. This step is performed in an unsupervised manner, meaning that it does not depend on a labeled set of questions with known similarity values to determine the similarity between each pair of questions.

2. Threshold Definition: based on the similarity values distribution, define (i.e., choose) a threshold value σ.

3. Graph Construction: create the graph G_σ, whose nodes corresponds to the questions and and there is an edge between two nodes if and only if the similarity of the corresponding questions is greater than σ.

4. Graph Exploration: use graph visual analysis to explore the relationships between the questions.

4.4 Managing Large MCQs Repositories: an example

In this section, we present an illustrative application of the workflow described in the previous section. Our aim is to investigate and answer three specific research

questions using the proposed methodology.

- R1: Can Natural Language Processing techniques, in particular language models, be utilized to compute semantic similarities between MCQs?
- R2: What are the most effective preprocessing strategies for MCQs prior to similarity computation: removing the list of candidate answers or augmenting each question with the correct answer?
- R3: Can graph exploration techniques, specifically graph communities, assist in identifying potential issues and improving the quality of the repository?

In this section we provide an example using the MCQs database of the *Competenze Digitali* program[1]; this program aims to provide public employees (non-IT specialists) with personalized training, in e-learning mode, on basic digital skills starting from a structured and homogeneous survey of training needs, in order to increase involvement and motivation, performance, diffusion and quality of online services, simple and fast, for citizens and businesses. The implementation of the program is based on the following elements:

- The Syllabi that describes the minimum skills required to public employees to operate in an increasingly digital PA ("Pubblica Amministrazione", i.e. Public Administration)
- The online platform that supports processes for detecting individual skills gaps, defining training courses and providing training
- The catalog of quality training, thanks to the collaboration of large players, public and private

The MCQ dataset of the *Competenze Digitali* program comprises 798 Italian language questions, each presenting four candidate answers, of which only one is correct. Every question corresponds to a particular syllabus, which groups together questions related to the same topic (e.g. computer networks). In total, there are 11 distinct syllabi in the dataset, with their respective question counts listed in Table 4.1. We also provide a sample question in Figure 4.2. Please note that this question and the answers are based on a question that bears similarity to one found in the MCQ dataset used for this study. However, in order to maintain the privacy and confidentiality of the dataset, the original question and its corresponding answers cannot be provided.

Moreover, our MCQ dataset lacks labels, meaning that we do not have access to a definitive measure of similarity (binary or real-valued) for each question pair, indicating their true level of similarity.

Similarity Computation. In our case, as mentioned before, we use and compare the two representative methods below:

- the method `CountVectorizer` from the Scikit-learn library [179]; we used this library because it can be considered as a standard de facto for a large number

[1]The online platform and details of the program are online at the url https://www.competenzedigitali.gov.it/.

Syllabus	S1.1	S1.2	S1.3	S2.1	S2.2	S3.1	S3.2	S4.1	S4.2	S5.1	S5.2	**Total**
Count	70	94	50	68	134	49	40	57	97	62	77	798

Table 4.1. Number of questions in the MCQ database for each syllabus

Question: What is the definition of "cookies"?
A. Small files that store information about online browsing on the user's computer or device.
B. Small files that track and collect user's online behaviors for targeted advertising purposes.
C. Tokens generated by web applications to authenticate user sessions and enable personalized features.
D. Privacy settings that allow users to control the information shared with websites they visit.

Figure 4.2. Sample question and answers inspired by a similar question in the MCQ Dataset. For clarity, the question has been translated into English

of Machine Learning tasks, including basic Natural Language Processing, and it provides a rich and well designed API [28].

- Multilingual Sentence Transformer models from the Sentence Transformers library [192]. These models, which can handle multiple languages, have the ability to encode input sentences into dense vectors that capture semantic meaning. This makes them particularly effective for tasks involving sentence similarity and text classification, regardless of the language used.

We now describe in more details the Sentence Transformer models used in the experiments. We considered four multilingual models that were fine-tuned also with Italian language data, thus allowing them to handle Italian language texts with higher accuracy.

- The first one, dubbed `ST-Roberta`, is a variant of the XLM-RoBERTa transformer model [46] that was initially finetuned on a large scale paraphrase dataset consisting of more than 50 languages [194] and then further finetuned for English and Italian language. The model is available in the Huggingface repository under the name `T-Systems-onsite/cross-en-it-roberta-sentence-transformer`.

- The second model, dubbed `ST-DistilUSE` is a multilingual knowledge distilled version of of multilingual Universal Sentence Encoder (mUSE) [247, 194] that supports 15 languages, including Italian. The model is listed as `distiluse-base-multilingual-cased-v1` in the Huggingface repository.

- The third model, named `ST-MiniLM`, is a multilingual variant of MiniLM [232] that has been fine-tuned on a vast collection of paraphrase data spanning over 50 languages [194]. The model can be found as `paraphrase-multilingual-MiniLM-L12-v2` in the Hugginface repository.

- Finally, the fourth model, dubbed `ST-MPNet`, is a multilingual variant of MPNet [210] that has been trained on parallel data for more than 50 languages [194]. The model is available in the Huggingface repository as `paraphrase-multilingual-mpnet-base-v2`.

In our experiments, we considered the above listed four Sentence Transformer models that were fine-tuned on Italian language data, as this was the specific language of interest. However, in general, the model selection phase can have a significant impact on the quality of the results. For instance, a model trained on legal documents may perform better on legal texts, while a model trained on social media data may perform better on social media texts. Therefore, it is essential to consider the specific language and domain of the text and choose the most appropriate model accordingly.

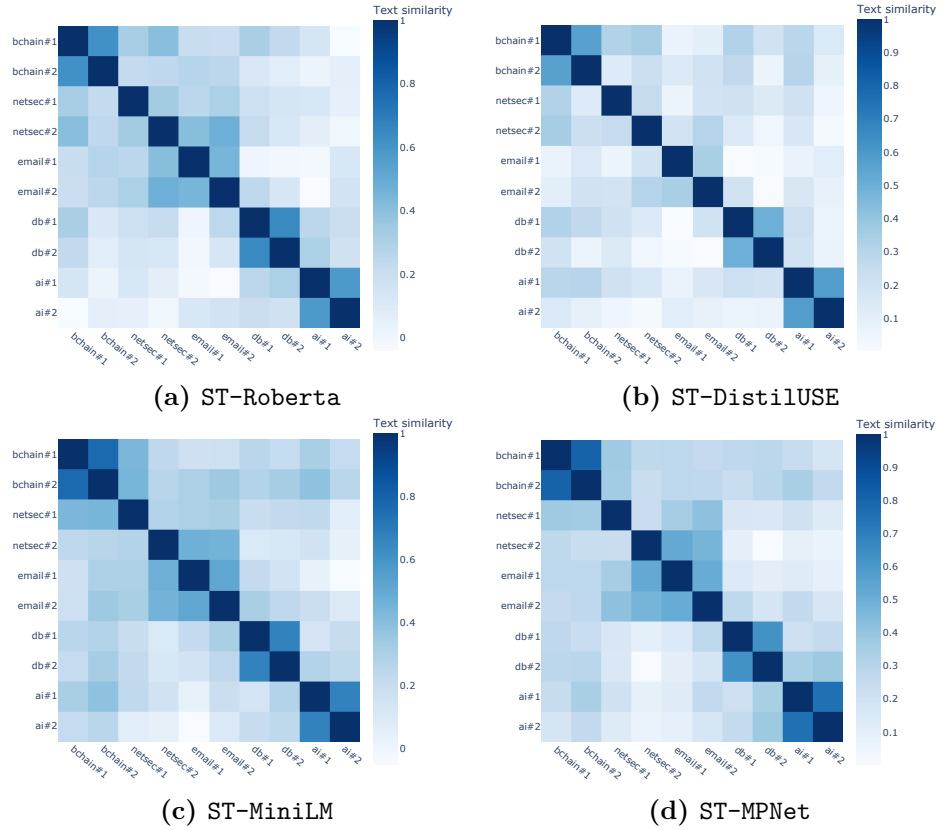
From this initial selection of Sentence Transfomer models we empirically evaluated their performance on a subset of our MCQ dataset and selected the one that achieved the best results. The subset of data used in our experiments was created by handpicking pairs of similar questions (note that these questions are not duplicate) that were related to the same syllabus. We recall that each syllabus focuses on a specific topic, such as blockchains. The goal was to evaluate how well the selected models could capture semantic similarities between questions on the same topic. We report the results in Figure 4.3, where we show the similarity matrix produced by each model on the select pairs of sentences.

We indicate for each question in Figure 4.3 its corresponding topic(e.g., `ai#1` and `ai#2` are two sentences related to AI). Ideally, we would expect to see a pattern of dark blue squares of size 2 on the diagonal of the similarity matrix, indicating that each pair of questions related to the same topic has a high similarity score and is also dissimilar from the other questions in the dataset. However, the results on the subset of data varied across different models. For instance, `ST-Roberta` and `ST-DistilUSE` gave a modest similarity score (ranging from 0.55 to 0.61) to the first pair of questions related to blockchains (i.e. `bchain#1` and `bchain#2`) compared to `ST-MiniLM` and `ST-MPNet`, which were more prone to give higher scores to questions related to the same topic. After comparing the results across different models, we observed that `ST-MPNet` performed better compared to the others, and hence we chose it for the subsequent stages of our pipeline.

It is important to mention that, irrespective of the choice between the above similarity methods, there are different approaches for computing the similarity of the questions in the case of Multiple Choice Test Items, based on the the following observation: the text of the question should include the possible answers or not? We do believe this choice depends on the repository (and, in some sense, on the domain); for example, our repository included several questions whose text was simply "*Which of the following statement is false?*" that, without adding the four response options, are indeed completely identical.

We experimented three approaches, dubbed question-only, all-answers, correct-only.

1. **Question-only.** The text for similarity computation is only the text of the question,



(a) `ST-Roberta` (b) `ST-DistilUSE`

(c) `ST-MiniLM` (d) `ST-MPNet`

Figure 4.3. Similarity matrices produced by each Sentence Transformer model on the handpicked dataset of similar questions pairs

2. **All-answers.** The text for similarity computation is the text of the question together with the texts of the four answers provided,

3. **Correct-only.** The text for similarity computation is the text of the question together with the texts of the correct answer.

In the following, when not stated explicitly, we report the results of the second case, i.e. the all-answers approach.

In our case, we had 798 MCQs in our repository, thus the number of pairs of questions is $798 \cdot (798 - 1)/2 = 318,003$.

In Figure 4.4 we can see the similarity values sorted by their value; as we can expect, on the one side of the table we have completely unrelated questions (similarity equals to 0) and, on the other side, we can spot some suspicious items definitely too similar (similarity values in the range $0.98 - 0.99$). Since, as we mentioned before, we have more than three hundred thousands similarity values, we need to analyze them using some plots, as we see in the following.

Threshold definition. In Figures 4.5–4.7 we show the distribution of similarity values for all the pairs of questions computed by the five similarity computation methods (`CountVectorizer`, `ST-Roberta`, `ST-DistilUSE`, `ST-MiniLM` and `ST-MPNet`) and the three approaches considered (question-only, all-answers, correct-only).

	Q1	Q2	similarity
281952	Q-4.1.2.2-3	Q-5.2.3.5-4	0.000000
231680	Q-2.2.2.6-11	Q-5.2.3.5-6	0.000000
210280	Q-2.2.2.1-1	Q-3.2.2.3-1	0.000000
231679	Q-2.2.2.6-11	Q-5.2.3.5-5	0.000000
225331	Q-2.2.2.5-4	Q-5.2.3.5-2	0.000000
...	...	...	...
311105	Q-5.1.2.2-1	Q-5.1.2.2-7	0.985793
216978	Q-2.2.2.3-7	Q-2.2.2.3-8	0.987500
215176	Q-2.2.2.3-3	Q-2.2.2.3-8	0.987500
215175	Q-2.2.2.3-3	Q-2.2.2.3-7	0.987500
311104	Q-5.1.2.2-1	Q-5.1.2.2-6	0.990566

318003 rows × 3 columns

Figure 4.4. Question similarities sorted sorted by values (`CountVectorizer`, all-answers).

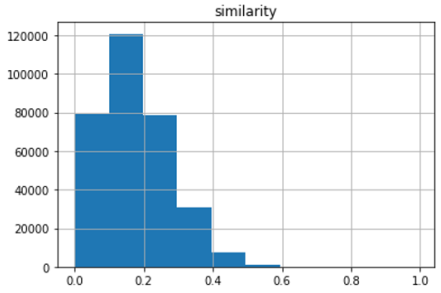
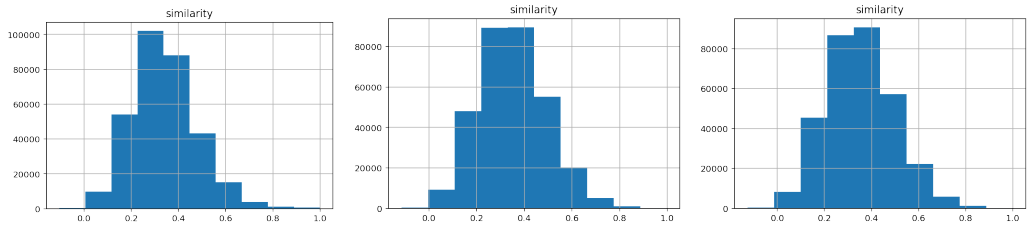


Figure 4.5. Similarity values distribution (`CountVectorizer`, all-answers)

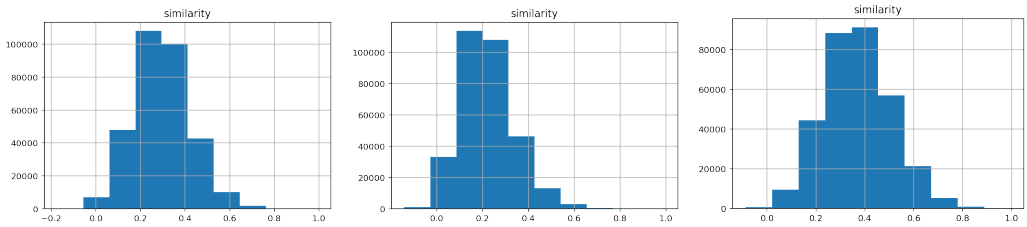


(a) question-only **(b)** all-answers **(c)** correct-only

Figure 4.6. Similarity values distribution (`ST-MPNet`).

In particular, Figure 4.5 reports the distribution for the `CountVectorizer` method in the all-answers approach. We can see that approximately two hundred thousands pair have a similarity score less than 0.2.

Figure 4.6 shows the similarity values distribution for the `ST-MPNet` methods in the three approaches considered (question-only, all-answers, correct-only). With respect to `CountVectorizer` we have generally higher similarity scores, because `ST-MPNet` can better capture synonyms (e.g., data base management system and



(a) `ST-Roberta` (b) `ST-DistilUSE` (c) `ST-MiniLM`

Figure 4.7. Similarity values distribution (all-answers)

Value	# of pairs	percentage
$0 <= \sigma < 0.4$	309,379	97.29%
$0.4 <= \sigma < 0.5$	7,192	2.26%
$0.5 <= \sigma < 0.6$	1,084	0.34%
$0.6 <= \sigma < 0.7$	125	0.04%
$0.7 <= \sigma < 1.0$	223	0.07%
total	318,003	

(a) `CountVectorizer`, all-answers

Value	# of pairs	percentage
$0 <= \sigma < 0.4$	222,310	69.91
$0.4 <= \sigma < 0.5$	57,791	18.17
$0.5 <= \sigma < 0.6$	25,335	7.97
$0.6 <= \sigma < 0.7$	8,786	2.76
$0.7 <= \sigma < 1.0$	3,781	1.19
total	318,003	

(b) `ST-MPNet`, question-only

Value	# of pairs	percentage
$0 <= \sigma < 0.4$	205,771	64.71%
$0.4 <= \sigma < 0.5$	64,958	20.43%
$0.5 <= \sigma < 0.6$	32,564	10.24%
$0.6 <= \sigma < 0.7$	10,874	3.42%
$0.7 <= \sigma < 1.0$	3,836	1.21%
total	318,003	

(c) `ST-MPNet`, all-answers

Value	# of pairs	percentage
$0 <= \sigma < 0.4$	204,254	64.23%
$0.4 <= \sigma < 0.5$	64,310	20.22%
$0.5 <= \sigma < 0.6$	33,182	10.43%
$0.6 <= \sigma < 0.7$	11,897	3.74%
$0.7 <= \sigma < 1.0$	4,360	1.37%
total	318,003	

(d) `ST-MPNet`, only-correct

Table 4.2. Similarity values distribution for different values of σ

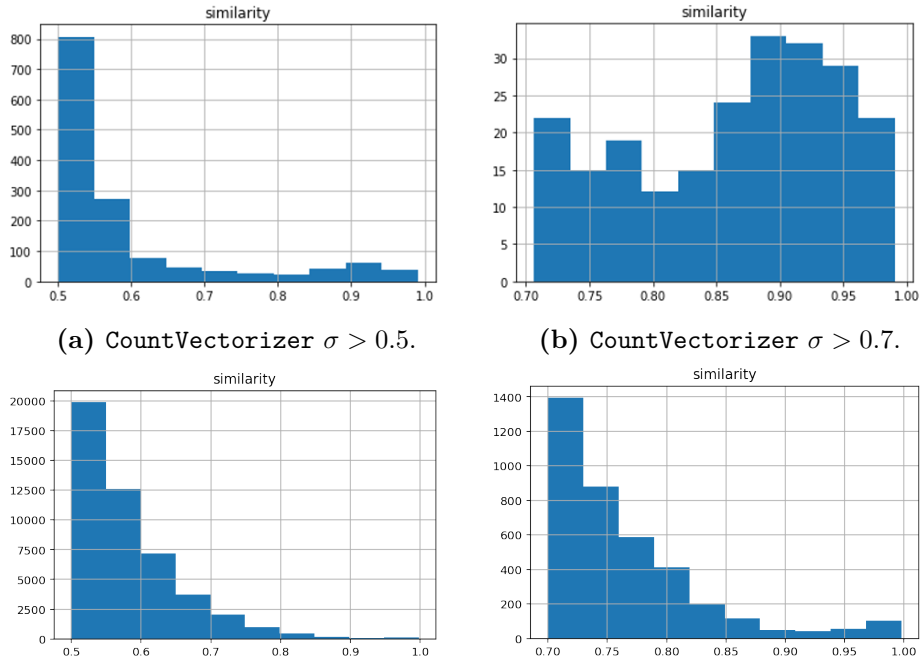
DBMS) and cross-lingual references (e.g., Open Data and Dati Aperti).

Finally, in Figure 4.7, we plot the results for the other selected models from the Sentence-Transformer library (all-answers). The similarity distribution of these methods is analogous to `ST-MPNet`.

From now on we consider `CountVectorizer` and `ST-MPNet`, respectively because it is simple and because it performed better compared to the other models. In Table 4.2 we have more detailed picture numbers for these methods. In particular, for the `CountVectorizer` method we can see that the large majority ($\approx 97\%$) of pairs have a similarity score less than 0.4, and almost all the pairs ($\approx 99\%$) have a similarity score less than 0.5. For `ST-MPNet`, the same buckets contain $\approx 70\%$ and $\approx 88\%$ of the pairs respectively, with more than $\approx 10\%$ of pairs having similarity larger or equal than 0.5.

For the subsequent steps, we consider as thresholds the natural value 0.5 and a slightly higher value 0.7. In Figure 4.8, we show the filtered similarity value distributions for the `CountVectorizer` and `ST-MPNet` method, with respect to the two threshold values.

Graph construction with `CountVectorizer`. As we mentioned before, we refer to two different values of the threshold σ, obtaining two different graphs.



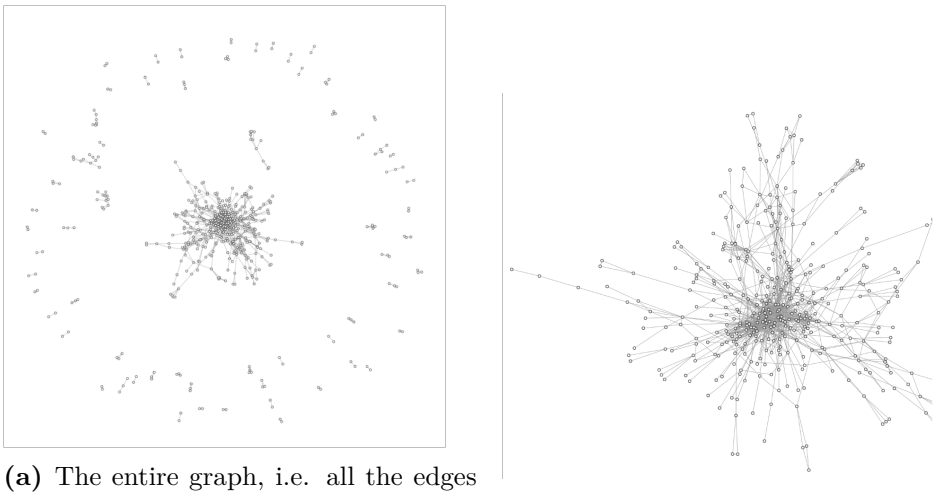
(a) `CountVectorizer` $\sigma > 0.5$.

(b) `CountVectorizer` $\sigma > 0.7$.

(c) `ST-MPNet` $\sigma > 0.5$.

(d) `ST-MPNet` $\sigma > 0.7$.

Figure 4.8. Filtered similarity values distribution (all-answers)



(a) The entire graph, i.e. all the edges with similarities $\sigma > 0.5$

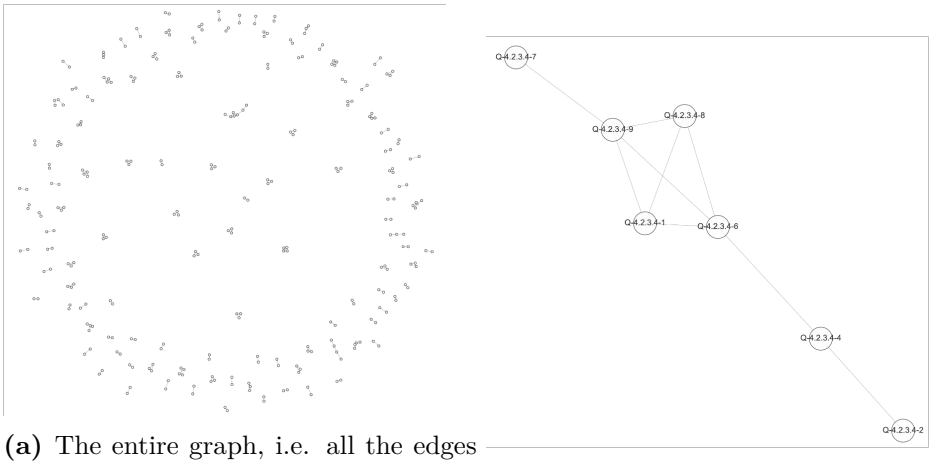
(b) The largest connected component.

Figure 4.9. Topology layout of G_5 (`CountVectorizer`, all-questions)

Indeed, we should aim at one threshold value and one graph, but here we want to provide two different examples. In particular, we now consider the similarity values computed by `CountVectorizer` and build

- G_5, with all the edges whose similarities values are > 0.5, and
- G_7, with all the edges whose similarities values are > 0.7.

Note that, by definition, all the edges of G_7 are also edges of G_5.



(a) The entire graph, i.e. all the edges with similarities $\sigma > 0.7$

(b) The largest connected component.

Figure 4.10. Topology layout of G_7 (`CountVectorizer`, all-questions)

Note that we do not consider all the questions as nodes of the graph, but only the ones that have at least one edge, i.e., the ones that have a similarity value (greater than the threshold) with another question. Thus, our graphs do not have 798 nodes, but, respectively, G_5 has 569 nodes and 1430 edges, and G_7 has 296 nodes and 223 edges.

We can now visually explore the similarities of the questions in the repository. Let us start with G_5, shown in Figure 4.9a. As we can see, this graph is still a (in this context) a moderately large graph, with more than one thousand edges. If we focus on its structure, we find that it has 66 connected components (CC); the largest CC is shown in Figure 4.9b: it has 384 nodes and 1271 edges; indeed, looking at Figure 4.9a we can see that the graph is made by a single large connected component and several tiny ones.

Now let us focus on G_7, shown in Figure 4.10a. In this case we have a relatively small graph, made of several tiny connected components; indeed, it has 122 CCs, and the largest one, shown in Figure 4.10b, is small: it has only 7 nodes and 9 edges.

Graph construction with `ST-MPNet`. We now follow the same graph construction protocol but use the similarity values computed by the `ST-MPNet` method. The resulting G_5 graph has 797 nodes and 47274 edges, while G_7 has 718 nodes and 3836 edges. Given the higher similarity values computed by `ST-MPNet` with respect to `CountVectorizer`, both G_5 and G_7 are much more connected, with 1 and 30 connected components respectively. Also, the largest CCs are larger, with 797 and 614 nodes respectively. In this scenario, the CC-based analysis may be too coarse and further decomposition may be needed. To this, we show in Figure 4.11 the result of the traditional Clauset-Newman-Moore community detection algorithm [44] on the largest CC of G_7. In many applications, the communities resulting from the latter step are fine-grained enough to be visually explored and groups of similar questions can be identified manually. Otherwise, it is also possible to apply node-centrality methods like [13, 12] to identify salient questions and boost visual exploration.

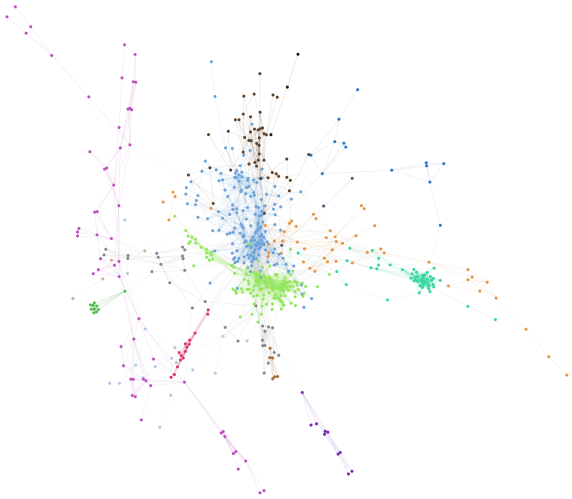


Figure 4.11. The largest CC in G_7 (`ST-MPNet`, all-answers) with node colors highlighting the community structure.

4.5 Application Tasks

Now we discuss some applications of the above mentioned approach.

4.5.1 Syllabus Coherence

Using the similarity techniques described before it is also possible, if a Syllabus is available, to check the coherence of the questions to specific areas of the Syllabus. In the case of the Competenze Digitali program, our syllabus is made of 5 (major) areas and 11 sub-areas; in particular, the first area has three sub areas (denoted with S1.1, S1.2 and S1.3) whilst the other four areas have only two sub areas (for example, for the second area we have S2.1, S2.2). The questions in the database have an unique identifier that starts with the number of the area and the number of the sub-area.

Using this, we can easily check whether a question is similar to the (text description of the) sub-area of the Syllabus it belongs. In our dataset we have 798 questions and 11 sub-areas, so we computed $798 \cdot 11 = 8778$ values of similarity. Look at Figure 4.12, where we can see the head and the tail of the values; the first eleven items (we do see only 5 of them) are the similarity score of the first question in the dataset against the first five sub-areas of the Syllabus; it is easy to see that the largest score, among the ones shown in the table, is exactly for the sub-area of the Syllabus it belongs (i.e., Q-**1.1**.1.1-1 belongs to S**1.1**), and the same happens for the last question (i.e., Q-**5.2**.3.5-8 belongs to S**5.2**).

In our dataset, approximately half of the questions have the largest similarity score, among the eleven sub-areas of the Syllabus, with the one they belong to.

This finding shows the capability of the proposed approach to evaluate the coherence between questions and the syllabus, enabling educators to ensure alignment between assessment items and specific areas of the curriculum. By examining the similarity scores and the corresponding sub-areas of the syllabus, educators can

	Q	S	similarity
0	Q-1.1.1.1-1	S1.1	0.415891
1	Q-1.1.1.1-1	S1.2	0.147890
2	Q-1.1.1.1-1	S1.3	0.254660
3	Q-1.1.1.1-1	S2.1	0.158570
4	Q-1.1.1.1-1	S2.2	0.202694
...	...	...	...
8773	Q-5.2.3.5-8	S3.2	0.041329
8774	Q-5.2.3.5-8	S4.1	0.096101
8775	Q-5.2.3.5-8	S4.2	0.073620
8776	Q-5.2.3.5-8	S5.1	0.106045
8777	Q-5.2.3.5-8	S5.2	0.175479

8778 rows × 3 columns

Figure 4.12. The similarity values between the questions and the areas of the Syllabus.

identify areas that may require additional focus or clarification (e.g. lack of questions related to a specific syllabus). Moreover, the evaluation of coherence between questions and the syllabus also helps educators identify potential redundancies in their question sets. By comparing the similarity scores across different sub-areas, educators can pinpoint instances where multiple questions assess the same or similar concepts.

In these scenarios, leveraging modern NLP-based models (such as deep-learning architectures) instead of relying solely on lexical approaches can effectively allow such use case. By considering the semantic and contextual relationships between words and phrases, NLP models can provide a more comprehensive understanding of the content and meaning of the text, compared to our previous approach that was limited to the co-occurence of words in both questions and the syllabus.

4.5.2 Syllabus Refactoring

Our proposed approach is also useful in the syllabus refactoring process, which involves making changes to one or more parts of the syllabus. In this context, the challenge lies in identifying the questions that pertain to the modified parts of the syllabus. In contrast to our previous based work word-occurrence based similarity methods, our new approach incorporates modern NLP models, allowing us to analyze the textual content of both the modified syllabus components and the questions in the repository to accurately measure semantic similarity and capture more nuanced relationships between the two. With the assistance of our approach, educators can easily identify the questions that are related to the specific parts of the syllabus that have undergone changes. By analyzing the semantic similarities and contextual

relevance, the models can accurately pinpoint the questions that align with the modified syllabus sections.

One notable advantage of our approach in syllabus refactoring is its efficiency. Manual methods of tracking and updating questions in response to syllabus changes can be time-consuming and error-prone. Our approach can help in identifying and updating the questions that align with the modified syllabus, reducing the burden on educators and ensuring a more streamlined and accurate identification of relevant questions.

4.5.3 Merging Repositories

The similarity based approach could be useful also when one has to merge two different MCQs repositories that are based on the same topic but refer to different syllabi: in this case, whilst a "natural" approach is usually a manual work in which one looks how to match the areas of the syllabi, in practice usually there is an overlap in several areas, that makes not possible to find a direct correspondence. In this case, it might be more practical to simply compute the similarity between the new set of questions and the areas of the Syllabus they are merged into; this approach is not aimed to replace a human work but to simplify it by a quick preprocessing. The use of NLP-based models can provide useful in this application scenario. Instead of relying solely on lexical similarities or manual matching (as in our previous work), NLP models can consider the underlying meaning and conceptual overlap between questions and syllabus areas. This enables a more accurate assessment of the alignment and facilitates the identification of related content, even when there is no direct correspondence between the two repositories.

4.6 Discussion

In this section we revisit our three research questions and envision further development, starting from current limitations of the proposed approach and how end users (e.g. educators) could use the proposed approach in real-world scenario.

4.6.1 Answer to Research Questions

We now provide an answer to our research questions based on our experimental observations.

R1 *Can Natural Language Processing techniques, in particular language models, be utilized to compute semantic similarities between MCQs?* Novel Language models proved to be a very effective tool to compute semantic similarities between MCQs, thanks to their strong capabilities in understanding the contextual meaning of text, including questions. In contrast, traditional methods such as *CountVectorizer* fail to identify many of the similar question pairs, thus being less valuable in tasks like identifying duplicate questions, question clustering, or evaluating the overall diversity of question sets.

R2 *What are the most effective preprocessing strategies for MCQs prior to similarity computation: removing the list of candidate answers or augmenting each question with the correct answer?* When it comes to preprocessing MCQs for

similarity computation, both removing the list of candidate answers and augmenting each question with the correct answer have their advantages and considerations. Removing the list of candidate answers can help focus on the core content of the question, reducing potential noise introduced by the options. On the other hand, augmenting each question with the correct answer can enhance the semantic representation of the question, providing additional context.

R3 *Can graph exploration techniques, specifically graph communities, assist in identifying potential issues and improving the quality of the repository?* Graph exploration techniques, particularly those involving graph communities, can play a valuable role in identifying potential issues and enhancing the quality of a repository. By representing the repository as a graph, where questions and their similarity are nodes and edges respectively, communities within the graph can reveal clusters of related or similar questions, aiding in tasks such as identifying redundant or overlapping content.

4.6.2 Comparison with Similar Works

In this section, we discuss other works that share the objective of identifying duplicate questions and draw comparison with them. The task of finding duplicate questions has been particularly investigated in the Q&A field with the aim of answering queries by finding other similar questions in Q&A forums that have been already answered. One of such works is [134], which presents a deep-learning based system designed to recognize similar questions in the context of unsolved medical queries. The primary focus of this work is to assist users of medical Q&A platforms in finding answers to health-related questions by identifying duplicate queries that have already been answered. The approach involves training a Long-Short Term Memory (LSTM) [92] neural network on pairs of questions, mapping semantically analogous queries closer together within the LSTM network's vector space. The system then employs this trained network to embed unanswered queries into vectors, subsequently comparing these vectors with those generated by the LSTM network for previously answered questions on the Q&A platform. The authors in [107] also proposes a analogous concept to [134] for Q&A platforms focused on game development. The authors propose the use of a combination of large pretrained deep learning models and unsupervised techniques to identify duplicate questions. Similarly to [134], the primary goal is to help game developers in finding answers to their queries by identifying similar previously posted questions that have been answered. More specifically, the proposed methodology involves using outputs from various unsupervised and semi-supervised methods, including large language models such as MPNet [210], as features to train a supervised model for predicting the similarity score between sentence pairs.

Both [134] and [107] differ from our work in that they use a supervised approach to learn a model that predicts whether a pair of questions is duplicated or not. In contrast, since we do not have access to the ground-truth labels that denote which MCQs are duplicated, we opted for an unsupervised approach by leveraging existing pretrained large language models to discover duplicate MCQs.

On the other hand, to the best of our knowledge, only few works have investigated the task of finding duplicate questions in the learning analytics field. In this context,

we mention [163], which presents a machine learning-based system for managing extensive question paper databases. This system identifies pairs of English sentences exhibiting high semantic similarity by training an XGBoost model [38] with diverse manually chosen features, including structural attributes (e.g., token count) and word embeddings. The training data consists of labeled duplicate question pairs from Quora. Although sharing similar goals, this work does not incorporate further processing of the model's output to enable accessible navigation of the similarity model's results. In contrast, our approach integrates a graph construction phase intended to simplify duplicate identification and reasoning over model outcomes.

Finally, we mention [153], which introduces an efficient system for detecting and managing duplicate questions within a Bengali-language MCQ database of a Problem-Based e-Learning (PBeL) system. In contrast to the previously mentioned works, this study does not employ deep learning techniques. Instead, the authors represent each MCQ with a vector via the TF-IDF algorithm, identifying similar MCQs using cosine similarity. We observed in our work that TF-IDF has not proved effective in finding duplicated MCQs.

4.6.3 Current Limitations

We now discuss limitations of this work and future research directions. Firstly, we acknowledge that selecting the right NLP model may not be straightforward as it depends on various factors such as the language of the dataset and its domain. For instance, the Sentence Transformer models mentioned earlier may not perform optimally for low resource languages, specific domains like the legal domain or dataset that have specific characteristics that differs from the Competenze Digitali MCQ dataset we used in our example. In such cases, developing custom Sentence Transformer models tailored to the specific language or domain may be necessary to effectively identify pairs of similar questions. Additionally, when multiple models are suitable for the task, defining criteria to select the best-performing model becomes crucial. This may involve manually selecting questions that are known to be duplicates or very similar to assess and compare the performance of different models. Exploring the appropriate model selection process and criteria among different models is an avenue for future research.

Another current limitation of our work is the lack of integration of human annotators within the pipeline. Currently, our approach focuses on identifying pairs of sentences that are likely duplicates or highly similar, with the final decision left to the human expert. However, a more end-to-end strategy could involve integrating human feedback from the initial stages, leveraging both human judgment and the NLP approach to validate and reinforce each other. This approach would involve actively incorporating human annotators into the process, utilizing their expertise to assess and validate the results generated by the NLP models. Investigating the integration of human feedback and collaboration within the pipeline is a promising area for future research.

4.6.4 Envisioned Workflow for MCQ Management

To fully leverage the proposed approach, we envision the development of a user-friendly interface that will serve as a central platform for managing MCQ repositories. Such interface will offer various features and functionalities to support educators in their task of finding duplicate questions. One of the key capabilities of the interface will be the ability to load an MCQ dataset, allowing users to easily import their existing question banks. Once the dataset is loaded, educators can browse through the questions, gaining a comprehensive understanding of the available content. To assist users in assessing the similarity between questions, the user interface (UI) will offer a range of similarity measures, such as those introduced in Section 4.4. Educators can select different measures, such as lexical or semantic similarity, and customize the corresponding thresholds based on their preferences and requirements. Upon selecting the desired similarity measures and thresholds, educators can apply them to a specific set of questions that they have chosen through the user interface. This enables them to evaluate the effectiveness of different methods and thresholds on a subset of questions, gaining insights into the best approach for their needs. Finally, the envisioned UI will allow to find relationships between questions by generating and exploring the graphs of communities within the MCQ repository using our proposed approach. Educators can select different community detection algorithms and interactively explore the graph, examining the connections between questions. This interactive exploration will allow users to manually review potential duplicates and determine their validity.

It is essential to highlight that our approach places the final decision-making authority in the hands of the human user. While the system provides recommendations based on the applied models, thresholds, and community detection algorithms, educators have the ultimate discretion to choose the most suitable model, threshold, and community detection algorithm for their specific context.

In this work, our primary focus was on developing an effective pipeline for identifying duplicate questions within MCQ repositories. Although we recognize the importance of an appropriate user interface for facilitating this task, we acknowledge that its design was beyond the scope of this work. Therefore, the development of a user-friendly interface for managing MCQ repositories is left as an avenue for future research.

4.6.5 Ethical Issues

Our study does not involve human subjects or sensitive data, neither from the participants to the *Competenze Digitali* program nor from the experts compiling the question database. The main ethical aspect that we considered is therefore the confidentiality of the question database itself. However, we can share the similarity scores used in our experiments and, upon request, we can also share the main codebase that implements the described workflow and the demonstrative questions that where used for the examples throughout this chapter.

When implementing our workflow in authentic classrooms, the same confidentiality aspect could exist. Since our workflow is meant to be used mainly by educators, it is reasonable to assume that human users of our system would have permission to

access the question databases to review the provided recommendations.

Future versions of our workflow could involve data from the experts compiling the question database to mitigate the risk of discriminating certain users' groups when performing the duplicate detection task. Identifying and mitigating potential biases towards protected categories when performing analogous data cleaning tasks is nowadays an active area in data management and further analysis in this direction is left as an interesting future work. We refer the interested reader to recent works in the related Entity Resolution area [64].

4.7 Concluding Remarks

In this chapter, based on our experience in a real large scale assessment project by Italian Government, we proposed a novel approach for the maintanance of a large repository of multiple-choice questions.

Firstly, a new method for identifying pairs of similar questions in a Multiple-Choice Questions (MCQs) repository has been proposed. The approach uses deep learning-based natural language processing (NLP) techniques based on recent transformer architectures to find similarities based on the semantics of the questions, instead of relying on the co-occurrence of words. This method is expected to identify more similar pairs but can also produce false positives.

Secondly, a new graph exploration approach based on graph communities has been proposed to handle the issue of false positives. This method can potentially reduce the number of false positives by only considering the similarity between questions within the same community. By doing so, the method can better capture the underlying structure of the MCQs repository and identify clusters of related questions.

Lastly, new application settings have been introduced for identifying similar records. These settings include options to strip candidate answer lists from each question or to enrich each question with the correct answer only. The former option can be useful in scenarios where the list of candidate answers may contain misleading or irrelevant information, while the latter can provide additional context and meaning for each question.

Our prototype tool, based on NLP and graph visualization, has been effective for different tasks, including finding questions that are too similar and checking the coherence between a question and the area of the syllabus it belongs to.

Furthermore, this tool could be employed in other tasks, such as the refactoring of a syllabus, i.e., when the syllabus changes and thus the questions need to be rearranged, and the merging of two repositories, by matching the imported questions against the new Syllabus they are moved to.

Part II

Build High Quality Data

Chapter 5

A Bayesian Approach for Crowdsourcing the Truths from LLMs

In this chapter we shift our focus to uncertainty estimation, which we study in the context of unstructured data, specifically factual open-domain QA datasets. We summarize the concepts covered in this chapter in Figure 5.1.

Figure 5.1. Tasks, domains and data types covered in this chapter

Concerns persist over the trustworthiness of LLMs due to the generation of plausible but incorrect information, known as hallucination. Existing approaches focus on identifying false answers or improving correctness by sampling responses from a single LLM. However, querying multiple LLMs, which exhibit complementary strengths, remains largely unexplored. In this chapter, we propose a Bayesian crowdsourcing approach to aggregate multiple answers from multiple LLMs and quantify their uncertainty. Extending the Dawid-Skene model, we treat LLMs as annotators, using their answer probabilities as noisy observations of truthfulness and modeling semantic relations between answers in the covariance structure, and jointly learn about LLM's reliability and calibration as parameters. Validated across three open-domain question answering dataset, results show that our approach outperforms existing statistical or agentic methods in abstaining from false answers and identifying truthful ones, offering a robust, scalable solution for uncertainty quantification and truth discovery in LLM outputs.

5.1 Introduction

LLMs [170, 57, 219, 103] are pre-trained on web-scale language data that make them excel at generating human-like responses and storing extensive world knowledge

[183]. They have demonstrated remarkable capabilities across a wide range of tasks that require conceptual knowledge, from recalling answers to trivia questions [148] to performing complex multi-hop reasoning [119]. However, their wide applications have raised concerns over the trustworthiness and reliability of their outputs [141, 97], exemplified by the generation of plausible but incorrect information (i.e. "hallucination" [257]) and the lack of self-awareness of limitations in knowledge [106], which both remain largely unaddressed.

To improve the reliability of LLM answers at test time, strategies that work on one of the two complementary fronts have been proposed: improving the correctness of answers [147, 233, 97, 56], or identifying answers that are more likely to be false to abstain from them [86, 125, 138, 70]. The underlying ideas behind strategies from both fronts are similar: they rely on sampling multiple answers from a *single* LLM and aggregating them based on consistency [233, 125, 138], or following an agentic workflow where LLMs verbally evaluate or improve an answer, similarly to human interactions [147, 56, 70].

We study a more **general scenario**, where *multiple* LLMs are each queried *multiple* times to generate a set of candidate answers for *multiple* questions, that to our knowledge has not been extensively and systematically studied. Querying multiple LLMs, rather than a single one, could be helpful because they are known to have different strengths and weaknesses [105, 230], and their outputs can be complementary [228]. The goals are, at the same time, (1) to quantify the uncertainty (**UQ**) of these answers, as a base for abstaining; and (2) to aggregate these answers to infer the truthful answer for each question, a problem known as truth discovery (**TD**) in the data management literature [133, 260].

This task can be seen as a special case of inferring the ground truths from multiple annotators, where the annotators are all LLMs. The base problem, known as crowdsourcing [260], is well-established with Bayesian models that go beyond simple consistency-based aggregation [48, 238], leading to applications such as in NLP tasks [176, 178, 209]. Despite precursory works [52] on crowdsourcing with weak systems as annotators, not much has been done to extend these models to LLM-based annotators. Crowdsourcing models are mainly limited to classification tasks with predefined classes, while the typical use cases of LLMs require free-form text answers. Moreover, classical crowdsourcing models expect a single answer from each annotator, while LLMs can generate multiple answers for a single question with different probabilities, which could additionally inform crowdsourcing models.

To address these challenges, we propose a probabilistic generative model that extends the Dawid-Skene model [48] to the LLM-based setting. For each question, the *truthfulness* of candidate answers from multiple LLMs is modeled as a multivariate latent variable whose correlation structure is tied to the semantic relations between the answers, and whose marginal distribution determined by the *reliability* of the LLMs. The observed data are the probabilities of the candidate answers being generated by the LLMs, which are treated as noisy observations of the truthfulness and are modeled as dependent on the latent truthfulness variable via a *calibration* process. Parameters for reliability and calibration are learned by the EM algorithm, and the truthfulness (inverse of uncertainty) of the candidate answers is inferred by the posterior of the latent variable. We validate our model on three open-domain question answering datasets, where we show that our model can even outperform

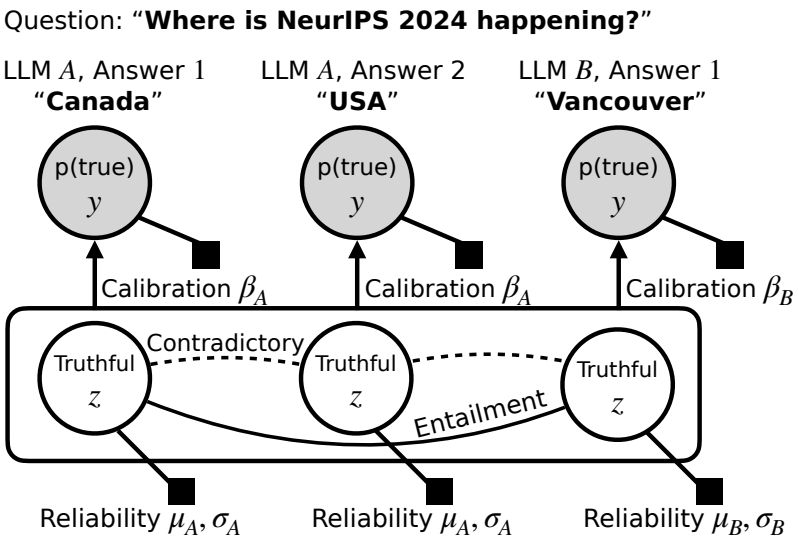


Figure 5.2. We propose a Bayesian model for crowdsourcing from free-text outputs of multiple LLMs.

costly agentic methods in effectively identifying the truthful answers and abstaining from the false ones.

5.2 Related Work

Crowdsourcing. Combining multiple answers by simple majority voting is commonly used to improve LLMs at test time [233, 131]. Crowdsourcing models such as Dawid-Skene [48] opt for iterative weighted majority voting by assuming different reliability of workers. They are comprehensively reviewed by [260, 178, 176] and implemented by [224]. Typically, these models require a known and fixed set of possible answers, while recently studies [130] operate with free-text by a weighted majority voting in the embedding space. Our approach eliminates the need for embedding and provides more flexibility and interpretability. Weak supervision [196], where the truths are not fully available and truth inference is integrated with model learning, is closely related. We only consider the case when the ground truth is completely unavailable, and do not consider learning. Second-order information such as worker's predictions [121] is useful but beyond the scope of this work.

Uncertainty quantification. Many UQ methods work by quantifying how consistent the answers are, with [125, 138] providing methods for a semantic-aware consistency measure. Picking the most consistent answer is also the underlying strategy to improve the factuality of LLMs [233, 131]. [94] infers about the uncertainty of a statement using a Hidden Markov Model by considering the logical relations with other statements conditionally generated from the initial statement. Our model offers more relaxations and consider arbitrary answers sampled independently from multiple LLMs. An separate step in UQ is to calibrate the uncertainty estimates to match accuracy using labeled dataset [90], while our model allows the learning of the calibration and reliability parameters without labelling.

5.3 Methodology

Suppose we have a set of N questions and J LLMs. For each question $q^{(i)}$, we sample K answers from each LLM, resulting in $J \times K$ answers $\{a_{11}, \ldots, a_{JK}\}$, and record their probabilities of being generated $\boldsymbol{y}^{(i)} = \{y_{11}, \ldots, y_{JK}\}$, which are provided as `logprob` or can be estimated by sampling black-box models, and are known as p(True) [106]. For simplicity, we omit the superscript (i) as the parameters are shared across questions.

We introduce a continuous latent variable $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ of dimension $(J \times K)$ to model the real truthfulness of the answers. The marginal distribution of z_{jk} has shape $z_{jk} \sim \mathcal{N}(\mu_j, \sigma_j)$ and is determined by the reliability of the LLM j. The covariance $\boldsymbol{\Sigma}$ is determined by the semantic relations between answers. Following [94], we posit that semantically similar answers would have correlated truthfulness, while contradictory answers would have negating truthfulness. As such, $\boldsymbol{z}$'s correlation matrix $\boldsymbol{R}$ is determined by an NLI model [193] that classifies a pair of answers:

$$E_{jk,j'k'} = \text{Entails}(a_{jk}, a_{j'k'}) \quad C_{jk,j'k'} = \text{Contradicts}(a_{jk}, a_{j'k'})$$
$$R_{jk,j'k'} = (E_{jk,j'k'} + E_{j'k',jk})/2 - (C_{jk,j'k'} + C_{j'k',jk})/2$$

which ensures a correlation of 1 for equivalent answers and -1 for mutually exclusive ones. The covariance matrix $\boldsymbol{\Sigma}$ is then computed as $\boldsymbol{\Sigma} = \boldsymbol{R} \odot \boldsymbol{\sigma}\boldsymbol{\sigma}^T$ and is projected to the nearest positive semi-definite matrix in the Frobenius norm [41].

The data $\boldsymbol{y}$ is assumed to the noisy and uncalibrated observation of the truthfulness $\boldsymbol{z}$, which is calibrated by logistic regression $y_{jk} \sim \text{Bernoulli}(\text{sigmoid}(\beta_{0j} + \beta_{1j} \cdot z_{jk}))$ such that $\beta_1 \geq 0$. The parameters $\boldsymbol{\mu}$, $\boldsymbol{\sigma}$, and $\boldsymbol{\beta}$ are unique for each LLM and are shared across questions. An example of the full generative model is shown in 5.2.

For the ease of implementation, we use EM algorithm to learn the parameters by maximizing log-likelihood using stochastic gradient descent and sampling from $p(\boldsymbol{z}|\boldsymbol{y})$ using NUTS [93, 184, 22]. As with most baseline methods to be compared in 5.4, we select the best answer with the lowest uncertainty as the final answer for TD. For our model, the uncertainty of answer a_{jk} is quantified by the lower bound of 99.9% confidence interval of the mean of the posterior distribution $p(z_{jk}|\boldsymbol{y})$.

5.4 Experiments and Results

Baselines. We perform a comprehensive comparison of our Bayesian approach against recent *statistical* and *agentic* methods for UQ and TD related tasks. In addition, we include *oracle* baselines that always choose answers from a single LLM or choose a correct answers if one exists.

Statistical methods are based on the frequency or likelihood of candidate answers and direct consistency measures, with the most commonplace method being **simple majority voting**. [106] assumes the uncertainty of an answer to be the complement of the probability of it being generated, denoted as **p(True)**. **Semantic entropy** [125] clusters candidate answers based on semantic similarity and assigns uncertainty based on the entropy of the clusters. **Lin et al. (2024)** [138] provides a graph Laplacian-based remedy to semantic entropy when p(True) is not available. The **random** baseline assigns uniform uncertainty to all candidate answers.

AbstainQA [70] provides two agentic methods for UQ. The **cooperate** method asks LLMs to provide feedback on candidate answers, which is then summarized by a judge for a true/false judgment. The **compete** method asks LLMs to draft paragraphs supporting alternative answers and reconsider the question, and the uncertainty is estimated by the consistency of final answers. **Debate** [56] is a multi-round debating framework where LLMs refine their answers based on explanations from other LLMs. An LLM judge selects the final answer based on the responses from the last round, which does not provide UQ and is only applicable to TD.

Tasks and evaluation metrics. Experiments are done on three open-domain question answering datasets pertaining to different levels of aleatoric and epistemic uncertainty [98]: FreebaseQA [105], AmbigQA [156], and IMDB-Torso [214], with FreebaseQA downsampled to match the size of the other datasets of around 1k. Using 10 different seeds, we sample a total of 40 candidate answers per question from four LLMs: Gemma-2-9B [219], GPT-3.5 [170], Llama-3-8B [57], and Mistral-7B [103].

Given a question and its candidate answers, the goal of UQ is to estimate the uncertainty of the candidate answers and it is evaluated by UQ using the area under the receiver operating characteristic curve (AUROC) that measures how well the uncertainty of the correct answer is separated from the uncertainty of the incorrect answers (Table 5.1). The goal of TD is to select a correct answer (if any) from the candidate answers and it is evaluated by the final question-answering accuracy (Table 5.2).

		FreebaseQA	AmbigQA	IMDB-Torso
Agentic	AbstainQA [70] (cooperate)	0.58	0.57	0.53
	AbstainQA [70] (compete)	0.79	0.71	0.70
Statistical	Random	0.50	0.50	0.50
	p(True) [106]	0.55	0.53	0.57
	Simple majority voting	0.89	0.68	0.83
	Semantic entropy [125]	0.83	0.65	0.82
	Lin et al. (2024) [138]	0.52	0.51	0.53
	Bayesian network (ours)	0.88	0.74	0.85

Table 5.1. Bayesian network outperforms baseline methods on all datasets in terms of AUROC for uncertainty quantification. Top non-oracle results are highlighted in green in this Table and Table 5.2.

5.5 Discussion

Our Bayesian model consistently performs the best among all baseline methods when ranking candidate answers to align with truthfulness, making it a promising approach for uncertainty quantification. For improving question answering (TD), the Bayesian model outperforms other statistical models on harder datasets. Despite the relative success of multi-agent models in TD, it is noteworthy that the fast

		FreebaseQA	AmbigQA	IMDB-Torso
Oracle	Gemma-2-9B	83.9	57.3	39.4
	GPT-3.5	92.2	69.7	55.1
	Llama-3-8B	79.4	52.5	33.7
	Mistral-7B	84.1	69.4	54.7
	Best answer	98.9	97.9	92.0
Agentic	AbstainQA [70] (cooperate)	89.5	71.8	50.4
	AbstainQA [70] (compete)	88.5	66.1	48.1
	Debate [56]	88.2	77.3	57.4
Statistical	Random	83.4	61.1	42.7
	p(True) [106]	79.2	65.1	45.3
	Simple majority voting	90.7	68.9	55.3
	Semantic entropy [125]	92.2	68.5	53.2
	Lin et al. (2024) [138]	83.0	63.1	43.4
	Bayesian network (ours)	88.2	69.6	56.2

Table 5.2. Bayesian network has high accuracy when picking a single correct answer for AmbigQA and IMDB-Torso questions, compared to other statistical methods.

growth of context length and computation constrains scaling with the number of agents, which is crucial to performance [131]. We suspect that verbal reasoning and world knowledge are important when dealing with high aleatoric uncertainty as demonstrated in AmbigQA, where LLM excels but statistical method suffers. The relative high gap between the best performing model and the "best answer" oracle suggests that relying solely on "wisdom of the crowd" is not always reliable, and the best answer is not always the most popular one [121].

Although we are exploring methods to rank answers without ground truths, we observe from the oracles that unsurprisingly, having labelled data for benchmarking and calibration might be more beneficial in practical applications. For example, evidence from the oracles and our calibration model ($\beta_1 = 1.3$ for GPT-3.5 and $0.5 - 0.7$ for other LLMs) both show that GPT-3.5 is more reliable and confident than other LLMs, an important clue that current methods have not fully leveraged.

In the typical crowdsourcing setting, annotators work on instances of the same task, while in question answering the questions can be heterogeneous and diverse, which our current model fail to consider. The Bayesian framework allows for the incorporation of factors such as question domain or difficulty, which could be added in the future to improve our model's performance.

Chapter 6

Composing Smart Data Services in Shop Floors through Large Language Models

In this chapter we focus on the task of data-on-demand, which we study in the context of sensor data, which are used to dynamically generate structured data on request. We summarize the concepts covered in this chapter in Figure 6.1.

Figure 6.1. Tasks, domains and data types covered in this chapter

Recent years have witnessed an ever-growing use of Large Language Models (LLMs) to lower the technical barrier for several tasks, ranging from coding to querying relational databases to composing services. In this work, we focus on using LLMs to simplify access to data in the industrial scenario, by allowing humans operating on the shop floor to submit a query in natural language and then materializing a table integrating data gathered from different data sources including machines and information systems. In particular, we introduce in this chapter *COSMADS*, a system that takes as input a query from an operator on the shop floor and automatically synthesizes a pipeline that leverages existing data sources accessible as services (data services), to compose a table output fulfilling the user's information need. The proposed solution is evaluated using a real case study, showing that results obtained by taking into account available data service descriptions and previous pipelines outperform those obtained by naively employing a state-of-the-art code generation tool.

6.1 Introduction

Manufacturing companies have access to massive amounts of disparate data sources spread across various departments, from the shop floor to the warehouse and the

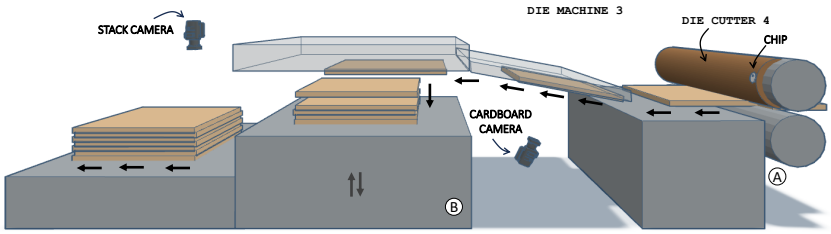


Figure 6.2. The cardboard manufacturing case study

administrative offices. These include traditional information systems (e.g., Enterprise Resource Planning systems - ERPs) and sensors provided by machines and physical spaces. These data can be accessed through various mechanisms, including direct access (e.g., a connection to a DBMS) and Application Programming Interfaces (APIs) provided, for example, in the form of Web services. We refer to these mechanisms as *smart data services* (or simply *data services* – DSs).

Single data services can be combined to extract information, benefiting from the integration of multiple data sources. We use the term *information extraction pipeline* (or simply *pipeline*) to denote any computing procedure that *compose* new information from available data services. Without loss of generality, we can consider this composition as representable by means of relational tables.

Despite this huge potential, the kind of information needed to make decisions on the shop floor, where value is created, changes quickly, leaving operators with limited access to timely data. Continuously developing novel pipelines to address these unpredictable data needs can incur significant costs, especially for non-core IT companies, such as manufacturing ones, that usually rely on an external workforce for data management. Data governance and management platforms can mitigate this problem only partially: maintaining large back-end data lakes they require is often unfeasible due to a lack of know-how, human, and computational resources. This is especially true for Small and Medium Enterprises (SMEs), which cannot sustain the costs of such solutions [251].

On the other hand, chat-enabled digital assistants are emerging as a novel form of interaction between humans and machines, garnering increasing interest in the industrial sector [45]. By enabling shop floor operators to interact using natural language and offering fast, intuitive, and potentially hands-free access to data, these assistants have the potential to streamline, expedite, and improve product-related activities. Large Language Models (LLMs), in particular, have recently demonstrated remarkable capabilities to solve complex questions, especially through the use of external knowledge and tools (e.g., search engines, DBMS, etc.) to ground their responses [95].

Example case study. Consider the case of a cardboard box factory, as shown in Figure 6.2. The core production process consists of two main machines: Ⓐ a *die cutting machine* that cuts the cardboard by means of interchangeable *die cutters* and Ⓑ a *stacker*, which groups the cut-out cardboard into bunches. Each die cutter features a chip that tracks its speed in rotations per second. A high-resolution camera records the stream of cardboards and detect defective cardboards. A second camera

is placed after the stacker and records when a bunch is ready for the subsequent processing steps. Several data services are available in this scenario, such as:

- `DS1`, returning the speed of the currently installed die cutter;
- `DS2`, returning the id of the camera streaming the production of a die cutter;
- `DS3`, capturing a cardboard frame from the camera with a given id;
- `DS4`, returning the defects encountered in a cardboard;
- `DS5`, detecting the completion of a new bunch.

Suppose the operator proposes the following natural language query:

Q: "Consider the current session of the die cutter with id 4. Generate a table containing *(i)* the number of cardboards with no defects and *(ii)* those with errors."

We can imagine a system that builds, leveraging LLMs capabilities, a pipeline returning a relational table that satisfies the query **Q** by composing available data services and treating the data sources (e.g., cameras, sensors, tools) as external knowledge that can be queried and processed. To answer query **Q**: *(i)* `DS2` is invoked to retrieve the id (e.g., 54239) of the camera streaming the production of die cutter 4, *(ii)* `DS3` captures a frame from camera 54239 of cardboard produced, and *(iii)* `DS4` detects possible defects in the captured cardboard frame.

Contribution. In this paper, we present *COSMADS* – COmposing SMArt Data Services, a tool that synthesizes information extraction pipelines, in the form of Python scripts, starting from natural language queries and a documented *codebase* consisting of available data services and possibly other, previously defined, information extraction pipelines. Such pipelines, in particular, can be either manually defined or formerly generated with *COSMADS*.

Outline. This paper is organized as follows. Section 6.2 introduces background and related works. In Section 6.3, we discuss the main components of *COSMADS*. In Section 6.4, we provide technical details. Experimental results are presented in Section 6.5. Finally, an outline of future challenges together with concluding remarks are presented in Section 6.6

6.2 Background and Related Works

LLMs are computational models capable of operating on and generating sentences in natural language [33]. State-of-the-art large pre-trained language models (PLMs) such as BERT [49], GPT [4, 25] and LLaMA [222] are not domain-specific and can produce hallucinations (wrong and ambiguous outputs) when used in specialized domains like the one presented in this work. However, the development of a domain-specific model requires a proper dataset and significant computing power for LLM training (or fine-tuning, transfer-learning) which may not always be available. In this context, prompt engineering has emerged as an approach to enhance the In Context Learning (ICL) ability of LLMs [25], enabling zero-shot or few-shot learning without additional training data [47, 258]. Users design specific prompt text, sometimes with

few examples (few-shot prompting), to guide an LLM in generating desired responses. In the following, we review LLM literature that is relevant to our scenario.

LLM Agent. An emerging approach to solve complex problems and mitigate hallucinations is the development of LLMs as agents, also referred to as tool-augmented LLMs or LLM Agents, which can generate reasoning plans involving the invocation of external services (tools) [95, 96, 177, 175]. These tools can access external information (e.g., documents, calendars) or affect the virtual or physical world (e.g., robotic arm) [154]. The general technique leverages the ICL ability by including in the prompt *(i)* tool demonstrations, *(ii)* tool documentation, or *(iii)* even a combination of both [95]. Tool demonstrations consist of providing examples (e.g. code snippets) that show how a tool can be used to accomplish certain tasks in combination with other tools. On the other hand, tool documentation contains general usage details about what the tool can be used for and how the LLM can interact with the tool.

In this paper, we leverage the LLM Agent concept to generate the solutions. Notable differences with existing works are that the tools, i.e., the data services, in our scenario are cyber-physical assets, in contrast to works that focus on general-purpose tools (e.g. search engines), jointly used within a production line, thus dependencies between these assets may arise and should be taken into account when generating and running the reasoning plan. Finally, another notable difference is in the generated output, since the above works can generate any kind of textual data, while we focus on the generation of structured (i.e. relational) data.

LLM in data management. LLMs are rapidly transforming data management, with a wide landscape of methods for querying relational databases [89, 5, 110], reasoning over structured data [104, 235, 145] and performing data cleaning operations [71]. These works are orthogonal to our approach, where we envision a scenario where relational data are constructed from scratch rather than queried. The idea of creating tables via LLM was recently introduced in [197], where the authors use LLMs not only to query a relational database but also to potentially augment the selected records by tapping the information in the LLM, which has been in turn extracted from massive corpora of text documents during training. In our case, the LLM is not used as a source of information but as a way to identify and compose data services.

LLM for code generation. Due to the similarity between natural language and source code, LLMs are extensively used in software engineering tasks like code understanding and generation [33, 42, 135]. In this setting, the LLM is prompted to take a natural language description of a problem and generate an executable code in a certain programming language to solve the problem. The surprising performance has also prompted research on using LLM to generate code to express the necessary logic to answer a given question [39].

Recently, a variety of code-generating LLMs have been introduced, including Codex [37], AlphaCode [135], Codegen [166] and CodeT [35]. These models feature billions of parameters and benefit from the availability of vast amounts of code-related datasets that are publicly available. The use of the LLM agent paradigm and self-reflection has further improved the accuracy of these models on code generation [195].

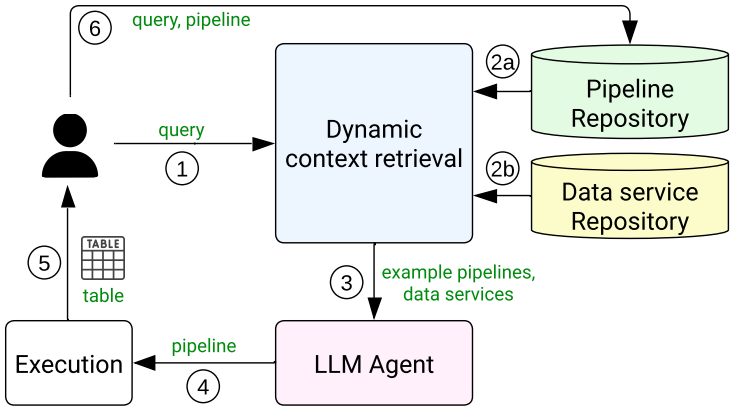


Figure 6.3. *COSMADS* high-level architecture

LLM for service composition. An LLM Agent which leverages external tools can be thought of as a service composition engine. As explained in [18], service composition addresses the situation when a complex task, also called *target service*, cannot be realized by calling a single service, but a composite service, obtained by combining "parts of" available *component services*, might be used. Such a definition perfectly aligns with the definition of LLM Agent which leverages external tools to derive the output. Authors in [159] take advantage of such an ability to develop a solution where an LLM Agent composes external tools performing process activities, resulting in a code representing a business process (a.k.a. orchestration).

In service composition, target and component services are often modeled as state machines, with automated synthesis techniques employed to compute an *exact* solution to the problem. The main limitation of classical service composition methods is that the formal definition of the target and component services is a demanding task that must be done manually. Also, symbolic formalisms employed usually make it hard to model complex behaviors, especially when the data flow is considered. Additionally, the computational complexity of the solving methods is exponential, quickly making solutions hard to compute. As pointed out in [182, 6], LLMs can help address many of these limitations, although domain expert supervision is still necessary due to potential inaccuracies in the output.

6.3 COmposing SMArt Data Services

Figure 6.3 depicts the architectural components of *COSMADS*. A new execution is spawned as soon as a human operator specifies a natural language query ① to retrieve information from the ongoing manufacturing process. A query can be parametric, meaning that it can provide a set of input arguments (e.g., a time range, or a specific kind of defect to be monitored). We can imagine the human operator to have little or zero knowledge about the available data services. As a consequence, we suppose the query to only express the required information without technical details on how to compute it.

The core of the architecture is represented by an *LLM Agent* that instructs a pre-trained LLM by feeding a query-specific prompt, which is built according to the

output of the *dynamic context retrieval* component. This module analyzes the query and retrieves ②a a set of example pipelines from a *pipeline repository* and ②b a set of data services from a *data service repository* that can be used to answer the query.

Data services exposed by manufacturing assets in a factory can range from operational/actuating services (e.g., turning on the camera) to services that generate data (e.g., getting the current speed of the die cutting machine using its embedded chip). Data services can be accessed using different paradigms and communication protocols, but for simplicity, we assume they can be called through function calls wrapping the actual calling mechanism. A data service can expect a set of parameters and returns an output, which can be either structured or unstructured. For each data service, the *data service repository* contains a *documentation*, i.e., a textual description of the functioning and usage of a data service.

The *pipeline repository* consists of all the pipelines already available. These pipelines can be manually defined or obtained from previous executions of *COSMADS*. Each pipeline is associated with the query it fulfills. In general, pipelines can be defined using various modeling formalisms, such as programming languages and scientific workflow scripting languages [168], among others. In *COSMADS*, pipelines are software (Python) scripts that *(i)* produce a table as an output, and *(ii)* make use of data services.

Examples of pipelines and documentation of relevant data services are fed ③ to the *LLM Agent* together with the original query. Such input data is incorporated into a prompt, according to a specific prompt template. The output pipeline generated by the *LLM Agent* is then sent ④ to the execution module (e.g., the Python interpreter).

The execution of the pipeline finally ⑤ produces a table answering the query. If the human operator thinks the produced pipeline can be helpful as a future reference for future queries, the pipeline together with the originating query can be stored ⑥ in the pipeline repository.

6.4 Realizing COSMADS

In this section, we describe the realization details of *COSMADS*, of which a prototype[1] has been implemented in Python leveraging the Langchain framework[2]. The base LLM model utilized for the LLM Agent relies on GPT-4 (`gpt-4-turbo`) by OpenAI[3].

6.4.1 Dynamic Context Retrieval

The contextual information required for the prompt of the LLM agent consists of *(i)* relevant previous queries with corresponing pipelines to be used as *few-shot examples*, and *(ii)* the set of data services needed for solving the input query. As discussed in Section 6.3, these information can be obtained from the *pipeline repository* and the *data service repository* respectively.

[1] For repeatability, both the source code and the experimental results are openly available at the following link: https://anonymous.4open.science/r/COSMADS-0735

[2] Cf. https://www.langchain.com/

[3] The base LLM model for the LLM Agent can be any LLM model available in the literature. Some changes in the prompt may be necessary if another model is selected.

```
{
    "query": "Consider the next 5 carboard of the first diecutter. Generate
        ↪ a table containing: (i) the number of cardboards with no defects,
        ↪ (ii) those with errors, (iii) how many fold errors, and (iv) how
        ↪ many hole errors.",
    "metadata":{
        "pipeline": "pipelines/04_07_2024-21_43_24.py"
    }
}
```

Figure 6.4. Example of stored query and corresponding pipeline

The *pipeline repository* is implemented as a vector store containing the vector representations of queries already solved. Each vector contains the embedding of the `query` and the metadata, which include the path to the Python script containing the `pipeline` that solves that query. Figure 6.4 shows an example of query vector. The figure does not show, for the sake of space, the actual embedding, which is computed by applying the `text-embedding-ada-002` model by OpenAI. For the vector store, we rely on DocArray's DocIndex[4], which is well integrated into Langchain and allows to efficiently access stored data.

Given a natural language query, the *dynamic context retrieval* component computes its embedding and retrieves the top-K[5] similar queries from the vector store. Similarity is computed according to cosine similarity. For example, considering the query $\mathbf{Q}$, Figure 6.4 shows the most similar query.

The set of data services needed for solving the input query is stored in the *data service repository*. A data service is used to retrieve historical or online data. Online data is often associated with the execution of some operation (e.g., taking a picture). As a consequence, any asset of the company exposing services to support its operational functionalities can be considered a data service. In our example case study, an example of data service is the service related to the camera asset which capture frames (i.e., `DS3`). In *COSMADS*, each data service is realized as a Python class having two main components: *(i)* a function wrapping the existing service of the asset and representing the actual execution logic, and *(ii)* a class variable containing the documentation. Our LLM Agent relies on the documentation of the data services, which summarizes their capabilities, how they need to be used, a one-shot example, and a specification of the input and output parameters. Figure 6.5 provides the data service documentation of the camera asset capturing frames. Notably, this documentation corresponds to what in traditional service composition was referred to as service description.

Noteworthy, while the *dynamic context retrieval* select only K example pipelines to be included in the prompt, all data services are considered. This approach is justified by the assumption that the number of services in the entire repository remains relatively stable, as the codebase of a company typically grows slowly and is also relatively small compared to the maximum prompt length allowed by the LLM. Conversely, the number of pipelines is expected to grow more significantly over time, as new pipelines are added to the repository, either through manual definition or

[4] Cf. https://github.com/docarray/docarray

[5] Experiments in Section 6.5 are based on $K = 1$.

```
"brief_description": "Data service that, given the id of a camera1, provides
    ↪ a frame captured from that camera1.",
"detailed_description":
    """Data service that, given the id of a camera1, provides a frame
        ↪ captured from that camera1.
    In general instances of camera1 point downwards to a conveyor belt of a
        ↪ specific production line that transports single cutout cardboards
        ↪ produced by a specific die machine.
    The data service takes a single parameter, namely the id of the camera1
        ↪ (an integer) and returns a frame captured from that camera1 as a
        ↪ numpy matrix.
    The matrix is a 2D array having a shape of (1080, 1920, 3) where 1080 is
        ↪ the height, 1920 is the width and 3 is the number of channels
        ↪ (RGB).
    Example usage:
    - If the id of the camera1 is 123, then the data service would be called
        ↪ as follows:
    camera1_id = 123
    frame = GetFrameFromCamera1.call(camera1_id=123)
    # assuming the frame is a numpy matrix
    print(frame.shape)  # (1080, 1920, 3)
    Things to keep in mind:
    - The refresh rate of the camera is 1 second, i.e. the frame is updated
        ↪ every second, so if the data service is called multiple times
        ↪ within a second, it will return the same value.
    - The frame is a numpy matrix, so avoid trying to access it as a
        ↪ dictionary.""",
"input_parameters": ["camera1_id:int"],
"output_values": ["frame:np.matrix"],
"module": "camera1"
```

Figure 6.5. Example of data service contextual information

automatic generation by *COSMADS*. Also, this excludes the possibility an incorrect pipeline is generated, simply because information about needed data services is not available.

6.4.2 LLM Agent

Our LLM Agent leverages the ICL ability of LLMs [25]. The quality of the output though is strongly dependent on the quality of the provided prompt [237]. For *COSMADS*, in particular, we designed a *prompt template* to be filled with the output of the *dynamic context retrieval* module, which follows the most common best practices [259]. These include *(i)* expressing the goal task clearly, *(ii)* including contextual information, *(iii)* providing demonstrations, and *(iv)* utilizing model-friendly format style.

The prompt template is described in the following [6]. Blue highlighted sentences represent the invariable parts of the prompt. They include *(i)* a system header describing the skills of the agent, *(ii)* a description of the specific goal to be fulfilled, *(iii)* the description of the structure of the documentations of the data services, and *(iv)* a set of guidelines the output of the agent must respect. Yellow highlighted texts represent the output of the *dynamic context retrieval* component, i.e., the set of data services the pipeline can call, and the example queries. Red highlighted text is instead the input natural language query to be answered.

Prompt template

```
<LLM AGENT EXPERTISE>
Query: {query}
<GOAL DESCRIPTION>
{data_services}
<DATA SERVICES DOCUMENTATIONS STRUCTURE>
<GUIDELINES>
Here examples of pipelines that may help you in generating a new
pipeline:
Query: {example_query}
Pipeline: {example_pipeline}
...other examples...
Answer:
```

6.5 Experimental Validation

In order to evaluate our approach, we used a data service repository adapted from a proprietary codebase deployed in a real cardboard manufacturing domain, as part of a smart manufacturing pilot project. The data services have been simplified for our experiments, by simulating the interaction with the physical assets instead of really interacting with actual industrial devices. In particular, the repository consisted of 12 data services.

[6] The full prompt is available at https://anonymous.4open.science/r/COSMADS-0735/src/pipeline_chain.py.

Query	Col	Rows	DS	Text
q0	3	6	3	Give me the serial number of camera 1 and camera 2 of all the diecutters in the factory.
q1	2	1	3	Consider the next 30 cardboard of the diecutter with id 7. Generate a table containing (i) the number of cardboards with no defects and (ii) those with errors.
q2	3	1	2	Generate a table containing the max speed of the diecutter with id 25 over a time span of 30 seconds.
q3	4	10	2	Generate a list of average speed values for the current session of the diecutter 7, where each average value is computed over 10 consecutive, non-overlapping time windows of 10 seconds.
q4	6	10	6	Compute a table that gives some information about the current session of diecutter 14. For each stack produced, the table should tell me (i) if the stack contains errors, (ii) if it contains errors, the type of error, (iii) the current speed of the die cutter, and (iii) the current temperature of the die cutter. The table should contain data for the next 10 stacks.

Table 6.1. Queries in our experiments

Moreover, we manually defined 5 queries with corresponding solved pipelines to be used as examples in the LLM Agent prompt and stored them in the *pipeline repository*.

We measured performance results on 5 manually defined queries. The queries were suggested by operators working in the manufacturing factories where the proprietary codebase is deployed. For each query, we produced a total of 10 variants by rephrasing the original query text using an LLM[7]. Table 6.1 reports *(i)* the number of columns (*Col*) and *(ii)* rows (*Rows*) in the original table, *(iii)* the number of data services (*DS*) required to generate the table, and, finally, *(iv)* the query text. The queries are defined by increasing their complexity by means of how DSs need to be composed to compute the output. Besides, the queries are defined such that at least one similar query and corresponding solving pipeline are stored in the *pipeline repository*.

Evaluation details. We evaluated the quality of the tables generated by *COSMADS* by comparing them with ground truth data generated from manually-designed Python scripts. The evaluation has been performed at two levels, i.e., intentional (at the schema level) and extensional (at the objects/tuples level).

At the intentional level, we assessed the generated table by comparing its schema to the ground truth schema, i.e., the correct relational definition that the *COSMADS* should ideally provide to the user. This has been done by employing Valentine tool[8] [123] for schema matching based on COMA instance-based method [51]. *Pre-*

[7]The LLM prompt for rephrasing the queries is available at https://anonymous.4open.science/r/COSMADS-0735/src/evaluation/rephrase.py

[8]Cf. https://github.com/delftdata/valentine

Query	Precision	Recall	Accuracy (cell)	Accuracy (row)
q0	1.0	1.0	1.0	1.0
q1	1.0	1.0	1.0	1.0
q2	1.0	0.66	1.0	1.0
q3	1.0	0.67	0.83	0.80
q4	0.9	0.81	0.76	0.55

Table 6.2. Evaluation results of *COSMADS* (w/ similar query-pipeline example)

cision and *recall* metrics are computed for the matching results. A high precision value indicates a low rate of false positives, meaning the generated table contains the correct columns among those it generated. A high recall value indicates a low rate of false negatives, meaning the generated table contains most of the correct columns that should have been included. Note that in general, the columns of the generated schema and the ground truth schema may have different wordings due to different rephrasing of the queries. Valentine COMA instance-based considers such aspects.

At the extensional level, we computed the *instance accuracy* by assessing if the resulting match between the ground truth and generated tables contains the same data. We computed the instance accuracy by measuring if the tables have both the same cells (*Accuracy (cell)*) and the same rows (*Accuracy (row)*). Noteworthy, the accuracy value is computed only on the matched columns. This means that, in principle, high values of accuracy can be achieved with low values of precision and recall if only a subset of the columns are matched but the quality of them is high.

Evaluation results. We report the results of the evaluation in Table 6.2. Results are averaged over the set of variants for each query.

The results indicate a decreasing trend in the measured metrics as queries become more complex. While the precision of the schema matching remains relatively stable despite query complexity, recall significantly drops for *q2* and *q3*. By manually inspecting the results, we found that missing columns in the generated tables often relate to the timestamps of the time windows. In those cases, accuracy is high for *q2* but not for *q3*, where the main issue lies in incorrect computation of aggregated results within time windows. Metrics of *q4* result to be the worst. In this case, indeed, the composition of the data services is more complex, and *COSMADS* does not always produce accurate results. Manually inspecting the results, in some cases, generated tables contain correct data but are badly structured, e.g., usage of arrays or textual representation instead of numerical data.

Ablation study. To confirm the relevance of the contextual information generated by the *dynamic context retrieval* for the quality of the results, we performed an ablation study. Specifically, we ran the experiments on the variants of the queries in Table 6.1 considering two versions of *COSMADS*, where *(a)* instead of feeding the LLM Agent with similar query-pipeline examples, we used unrelated query-pipeline examples, and *(b)* where we removed the query-pipeline example from the prompt.

Table 6.3 and Table 6.4 provide the results of the *(a)* and *(b)* versions, respectively. The results show worse performance than those in Table 6.2, highlighting the importance of query-pipeline few-shot examples. Interestingly, version *(a)*, which uses unrelated query-pipeline examples to fill the prompt template, simulating a

Query	Precision	Recall	Accuracy (cell)	Accuracy (row)
q0	1.0	1.0	1.0	1.0
q1	1.0	1.0	1.0	1.0
q2	1.0	0.33	1.0	1.0
q3	1.0	0.42	0.54	0.23
q4	0.4	0.31	0.28	0.0

Table 6.3. *COSMADS* w/o similar pipeline example – *(a)* version

Query	Precision	Recall	Accuracy (cell)	Accuracy (row)
q0	1.0	1.0	1.0	1.0
q1	1.0	1.0	1.0	1.0
q2	1.0	0.4	0.6	0.6
q3	0.9	0.32	0.36	0.23
q4	0.6	0.23	0.23	0.14

Table 6.4. *COSMADS* w/o pipeline example – *(b)* version

possible initial deployment of *COSMADS*, does not perform very poorly.

Baseline: GitHub Copilot. Finally, we compared *COSMADS* to an end-to-end baseline consisting of a code-generating LLM, namely Github Copilot[9], which was prompted to access the documentation of the data services and answer the query by generating a JSON file with the tabular data. Copilot is utilized through the chat mode provided by Visual Studio Code[10]. Given a workspace containing the set of implemented tools, we invoke the `@workspace` agent, which leverages a meta-prompt to determine what information to collect from the workspace to help answering a question.

Table 6.5 reports the results of the experiments. We note that while Copilot can generate tables for simpler queries, it struggles with more complex ones. Surprisingly, in some cases, it outperforms *COSMADS*. This is likely because it is directly trained on generating code, which gives it an advantage over GPT-4. Unfortunately, access to Copilot cannot be fully automatized as web APIs are not for public access, but the obtained results suggest that replacing it to GPT-4 would further improve the performance on *COSMADS*.

[9] Cf. https://github.com/features/copilot/
[10] Cf. https://code.visualstudio.com/docs/copilot/overview.

Query	Precision	Recall	Accuracy (cell)	Accuracy (row)
q0	1.0	1.0	1.0	1.0
q1	0.7	0.55	0.15	0.1
q2	0.9	0.33	0.8	0.8
q3	1.0	0.45	0.65	0.63
q4	0.2	0.15	0.09	0.0

Table 6.5. Copilot

6.6 Concluding Remarks

In this paper, we introduce *COSMADS*, a tool that takes as input a query from an operator on the shop floor and automatically synthesizes a Python script that leverages existing data sources accessible as services to produce a table output fulfilling the user's information need. The proposed solution is evaluated using a real case study, showing that results obtained by taking into account available data service descriptions and previous pipelines outperform those obtained by naively employing a state-of-the-art code generation tool. The employment of the tool is not straightforward though, opening to a set of research challenges.

In first place, the good quality of results strongly depends on the quality of data services documentation. As a consequence, considerable effort must be devoted to defining data services documentation that can be effectively exploited by the LLM Agent. Similar issues happened in the past for traditional service composition, which required semantic service descriptions. Whereas, to this aim, automatic code documentation techniques can be used [113], assessing whether the LLM Agent correctly uses data services and pipelines based on their documentation and demonstrations is an open question. In this scenario, a *profiling* of data services might be needed. Such profiling could involve letting the LLM imagine plausible scenarios where the data services can be used, and letting the LLM fail and learn from its errors, similarly to [229], or using a human-in-the-loop approach where users define exemplary tool usage scenarios and refine the documentation if necessary.

Linked to the previous point, the proposed approach aims at generating an entire pipeline at once, similar to other proposed approaches such as [231]. An alternative is to generate it iteratively using paradigms such as Chain of Thoughts [236]. Both approaches can be further refined using *self-reflection* based strategies [250, 206, 139], that allows the LLM Agent to reason over the generated output and refine it progressively. Generating a preview of the final table can be useful to *(i)* detect potential syntax and runtime errors that can hinder the generation of the table and *(ii)* allow the user to find potential missing or incomplete information in the table without providing direct access to the pipeline.

Also, after generating the pipeline, *COSMADS* immediately executes it. While this is generally safe, if the generated pipeline is partly or fully wrong, these could lead to manufacturing sessions that are not well monitored. To this aim, pipeline simulation strategy could help. A possible approach involves *digital twins*, which represent digital replicas of the industrial assets, and have been used either to improve the product development or for monitoring its underlying physical assets [143].

Part III

Ethical Data Quality

Chapter 7

R-Fairness: Assessing Fairness of Ranking in Subjective Data

In this chapter we focus on the last task covered in this thesis, namely fairness, which we study in the context of subjective data crawled from e-commerce platforms and reviews-rich recommendation systems. We summarize the concepts covered in this chapter in Figure 7.1.

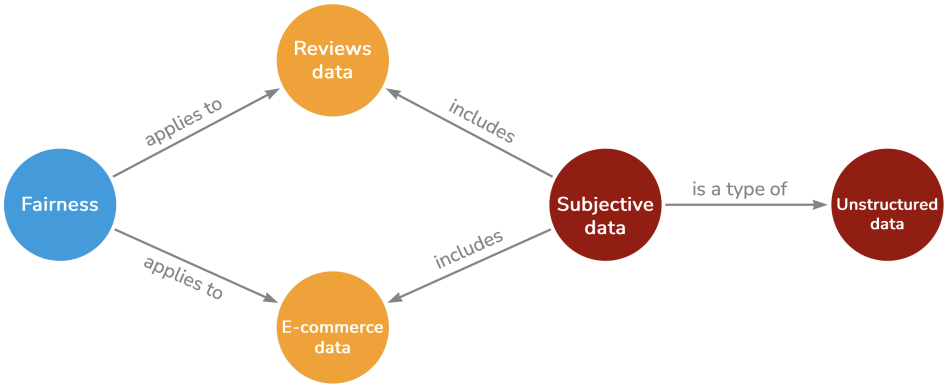


Figure 7.1. Tasks, domains and data types covered in this chapter

Subjective data, reflecting individual opinions, permeates platforms like Yelp and Amazon, influencing everyday decisions. Upon a user query, collaborative rating platforms return a collection of items ranked in an order that is often not transparent to the users. Then, each item is presented with a collection of reviews in an order that typically is, again, rather opaque. Despite the prevalence of such platforms, little attention has been given to fairness in their context, where groups writing best-ranked reviews for best-ranked items have more influence on users' behavior. We design and evaluate a fairness assessment pipeline that starts with a data collection phase to gather reviews from real-world platforms, by submitting artificial user queries and iterating through rated items. Following that, a group assignment phase computes and infers relevant groups for each review, based on review content and user data. Finally, the third step assesses and evaluates the fairness of rankings for different user groups. The key contributions are comparing group exposure

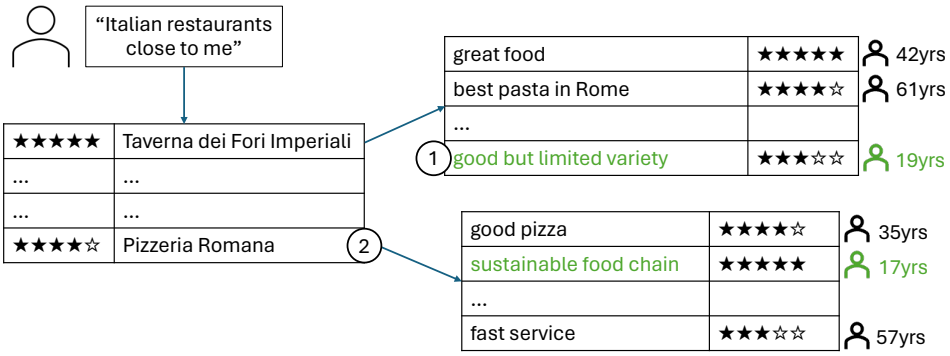


Figure 7.2. Ranking of items and reviews.

for different queries and platforms and comparing how popular fairness definitions behave in different settings. Experiments on real datasets reveal insights into the impact of item ranking on fairness computation and the varying robustness of these measures.

7.1 Introduction

Subjective data represents people and their opinions [216, 136]. Many subjective datasets are available nowadays and are used in decision-making, studying online behavior, or providing recommendations. *Collaborative rating* platforms such as Amazon for products, Yelp for restaurants, and Netflix for movies, are a special case of subjective data and are permeating our everyday decisions. In these platforms, reviews are usually presented in nested ranked lists that intrinsically favor the perspectives of certain groups of reviewers over others.

In this work, we want to assess *r-fairness*, that is, how fairly those systems behave toward reviewers. We consider reviewers' groups based on the traditional notion of demographics such as gender or age, as in the example below, as well as opinions, such as positive reviewers.

Example 5. *In Figure 7.2 we show how the group of reviewers with younger age can suffer under-exposure for a certain query, such as "Italian restaurant close to me". When users submit a query to a collaborative rating system such as Yelp, they get in return a list of ranked items (i.e., restaurants), each of which is associated with ranked reviews. Reviews of certain groups of reviewers are unlikely to be read, either because low ranking is associated with their reviews ①, or to the item that is the object of the review ② or both. As a direct consequence, reviewers who wrote those reviews (in this case: reviewers with younger age) have less chance of impacting other users' choices.*

Fairness of ranking has been trending in research for the last few years as we increasingly rely on algorithms for decision-making (e.g., [117, 223, 253, 19, 31, 246]). Most of these works focus on *group fairness*, stating that individuals in protected groups with comparable utilities should have equal probability of being ranked at the same position. Fairness in ranking has been applied in various scenarios, from

search engines [207] to recommendation systems [234]. Surprisingly, little attention has been paid to the collaborative rating scenarios, despite their ubiquitous presence in our everyday lives.

Contribution. In this paper we develop a reproducible pipeline[1] to assess r-fairness in subjective data, that is the fairness of ranking in collaborative rating systems. We also provide insights from its application on real-world platforms such as Yelp and Amazon. We build on the fairness measures proposed in the literature that formulate fairness constraints on rankings in terms of *exposure* allocation [207, 111]. The main idea of exposure is to assess how a particular ranking balances the fairness of items with the utility that the ranking provides to the users.

Intuitively, restaurants in different areas of a city, such as business areas of residential neighborhoods, may display different group exposures. Analogously, some groups could receive fair treatment on Yelp but not on Amazon, according to a specific fairness measure, due to the different contents and policies of the two platforms.

In this framework, we address the following research issues.

- **RI 1** Compare the group exposure at the *item level*, that is, by considering in isolation the rankings of reviews for each individual item.
- **RI 2** Examine how groups are treated in different platforms under different fairness measures.
- **RI 3** Measure the group exposure at the *query level*, that is, for different queries returning a ranking of items, with each item returning a ranking of reviews.

Methodology. As we do not have direct access to the underlying algorithms and datasets used by the platforms, we developed a pipeline with 3 phases: *data collection, group assignment, and fairness quantification.* Data collection gathers the rankings available for each search query. For instance, for Yelp we collect the ranking for each restaurant search. Group assignment is the process of assigning each review to a group, using automatic tools that guess each group based on the text of the review and the user data. Fairness quantification implements the exposure allocation measures to quantify two types of r-fairness, dubbed *item-level fairness* and *query-level fairness*. For the former, we consider one item at a time. For the latter, we consider different items together, as returned to a user query such as the one in Figure 7.2.

Our key observations can be summarized as follows.

- In real-world data taken from well-known platforms there is high variability of item-level fairness, with some items showing reviews in a fair ranking while others overexposing certain categories of users (**RI1**).
- Different platforms treat groups very differently, with some more relying on utility than others for producing the ranking (**RI2**).

[1] Code and experiments can be found at the following link: https://github.com/jermathew/R-Fairness

- Different queries over the same platform produce rather different exposures for the same groups of individuals, even when the queries return the same set of items (**RI3**).
- Additionally, we observed that the ranking of reviews is highly sensitive to the activity of reviewers and that the most active reviewers are not necessarily diverse in demographics. This calls for the design of novel fairness policies in the platforms that incorporate reviewer activity and give voice to less active groups.

These results highlight: (i) the importance of injecting methods for evaluating and assessing fairness in Collaborative Rating Platforms to guarantee that the influence of all user groups is balanced and transparent, and (ii) the effectiveness of the methodology proposed in this paper to address this problem.

Outline. The paper is organized as follows. Section 7.2 discusses related works in the literature about fairness in ranking. Section 7.3 provides preliminary notions and Section 7.4 introduces the fairness measures used in this paper. Section 7.5 reports our experimental results. Finally, Section 7.6 presents conclusive remarks.

7.2 Related Work

Fairness is a rapidly expanding topic, from the seminal works of Dwork et al. [60] discussing the fundamental notion of *fairness in classification* to more recent works applying and extending such concept in a wide variety of data management tasks [65, 54]. An early discussion of fairness with a data management perspective is available in the works by Stoyanovich et al. [213, 3].

The notion of r-fairness explored in this paper is mostly related to the notion of *fairness in ranking*, which refers to the principle of incorporating fairness requirements into algorithmic rankers. This concept is crucial in applications such as search engines [207] and recommendation systems [234] where biased ranking can lead to unfair outcomes. A popular way of achieving fairness in ranking is the optimization of *exposure*, formulated in terms of position allocation, that is, positions in the ranking that are occupied by a certain group [207]. For an extensive discussion on fairness optimization in ranking, we refer the reader to the two-parts survey [254, 255].

We build on these concepts and address the case of collaborative review platforms, where items (e.g., a restaurant, a product or a place) represent multiple rankings of reviews, and items themselves appear ranked (e.g., best-rated restaurants first) upon a user's query. In this scenario, groups that write high-ranked reviews for high-ranked items have the most influence on online customers' decisions. Note that this is fundamentally different from notions such as fairness with respect to the users browsing the site (that one could call "c-fairness", consumer-fairness) and fairness with respect to items being ranked ("p-fairness", product-fairness). These related but different notions are partly included in the notion of multi-sided fairness [2]. Multi-sided fairness considers the impact of the ranking on all the stakeholder, including the users who receive the recommendations and the individuals being ranked.

$\mathcal{I}$ (Items)

Iid	Name	Rating	City
i1	Ace	4	New York
i2	Kixby	3	New York
i3	Porto	4	Boston

$\mathcal{U}$ (Users)

Uid	Age	Ethnicity
u1	28	white
u2	23	middle east
u3	$\perp$	$\perp$
u4	32	white

$\mathcal{R}$ (Reviews)

Item	User	Text	Rating	Date	Like
i1	u1	"Welcome to C..."	4	2023-12-08	11
i2	u2	"The food was ..."	3	2023-12-09	0
i3	u1	"As expected, ..."	5	2023-12-10	3
i1	u4	"Was pretty da..."	3	2023-12-12	1

Figure 7.3. Example of items, users, and reviews from Yelp

There is a number of works addressing exposure over multiple rankings [208, 243, 50]. Differently from our case, the rankings are drawn from a stochastic ranking process and the fairness measures represent the expected exposure of items. More specifically, in [208] the stochastic process is the training algorithm for learning a ranking policy. In [243] user feedback is also considered when learning the policy. In [50] the stochastic ranking processes represent information retrieval systems. We also mention [20] where the authors propose a way to amortize fairness over time. None of these works address the specific case of nested rankings occurring in collaborative rating systems.

Finally, since our focus is building a *fairness assessment* tool rather than a fairness optimizer method, we mention for the sake of completeness the existence of a number of fairness assessment tools.[2] However, all these tools are either not applicable to assess fairness in ranking or do not provide the expressivity we need to assess ranking fairness for different demographic groups.

7.3 Preliminaries

As illustrated in Figure 7.3, a subjective database is typically used to represent user opinions, expressed through reviews over a set of items and it involves three main components: a set of users $\mathcal{U}$, a set of items $\mathcal{I}$ (e.g. restaurants on Yelp or products on Amazon) and a set of reviews $\mathcal{R}$ for those items.

We model $\mathcal{U}$ and $\mathcal{I}$ as relational tables with their own set of identifiers and attributes (such as restaurant location for items and username for users), and $\mathcal{R}$ with a relational table in which each tuple includes (i) an item I, (ii) the user who wrote a review for I, and (iii) rating information, which can include: a numerical value representing how much the user has appreciated the item I, a detailed textual description motivating such rating, and the date of the review. A review may also contain other details, such as the number of users that liked that review. The only assumption here is that a user can write at most one review per item, as it usually happens in collaborative rating systems. Finally, a review r can be associated with a measure of *utility* $u(r)$. Such a measure can be directly available in the subjective

[2] https://blog.einstein.ai/frameworks-tool-kits-principles-and-oaths-oh-my/

Name	Symbol	Property	Level
Exposure	Exp_{avg}	The measure defined in [207]	Item
Exposure-Ratio	Exp_{ratio}	A variant of Exp_{avg} for long lists that will be likely not exhaustively consulted.	Item
Treatment-Ratio	$Treat_{ratio}$	Exposure-Ratio divided by average utility	Item
Weighted Rank Equality	WRE	Distance from rank by Utility	Item
Query Fairness	$\mathcal{F}$	For query results containing multiple items.	Query

Table 7.1. Fairness metrics used in this paper.

dataset or can be derived from attributes of the review. For instance, in Figure 7.3 we could define the utility u of a review as the number of "Like" it received.

Subjective datasets are mainly used in online collaborative rating systems to help users find items of interest [186]. As illustrated in Figure 7.2, a user can interact with one of these systems by issuing a query Q that specifies the user's desiderata, such as "Italian restaurants close to me", upon which the system outputs an ordered set of k items $\langle I_1, ..., I_k \rangle$. For each item, I_i, the system shows an ordered list of its n_i reviews $\langle r_{i,1}, ..., r_{i,n_i} \rangle$. Both the rankings on items and reviews are sorted according to a *relevance score*, which is usually application-specific and opaque to the end-user, or other more transparent criteria, such as the cost for an item and the date of issue for a review.

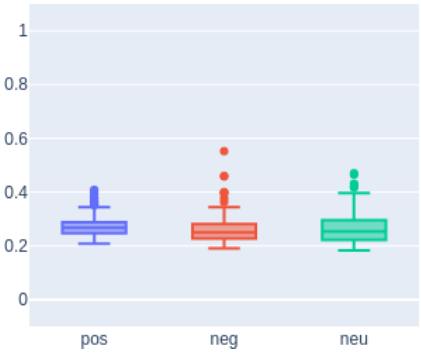
Users can be part of a variety of groups based on (i) their protected attributes, such as `Age` or `Ethnicity`, and (ii) the information they expose through the review, such as the rate given to an item or the sentiment towards an item expressed with a review. We denote with $g \subseteq \mathcal{R}$ a group of reviews written by users that share similarities for a specific attribute (or combination of multiple attributes). For instance, with respect to the subjective dataset in Figure 7.3, the group $g = \{r_1, r_4\}$ denotes the set of reviews written by females. Given two groups g_1 and g_2, we say that they are *comparable* if both are defined over the same (set of) attributes and their corresponding set of reviews do not overlap.

We can think of rankings of items and reviews as fundamentally a *ranking of users*, where users who write high-ranked reviews for high-ranked items obtain more visibility and have therefore more possibility to influence the choices of other users. Several methods have been proposed in the literature to address fairness in ranking [254, 255, 186]. One naive application of such methods is to directly use, for each item, the fairness metrics on $\mathcal{R} \bowtie \mathcal{U}$. Although this could highlight fairness issues on specific items, it would not address the fairness of a query Q, where a high-ranked review for an item i can lose its visibility if i is low-ranked in the result of Q.

7.4 R-Fairness

In this work, we consider r-fairness, that is, fairness with respect to reviewers. Specifically, we consider two types of fairness, referred to as *item-level fairness* and

(a) Exp_{avg} on Yelp for ethnicity



(b) Exp_{avg} on Amazon for sentiment

Figure 7.4. Exposure avg does not capture variations of exposure in different items because avg flattens for long lists.

query-level fairness. For the former type, we identify basic methods for fairness of individual items, based on metrics from the literature: dubbed Exposure [207], treatment [207] and rank equality [111]. Then, we introduce variants of such methods to better capture the properties of our scenario. For the latter type, we introduce a principled aggregation strategy to combine individual items' fairness into query-level fairness.

We summarize the metrics used for our analysis in Table 7.1. Detailed descriptions are provided below.

Item-level fairness. We start from the popular fairness measures of exposure defined in [207].

Definition 7.4.1 (Exposure [207])**.** *The exposure of a group g in a ranking $\mathcal{L}$ is defined as*

$$Exp_{avg}(g|\mathcal{L}) = \frac{1}{|g|}\sum_{e\in g} Exp(e|\mathcal{L})$$

where $Exp(e|\mathcal{L}) = \frac{1}{log_2(1+pos(e|\mathcal{L}))}$ *and* $pos(e|\mathcal{L})$ *denotes the position of* e *in* $\mathcal{L}$*.*

For instance, given the list of reviews related to the item $i1$ in Figure 7.3 the exposure of users age $25-35$ is

$$Exp_{avg}(g|\mathcal{L}) = \frac{1}{log_2(1+1)} + \frac{1}{log_2(1+4)}.$$

One limitation of the above notion of exposure is that, for large lists, this measure tends to converge to a low value, which becomes indistinguishable for different groups of relevant size. This behavior does not allow the identification of groups that are more visible than others. This is confirmed by the experiments of real-world data reported in Figure 7.4, in which the exposure is computed for different groups based on the ethnicity (Figure 7.4(a)) and sentiment (Figure 7.4(b)).

For this reason, we introduce another measure that compares the exposure of the group in a ranking with its best possible outcome.

Let $Exp_{best}(g|\mathcal{L})^*$ denote the best exposure a group g can achieve, i.e. the exposure that g would get if members of g were placed in the top $|g|$ positions of

the ranked list $\mathcal{L}$:

$$Exp_{best}(g|\mathcal{L})^* = \sum_{i=1}^{|g|} \frac{1}{log_2(1+i)}.$$

We call the new measure *ratio* exposure, defined as follows:

Definition 7.4.2 (Ratio exposure)**.** *The ratio exposure of a group g in a ranking $\mathcal{L}$ is defined as:*

$$Exp_{ratio}(g|\mathcal{L}) = \frac{\sum_{e \in g} Exp(e|\mathcal{L})}{Exp_{best}(g|\mathcal{L})^*}.$$

Item-level treatment. The next measure takes into consideration the *utility* of (groups of) elements, under the rationale that it is reasonable for an element in a ranking to have an exposure that is proportional to its utility. For this, let U be an application-dependent function that associates with each element in play its utility, under the hypothesis that this function is the same for all queries.

We build here on the treatment metric in [207] but with Exp_{ratio}. The *utility* of the review according to the opinion of other users. Rating and utility are usually records of values (e.g., for ratings: scores for staff and location plus a descriptive text, in the case of hotels; for utility: positive votes received by other reviewers) and application-dependent. However, the way in which they are implemented does not affect our model.

Definition 7.4.3 (Ratio treatment)**.** *Analogously we define*

$$Treat_{ratio}(g|\mathcal{L}) = \frac{Exp_{ratio}(g|\mathcal{L})}{U(g)}$$

where $U(g)$ is the average utility of members of g.

Item-level rank-equality. Our last item-level fairness metric is a weighted version of the rank equality metric in [111, 124]. This metric relies on the assumption that there exists a ground truth ranking and measures pairwise disagreements with such ranking using the Kendall-Tau distance. Since a "truly fair" ranking may not be available, we reformulate this metric to distinguish whether the proposed ranking is more or less fair than the utility-based ranking.

Definition 7.4.4 (Weighted rank equality)**.** *Given two groups g_1, g_2 we define*

$$WRE\ (g_1, g_2|\mathcal{L}) = \frac{1}{N_{g_1,g_2}} \sum_{(e,e') \in g_1 \times g_2} Inv(e, e')$$

where function $Inv(e, e')$ measures the weighted inversion as

$$Inv(e, e') = \begin{cases} +W(e, e') & \text{if } W(e, e') < 0 \text{ and } Exp(e) < Exp(e'), \\ -W(e, e') & \text{if } W(e, e') < 0 \text{ and } Exp(e) > Exp(e'), \\ 0 & \text{if } W(e, e') \geq 0, \end{cases}$$

and $W(e, e') = (U(e) - U(e'))(Exp(e|\mathcal{L}) - Exp(e'|\mathcal{L}))$ is the weight of the (possible) inversion of the couple. Finally, $N_{g_1,g_2} = \sum_{(e,e') \in g_1 \times g_2} |W(e, e')|$ is the normalization factor.

Query-level fairness. For query-level, we design a dedicated metric. Assume we are given a query $q(\mathcal{R})$ which is associated to a ordered set of ranked lists $\langle \mathcal{L}_1, \mathcal{L}_2, ..., \mathcal{L}_n \rangle$.

Definition 7.4.5 (Query-level fairness)**.** *Given a measure $\mathcal{F}$ (e.g., Exp_{cov}) and a group g, we define $\mathcal{F}(g|q(\mathcal{R}))$ as follows:*

$$\mathcal{F}(g|q(\mathcal{R})) = \frac{1}{count(g, q(\mathcal{R}))} \sum_{i=1}^{n} \mathcal{F}(g|\mathcal{L}_i) \cdot Exp(\mathcal{L}_i|q(\mathcal{R}))$$

where $count(g, q(\mathcal{R}))$ is the number of items in $q(\mathcal{R})$ in which at least one member of g exists and $Exp(\mathcal{L}_i|q(\mathcal{R}))$ is the individual exposure of the i-th ranked list $\mathcal{L}_i$ in $q(\mathcal{R})$, that is:

$$Exp(\mathcal{L}_i|q(\mathcal{R})) = \frac{1}{log_2(1+i)}.$$

In order to combine the values of the fairness measure $\mathcal{F}(g)$ across the different ranked lists $\mathcal{L}_1, ..., \mathcal{L}_n$ we use the following approach: first, we weight each fairness measure $\mathcal{F}(g|\mathcal{L}_i)$ by the exposure of the corresponding item (e.g. $Exp(\mathcal{L}_i|q(\mathcal{R}))$. This allows us to mitigate the effect of the fairness measure of group associated to an low-ranked item. Then, we sum up those values and take the average. The latter step is required since each group g may not occur in every item of $q(\mathcal{R})$.

7.5 Experiments

We run extensive experiments to examine how the various measures of r-fairness vary. For our experiments, we use two real-world datasets, dubbed Yelp and Amazon, collected during these studies and available on demand.

- `Yelp` consists of restaurant reviews from the `yelp.com` website. Using the Yelp API we selected 200 restaurants, collecting 488789 reviews from 324228 users with an average 2338.70 reviews per item. For each restaurant, we collect name, address, average rating, and category. For each review, we collect text, rating, publication date, as well as the user name and profile picture (which we do not share). We also collected, if any, the feedback of each review given by other users, who can assign a vote among useful, funny, and cool. The selected restaurants are chosen from Yelp Dataset according to two criteria: location and type of cuisine. We chose 4 different American states and about 7 different types of cuisine.

- `Amazon` consists of product reviews from the `Amazon.com` website. We selected 245 items, collected 774760 reviews from 608826 users, with an average 3637.37 reviews per item, and collected a total of around 500k reviews. Items for `Amazon` are selected from different categories. In both cases, the items returned by the query are ranked by the aggregate rating (e.g., stars) assigned to the item. For each review, we collect text, rating, publication date, and user name. We also collected the feedback of each review given by other users.

In both datasets, users' demographic data are predicted by user name and picture (on Yelp) and by user name only (on Amazon) and picture using the Clarifai[3] system. In case of uncertain prediction, we set a null value. The users' activity in both Yelp and Amazon platforms is reported in Table 7.2.

Summary of results. We demonstrate three main things.

- **RI 1** Compare the *item-level* exposure of groups of subjective data, for each individual item: in both platforms, there is a significant variation in fairness. Some items are reviewed in a balanced manner, while others disproportionately highlight specific user categories. Yelp more than Amazon or vice versa.
- **RI 2** Examine how groups are treated in different platforms under different fairness measures: Amazon and Yelp treat groups very differently, with Amazon prioritizing utility more heavily in their rankings.
- **RI 3** Measure the *query-level* of groups of subjective data for different queries, where each query returns a ranking of items: On the same platform, different search queries can lead to varying levels of exposure for the same groups of individuals, even when the items returned by these queries are identical.

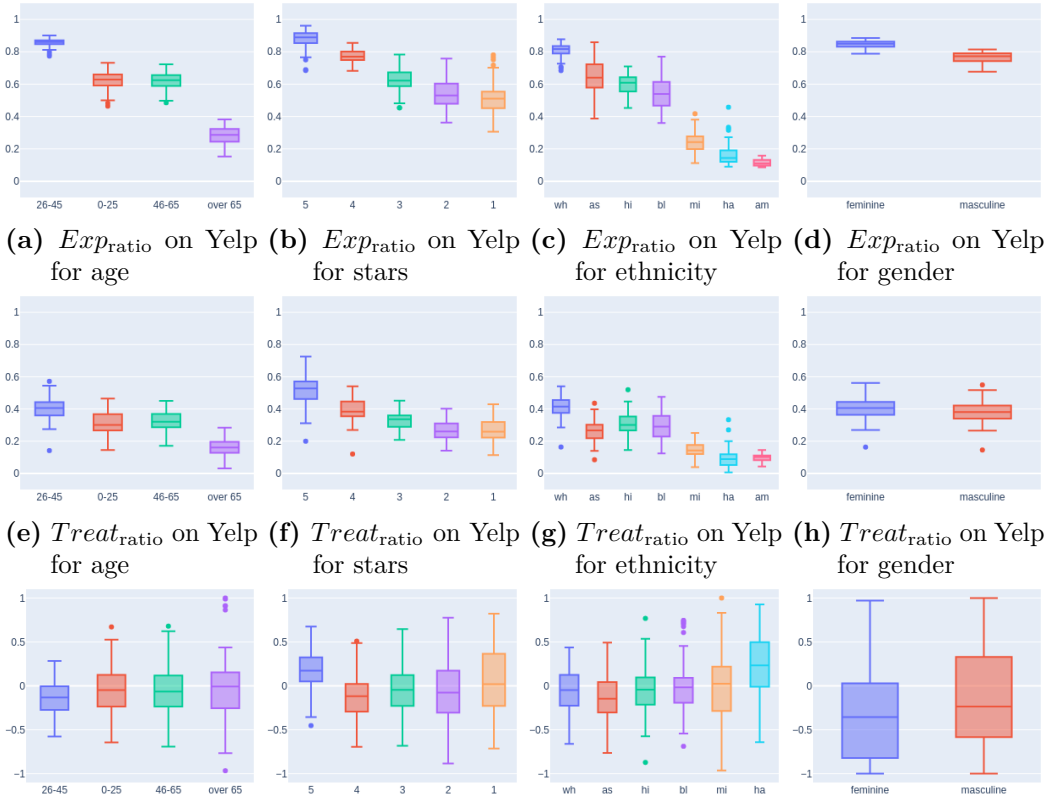
7.5.1 Item-level Exposure

This section elaborates on **RI 1**. As reported in Table 7.2, for almost all group families, the percentage of the activity of reviews without any group (the missing values) is high. If the groups are inferred respecting the marginal activity of each group, then the Exp_{ratio} and $Treat_{ratio}$ change a little. Indeed, being that these two measures take more consideration of the top reviews, after inferring the groups, there are more new top reviews for groups with high activity and few or none new top reviews for groups with low activity. For this reason, we prefer to exclude the group of missing values from our plot rather than inferring the groups. In these experiments, reported in Figure 7.5 on Yelp's dataset and Figure 7.6 on Amazon's dataset, the barplots related to a group are with respect to the items in which the group is present. So the barplots for the ethnicity 'white' are with respect to all the items, being this ethnicity present in every ranking, while for the ethnicity 'american indian or Alaska native' the measures are with respect to about half of the items. The barplots highlight that the introduced Exp_{ratio} measure is more discriminating than the Exp_{avg} one. Indeed, with long rankings, the Exp_{avg} tends to flatten out, and all groups have very similar Exp_{avg} values. This is due to the decreasing values of the exposure function. The same happens even if we consider a few reviews per item. On the counterparts, the Exp_{ratio} discriminates among different groups, by giving more relevance to the top positions. This implies that groups with larger cardinality tend to have higher values of Exp_{ratio}; indeed, the larger the cardinality of a group, the higher the probability that a review of the group is in the first positions. We remark that the values of Exp_{ratio} can not be inferred by exclusively looking at the cardinality of the groups. In fact, the Exp_{ratio} of the ethnicity 'white' is about ten times larger than the value of the ethnicity 'american indian or alaska native', although the cardinality of the ethnicity 'white' is thousands of times larger.

[3] https://www.clarifai.com

Group	Symbol	Activity (%)
Yelp		
unknown ethnicity	/	42.67
white	wh	29.33
asian	as	13.58
hispanic, latino, or spanish origin	hi	8.61
black or african american	bl	5.16
middle eastern or north african	mi	0.48
native hawaiian or pacific islander	ha	0.13
american indian or alaska native	am	0.03
unknown age	/	42.67
26-45	26-45	38.17
0-25	0-25	9.18
45-65	45-65	9.17
over 65	over 65	0.01
5 stars	5	48.89
4 stars	4	26.24
3 stars	3	11.58
2 stars	2	7.06
1 star	1	6.23
unknown gender	/	42.68
feminine	feminine	33.58
masculine	masculine	23.75
Amazon		
unknown gender	/	46.63
female	female	28.14
male	male	25.21
andy[4]	andy	0.03
positive	pos	49.54
negative	neg	30.85
neutral	neu	19.61

Table 7.2. Summary of users' activity.



(a) Exp_{ratio} on Yelp for age **(b)** Exp_{ratio} on Yelp for stars **(c)** Exp_{ratio} on Yelp for ethnicity **(d)** Exp_{ratio} on Yelp for gender

(e) $Treat_{\text{ratio}}$ on Yelp for age **(f)** $Treat_{\text{ratio}}$ on Yelp for stars **(g)** $Treat_{\text{ratio}}$ on Yelp for ethnicity **(h)** $Treat_{\text{ratio}}$ on Yelp for gender

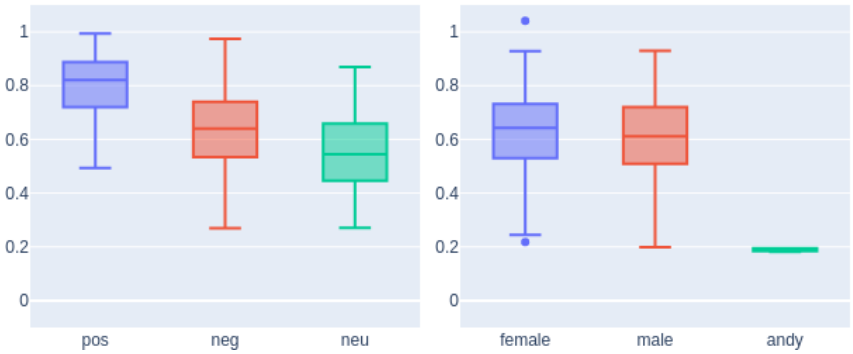
(i) *WRE* on Yelp for age **(j)** *WRE* on Yelp for stars **(k)** *WRE* on Yelp for ethnicity **(l)** *WRE* on Yelp for gender

Figure 7.5. Box-plots showing how each group is treated in different items under different fairness measures on Yelp. Different group families on columns and different measures on rows. Each datapoint is an item.

Another sign of the flattening of Exp_{avg} is that all the groups have almost the same range, while in Exp_{ratio}, $Treat_{ratio}$ and *WRE* different groups have different ranges.

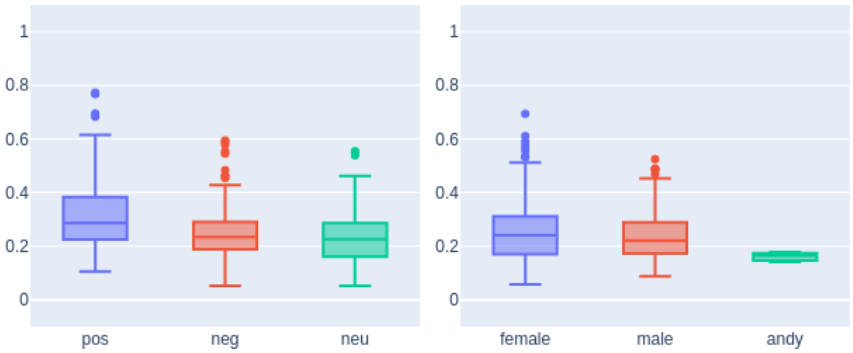
Weighted rank equality requires much more computational effort to calculate than Exp_{avg}, Exp_{ratio} and $Treat_{ratio}$. Indeed, to compute the weighted rank equality of a group g, we consider all possible couples between reviews of group g and reviews of other groups, implying that in the worst case, we need $O(|\mathcal{L}|^2)$ operations; while other measures only consider reviews of group g, requiring $O(|g|)$ operations. For this reason, we computed weighted rank equality for the first 100 reviews per item reported in Figure 7.5 (the value of the ethnicity 'american indian or alaska native' in Figure 7.5k is missing because there are no reviews of this ethnicity in the first 100 positions of any item).

Figure 7.5 and Figure 7.6 allow us to draw some conclusions about **RI 2**. There are no evident differences between Yelp's and Amazon's datasets when the fairness measure is Exposure Average or Treatment Ratio, and the general trend is that the higher the activity, the greater the exposure/treatment. Contrarily, there is a clear discrepancy with the rank equality measure. The weighted rank equality in



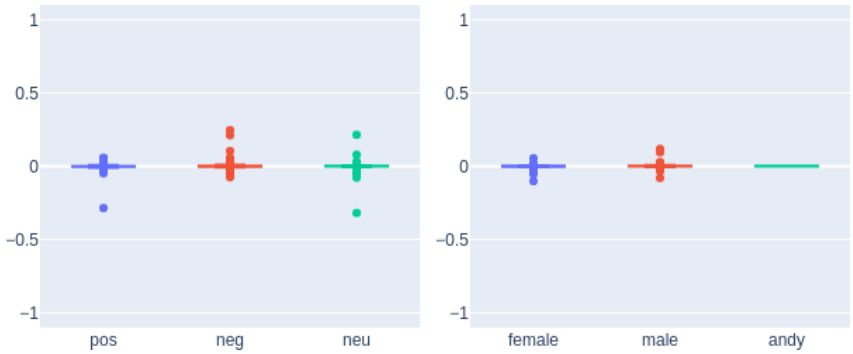
(a) Exp_{ratio} on Amazon for sentiment

(b) Exp_{ratio} on Amazon for gender



(c) $Treat_{ratio}$ on Amazon for sentiment

(d) $Treat_{ratio}$ on Amazon for gender



(e) *WRE* on Amazon for sentiment

(f) *WRE* on Amazon for gender

Figure 7.6. Box-plots showing how each group is treated in different items under different fairness measures on Amazon. Each datapoint is an item.

Amazon is approximately zero for each group. This happens because the rankings are close to being sorted by utility, and all the inversions concerning this sorting are among reviews with close utility values. Instead, on Yelp dataset, the weighted rank equality varies between -1 and 1 for each group. This means that for each group there exist items in which it is over-represented with respect to its utility, and other items in which it is under-represented. Moreover, this implies that Yelp's reviews are not ranked by utility.

7.5.2 Query-level Exposure

This section addresses **RI 3**, which aims to measure the query-level exposure of groups for different queries by using the formula stated in Definition 7.4.5. Different queries may return different sets of items or the same set of items with different rankings. We focus on the latter case, indeed, the fairness of groups calculated on different items is expected to be different. In these experiments, we focus on Yelp's dataset because it is more versatile for generating real-world queries than Amazon's. First, we chose a set of restaurants in the same city (Los Angeles), and then we ordered them according to different queries. The real-world queries reported in Figure 7.7 are generated by the proximity of random points into the city (*distance asc*), overall evaluation (*stars desc*), number of reviews (*#reviews*) and price (*price asc* and *price desc*). The goal is to show that the query-level fairness of a group changes as the queries change, even if the set of the restaurant is the same. We stress that a direct comparison with the plots in Subsection 7.5.1 is not possible because we now consider only a subset of restaurants. Moreover, we discarded Rank-equality in the query-level fairness because the range of the barplots in Figure 7.5 are often overlapping. To make the experiments more realistic, we consider only the first fifteen reviews per restaurant. Indeed, when someone consults rating platforms, he/she usually considers only the first reviews per item, and the items are read by following the order given by the platform. It is unrealistic for someone to read all the reviews of an item before reading the reviews of the following item. As a consequence, in Figure 7.7 there are fewer ethnicities than those reported in Table 7.2.

Let us focus on Figure 7.7. First of all, we note that the exposure of a group varies depending on the chosen query. Clearly, the range of each group is much narrower compared to the plots in Figure 7.5, as the total number of reviews considered is considerably smaller (hundreds of thousands compared to about a hundred). Another difference concerns the trend of activity: while in the plots in Figure 7.5, higher activity almost always corresponded to higher exposure or treatment, this does not apply with query-level fairness. This is evident in Figure 7.7, where the group with the highest $Treat_{ratio}$ corresponds to the ethnicity 'black or African American', which has considerably smaller activity compared to other ethnicities.

Finally, we observe that, given two groups, we cannot establish *a priori* which one has the higher exposure or treatment, as there are some inversions for different queries. Similar results are obtained with Amazon's dataset and with Yelp's dataset for the other groups. The plots are omitted due to space limitations.

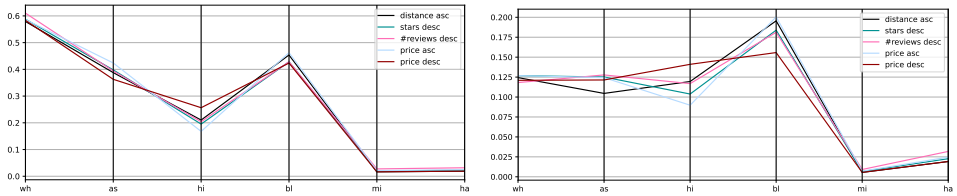


Figure 7.7. Query-level fairness for Exp_{ratio} (top) and $Treat_{ratio}$ (bottom) on Yelp for ethnicity.

7.6 Concluding Remarks

In this paper, we have addressed the problem of r-fairness in collaborative rating platforms, focusing on understanding how the ranking of subjective data impacts fairness among different reviewers' groups. We introduced a comprehensive fairness assessment pipeline, which begins with data collection from real-world platforms via user queries, and ends with the evaluation of the fairness of these rankings, offering insights into the exposure of various user groups. Our experiments, conducted on real-world datasets from platforms like Yelp and Amazon, have shown that the item ranking process impacts fairness outcomes, often amplifying the visibility of certain user groups over others. This highlights the importance of transparency in ranking algorithms to ensure equitable exposure across diverse user demographics. The insights gained from our study pave the way for more equitable design and evaluation of collaborative rating systems. Future work aims at exploring additional platforms and developing more sophisticated fairness metrics that better capture subjective data.

Chapter 8

Conclusions and Future Work

In this thesis we studied and explored ways to improve information quality across a range of tasks, data types, and domains using ML techniques, particularly Large Language Models (LLMs).

In Chapter 2, we addressed *entity count estimation*, designing scalable solutions through sampling, which we evaluated using ML and statistical techniques. This analysis offered insights into balancing computational efficiency with accuracy, which is essential for large-scale applications. Chapter 3 focused on entity resolution (ER), examining the potential of integrating knowledge graphs (KGs) to ground and enhance predictions, revealing strengths and limitations in using external knowledge for ER tasks. In Chapter 4, we turned to managing large repositories of multi-choice questions, employing Transformer models combined with graph community detection. Chapter 5 addressed uncertainty in data, leveraging LLM-based crowdsourcing methods to self-assess answer reliability. We evaluated the performance of these methods in both accuracy and uncertainty estimation, particularly in tasks like factual question-answering. In Chapter 6, we explored on-demand data building within the manufacturing domain, using LLMs to interact with structured data via code prompts. Finally, Chapter 7 examined fairness in ethical data quality, constructing a pipeline to assess ranking fairness across groups, using real-world data from sources like Yelp and Amazon.

While ML and specifically LLM-based approaches have opened up new avenues for many tasks, questions surrounding reliability and trustworthiness persist. Chapter 5 partially addresses these concerns, evaluating the ability of LLMs to estimate uncertainty, suggesting a direction for improving trust in model-generated answers.

Future works. Regarding entity count estimation (Chapter 2), a more comprehensive evaluation covering recent ER ML and statistical approaches (e.g., [149, 242]) and extending computational caps would allow a better assessment of the scalability of these methods. Additionally, capture-recapture techniques [73], which estimate population size through iterative sampling, could offer an alternative for scaling entity estimation to even larger datasets. Regarding the integration of KGs for ER (Chapter 3), though conceptually promising, requires practical implementation to validate its theoretical benefits. Equipping an LLM agent with KG access, potentially via SPARQL or natural language interfaces, could allow better use of contextual knowledge. Regarding the use of ER for MCQ management (Chapter 4),

an open direction is automating the clustering of duplicated answers, potentially alleviating manual work and improving clustering accuracy. Likewise, LLM-based crowdsourcing techniques (Chapter 5) could benefit from more extensive scalability tests, adjusting crowd size and response diversity, to assess performance consistency. The COSMADS framework proposed in Chapter 6 would benefit from real-world manufacturing evaluation, ideally with access to live industrial data to validate the effectiveness of LLM interactions with physical assets. Lastly, regarding the fairness pipeline (Chapter 7), investigating misclassification in group assignment due to errors in sensitive attribute prediction remains an open research question, essential for refining fairness metrics and enhancing ethical data quality.

Bibliography

[1] Abdar, M., Pourpanah, F., Hussain, S., Rezazadegan, D., Liu, L., Ghavamzadeh, M., Fieguth, P., Cao, X., Khosravi, A., Acharya, U.R., et al.: A review of uncertainty quantification in deep learning: Techniques, applications and challenges. Information fusion 76, 243–297 (2021) (Cited on page vii)

[2] Abdollahpouri, H., Burke, R.: Multi-stakeholder recommendation and its connection to multi-sided fairness. arXiv preprint arXiv:1907.13158 (2019) (Cited on page 86)

[3] Abiteboul, S., Stoyanovich, J.: Transparency, fairness, data protection, neutrality: Data management challenges in the face of new regulation. J. Data and Information Quality 11(3) (jun 2019), https://doi.org/10.1145/3310231 (Cited on page 86)

[4] Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F.L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al.: Gpt-4 technical report. arXiv preprint arXiv:2303.08774 (2023) (Cited on pages 6, 7, and 71)

[5] Affolter, K., Stockinger, K., Bernstein, A.: A comparative survey of recent natural language interfaces for databases. The VLDB Journal 28, 793–819 (2019) (Cited on page 72)

[6] Aiello, M., Georgievski, I.: Service composition in the chatgpt era. SOCA (2023) (Cited on page 73)

[7] Albano, V., Firmani, D., Laura, L., Mathew, J.G., Paoletti, A.L., Torrente, I.: Nlp-based management of large multiple-choice test item repositories. Journal of Learning Analytics 10(3), 28–44 (2023) (Cited on pages ix and xi)

[8] Albano, V., Firmani, D., Laura, L., Mathew, J.G., Paoletti, A.L., Torrente, I.: Resolving duplicates in large multiple-choice questions repositories. In: IRCDL. pp. 34–40 (2024) (Cited on pages ix and xi)

[9] Amadori, F., Bardani, M., Bernasconi, E., Cappelletti, F., Catarci, T., Drudi, G., Ferretti, M., Foschini, L., Galli, P., Germani, M., et al.: Electrospindle 4.0: Towards zero defect manufacturing of spindles. In: CEUR Workshop Proceedings. vol. 3144, pp. 1–7 (2022) (Cited on page xiii)

[10] Andreou, A.S., Firmani, D., Mathew, J.G., Mecella, M., Pingos, M.: Using knowledge graphs for record linkage: Challenges and opportunities. In: In-

ternational Conference on Advanced Information Systems Engineering. pp. 145–151. Springer (2023) (Cited on pages ix, x, and xii)

[11] Ankerst, M., Breunig, M.M., Kriegel, H.P., Sander, J.: Optics: Ordering points to identify the clustering structure. ACM Sigmod record 28(2), 49–60 (1999) (Cited on page 23)

[12] Ausiello, G., Firmani, D., Laura, L.: Real-time monitoring of undirected networks: Articulation points, bridges, and connected and biconnected components. Networks 59(3), 275–288 (2012), https://doi.org/10.1002/net.21450 (Cited on page 54)

[13] Ausiello, G., Firmani, D., Laura, L.: The (betweenness) centrality of critical nodes and network cores. In: 2013 9th International Wireless Communications and Mobile Computing Conference (IWCMC). pp. 90–95. IEEE (2013) (Cited on page 54)

[14] Azevedo, J.M., Oliveira, E.P., Damas Beites, P.: Using learning analytics to evaluate the quality of multiple-choice questions: A perspective with classical test theory and item response theory. The International Journal of Information and Learning Technology 36(4), 322–341 (Jan 2019) (Cited on pages 44 and 45)

[15] Balzotti, L., Firmani, D., Mathew, J.G., Torlone, R., Amer-Yahia, S.: R-fairness: Assessing fairness of ranking in subjective data. ACL Rolling Review (2024), https://openreview.net/forum?id=U6Kyzivd7v, under minor revision (Cited on pages x and xi)

[16] Bansal, N., Blum, A., Chawla, S.: Correlation clustering. Machine learning 56, 89–113 (2004) (Cited on page 15)

[17] Bengio, Y., Ducharme, R., Vincent, P.: A neural probabilistic language model. Advances in neural information processing systems 13 (2000) (Cited on page 6)

[18] Berardi, D., Calvanese, D., De Giacomo, G., Lenzerini, M., Mecella, M.: Automatic service composition based on behavioral descriptions. IJCIS 14(04), 333–376 (2005) (Cited on page 73)

[19] Biega, A.J., Gummadi, K.P., Weikum, G.: Equity of attention: Amortizing individual fairness in rankings. In: The 41st international acm sigir conference on research & development in information retrieval. pp. 405–414 (2018) (Cited on page 84)

[20] Biega, A.J., Gummadi, K.P., Weikum, G.: Equity of attention: Amortizing individual fairness in rankings. In: The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval. p. 405–414. SIGIR '18, Association for Computing Machinery, New York, NY, USA (2018), https://doi.org/10.1145/3209978.3210063 (Cited on page 87)

[21] Bilenko, M., Kamath, B., Mooney, R.J.: Adaptive blocking: Learning to scale up record linkage. In: Sixth International Conference on Data Mining (ICDM'06). pp. 87–96. IEEE (2006) (Cited on pages 20 and 21)

[22] Bingham, E., Chen, J.P., Jankowiak, M., Obermeyer, F., Pradhan, N., Karaletsos, T., Singh, R., Szerlip, P.A., Horsfall, P., Goodman, N.D.: Pyro: Deep universal probabilistic programming. J. Mach. Learn. Res. 20, 28:1–28:6 (2019), http://jmlr.org/papers/v20/18-403.html (Cited on page 66)

[23] Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching word vectors with subword information. Transactions of the association for computational linguistics 5, 135–146 (2017) (Cited on pages 3 and 45)

[24] Brown, P.F., Della Pietra, V.J., Desouza, P.V., Lai, J.C., Mercer, R.L.: Class-based n-gram models of natural language. Computational linguistics 18(4), 467–480 (1992) (Cited on page 6)

[25] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al.: Language models are few-shot learners. NeurIPS 2020 (2020) (Cited on pages 7, 8, 10, 71, and 77)

[26] Brunner, U., Stockinger, K.: Entity matching with transformer architectures-a step forward in data integration. In: EDBT 2020 (2020) (Cited on page 46)

[27] Bruno, N., Kwon, Y., Wu, M.C.: Advanced join strategies for large-scale distributed computation. Proceedings of the VLDB Endowment 7(13), 1484–1495 (2014) (Cited on page 12)

[28] Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., Layton, R., VanderPlas, J., Joly, A., Holt, B., Varoquaux, G.: API design for machine learning software: experiences from the scikit-learn project. In: ECML PKDD Workshop: Languages for Data Mining and Machine Learning. pp. 108–122 (2013) (Cited on page 48)

[29] Calamo, M., De Franceschi, A., De Santis, G., Leotta, F., Mazzaroppi, C., Mathew, J.G., Mecella, M., Monti, F., Sabatino, C., Visani, L., et al.: Topkontrol: a monitoring and quality control system for the packaging production (2023) (Cited on page xiii)

[30] Calvanese, D., De Giacomo, G., Lenzerini, M., Mecella, M., Patrizi, F., et al.: Automatic service composition and synthesis: the roman model. IEEE Data Eng. Bull. 31(3), 18–22 (2008) (Cited on page vii)

[31] Celis, L.E., Straszak, D., Vishnoi, N.K.: Ranking with fairness constraints. arXiv preprint arXiv:1704.06840 (2017) (Cited on page 84)

[32] Ch, D.R., Saha, S.K.: Automatic multiple choice question generation from text: A survey. IEEE Transactions on Learning Technologies 13(1), 14–25 (2018) (Cited on page 45)

[33] Chang, Y., Wang, X., Wang, J., Wu, Y., Yang, L., Zhu, K., Chen, H., Yi, X., Wang, C., Wang, Y., et al.: A survey on evaluation of large language models. TIST (2023) (Cited on pages 71 and 72)

[34] Chaturvedi, A., Pandit, O., Garain, U.: Cnn for text-based multiple choice question answering. In: ACL 2018-56th Annual Meeting of the Association for Computational Linguistics. pp. 272–277 (2018) (Cited on page 45)

[35] Chen, B., Zhang, F., Nguyen, A., Zan, D., Lin, Z., Lou, J.G., Chen, W.: Codet: Code generation with generated tests. In: ICLR (2022) (Cited on page 72)

[36] Chen, D.: Reading wikipedia to answer open-domain questions. arXiv preprint arXiv:1704.00051 (2017) (Cited on page 10)

[37] Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H.P.d.O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., et al.: Evaluating large language models trained on code. arXiv preprint arXiv:2107.03374 (2021) (Cited on page 72)

[38] Chen, T., Guestrin, C.: Xgboost: A scalable tree boosting system. In: Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining. pp. 785–794 (2016) (Cited on page 59)

[39] Chen, W., Ma, X., Wang, X., Cohen, W.W.: Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. arXiv preprint arXiv:2211.12588 (2022) (Cited on page 72)

[40] Chen, Y., Yi, K.: Two-level sampling for join size estimation. In: Proceedings of the 2017 ACM International Conference on Management of Data. pp. 759–774 (2017) (Cited on page 14)

[41] Cheng, S.H., Higham, N.J.: A modified cholesky algorithm based on a symmetric indefinite factorization. SIAM Journal on Matrix Analysis and Applications 19(4), 1097–1110 (1998), https://doi.org/10.1137/S0895479896302898 (Cited on page 66)

[42] Chirkova, N., Troshin, S.: Empirical study of transformers for source code. In: Proceedings of ESEC/FSE 2021 (2021) (Cited on page 72)

[43] Christen, P.: A survey of indexing techniques for scalable record linkage and deduplication. IEEE transactions on knowledge and data engineering 24(9), 1537–1555 (2011) (Cited on page 5)

[44] Clauset, A., Newman, M.E.J., Moore, C.: Finding community structure in very large networks. Physical Review E 70(6) (dec 2004) (Cited on page 54)

[45] Colabianchi, S., Tedeschi, A., Costantino, F.: Human-technology integration with industrial conversational agents: A conceptual architecture and a taxonomy for manufacturing. JIII (2023) (Cited on page 70)

[46] Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., Grave, E., Ott, M., Zettlemoyer, L., Stoyanov, V.: Unsupervised cross-lingual representation learning at scale. arXiv preprint arXiv:1911.02116 (2019) (Cited on page 48)

[47] Dai, D., Sun, Y., Dong, L., Hao, Y., Sui, Z., Wei, F.: Why can gpt learn in-context? language models secretly perform gradient descent as meta optimizers. arXiv preprint arXiv:2212.10559 (2022) (Cited on page 71)

[48] Dawid, A.P., Skene, A.M.: Maximum likelihood estimation of observer error-rates using the em algorithm. Journal of the Royal Statistical Society. Series C (Applied Statistics) 28(1), 20–28 (1979), http://www.jstor.org/stable/2346806 (Cited on pages 64 and 65)

[49] Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: Burstein, J., Doran, C., Solorio, T. (eds.) 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). pp. 4171–4186. Association for Computational Linguistics, Minneapolis, Minnesota (Jun 2019), https://aclanthology.org/N19-1423 (Cited on pages 4, 7, 15, 23, and 71)

[50] Diaz, F., Mitra, B., Ekstrand, M.D., Biega, A.J., Carterette, B.: Evaluating stochastic rankings with expected exposure. In: Proceedings of the 29th ACM International Conference on Information & Knowledge Management. p. 275–284. CIKM '20, Association for Computing Machinery, New York, NY, USA (2020), https://doi.org/10.1145/3340531.3411962 (Cited on page 87)

[51] Do, H.H., Rahm, E.: Coma—a system for flexible combination of schema matching approaches. In: VLDB Proceedings. pp. 610–621. Elsevier (2002) (Cited on page 78)

[52] Dong, X., Gabrilovich, E., Heitz, G., Horn, W., Lao, N., Murphy, K., Strohmann, T., Sun, S., Zhang, W.: Knowledge vault: a web-scale approach to probabilistic knowledge fusion. In: The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014. pp. 601–610. ACM (2014), https://doi.org/10.1145/2623330.2623623 (Cited on page 64)

[53] Dong, X.L., Srivastava, D.: Big data integration. In: 2013 IEEE 29th international conference on data engineering (ICDE). pp. 1245–1248. IEEE (2013) (Cited on page 2)

[54] Dong, Y., Ma, J., Wang, S., Chen, C., Li, J.: Fairness in graph mining: A survey. IEEE Transactions on Knowledge and Data Engineering 35(10), 10583–10602 (2023) (Cited on page 86)

[55] Du, Y., Li, S., Torralba, A., Tenenbaum, J.B., Mordatch, I.: Improving factuality and reasoning in language models through multiagent debate. In: Forty-first International Conference on Machine Learning (Cited on page 8)

[56] Du, Y., Li, S., Torralba, A., Tenenbaum, J.B., Mordatch, I.: Improving factuality and reasoning in language models through multiagent debate. In:

Proceedings of the 41st International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 235, pp. 11733–11763. PMLR (21–27 Jul 2024), https://proceedings.mlr.press/v235/du24e.html (Cited on pages 64, 67, and 68)

[57] Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Yang, A., Fan, A., Goyal, A., et al.: The llama 3 herd of models (2024), https://arxiv.org/abs/2407.21783 (Cited on pages 63 and 67)

[58] Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Yang, A., Fan, A., et al.: The llama 3 herd of models. arXiv preprint arXiv:2407.21783 (2024) (Cited on pages 6 and 7)

[59] Dubey, M., Banerjee, D., Abdelkawi, A., Lehmann, J.: Lc-quad 2.0: A large dataset for complex question answering over wikidata and dbpedia. In: The Semantic Web–ISWC 2019: 18th International Semantic Web Conference, Auckland, New Zealand, October 26–30, 2019, Proceedings, Part II 18. pp. 69–78. Springer (2019) (Cited on page 9)

[60] Dwork, C., Hardt, M., Pitassi, T., Reingold, O., Zemel, R.: Fairness through awareness. In: Proceedings of the 3rd innovations in theoretical computer science conference. pp. 214–226 (2012) (Cited on page 86)

[61] Ebraheem, M., Thirumuruganathan, S., Joty, S., Ouzzani, M., Tang, N.: Distributed representations of tuples for entity resolution. Proceedings of the VLDB Endowment 11(11), 1454–1467 (2018) (Cited on pages 15 and 41)

[62] Ebraheem, M., Thirumuruganathan, S., Joty, S.R., Ouzzani, M., Tang, N.: Distributed representations of tuples for entity resolution. Proc. VLDB Endow. 11(11), 1454–1467 (2018), http://www.vldb.org/pvldb/vol11/p1454-ebraheem.pdf (Cited on pages 5 and 46)

[63] Efthymiou, V., Papadakis, G., Papastefanatos, G., Stefanidis, K., Palpanas, T.: Parallel meta-blocking: Realizing scalable entity resolution over large, heterogeneous data. In: 2015 IEEE International Conference on Big Data (Big Data). pp. 411–420. IEEE (2015) (Cited on page 35)

[64] Efthymiou, V., Stefanidis, K., Pitoura, E., Christophides, V.: Fairer: Entity resolution with fairness constraints. p. 3004–3008. CIKM '21, Association for Computing Machinery, New York, NY, USA (2021), https://doi.org/10.1145/3459637.3482105 (Cited on page 61)

[65] Efthymiou, V., Stefanidis, K., Pitoura, E., Christophides, V.: Fairer: entity resolution with fairness constraints. In: Proceedings of the 30th ACM International Conference on Information & Knowledge Management. pp. 3004–3008 (2021) (Cited on page 86)

[66] Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proceedings

of the Second International Conference on Knowledge Discovery and Data Mining. p. 226–231. KDD'96, AAAI Press (1996) (Cited on pages 15, 20, 23, and 28)

[67] Fahad, A., Alshatri, N., Tari, Z., Alamri, A., Khalil, I., Zomaya, A.Y., Foufou, S., Bouras, A.: A survey of clustering algorithms for big data: Taxonomy and empirical analysis. IEEE transactions on emerging topics in computing 2(3), 267–279 (2014) (Cited on page 35)

[68] Fellegi, I.P., Sunter, A.B.: A theory for record linkage. Journal of the American Statistical Association 64(328), 1183–1210 (1969) (Cited on pages 13, 15, 20, and 22)

[69] Feng, S., Shi, W., Wang, Y., Ding, W., Balachandran, V., Tsvetkov, Y.: Don't hallucinate, abstain: Identifying llm knowledge gaps via multi-llm collaboration. arXiv preprint arXiv:2402.00367 (2024) (Cited on page 8)

[70] Feng, S., Shi, W., Wang, Y., Ding, W., Balachandran, V., Tsvetkov, Y.: Don't hallucinate, abstain: Identifying LLM knowledge gaps via multi-LLM collaboration. In: Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 14664–14690. Association for Computational Linguistics, Bangkok, Thailand (Aug 2024), https://aclanthology.org/2024.acl-long.786 (Cited on pages 64, 67, and 68)

[71] Fernandez, R.C., Elmore, A.J., Franklin, M.J., Krishnan, S., Tan, C.: How large language models will disrupt data management. VLDB Proceedings (2023) (Cited on page 72)

[72] Fernandez, R.C., Subramaniam, P., Franklin, M.J.: Data market platforms: trading data assets to solve data problems. Proceedings of the VLDB Endowment 13(12), 1933–1947 (2020) (Cited on page vii)

[73] Fienberg, S.E.: The multiple recapture census for closed populations and incomplete 2k contingency tables. Biometrika 59(3), 591–603 (1972) (Cited on pages 13 and 98)

[74] Firmani, D., Galhotra, S., Saha, B., Srivastava, D.: Robust entity resolution using a crowdoracle. IEEE Data Eng. Bull. 41(2), 91–103 (2018), http://sites.computer.org/debull/A18june/p91.pdf (Cited on page 46)

[75] Firmani, D., Leotta, F., Mathew, J.G., Rossi, J., Balzotti, L., Song, H., Roman, D., Dautov, R., Husom, E.J., Sen, S., et al.: Intend: Intent-based data operation in the computing continuum. In: CEUR WORKSHOP PROCEEDINGS. vol. 3692, pp. 43–50. CEUR-WS (2024) (Cited on page xiii)

[76] Firmani, D., Mathew, J.G., Santoro, D., Simonini, G., Zecchini, L., et al.: Bridging the gap between buyers and sellers in data marketplaces with personalized datasets. In: CEUR Workshop Proceedings. vol. 3478, pp. 525–534. CEUR-WS (2023) (Cited on page xiii)

[77] Firmani, D., Mathew, J.G., Srivastava, D.: Evaluating methods for efficient entity count estimation (2024), manuscript in preparation (Cited on pages ix and x)

[78] Firmani, D., Saha, B., Srivastava, D.: Online entity resolution using an oracle. Proceedings of the VLDB Endowment 9(5), 384–395 (2016) (Cited on pages 21, 22, and 27)

[79] Firmani, D., Tanca, L., Torlone, R.: Ethical dimensions for data quality. Journal of Data and Information Quality (JDIQ) 12(1), 1–5 (2019) (Cited on page vi)

[80] Frank, O.: Estimation of the number of connected components in a graph by using a sampled subgraph. Scandinavian Journal of Statistics pp. 177–188 (1978) (Cited on pages 13, 15, 19, 23, and 34)

[81] Galhotra, S., Firmani, D., Saha, B., Srivastava, D.: Robust entity resolution using random graphs. In: Proceedings of the 2018 International Conference on Management of Data. pp. 3–18 (2018) (Cited on pages 21, 22, 24, and 28)

[82] Galhotra, S., Firmani, D., Saha, B., Srivastava, D.: Beer: blocking for effective entity resolution. In: Proceedings of the 2021 International Conference on Management of Data. pp. 2711–2715 (2021) (Cited on page 20)

[83] Galhotra, S., Firmani, D., Saha, B., Srivastava, D.: Efficient and effective er with progressive blocking. The VLDB Journal 30(4), 537–557 (2021) (Cited on page 46)

[84] Galhotra, S., Firmani, D., Saha, B., Srivastava, D.: Hierarchical entity resolution using an oracle. In: Proceedings of the 2022 International Conference on Management of Data. pp. 414–428 (2022) (Cited on page 17)

[85] Gautam, B., Terrades, O.R., Pujadas-Mora, J.M., Valls, M.: Knowledge graph based methods for record linkage. Pattern Recognition Letters 136, 127–133 (2020) (Cited on page 40)

[86] Geng, J., Cai, F., Wang, Y., Koeppl, H., Nakov, P., Gurevych, I.: A survey of confidence estimation and calibration in large language models. In: Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers). pp. 6577–6595. Association for Computational Linguistics, Mexico City, Mexico (Jun 2024), https://aclanthology.org/2024.naacl-long.366 (Cited on page 64)

[87] Getoor, L., Machanavajjhala, A.: Entity resolution: theory, practice & open challenges. Proceedings of the VLDB Endowment 5(12), 2018–2019 (2012) (Cited on page 45)

[88] Goldberg, Y., Levy, O.: word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. arXiv preprint arXiv:1402.3722 (2014) (Cited on page 45)

[89] Gu, Z., Fan, J., Tang, N., Cao, L., Jia, B., Madden, S., Du, X.: Few-shot text-to-sql translation using structure and content prompt learning. PACMMOD 1(2) (2023) (Cited on page 72)

[90] Guo, C., Pleiss, G., Sun, Y., Weinberger, K.Q.: On calibration of modern neural networks. In: Precup, D., Teh, Y.W. (eds.) Proceedings of the 34th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 70, pp. 1321–1330. PMLR (06–11 Aug 2017), https://proceedings.mlr.press/v70/guo17a.html (Cited on page 65)

[91] He, P., Liu, X., Gao, J., Chen, W.: Deberta: Decoding-enhanced bert with disentangled attention. In: International Conference on Learning Representations (Cited on page 7)

[92] Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation 9(8), 1735–1780 (1997) (Cited on pages 6, 15, and 58)

[93] Hoffman, M.D., Gelman, A.: The no-u-turn sampler: Adaptively setting path lengths in hamiltonian monte carlo. Journal of Machine Learning Research 15(47), 1593–1623 (2014), http://jmlr.org/papers/v15/hoffman14a.html (Cited on page 66)

[94] Hou, B., Zhang, Y., Andreas, J., Chang, S.: A probabilistic framework for LLM hallucination detection via belief tree propagation. CoRR abs/2406.06950 (2024), https://doi.org/10.48550/arXiv.2406.06950 (Cited on pages 65 and 66)

[95] Hsieh, C.Y., Chen, S.A., Li, C.L., Fujii, Y., Ratner, A., Lee, C.Y., Krishna, R., Pfister, T.: Tool documentation enables zero-shot tool-usage with large language models. arXiv preprint arXiv:2308.00675 (2023) (Cited on pages 70 and 72)

[96] Huang, W., Abbeel, P., Pathak, D., Mordatch, I.: Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In: ICML (2022) (Cited on page 72)

[97] Huang, Y., Sun, L., Wang, H., Wu, S., Zhang, Q., Li, Y., Gao, C., Huang, Y., Lyu, W., Zhang, Y., Li, X., Sun, H., Liu, Z., Liu, Y., Wang, Y., Zhang, Z., Vidgen, B., Kailkhura, B., Xiong, C., Xiao, C., Li, C., Xing, E.P., Huang, F., Liu, H., Ji, H., Wang, H., Zhang, H., Yao, H., Kellis, M., Zitnik, M., Jiang, M., Bansal, M., Zou, J., Pei, J., Liu, J., Gao, J., Han, J., Zhao, J., Tang, J., Wang, J., Vanschoren, J., Mitchell, J., Shu, K., Xu, K., Chang, K.W., He, L., Huang, L., Backes, M., Gong, N.Z., Yu, P.S., Chen, P.Y., Gu, Q., Xu, R., Ying, R., Ji, S., Jana, S., Chen, T., Liu, T., Zhou, T., Wang, W.Y., Li, X., Zhang, X., Wang, X., Xie, X., Chen, X., Wang, X., Liu, Y., Ye, Y., Cao, Y., Chen, Y., Zhao, Y.: Position: TrustLLM: Trustworthiness in large language models. In: Proceedings of the 41st International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 235, pp. 20166–20270. PMLR (21–27 Jul 2024), https://proceedings.mlr.press/v235/huang24x.html (Cited on page 64)

[98] Hüllermeier, E., Waegeman, W.: Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. Machine learning 110(3), 457–506 (2021) (Cited on page 67)

[99] Ioannidis, Y.E., Christodoulakis, S.: On the propagation of errors in the size of join results. In: Proceedings of the 1991 ACM SIGMOD International Conference on Management of data. pp. 268–277 (1991) (Cited on page 14)

[100] Jain, S., Khangarot, H., Singh, S.: Journal recommendation system using content-based filtering. In: Recent developments in machine learning and data analytics, pp. 99–108. Springer (2019) (Cited on page 45)

[101] Jelodar, H., Wang, Y., Yuan, C., Feng, X., Jiang, X., Li, Y., Zhao, L.: Latent dirichlet allocation (lda) and topic modeling: models, applications, a survey. Multimedia Tools and Applications 78(11), 15169–15211 (2019) (Cited on page 45)

[102] Ji, S., Pan, S., Cambria, E., Marttinen, P., Philip, S.Y.: A survey on knowledge graphs: Representation, acquisition, and applications. IEEE transactions on neural networks and learning systems 33(2), 494–514 (2021) (Cited on pages 38 and 41)

[103] Jiang, A.Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D.S., de Las Casas, D., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., Lavaud, L.R., Lachaux, M., Stock, P., Scao, T.L., Lavril, T., Wang, T., Lacroix, T., Sayed, W.E.: Mistral 7b. CoRR abs/2310.06825 (2023), https://doi.org/10.48550/arXiv.2310.06825 (Cited on pages 63 and 67)

[104] Jiang, J., Zhou, K., Dong, Z., Ye, K., Zhao, W.X., Wen, J.R.: Structgpt: A general framework for large language model to reason over structured data. arXiv preprint arXiv:2305.09645 (2023) (Cited on page 72)

[105] Jiang, K., Wu, D., Jiang, H.: FreebaseQA: A new factoid QA data set matching trivia-style question-answer pairs with Freebase. 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). pp. 318–323. Association for Computational Linguistics, Minneapolis, Minnesota (Jun 2019), https://aclanthology.org/N19-1028 (Cited on pages 64 and 67)

[106] Kadavath, S., Conerly, T., Askell, A., Henighan, T., Drain, D., Perez, E., Schiefer, N., Hatfield-Dodds, Z., DasSarma, N., Tran-Johnson, E., Johnston, S., Showk, S.E., Jones, A., Elhage, N., Hume, T., Chen, A., Bai, Y., Bowman, S., Fort, S., Ganguli, D., Hernandez, D., Jacobson, J., Kernion, J., Kravec, S., Lovitt, L., Ndousse, K., Olsson, C., Ringer, S., Amodei, D., Brown, T., Clark, J., Joseph, N., Mann, B., McCandlish, S., Olah, C., Kaplan, J.: Language models (mostly) know what they know. CoRR abs/2207.05221 (2022), https://doi.org/10.48550/arXiv.2207.05221 (Cited on pages 64, 66, 67, and 68)

[107] Kamienski, A., Hindle, A., Bezemer, C.P.: Analyzing techniques for duplicate question detection on q&a websites for game developers. Empirical Software Engineering 28(1), 17 (2023) (Cited on page 58)

[108] Kannan, A.V., Fradkin, D., Akrotirianakis, I., Kulahcioglu, T., Canedo, A., Roy, A., Yu, S.Y., Arnav, M., Al Faruque, M.A.: Multimodal knowledge graph for deep learning papers and code. In: Proceedings of the 29th ACM International Conference on Information & Knowledge Management. pp. 3417–3420 (2020) (Cited on page 41)

[109] Kanwal, S., Nawaz, S., Malik, M.K., Nawaz, Z.: A review of text-based recommendation systems. IEEE Access 9, 31638–31661 (2021) (Cited on page 45)

[110] Katsogiannis-Meimarakis, G., Koutrika, G.: A survey on deep learning approaches for text-to-sql. VLDB J. 32(4), 905–936 (2023) (Cited on page 72)

[111] Kendall, M.G.: A new measure of rank correlation. Biometrika 30(1/2), 81–93 (1938) (Cited on pages 85, 89, and 90)

[112] Khan, A., Hughes, J., Valentine, D., Ruis, L., Sachan, K., Radhakrishnan, A., Grefenstette, E., Bowman, S.R., Rocktäschel, T., Perez, E.: Debating with more persuasive llms leads to more truthful answers. arXiv preprint arXiv:2402.06782 (2024) (Cited on page 8)

[113] Khan, J.Y., Uddin, G.: Automatic code documentation generation using gpt-3. In: ASE (2022) (Cited on page 81)

[114] Khashabi, D., Min, S., Khot, T., Sabharwal, A., Tafjord, O., Clark, P., Hajishirzi, H.: Unifiedqa: Crossing format boundaries with a single qa system. arXiv preprint arXiv:2005.00700 (2020) (Cited on page 10)

[115] Khashabi, D., Min, S., Khot, T., Sabharwal, A., Tafjord, O., Clark, P., Hajishirzi, H.: UNIFIEDQA: Crossing format boundaries with a single QA system. In: Findings of the Association for Computational Linguistics: EMNLP 2020. pp. 1896–1907. Association for Computational Linguistics, Online (Nov 2020), https://aclanthology.org/2020.findings-emnlp.171 (Cited on page 45)

[116] Khetan, A., Shah, H., Oh, S.: Number of connected components in a graph: Estimation via counting patterns. arXiv preprint arXiv:1812.00139 (2018) (Cited on page 15)

[117] Kirkpatrick, K.: Battling algorithmic bias: how do we ensure algorithms treat us fairly? Commun. ACM 59, 16–17 (2016) (Cited on page 84)

[118] Klusowski, J.M., Wu, Y.: Estimating the number of connected components in a graph via subgraph sampling. Bernoulli 26(3), 1635 – 1664 (2020), https://doi.org/10.3150/19-BEJ1147 (Cited on pages 15, 19, and 23)

[119] Ko, M., Park, S.H., Park, J., Seo, M.: Investigating how large language models leverage internal knowledge to perform complex reasoning. CoRR abs/2406.19502 (2024), https://doi.org/10.48550/arXiv.2406.19502 (Cited on page 64)

[120] Kong, A., Zhao, S., Chen, H., Li, Q., Qin, Y., Sun, R., Zhou, X., Wang, E., Dong, X.: Better zero-shot reasoning with role-play prompting. arXiv preprint arXiv:2308.07702 (2023) (Cited on page 8)

[121] Kong, Y., Li, Y., Zhang, Y., Huang, Z., Wu, J.: Eliciting thinking hierarchy without a prior. In: Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., Oh, A. (eds.) Advances in Neural Information Processing Systems. vol. 35, pp. 13329–13341. Curran Associates, Inc. (2022), https://proceedings.neurips.cc/paper_files/paper/2022/file/56d7585405a534b3af91905650ce7f9e-Paper-Conference.pdf (Cited on pages 65 and 68)

[122] Köpcke, H., Thor, A., Rahm, E.: Evaluation of entity resolution approaches on real-world match problems. Proceedings of the VLDB Endowment 3(1-2), 484–493 (2010) (Cited on page 24)

[123] Koutras, C., Siachamis, G., Ionescu, A., Psarakis, K., Brons, J., Fragkoulis, M., Lofi, C., Bonifati, A., Katsifodimos, A.: Valentine: Evaluating matching techniques for dataset discovery. In: ICDE. pp. 468–479. IEEE (2021) (Cited on page 78)

[124] Kuhlman, C., VanValkenburg, M., Rundensteiner, E.: Fare: Diagnostics for fair ranking using pairwise error metrics. In: The world wide web conference. pp. 2936–2942 (2019) (Cited on page 90)

[125] Kuhn, L., Gal, Y., Farquhar, S.: Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. In: The Eleventh International Conference on Learning Representations (2023), https://openreview.net/forum?id=VD-AYtP0dve (Cited on pages 64, 65, 66, 67, and 68)

[126] Kumar, A.P., Nayak, A., Ghosh, K., et al.: A novel framework for the generation of multiple choice question stems using semantic and machine-learning techniques. International Journal of Artificial Intelligence in Education pp. 1–44 (2023) (Cited on page 45)

[127] Leotta, F., Mathew, J.G., Mecella, M., Monti, F.: Supporting zero defect manufacturing through cloud computing and data analytics: The case study of electrospindle 4.0. In: International Conference on Advanced Information Systems Engineering. pp. 119–125. Springer (2022) (Cited on page xiii)

[128] Lewis, M.: Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. arXiv preprint arXiv:1910.13461 (2019) (Cited on page 7)

[129] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.t., Rocktäschel, T., et al.: Retrieval-augmented generation for knowledge-intensive nlp tasks. NeurIPS 2020 (2020) (Cited on page 10)

[130] Li, J.: Crowdsourced text sequence aggregation based on hybrid reliability and representation. In: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval. p. 1761–1764. SIGIR '20, Association for Computing Machinery, New York, NY, USA (2020), https://doi.org/10.1145/3397271.3401239 (Cited on page 65)

[131] Li, J., Zhang, Q., Yu, Y., Fu, Q., Ye, D.: More agents is all you need. CoRR abs/2402.05120 (2024), https://doi.org/10.48550/arXiv.2402.05120 (Cited on pages 65 and 68)

[132] Li, P., Dong, X.L., Maurino, A., Srivastava, D.: Linking temporal records. Proceedings of the VLDB Endowment 4(11), 956–967 (2011) (Cited on page 41)

[133] Li, Q., Li, Y., Gao, J., Su, L., Zhao, B., Demirbas, M., Fan, W., Han, J.: A confidence-aware approach for truth discovery on long-tail data. Proc. VLDB Endow. 8(4), 425–436 (dec 2014), https://doi.org/10.14778/2735496.2735505 (Cited on page 64)

[134] Li, Y., Yao, L., Du, N., Gao, J., Li, Q., Meng, C., Zhang, C., Fan, W.: Finding similar medical questions from question answering websites (2018) (Cited on page 58)

[135] Li, Y., Choi, D., Chung, J., Kushman, N., Schrittwieser, J., Leblond, R., Eccles, T., Keeling, J., Gimeno, F., Dal Lago, A., et al.: Competition-level code generation with alphacode. Science 378(6624), 1092–1097 (2022) (Cited on page 72)

[136] Li, Y., Feng, A., Li, J., Mumick, S., Halevy, A., Li, V., Tan, W.C.: Subjective databases. Proceedings of the VLDB Endowment 12(11), 1330–1343 (2019) (Cited on page 84)

[137] Li, Y., Li, J., Suhara, Y., Doan, A., Tan, W.C.: Deep entity matching with pre-trained language models. Proceedings of the VLDB Endowment 14(1), 50–60 (2020) (Cited on pages 9, 13, 15, 20, 35, and 38)

[138] Lin, Z., Trivedi, S., Sun, J.: Generating with confidence: Uncertainty quantification for black-box large language models. Transactions on Machine Learning Research (2024), https://openreview.net/forum?id=DWkJCSxKU5 (Cited on pages 64, 65, 66, 67, and 68)

[139] Liu, H., Sferrazza, C., Abbeel, P.: Chain of hindsight aligns language models with feedback. arXiv preprint arXiv:2302.02676 (2023) (Cited on page 81)

[140] Liu, Y.: Multilingual denoising pre-training for neural machine translation. arXiv preprint arXiv:2001.08210 (2020) (Cited on page 7)

[141] Liu, Y., Yao, Y., Ton, J., Zhang, X., Guo, R., Cheng, H., Klochkov, Y., Taufiq, M.F., Li, H.: Trustworthy llms: a survey and guideline for evaluating large language models' alignment. CoRR abs/2308.05374 (2023), https://doi.org/10.48550/arXiv.2308.05374 (Cited on page 64)

[142] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V.: Roberta: A robustly optimized bert pretraining approach (2019) (Cited on page 7)

[143] Lo, C., Chen, C.H., Zhong, R.Y.: A review of digital twin in product design and development. Advanced Engineering Informatics 48, 101297 (2021) (Cited on page 81)

[144] Long, P., Siemens, G., Conole, G., Gasevic, D. (eds.): Proceedings of the 1st International Conference on Learning Analytics and Knowledge, LAK 2011, Banff, AB, Canada, February 27 - March 01, 2011. ACM (2011) (Cited on page 44)

[145] Ma, P., Ding, R., Wang, S., Han, S., Zhang, D.: Insightpilot: An llm-empowered automated data exploration system. In: EMNLP. pp. 346–352 (2023) (Cited on page 72)

[146] Maccioni, A., Torlone, R.: Kayak: a framework for just-in-time data preparation in a data lake. In: Advanced Information Systems Engineering: 30th International Conference, CAiSE 2018, Tallinn, Estonia, June 11-15, 2018, Proceedings 30. pp. 474–489. Springer (2018) (Cited on page 38)

[147] Madaan, A., Tandon, N., Gupta, P., Hallinan, S., Gao, L., Wiegreffe, S., Alon, U., Dziri, N., Prabhumoye, S., Yang, Y., Gupta, S., Majumder, B.P., Hermann, K., Welleck, S., Yazdanbakhsh, A., Clark, P.: Self-refine: Iterative refinement with self-feedback. In: Advances in Neural Information Processing Systems. vol. 36, pp. 46534–46594. Curran Associates, Inc. (2023), https://proceedings.neurips.cc/paper_files/paper/2023/file/91edff07232fb1b55a505a9e9f6c0ff3-Paper-Conference.pdf (Cited on page 64)

[148] Mallen, A., Asai, A., Zhong, V., Das, R., Khashabi, D., Hajishirzi, H.: When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In: Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 9802–9822. Association for Computational Linguistics, Toronto, Canada (Jul 2023), https://aclanthology.org/2023.acl-long.546 (Cited on page 64)

[149] Marchant, N.G., Kaplan, A., Elazar, D.N., Rubinstein, B.I., Steorts, R.C.: d-blink: Distributed end-to-end bayesian entity resolution. Journal of Computational and Graphical Statistics 30(2), 406–421 (2021) (Cited on pages 13, 15, 16, 20, 22, 36, and 98)

[150] Martinez-Gil, J., Freudenthaler, B., Tjoa, A.M.: Multiple choice question answering in the legal domain using reinforced co-occurrence. In: Interna-

tional Conference on Database and Expert Systems Applications. pp. 138–148. Springer (2019) (Cited on page 45)

[151] Mathew, J.G., Monti, F., Firmani, D., Leotta, F., Mandreoli, F., Mecella, M.: Composing smart data services in shop floors through large language models. In: 22nd International Conference on Service-Oriented Computing (ICSOC 2024). Tunis, Tunisia (2024), accepted, proceedings forthcoming (Cited on pages x and xi)

[152] MESTS, J., Tang, M.: Distributed representations of tuples for entity resolution. Proceedings of the VLDB Endowment 11(11) (2018) (Cited on page 9)

[153] Mia, M.R., Latiful Hoque, A.S.M.: Question bank similarity searching system (qb3s) using nlp and information retrieval technique. In: 2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT). pp. 1–7 (2019) (Cited on page 59)

[154] Mialon, G., Dessì, R., Lomeli, M., Nalmpantis, C., Pasunuru, R., Raileanu, R., Rozière, B., Schick, T., Dwivedi-Yu, J., Celikyilmaz, A., et al.: Augmented language models: a survey. arXiv preprint arXiv:2302.07842 (2023) (Cited on page 72)

[155] Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013) (Cited on page 3)

[156] Min, S., Michael, J., Hajishirzi, H., Zettlemoyer, L.: AmbigQA: Answering ambiguous open-domain questions. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 5783–5797. Association for Computational Linguistics, Online (Nov 2020), https://aclanthology.org/2020.emnlp-main.466 (Cited on page 67)

[157] Mitkov, R., Le An, H., Karamanis, N.: A computer-aided environment for generating multiple-choice test items. Natural language engineering 12(2), 177–194 (2006) (Cited on page 44)

[158] (MONICA.), C.S.B.: DATA AND INFORMATION QUALITY: Dimensions, Principles and Techniques. SPRINGER (2018) (Cited on page vi)

[159] Monti, F., Leotta, F., Mangler, J., Mecella, M., Rinderle-Ma, S.: Nl2processops: Towards llm-guided code generation for process execution. In: BPM. Springer (2024) (Cited on page 73)

[160] Monti, F., Mathew, J.G., Leotta, F., Koschmider, A., Mecella, M.: On the application of process management and process mining to industry 4.0. Software and Systems Modeling pp. 1–13 (2024) (Cited on page xiii)

[161] Mousselly-Sergieh, H., Botschen, T., Gurevych, I., Roth, S.: A multimodal translation-based approach for knowledge graph representation learning. In: Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics. pp. 225–234 (2018) (Cited on page 41)

[162] Mudgal, S., Li, H., Rekatsinas, T., Doan, A., Park, Y., Krishnan, G., Deep, R., Arcaute, E., Raghavendra, V.: Deep learning for entity matching: A design space exploration. In: Proceedings of the 2018 International Conference on Management of Data, SIGMOD Conference 2018, Houston, TX, USA, June 10-15, 2018. pp. 19–34. ACM (2018), https://doi.org/10.1145/3183713.3196926 (Cited on pages 9 and 46)

[163] Mukherjee, S., Kumar, N.S.: Duplicate question management and answer verification system. In: 2019 IEEE Tenth International Conference on Technology for Education (T4E). pp. 266–267 (2019) (Cited on page 59)

[164] Müller, M., Flachs, D., Moerkotte, G.: Memory-efficient key/foreign-key join size estimation via multiplicity and intersection size. In: 2021 IEEE 37th International Conference on Data Engineering (ICDE). pp. 984–995. IEEE (2021) (Cited on page 12)

[165] Neelakantan, A., Xu, T., Puri, R., Radford, A., Han, J.M., Tworek, J., Yuan, Q., Tezak, N., Kim, J.W., Hallacy, C., et al.: Text and code embeddings by contrastive pre-training. arXiv preprint arXiv:2201.10005 (2022) (Cited on page 20)

[166] Nijkamp, E., Pang, B., Hayashi, H., Tu, L., Wang, H., Zhou, Y., Savarese, S., Xiong, C.: Codegen: An open large language model for code with multi-turn program synthesis. arXiv preprint arXiv:2203.13474 (2022) (Cited on page 72)

[167] Obraczka, D., Schuchart, J., Rahm, E.: Embedding-assisted entity resolution for knowledge graphs. In: Second International Workshop on Knowledge Graph Construction (2021) (Cited on page 40)

[168] Oinn, T., Addis, M., Ferris, J., Marvin, D., Senger, M., Greenwood, M., Carver, T., Glover, K., Pocock, M.R., Wipat, A., Li, P.: Taverna: a tool for the composition and enactment of bioinformatics workflows. Bioinformatics 20(17), 3045–3054 (2004) (Cited on page 74)

[169] OpenAI: New embedding models and API updates (Jan 2024), https://openai.com/blog/new-embedding-models-and-api-updates, last accessed 2024-07-18 (Cited on page 28)

[170] Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P.F., Leike, J., Lowe, R.: Training language models to follow instructions with human feedback. In: Advances in Neural Information Processing Systems. vol. 35, pp. 27730–27744. Curran Associates, Inc. (2022), https://proceedings.neurips.cc/paper_files/paper/2022/file/b1efde53be364a73914f58805a001731-Paper-Conference.pdf (Cited on pages 63 and 67)

[171] Papadakis, G., Ioannou, E., Thanos, E., Palpanas, T.: The four generations of entity resolution. Springer (2021) (Cited on pages 6 and 35)

[172] Papadakis, G., Koutrika, G., Palpanas, T., Nejdl, W.: Meta-blocking: Taking entity resolutionto the next level. IEEE Transactions on Knowledge and Data Engineering 26(8), 1946–1960 (2013) (Cited on pages vii, 21, and 35)

[173] Papasalouros, A., Kanaris, K., Kotis, K.: Automatic generation of multiple choice questions from domain ontologies. e-Learning 1, 427–434 (2008) (Cited on page 45)

[174] Papenbrock, T., Heise, A., Naumann, F.: Progressive duplicate detection. IEEE Transactions on knowledge and data engineering 27(5), 1316–1329 (2014) (Cited on page 13)

[175] Parisi, A., Zhao, Y., Fiedel, N.: Talm: Tool augmented language models. arXiv preprint arXiv:2205.12255 (2022) (Cited on page 72)

[176] Passonneau, R.J., Carpenter, B.: The benefits of a model of annotation. Transactions of the Association for Computational Linguistics 2, 311–326 (2014), https://aclanthology.org/Q14-1025 (Cited on pages 64 and 65)

[177] Patil, S.G., Zhang, T., Wang, X., Gonzalez, J.E.: Gorilla: Large language model connected with massive apis. arXiv preprint arXiv:2305.15334 (2023) (Cited on page 72)

[178] Paun, S., Carpenter, B., Chamberlain, J., Hovy, D., Kruschwitz, U., Poesio, M.: Comparing Bayesian models of annotation. Transactions of the Association for Computational Linguistics 6, 571–585 (2018), https://aclanthology.org/Q18-1040 (Cited on pages 64 and 65)

[179] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research 12, 2825–2830 (2011) (Cited on page 47)

[180] Peeters, R., Bizer, C.: Entity matching using large language models. arXiv preprint arXiv:2310.11244 (2023) (Cited on page 9)

[181] Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). pp. 1532–1543 (2014) (Cited on pages 3 and 15)

[182] Pesl, R.D., Stötzner, M., Georgievski, I., Aiello, M.: Uncovering llms for service-composition: Challenges and opportunities. In: ICSOC. Springer (2023) (Cited on page 73)

[183] Petroni, F., Rocktäschel, T., Riedel, S., Lewis, P., Bakhtin, A., Wu, Y., Miller, A.: Language models as knowledge bases? 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). pp. 2463–2473. Association for Computational Linguistics, Hong

Kong, China (Nov 2019), https://aclanthology.org/D19-1250 (Cited on page 64)

[184] Phan, D., Pradhan, N., Jankowiak, M.: Composable effects for flexible and accelerated probabilistic programming in numpyro. arXiv preprint arXiv:1912.11554 (2019) (Cited on page 66)

[185] Pipino, L.L., Lee, Y.W., Wang, R.Y.: Data quality assessment. Communications of the ACM 45(4), 211–218 (2002) (Cited on page vi)

[186] Pitoura, E., Stefanidis, K., Koutrika, G.: Fairness in rankings and recommendations: an overview. The VLDB Journal pp. 1–28 (2021) (Cited on page 88)

[187] Pujara, J., Getoor, L.: Generic statistical relational entity resolution in knowledge graphs. arXiv preprint arXiv:1607.00992 (2016) (Cited on page 40)

[188] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P.J.: Exploring the limits of transfer learning with a unified text-to-text transformer. Journal of machine learning research 21(140), 1–67 (2020) (Cited on pages 7 and 10)

[189] Rahm, E., Do, H.H., et al.: Data cleaning: Problems and current approaches. IEEE Data Eng. Bull. 23(4), 3–13 (2000) (Cited on page vi)

[190] Reid, M., Savinov, N., Teplyashin, D., Lepikhin, D., Lillicrap, T., Alayrac, J.b., Soricut, R., Lazaridou, A., Firat, O., Schrittwieser, J., et al.: Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. arXiv preprint arXiv:2403.05530 (2024) (Cited on page 6)

[191] Reimers, N., Gurevych, I.: Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In: Inui, K., Jiang, J., Ng, V., Wan, X. (eds.) Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). pp. 3982–3992. Association for Computational Linguistics, Hong Kong, China (Nov 2019), https://aclanthology.org/D19-1410 (Cited on page 5)

[192] Reimers, N., Gurevych, I.: Sentence-bert: Sentence embeddings using siamese bert-networks. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics (11 2019), http://arxiv.org/abs/1908.10084 (Cited on pages 28, 45, and 48)

[193] Reimers, N., Gurevych, I.: Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In: Inui, K., Jiang, J., Ng, V., Wan, X. (eds.) Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). pp. 3982–3992. Association for Computational Linguistics, Hong Kong, China (Nov 2019), https://aclanthology.org/D19-1410 (Cited on page 66)

[194] Reimers, N., Gurevych, I.: Making monolingual sentence embeddings multilingual using knowledge distillation. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 4512–4525. Association for Computational Linguistics, Online (Nov 2020), https://aclanthology.org/2020.emnlp-main.365 (Cited on pages 48 and 49)

[195] Ridnik, T., Kredo, D., Friedman, I.: Code generation with alphacodium: From prompt engineering to flow engineering. arXiv preprint arXiv:2401.08500 (2024) (Cited on page 72)

[196] Rühling Cachay, S., Boecking, B., Dubrawski, A.: End-to-end weak supervision. In: Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., Vaughan, J.W. (eds.) Advances in Neural Information Processing Systems. vol. 34, pp. 1845–1857. Curran Associates, Inc. (2021), https://proceedings.neurips.cc/paper_files/paper/2021/file/0e674a918ebca3f78bfe02e2f387689d-Paper.pdf (Cited on page 65)

[197] Saeed, M., De Cao, N., Papotti, P.: Querying large language models with sql. arXiv preprint arXiv:2304.00472 (2023) (Cited on page 72)

[198] Saeedi, A., Peukert, E., Rahm, E.: Comparative evaluation of distributed clustering schemes for multi-source entity resolution. In: Advances in Databases and Information Systems: 21st European Conference, ADBIS 2017, Nicosia, Cyprus, September 24-27, 2017, Proceedings 21. pp. 278–293. Springer (2017) (Cited on pages 13 and 16)

[199] Saeedi, A., Peukert, E., Rahm, E.: Incremental multi-source entity resolution for knowledge graph completion. In: The Semantic Web: 17th International Conference, ESWC 2020, Heraklion, Crete, Greece, May 31–June 4, 2020, Proceedings 17. pp. 393–408. Springer (2020) (Cited on page 39)

[200] Sakoe, H., Chiba, S.: Dynamic programming algorithm optimization for spoken word recognition. IEEE transactions on acoustics, speech, and signal processing 26(1), 43–49 (1978) (Cited on page 5)

[201] Salloum, S.A., Khan, R., Shaalan, K.: A survey of semantic analysis approaches. In: The International Conference on Artificial Intelligence and Computer Vision. pp. 61–70. Springer (2020) (Cited on page 45)

[202] Samarinas, C., Zafeiriou, S.: Personalized high quality news recommendations using word embeddings and text classification models. EasyChair Preprint 1254, 2019 (2019) (Cited on page 45)

[203] Sarkhel, R., Nandi, A.: Cross-modal entity matching for visually rich documents. arXiv preprint arXiv:2303.00720 (2023) (Cited on page 41)

[204] Schütze, H., Manning, C.D., Raghavan, P.: Introduction to information retrieval, vol. 39. Cambridge University Press Cambridge (2008) (Cited on page 3)

[205] Seo, M.: Bidirectional attention flow for machine comprehension. arXiv preprint arXiv:1611.01603 (2016) (Cited on page 10)

[206] Shinn, N., Cassano, F., Gopinath, A., Narasimhan, K., Yao, S.: Reflexion: Language agents with verbal reinforcement learning. Advances in Neural Information Processing Systems 36 (2024) (Cited on pages 9 and 81)

[207] Singh, A., Joachims, T.: Fairness of exposure in rankings. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. pp. 2219–2228 (2018) (Cited on pages 85, 86, 88, 89, and 90)

[208] Singh, A., Joachims, T.: Policy learning for fairness in ranking. Advances in neural information processing systems 32 (2019) (Cited on page 87)

[209] Snow, R., O'Connor, B., Jurafsky, D., Ng, A.: Cheap and fast – but is it good? evaluating non-expert annotations for natural language tasks. In: Lapata, M., Ng, H.T. (eds.) Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing. pp. 254–263. Association for Computational Linguistics, Honolulu, Hawaii (Oct 2008), https://aclanthology.org/D08-1027 (Cited on page 64)

[210] Song, K., Tan, X., Qin, T., Lu, J., Liu, T.Y.: Mpnet: Masked and permuted pre-training for language understanding. Advances in Neural Information Processing Systems 33, 16857–16867 (2020) (Cited on pages 28, 49, and 58)

[211] Steorts, R.C.: Entity Resolution with Empirically Motivated Priors. Bayesian Analysis 10(4), 849 – 875 (2015), https://doi.org/10.1214/15-BA965SI (Cited on pages 13, 15, 20, 22, 28, and 30)

[212] Steorts, R.C.: Entity resolution with empirically motivated priors. Bayesian Analysis 10(4), 849–875 (2015) (Cited on page 38)

[213] Stoyanovich, J., Abiteboul, S., Miklau, G.: Data, responsibly: Fairness, neutrality and transparency in data analysis. In: International Conference on Extending Database Technology (2016) (Cited on pages vi and 86)

[214] Sun, K., Xu, Y., Zha, H., Liu, Y., Dong, X.L.: Head-to-tail: How knowledgeable are large language models (LLMs)? A.K.A. will LLMs replace knowledge graphs? In: Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers). pp. 311–325. Association for Computational Linguistics, Mexico City, Mexico (Jun 2024), https://aclanthology.org/2024.naacl-long.18 (Cited on page 67)

[215] Sun, Z., Vashishth, S., Sanyal, S., Talukdar, P., Yang, Y.: A re-evaluation of knowledge graph completion methods. In: ACL. pp. 5516–5522 (2020) (Cited on page 40)

[216] Tan, W.: Unleashing the power of subjective data: Managing experiences as first-class citizens. In: KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020. p. 3610 (2020) (Cited on page 84)

[217] Tancredi, A., Rebecca, S., Liseo, B., et al.: A unified framework for de-duplication and population size estimation (with discussion). BAYESIAN ANALYSIS 15(2), 633–658 (2020) (Cited on page 13)

[218] Tarrant, M., Ware, J.: A framework for improving the quality of multiple-choice assessments. Nurse Educator 37(3), 98–104 (2012) (Cited on page 45)

[219] Team, G.: Gemma 2: Improving open language models at a practical size (2024), https://arxiv.org/abs/2408.00118 (Cited on pages 63 and 67)

[220] Teofili, T., Firmani, D., Koudas, N., Martello, V., Merialdo, P., Srivastava, D.: Effective explanations for entity resolution models. In: 2022 IEEE 38th International Conference on Data Engineering (ICDE). pp. 2709–2721. IEEE (2022) (Cited on page 42)

[221] Thirumuruganathan, S., Li, H., Tang, N., Ouzzani, M., Govind, Y., Paulsen, D., Fung, G., Doan, A.: Deep learning for blocking in entity matching: a design space exploration. Proceedings of the VLDB Endowment 14(11), 2459–2472 (2021) (Cited on pages 13 and 15)

[222] Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al.: Llama: Open and efficient foundation language models. arXiv preprint arXiv:2302.13971 (2023) (Cited on page 71)

[223] Tramèr, F., Atlidakis, V., Geambasu, R., Hsu, D.J., Hubaux, J., Humbert, M., Juels, A., Lin, H.: Discovering unwarranted associations in data-driven applications with the fairtest testing toolkit. CoRR abs/1510.02377 (2015), http://arxiv.org/abs/1510.02377 (Cited on page 84)

[224] Ustalov, D., Pavlichenko, N., Tseitlin, B.: Learning from crowds with crowd-kit. Journal of Open Source Software 9(96), 6227 (2024), https://doi.org/10.21105/joss.06227 (Cited on page 65)

[225] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems 30 (2017) (Cited on pages 6 and 7)

[226] Vengerov, D., Menck, A.C., Zait, M., Chakkappen, S.P.: Join size estimation subject to filter conditions. Proceedings of the VLDB Endowment 8(12), 1530–1541 (2015) (Cited on page 14)

[227] Vesdapunt, N., Bellare, K., Dalvi, N.: Crowdsourcing algorithms for entity resolution. Proceedings of the VLDB Endowment 7(12), 1071–1082 (2014) (Cited on page 13)

[228] Wan, F., Huang, X., Cai, D., Quan, X., Bi, W., Shi, S.: Knowledge fusion of large language models. In: The Twelfth International Conference on Learning Representations (2024), https://openreview.net/forum?id=jiDsk12qcz (Cited on page 64)

[229] Wang, B., Fang, H., Eisner, J., Van Durme, B., Su, Y.: Llms in the imaginarium: tool learning through simulated trial and error. arXiv preprint arXiv:2403.04746 (2024) (Cited on page 81)

[230] Wang, H., Polo, F.M., Sun, Y., Kundu, S., Xing, E., Yurochkin, M.: Fusing models with complementary expertise. In: The Twelfth International Conference on Learning Representations (2024), https://openreview.net/forum?id=PhMrGCMIRL (Cited on page 64)

[231] Wang, L., Xu, W., Lan, Y., Hu, Z., Lan, Y., Lee, R.K.W., Lim, E.P.: Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. arXiv preprint arXiv:2305.04091 (2023) (Cited on page 81)

[232] Wang, W., Wei, F., Dong, L., Bao, H., Yang, N., Zhou, M.: Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers (2020) (Cited on page 48)

[233] Wang, X., Wei, J., Schuurmans, D., Le, Q.V., Chi, E.H., Narang, S., Chowdhery, A., Zhou, D.: Self-consistency improves chain of thought reasoning in language models. In: The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023. OpenReview.net (2023), https://openreview.net/forum?id=1PL1NIMMrw (Cited on pages 64 and 65)

[234] Wang, Y., Ma, W., Zhang, M., Liu, Y., Ma, S.: A survey on the fairness of recommender systems. ACM Transactions on Information Systems 41(3), 1–43 (2023) (Cited on pages 85 and 86)

[235] Wang, Z., Zhang, H., Li, C.L., Eisenschlos, J.M., Perot, V., Wang, Z., Miculicich, L., Fujii, Y., Shang, J., Lee, C.Y., Pfister, T.: Chain-of-table: Evolving tables in the reasoning chain for table understanding. ICLR (2024) (Cited on page 72)

[236] Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q.V., Zhou, D., et al.: Chain-of-thought prompting elicits reasoning in large language models. Advances in neural information processing systems 35, 24824–24837 (2022) (Cited on pages 8 and 81)

[237] White, J., Fu, Q., Hays, S., Sandborn, M., Olea, C., Gilbert, H., Elnashar, A., Spencer-Smith, J., Schmidt, D.C.: A prompt pattern catalog to enhance prompt engineering with chatgpt. arXiv preprint arXiv:2302.11382 (2023) (Cited on page 77)

[238] Whitehill, J., Wu, T.f., Bergsma, J., Movellan, J., Ruvolo, P.: Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In: Bengio, Y., Schuurmans, D., Lafferty, J., Williams, C., Culotta, A. (eds.) Advances in Neural Information Processing Systems. vol. 22. Curran Associates, Inc. (2009), https://proceedings.neurips.cc/paper_files/paper/2009/file/f899139df5e1059396431415e770c6dd-Paper.pdf (Cited on page 64)

[239] Winkler, W.E.: Matching and record linkage. Wiley interdisciplinary reviews: Computational statistics 6(5), 313–325 (2014) (Cited on page 12)

[240] Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., et al.: Transformers: State-of-the-art natural language processing. In: Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations. pp. 38–45 (2020) (Cited on page 45)

[241] Wood, T.J.: The effect of reused questions on repeat examinees. Advances in Health Sciences Education 14, 465–473 (2009) (Cited on page 45)

[242] Wu, R., Chaba, S., Sawlani, S., Chu, X., Thirumuruganathan, S.: Zeroer: Entity resolution using zero labeled examples. In: Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data. pp. 1149–1164 (2020) (Cited on pages 15 and 98)

[243] Yadav, H., Du, Z., Joachims, T.: Fair learning-to-rank from implicit feedback. In: SIGIR (2020) (Cited on page 87)

[244] Yaneva, V., et al.: Automatic distractor suggestion for multiple-choice tests using concept embeddings and information retrieval. In: Proceedings of the thirteenth workshop on innovative use of NLP for building educational applications. pp. 389–398 (2018) (Cited on page 45)

[245] Yang, J., McAuley, J., Leskovec, J.: Community detection in networks with node attributes. In: 2013 IEEE 13th international conference on data mining. pp. 1151–1156. IEEE (2013) (Cited on page 6)

[246] Yang, K., Stoyanovich, J.: Measuring fairness in ranked outputs. In: SSDM. p. 22 (2017) (Cited on page 84)

[247] Yang, Y., Cer, D., Ahmad, A., Guo, M., Law, J., Constant, N., Abrego, G.H., Yuan, S., Tar, C., Sung, Y.H., et al.: Multilingual universal sentence encoder for semantic retrieval. arXiv preprint arXiv:1907.04307 (2019) (Cited on page 48)

[248] Yao, P., Mathew, J.G., Singh, S., Firmani, D., Barbosa, D.: A bayesian approach towards crowdsourcing the truths from llms. In: NeurIPS Workshop on Bayesian Decision-making and Uncertainty. Vancouver, British Columbia, Canada (2024), accepted, proceedings forthcoming (Cited on pages ix, xi, and xii)

[249] Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T., Cao, Y., Narasimhan, K.: Tree of thoughts: Deliberate problem solving with large language models. Advances in Neural Information Processing Systems 36 (2024) (Cited on page 8)

[250] Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K.R., Cao, Y.: React: Synergizing reasoning and acting in language models. In: ICLR (2022) (Cited on pages 9 and 81)

[251] Yilmaz, G., Qurban, K., Kaiser, J., McFarlane, D.: Cost-effective digital transformation of smes through low-cost digital solutions. LoDiSA (2023) (Cited on page 70)

[252] Yoo, S., Huang, H., Kasiviswanathan, S.P.: Streaming spectral clustering. In: 2016 IEEE 32nd international conference on data engineering (ICDE). pp. 637–648. IEEE (2016) (Cited on page 15)

[253] Zehlike, M., Bonchi, F., Castillo, C., Hajian, S., Megahed, M., Baeza-Yates, R.: Fa* ir: A fair top-k ranking algorithm. In: CIKM. pp. 1569–1578 (2017) (Cited on page 84)

[254] Zehlike, M., Yang, K., Stoyanovich, J.: Fairness in ranking, part i: Score-based ranking. ACM Computing Surveys (CSUR) (2022) (Cited on pages 86 and 88)

[255] Zehlike, M., Yang, K., Stoyanovich, J.: Fairness in ranking, part ii: Learning-to-rank and recommender systems. ACM Computing Surveys (CSUR) (2022) (Cited on pages 86 and 88)

[256] Zhang, T., Ramakrishnan, R., Livny, M.: Birch: an efficient data clustering method for very large databases. ACM sigmod record 25(2), 103–114 (1996) (Cited on page 15)

[257] Zhang, Y., Li, Y., Cui, L., Cai, D., Liu, L., Fu, T., Huang, X., Zhao, E., Zhang, Y., Chen, Y., et al.: Siren's song in the ai ocean: a survey on hallucination in large language models. arXiv preprint arXiv:2309.01219 (2023) (Cited on page 64)

[258] Zhao, H., Chen, H., Yang, F., Liu, N., Deng, H., Cai, H., Wang, S., Yin, D., Du, M.: Explainability for large language models: A survey. ACM TIST (2023) (Cited on page 71)

[259] Zhao, W.X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., et al.: A survey of large language models. arXiv preprint arXiv:2303.18223 (2023) (Cited on page 77)

[260] Zheng, Y., Li, G., Li, Y., Shan, C., Cheng, R.: Truth inference in crowdsourcing: Is the problem solved? Proc. VLDB Endow. 10(5), 541–552 (2017), http://www.vldb.org/pvldb/vol10/p541-zheng.pdf (Cited on pages 64 and 65)