DataBlend: Efficient Discovery of Semantically Similar Tables in Data Lakes

Bachelor Thesis

Johannes Gesk

Supervisors:

Alexander Vielhauer
Prof. Dr. Thorsten Papenbrock

12.06.2024

Abstract

The development of advanced algorithms for dynamic data lakes has been prompted by the demand for efficient data management. This thesis introduces the DataBlend, which combines Latent Dirichlet Allocation, word embeddings, and Random Hyperplane Locality Sensitive Hashing, to address the problem of extracting information from dynamic data lakes. In an evaluation using the TREC and Common Web Table datasets, the DataBlend framework, which uses a Lambda architecture with Batch, Serving, and Speed Layers—showed notable gains in document ranking based on semantic similarity. The results reveal that DataBlend significantly improves data retrieval efficiency, providing a scalable solution for data lake applications. Future work will further refine the integration algorithm and exploring its potential applications.

Contents

1 Introduction

With an increasing amount of data and its growing complexity, organisations arein search of advanced data management solutions. Data lakes have become an effective solution for storing large amounts of unstructured data, providing the flexibility needed for modern data management. In contrast to conventional data warehouses, data lakes have the advantage of accepting data in its original format, making it suitable for a variety of data types and sources. Flexibility is essential in order to meet the dynamic data requirements of today's businesses and research.

There are still several challenges with data lakes that need to be resolved.. One of the main challenges is the attempt to find relevant datasets in a vast and unstructured repository. According to Nargesian et al. (2019)[20], it can be difficult for researchers to find and use datasets effectively due to the lack of comprehensive metadata in data lakes. In addition, there are significant challenges in integrating different data sources, ensuring data cleanliness, and maintaining data accuracy. These challenges often lead to data lakes becoming overwhelmed with data, making it challenging to effectively utilise and manage the stored information.

This thesis presents DataBlend, an innovative framework developed to tackle the main obstacles of dataset exploration and semantic similarity in data lakes. Through the utilisation of algorithms and a strong architectural design, DataBlend aims to enhance the efficiency and precision of data retrieval, thereby increasing the practical use of data lakes for organisations. DataBlend provides a scalable solution for efficiently operating and utilising the large amount of information stored in data lakes. By including techniques like Latent Dirichlet Allocation, word embeddings, and Locality Sensitive Hashing, DataBlend offers an advanced approach.

Through evaluations conducted using standard datasets, it has been shown that DataBlend greatly improves the process of discovering and merging tables that share similar semantic aspects. This thesis introduces a new approach to data lake difficulties and lays the foundation for future research and development in integrating data in data lakes.

In the upcoming sections, we will explore the conceptual foundations of data lakes, review previous research in the field, and conduct a evaluation of the DataBlend framework.

2 Preliminaries

In the Preliminaries section of this thesis, we will examine the fundamental concepts and approaches that are crucial for comprehending the DataBlend framework. We will start by giving a comprehensive explanation of data management structures, with a specific emphasis on data lakes and data lakehouses. Next, we will explore the fundamental algorithms that are essential to DataBlend, including Latent Dirichlet Allocation, word embeddings, Random Hyperplane Locality Sensitive Hashing and approximate k nearest neighbour search. These algorithms play a crucial role in extracting and capturing the meaning and context of information from documents. .

2.1 Data Management Architectures

In this thesis we will talk about *Information Retrival* and *Table Discovery* of publications the name of information retrival and table. Therefore will shortly define both terms and their connection

Information retrieval (IR) is the process of using user queries to extract pertinent information from massive datasets. In Introduction to Information Retrieval, Schütze [15]states that in order to improve search relevancy and accuracy, IR systems employ methods including machine learning, natural language processing, and keyword matching.

Table discovery is the process of finding and obtaining pertinent tables from data lakes. In order to comprehend the semantic content of tables, this procedure makes use of sophisticated techniques like word embeddings and Latent Dirichlet Allocation (LDA) for topic modelling.

Table discovery is essentially a subset of information retrieval (IR) in which tables are handled as structured documents. This viewpoint makes it possible to use IR principles to table retrieval, taking advantage of the organised form of the data to increase relevancy and accuracy. Because table discovery is a subset of information retrieval (IR), we can talk about data retrieval generally when we refer to "documents," and then talk specifically about "tables." This abstraction aids in keeping things clear as we go on to the particular difficulties with table finding.

Data Lake

James Dixon's 2010 blog article presents the notion of a "Data Lake" as a cutting-edge approach to managing substantial amounts of data [7]. Data lakes distinguish themselves from conventional data marts by storing data in its raw and unorganised state, allowing for more flexible and comprehensive analysis. This solution effectively addresses the limitations of pre-aggregated data marts, enabling users to thoroughly analyse and explore data at a granular level. Dixon's metaphor highlights the organic and unregulated influx of data into the lake, catering to a wide range of user requirements and potential inquiries in the future.

Data Lakehouse

A significant innovation in data management design is the *Data Lakehouse*, which combines data lakes and data warehouses according to Zaharia et al.[24] Data warehouses hold structured, queryable data, while data lakes contain raw, unstructured data. Data lakehouses use these technologies to build a unified platform that handles several data kinds and workloads, improving flexibility and performance.

Using scalable storage and flexible data input, data lakes allow organisations to store huge amounts of varied data kinds in their original formats. This is useful for handling modern organisations' rising unstructured data. Data lakes benefit from indexing, schema enforcement, and ACID transactions. These features protect data and speed up analysis.

Finally, the Data lakehouse architecture improves data management by providing a cost-effective, scalable, and flexible solution for modern companies. Data lakehouses promote innovation and better decision-making by combining data lakes and data warehouses.

Lambda Architecture

The Lambda architecture has been proposed by Nathan Marz [16].

- **Batch Layer:** This layer manages the master dataset and pre-computes batch views. It provides comprehensive and accurate views of the data by processing it in large, immutable batches.
- **Speed Layer:** This layer compensates for the high latency of the batch layer by processing data in real-time. It generates real-time views that are immediately accessible but might be less accurate or comprehensive.
- **Serving Layer:** This layer indexes and exposes the pre-computed views from both the batch and speed layers, allowing for fast queries that return the most up-to-date information.

The primary goal of the DataBlend algorithm, as outlined in this thesis, is to effectively oversee a constantly changing data lake, ensuring the real-time processing of workloads. More precisely, the algorithm is created to provide almost immediate solutions to queries that accurately represent the most up-to-date and comprehensive data, customized to the exact details of each individual request. The primary elements of this algorithm, namely the search and integration processes, account for the majority of the computing workload and are significantly impacted by the scale and intricacy of the data lake.

In order to tackle these difficulties, the thesis utilizes the Lambda Architecture, a framework developed by Nathan Marz [16] during his tenure at Twitter in 2011 . This design was a crucial milestone in effectively handling the rapid expansion of data on the social media platform. Marz first disseminated his knowledge on this architectural framework through blog posts and presentations, culminating in the release of his book "Big Data: Principles and Best Practices of Scalable Realtime Data Systems" in 2015. This study has subsequently been acknowledged as a groundbreaking addition to real-time big data systems, offering a resilient approach for managing large-scale data processing promptly.

The creation of DataBlend comprises basic parts from the Lambda Architecture which have been customised to meet the specific objectives and scenarios detailed in this thesis. This technique entails making deliberate decisions about whether to include or exclude specific architectural aspects, with a special emphasis on utilising components such as the batch layer. Implementing Nathan Marz's three-part architecture greatly simplifies the process of developing and improving the DataBlend method. By focusing on specific layers of the architecture, it is feasible to add targeted enhancements, which can increase both the accuracy and efficiency of the algorithm. This comprehensive strategy guarantees that DataBlend maintains its flexibility and effectiveness, enabling it to address the dynamic requirements of a data lake.

2.2 Algorithmic Foundations for DataBlend

This section explores the algorithmic strategies that support DataBlend, with a specific focus on Latent Dirichlet Allocation for extracting topics and word embeddings for representing semantics. The utilisation of these techniques is crucial for the identification and analysis of the main content within documents, hence enabling the efficient and precise retrieval of data in DataBlend.

Latent Dirichlet Allocation Keyword Search

Latent Dirichlet Allocation (LDA) is a statistical method used to extract topics from a document that describes the main content of the document. This is applicable to a variety of file types, but specifically to table files used in this thesis. According to [2]

the Dirichlet Distribution, upon which LDA is built, is a family of continuous multivariate probability distributions that are each given a vector of positive real numbers as their parameter. For a given set of parameters $\alpha_1, \alpha_2, \ldots, \alpha_K$, the Dirichlet distribution produces a probability distribution over the K-simplex, which is a K-dimensional space where each point represents a vector of probabilities summing to 1.

Probability Density Function (PDF):

The PDF of the Dirichlet distribution is given by:

$$f(x_1, \ldots, x_K; \alpha_1, \ldots, \alpha_K) = \frac{1}{B(\alpha)} \prod_{i=1}^{K} x_i^{\alpha_i - 1}$$

where $x_i \geq 0$ for all i and $\sum_{i=1}^{K} x_i = 1$. Here, $B(\alpha)$ is the multivariate Beta function, defined as:

$$B(\alpha) = \frac{\prod_{i=1}^{K} \Gamma(\alpha_i)}{\Gamma\left(\sum_{i=1}^{K} \alpha_i\right)}$$

where $\Gamma(\cdot)$ is the Gamma function.

Parameters:

The parameters $\alpha_1, \alpha_2, \ldots, \alpha_K$ are called concentration parameters and they control the distribution's shape:

- If all $\alpha_i > 1$, the distribution is unimodal with a peak at the center.
- If all $\alpha_i = 1$, the distribution is uniform over the simplex.
- If any $\alpha_i < 1$, the distribution has a higher density near the boundaries of the simplex.

According to Blei et al. [4], LDA follows a generative process:

1. For each document d, choose a distribution $\boldsymbol{\theta}_d$ over topics from a Dirichlet distribution with parameter $\boldsymbol{\alpha}$.
2. For each word w in document d:
   a) Choose a topic z from the multinomial distribution Mult($\boldsymbol{\theta}_d$).
   b) Choose a word w from the multinomial distribution Mult($\boldsymbol{\varphi}_z$), where $\boldsymbol{\varphi}_z$ is the word distribution for topic z.

Each word w is generated from the topic-specific word distribution $\boldsymbol{\varphi}_{z_{dw}}$.

As a result, the LDA algorithm extracts a predefined number of topics, each represented by a distribution of keywords within the document.

For a more precise explanation of the probability P, the overall formula for the extraction of topics out of documents is [1] and [4]:

$$P(\boldsymbol{W}, \boldsymbol{Z}, \boldsymbol{\theta}, \boldsymbol{\varphi}; \alpha, \beta) = \prod_{i=1}^{K} P(\varphi_i; \beta) \prod_{j=1}^{M} P(\theta_j; \alpha) \prod_{t=1}^{N} P(Z_{j,t} \mid \theta_j) P(W_{j,t} \mid \varphi_{Z_{j,t}}), \tag{2.1}$$

$\boldsymbol{W}$ is the set of observed words for all documents.
$\boldsymbol{Z}$ is the set of topic assignments for each word.
$\boldsymbol{\theta}$ is the set of topic distributions for each document.
$\boldsymbol{\varphi}$ is the set of word distributions for each topic.
α and β are the hyperparameters of the Dirichlet priors.

We can disassemble the forumula to better understand how it works. Every element is essential to the process of generating and explains how subjects and words are assigned and produced probabilistically. The model's elements are examined in detail here:

1. **Topic-Word Distribution**:

$$\prod_{i=1}^{K} P(\varphi_i; \beta) \tag{2.2}$$

This term represents the prior probability of the word distributions for each of the K topics, where φ_i is the word distribution for topic i. This distribution is drawn from a Dirichlet prior with parameter β. 2. **Document-Topic Distribution**:

$$\prod_{j=1}^{M} P(\theta_j; \alpha) \tag{2.3}$$

This term represents the prior probability of the topic distributions for each of the M documents, where θ_j is the topic distribution for document j.

3. **Topic Assignment for Each Word**:

$$\prod_{t=1}^{N} P(Z_{j,t} \mid \theta_j) \tag{2.4}$$

This term represents the probability of assigning a topic to each word in the documents. $Z_{j,t}$ is the topic assignment for the t-th word in document j, drawn from the topic distribution θ_j.

4. **Word Given Topic**:

$$\prod_{t=1}^{N} P(W_{j,t} \mid \varphi_{Z_{j,t}}) \tag{2.5}$$

This term represents the probability of a word given the assigned topic. $W_{j,t}$ is the t-th word in document j, drawn from the word distribution $\varphi_{Z_{j,t}}$ of the assigned topic $Z_{j,t}$.

In conclusion this formula provides a clear demonstration of how the LDA model generates themes and words, emphasising their dependence and the underlying statistical structure.

Word embeddings

Word embedding is a technique used in Natural Language Processing (NLP) to transform words into continuous vector representations in a multidimensional space. This representation utilises semantic links between words, allowing for the proximity of words with similar meanings or contexts. DataBlend is able to identify similar tables due to the advantage of the relationship between vector representation and semantic similarity.

Word embeddings are derived from the Distributional Hypothesis, which posits that words used in similar contexts share similar meanings. As stated in "Introduction to Information Retrieval"[15], word embeddings are commonly generated from extensive text corpora using methods like neural networks. These methods acquire word representations by making predictions about words in relation to their surrounding context, using a sliding window technique. This approach captures both syntactic and semantic information.

There are several popular pretrained models for word embeddings. One of them is Word2Vec, developed by Google in the paper [17], which utilises the Skip-gram and Continuous Bag of Words (CBOW) architectures. Another model is GloVe [23], created by Stanford University, which constructs embeddings based on global word-word co-occurrence statistics from a corpus. Lastly, there is FastText [13] by Facebook's AI Research (FAIR) lab, an extension of Word2Vec that takes into account subword information, making it better equipped to handle rare and out-of-vocabulary words.

Random Hyperplane Locality Sensitive Hashing

The algorithm of the *Random Hyperplane Locality Sensitive Hashing* consists of three underlying key concepts.

1. **Locality Sensitive Hashing (LSH)**: LSH is a technique that inserts simliar items with a high probability into the same bucket and thereby significantly reducing the search space.
2. **Cosine Similarity**: Cosine similarity measures the cosine of the angle between two vectors, providing a metric to determine how similar two vectors are, with a value ranging from -1 to 1.
3. **Hyperplanes**: In the context of machine learning, hyperplanes are decision boundaries that separate different classes of data in a high-dimensional space.

Cosine Similarity

$$S_C(A, B) := \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \cdot \sqrt{\sum_{i=1}^{n} B_i^2}} \tag{2.6}$$

Locality Sensitive Hashing

Locality Sensitive Hashing (LSH) is an algorithmic approach specifically developed to do effective searches for approximate nearest neighbours in spaces with a high number of dimensions. The main objective of Locality Sensitive Hashing (LSH) is to efficiently group similar items together in buckets, increasing the likelihood of performing similarity searches and clustering in huge datasets without the requirement of exhaustive comparisons.

LSH employs a set of hash functions H, where each hash function $h \in H$ possesses the characteristic that for two inputs x and y:

$$\Pr_{h \in H}[h(x) = h(y)] = sim(x, y) \tag{2.7}$$

The similarity function $sim(x, y)$ gives a number between 0 and 1 based on the level of how similar the vectors x and y are. The likelihood of hash collisions, which occurs when two different instances of data are assigned to the same hash bucket, allows us to estimate the similarity between two sets.

Hyperplanes

The hashing technique used in Random Hyperplane LSH involves the utilisation of hyperplanes. In a high-dimensional space, a hyperplane is defined by a normal vector and divides the space into two half-spaces. In Random Hyperplane LSH, several random hyperplanes are created by sampling normal vectors from a Gaussian distribution. Every data point is projected onto these hyperplanes. The position of each hyperplane determines the hash value of the data point, either as binary 0 or 1.[15, page 270]

All three components combined together

(1) Creating Random Hyperplanes: Hyperplanes are generated by sampling vectors from a Gaussian distribution.

(2) Calculating Hash Code: The binary hash code is generated by determining which side of the hyperplane each data point lies on, using the dot product with the normal vector of each hyperplane.

(3) Grouping into buckets: Data points that share the same hash code are placed together in a bucket.

(4) Querying: When determining the bucket for a query point, the process remains consistent. Only the points within the same bucket are taken into account for additional similarity evaluation.

This approach utilises cosine similarity to guarantee that vectors with similar angles (high cosine similarity) are more likely to be grouped together on a hyperplane, resulting in comparable hash codes. Efficient approximate nearest neighbour searches in high-dimensional spaces can be achieved using this probabilistic approach, striking a balance between accuracy and computational efficiency.

Approximate k-Nearest Neighbour Search

The *Approximate k-Nearest Neighbour (k-NN) Search* is a fundamental algorithm in the realm of machine learning and data mining, specifically tailored for high-dimensional spaces. Unlike the exact k-NN search, which guarantees the identification of the true k nearest neighbours, the approximate version aims to significantly enhance computational efficiency while delivering neighbours that are close enough to the actual nearest ones.

1. **k-Nearest Neighbour (k-NN)**: The k-NN algorithm finds the k closest data points to a given query point based on cosine similarity.
2. **Approximation for Efficiency**: By introducing approximation, the algorithm reduces the search space and computation time, making it feasible to handle large datasets and high-dimensional spaces.
3. **Data Structures and Techniques**: Utilizes specialized data structures and techniques like Locality Sensitive Hashing (LSH) to accelerate the search process.

Approximate k-NN Using Locality Sensitive Hashing

Locality Sensitive Hashing (LSH) is a pivotal technique in approximate k-NN searches, efficiently narrowing down the search space. The algorithm leverages hash functions that map similar data points to the same bucket with high probability. Here's how LSH integrates into the approximate k-NN search:

1. **Hashing Data Points**: Data points are hashed into buckets using LSH, with similar points likely falling into the same bucket.
2. **Reduced Search Space**: For a query point, only the data points in the corresponding bucket(s) are considered for the nearest neighbour search, drastically reducing the number of comparisons.

3. **Approximate Neighbours**: The search within the reduced subset provides approximate neighbours, balancing between precision and performance.

This methodology allows for efficient nearest neighbour searches in large and high-dimensional datasets by focusing computational resources on the most promising candidates, ensuring a practical trade-off between speed and accuracy.

2.3 Evaluation metrics

Throughout the related work and the DataBlend Algorithm follwing metrices are very important to evaluate the proposed systems.

Set-based evaluation metrics

The **Precision** is a measure of the proportion of relevant documents among the retrieved documents in the result list. The Formula is as follows:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \qquad (2.8) \qquad \text{Precision@}k = \frac{|\text{Rel}_q \cap \text{Ret}_q^k|}{|\text{Ret}_q^k|} \qquad (2.9)$$

Figure 2.1: The precision metric

In equation (2.8), *TP* represents the True Positive. In the context of semantic similarity, the algorithm accurately identified instances that are similar to the query file. On the other hand, *FP* refers to False Positive, which occurs when the algorithm incorrectly predicts instances as similar to the query instance when they are actually not. This explanation of Precision is most commonly used. [10, page 262]

Precision@K (2.9) also calculates the precision as in (2.8) but in the defintion of Alkhawaldeh the focus is on the ranking of the results, given the ranked list R_q of retrieved instances based on the query q, where Ret_q^k is the the top k documents of the query q and Rel_q are all relevant documents to q.

This difference between the two approaches will both be used in this thesis and are therefore highlighted here.

Position-based Evaluation Metrics

In [15, page 149] Manning et al. defines the **Normalised Discounted Cumulative Gain (NDCG)** as a metric to evaluate the performance of a search engine's ranking algorithm. It measures the relevance of retrieved documents by comparing them to an ideal ordering. This represents the main contrast between set-based evaluation criteria, which solely focuses on either the document is relevant or not. The approach based on two primary elements: **Discounted Cumulative Gain (DCG)** and **Ideal Discounted Cumulative Gain (IDCG)**.

DCG is a quantitative measure that evaluates the significance of retrieved documents, taking into account their position in the result list. The calculation for **Discounted Cumulative Gain** for a specific rank position

$$\mathrm{DCG}_p = \sum_{i=1}^{p} \frac{2^{rel_i} - 1}{\log_2(i+1)} \tag{2.10}$$

rel_i is the relevance score of the document at position i.
$\log_2(i+1)$ is used to discount the relevance score based on the position i.

The IDCG value is a measure of the highest achievable DCG for a specific set of documents, assuming they are arranged in the most ideal order based on relevance. For calculating the IDCG at position p, the documents are sorted in descending order based on their true relevance scores. The formula for the **Ideal Discounted Cumulative Discounted Gain** is:

$$\mathrm{IDCG}_p = \sum_{i=1}^{p} \frac{2^{rel_i} - 1}{\log_2(i+1)} \tag{2.11}$$

rel_i values are sorted in descending order to achieve the highest possible gain.

Normalising DCG with IDCG, the NDCG provides a more comprehensive evaluation of the ranking algorithm, allowing for a clearer understanding of its effectiveness. A higher NDCG value suggests a ranking that is more aligned with the ideal ranking, indicating improved performance of the system. The formula for the **Normalised Discounted Cumulative Gain** is:

$$\mathrm{NDCG}_p = \frac{\mathrm{DCG}_p}{\mathrm{IDCG}_p} \tag{2.12}$$

Degree of Relevance

Documents are classified as relevant or non-relevant in an Information Retrieval (IR) system evaluation based on their ability to satisfy a certain user information requirement. This classification is known as ground truth judgement. Manning et al. urge that specialists make these relevance assessments to ensure accuracy. As a result, experts' decisions will determine whether a result is True Positive, False Positive for set-based metrics, or the relevance score for position-based metrics [15, page 140]

3 Related Work

3.1 Inspirational Sources and Comparative Analysis

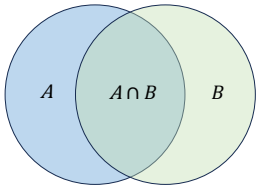
Throughout this project, the structure of DataBlend has undergone significant changes. Various publications have played an important part in shaping its evolution. In this section, we will showcase four important publications, starting with the master thesis by Víctor Díaz de Burgos Llaberia [9]. This thesis established the foundation for the concept of extracting and embedding keywords. In addition, the evaluation and proposals helped us determine which methods for extracting keywords to focus on for further development. In order to gain a comprehensive understanding of the different possibilities, we will provide a detailed presentation of these approaches. The publication "Topic Modelling for Expert Finding Using Latent Dirichlet Allocation" by Motazmi and Naumann [19] provided valuable insights into the performance of Latent Dirichlet Allocation and highlighted its advantages. In "Enhancing Data Lake Management Systems with LDA Approach" [6], additional research was conducted to explore the parameters of LDA. In addition, a recent publication by Edalatfard et al. [10] presented a new approach to ranking tables based on a modified word-embedding method. This paper, titled "A New Weighting Scheme for Document Ranking Based on the Modified Word-Embedding Method," introduced a novel method for ranking tables semantically. Furthermore we choose our evaluation metrics based on this paper. To improve runtime of the DataBlend the "Data Lake Architectures and Metadata Management" has presented valuable information for us to use the meta data.

Enhancing table discovery and similarity evaluation in data lakes

In his master thesis, Víctor Díaz de Burgos Llaberia [9] explores the enhancement of table discovery and similarity evaluation in data lakes through a combination of Jaccard similarity, keyword extraction (using YAKE and LDA), and word embeddings (using Word2Vec and BERT).

On three different datasets the algorithms have been evaluatated. First Díaz de Burgos Llaberia used the ***Simple Jaccard Similarity***.

The equation shown in (3.1) calculates the similarity score between two sets, specifically A and B. The similarity score between the query file and a specific file in the data lake determines by calculating the similarity scores for each pair of columns in the two CSV



$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \tag{3.1}$$

Figure 3.1: Jaccard Similarity

files. The overall similarity score is then obtained by taking the mean of all the pair scores. In terms of computational efficiency, this process is expensive due to its time complexity of $\mathcal{O}(N * M)$, where N represents the number of columns in the query table and M represents the number of columns in the file being compared.

Because of the expensive process runtime the author focused on different ways to determine the similarity scores between CSV files. Follwing approaches in the master thesis have been a combination of keyword extraction (using YAKE and Latent Dirichlent Allcotion), and word embeddings (using Word2Vec and BERT).

Keyword extraction

The ***Latent Dirichlent Allocation*** has been explained in detail in the premliminaries.(2.2)

The ***YAKE (Yet another Keyword Extraction)*** an innovative system that uses features to extract keywords from single documents in multiple languages. It can handle texts of various sizes, domains, or languages. Campos et al. [5]. Algorithm consists of six steps. (1) Text pre-processing; (2) Feature extraction; (3) Individual terms score; (4) Candidate keywords list generation; (5) Data Deduplication; and (6) Ranking. For further understanding, short explanation of the algorithm.

Text pre-processing: In this part the text does get split into words, the position of the each words get recorded and stop words removed. Stop words in the context of informaiton retrieval are words, that only add very little value to the document matching and therefore those words are excluded out of the vocabulary [15, page 25]. Besides the stop words the text gets split after each empty space.

Feature extraction: During the feature extraction process, a set of five attributes is utilised to capture the characteristics of each individual phrase. Camoe et. al presented these 5 features: (1) **Casing**. The casing refers to the grammatical feature that indicates the form of a word. (2) **Word Positional** which means the position of the first occurrence of the term. Early positions are considered more important based on the assumption that important words are at the beginning of the document. (3) **Word and Phrase Frequency** adds a higher score to a phrase the more often it appears.

The (4) **Word Relatedness** calculates the likelihood of the term appearing in similar contexts. This is been done by calculating the numbers of terms to the left and right of the current phrase. The greater the number of different terms that appear beside the candidate word, the more likely it is that the candidate word is meaningless. (5 **Word DifSentence** scores a phrase or word higher, if it occurs more often. This means in combination with the Word Relatedness that that this score is only higher if it appears more often in different sentences.

Individual terms score: In this section, the term **n-gram** is used and therefore is briefly defined here. The n-gram or also commonly often used k-gram is a contiguous sequence of k or n items from a given string or text sample. In text processing, k-grams typically refer to sequences of k characters, such as the 3-grams of "castle": $ca, cas, ast, stl, tle, le$. N-grams can be sequences of phonemes, syllables, letters, words, or base pairs and are used in computational linguistics and natural language processing for tasks like text analysis, machine learning, and statistical modeling. Both are useful for analyzing and indexing terms. [15, page 50]

On the third phase, the individual terms score, heuristically all the attributes of feature extraction merge into one measure and score each term. The fourth phase the **Candidate keywords list generation** creates keywords using this weight. A contiguous 1, 2, and 3-gram candidate keyword sequences is build using a sliding window of 3-grams.

$$S(kw) = \frac{\prod_{w \in kw} S(w)}{TF(kw) \cdot \left(1 + \sum_{w \in kw} S(w)\right)} \tag{3.2}$$

where $S(kw)$ is the score of a candidate keyword, $S(w)$ is the score of each term, and $TF(kw))$ is the term frequency of the keyword.
The candidate keywords receive a final score $S(kw)$, with lower scores indicating more significant keywords.

Data duplicatio and Ranking In the fifth stage, the Levenshtein distance is employed to eliminate candidates that are similar to those from the previous steps. The **Levenshtein Distance** is a measure of the least number of edit operations needed to turn one character string, s1, into another character string, s2. The allowed editing actions are character insertion, character deletion, and character replacement. As an illustration, the Levenshtein distance between the words "cat" and "dog" is three. [15, page 53]

At last, candidates are evaluated based on their scores in the ranking and selection process, with lower scores indicating stronger keywords. This grading system is designed to prioritise fewer commonly used and more contextually relevant concepts. The system will generate a list of relevant keywords, which will consist of 1, 2, and 3-grams. The most significant keywords will have the lowest S(kw) scores.

Following the keyword extraction part two embedding methods (2.2) have been focused on by Díaz de Burgos Llaberia. The Word2Vec has been explained in detail in the preliminaries (2.2). The socond embedding

BERT, a cutting-edge language representation model, was introduced by Devlin et al. (2018). The field of natural language processing (NLP) has been significantly improved by BERT, which offers pre-trained representations that can be customised for various subsequent tasks, including question answering, sentiment analysis, and identifying named entities. we want to highlight the key innovation of BERT - its use of a both directions transformer architecture. This unique approach allows BERT to consider language context from both directions simultaneously, resulting in a more comprehensive understanding compared to previous models that only focused on one direction at a time.

A multi-layer bidirectional Transformer encoder is where the embeddings in BERT come from. There are several layers (called L) of Transformer blocks in this encoder. Each one has a secret size (H) and a number of self-attention heads (A). For instance, BERTBASE has 12 layers, a secret size of 768, and 12 self-attention heads, which adds up to 110 million parameters. BERTLARGE, on the other hand, has 340 million parameters, 24 layers, a secret size of 1024, and 16 self-attention heads.

BERT utilises a vocabulary of 30,000 tokens, each of which is represented by WordPiece embeddings. Each input sequence starts with a special classification token ([CLS]) and ends with a separator token ([SEP]), whether it's a single sentence or a pair of sentences. The input representation of each token is determined by the sum of its token, segment, and position embeddings. When it comes to classification tasks, the last hidden state of the [CLS] token is used. On the other hand, for token-level tasks like named entity recognition, the hidden states of other tokens can be utilised.

During the pre-training phase, BERT utilises two unsupervised tasks: Masked Language Modelling (MLM) and Next Sentence Prediction (NSP). During the training process, a certain percentage of input tokens are masked in MLM, and the model is then tasked with predicting these masked tokens using the surrounding context. BERT is able to acquire bidirectional representations through this process. The process of NSP entails the prediction of whether sentence B follows sentence A, allowing the model to gain a deeper understanding of the relationship between pairs of sentences. These pre-training tasks allow BERT to capture intricate language patterns, making it incredibly powerful for a wide range of NLP applications.

The utilisation of BERT embeddings establishes a strong basis for numerous NLP tasks, harnessing the power of deep bidirectional context to attain exceptional results across a diverse set of benchmarks. By leveraging pre-trained models and their ability to be fine-tuned, the development and deployment of NLP systems becomes much more efficient, eliminating the requirement for task-specific architectures.

The evaluation metric selected for evaluating the accuracy of the proposed models is the Normalised Discounted Cumulative Gain (NDCG)(2.3.2). The evaluation examines the

output ranked tables of the models and compares them to the expected results of an ideal table search engine. Given the absence of a golden rule for determining document similarity in the dataset, the author annotated the similarity of the query tables with the tables in the data lake, based their judgements on relevance. Three levels of similarity were defined: 0 indicating no similarity, 1 indicating a similar topic, and 2 indicating the exact same topic.

The evaluation procedure involves analysing all candidate tables and the query table to extract keywords or embeddings, determining similarity scores, and ranking outcomes. Tracking execution time at these steps measures model efficiency. The initial run, which processes the full data lake by extracting 30 random tables, takes longer than runs with a preprocessed data lake that handle only the query table.

After analysing the results of this research project, it turns out that the Latent Dirichlet Allocation algorithm in combination the the word embedding achieved the most optimal balance between accuracy and runtime, both on preprocessed and non-preprocessed data.

Topic modeling for expert finding using latent Dirichlet allocation

In the approach of the algorithm we present, the latent Dirichlent Alloction (LDA) is being used. As in (2.2) mentioned the Allocation statistically extracts the main top-k topics of an defined file based of predefined number of keywords.

Already in 2013 [19] Momtazi et al. propose a topic modeling approach to the task of expert finding, aiming to rank experts based on a given query. They utilize Latent Dirichlet Allocation (LDA) to extract probabilistic topics from a document collection. Within their work they worked through the following phases:

Topic Extraction: LDA is used to extract latent topics from the document collection. Every document is depicted as a blend of topics, where each topic is a probability distribution across words. This process uncovers the core themes found in the documents, making it easier to pinpoint the areas of expertise linked to each candidate.

Calculating Probability: The topics that have been extracted are utilised to determine the likelihood of a candidate being knowledgeable about a specific query. The probability of each candidate is assessed by evaluating the alignment between the topics related to the candidate and the topics of the query.

Ranking Candidates: Candidates are evaluated and assigned rankings using calculated probabilities. This ranking demonstrates the probability that a candidate possesses expertise in the field of interest, utilising the connections between the search terms and topics relevant to the candidate.

The experimental results showcased by Momtazi and Naumann highlight the efficacy of their LDA-based approach. Experiments were conducted using the Text REtrieval Conference (TREC) Enterprise track datasets from 2005 and 2006. Based on the findings,

their method focusing on the topic proved to be more effective than the current profile- and document-based models. In particular, the LDA approach proved to be more effective in capturing the semantic connections between queries and candidate expertise, resulting in improved accuracy in rankings.

Benefits of LDA in Expert Finding according to Momtazi et al.:

Latent Relationships: LDA uncovers hidden connections between words in the document storage, allowing the model to better understand the underlying topics that may not be easily identifiable through basic keyword matching.

Through the representation of documents as mixtures of topics, LDA offers a deeper and more detailed knowledge of the content. This enables a more accurate identification of expert candidates by considering their connection to pertinent subjects.

Better Accuracy: The probabilistic approach of the is a key part of reducing unwanted noise and making the expert finding system more accurate overall. By focusing on topic ranges instead of single words, the model can tell the difference between information that is important and information that is not.

The work by Momtazi and Naumann highlights the potential of LDA in improving expert finding systems. Their approach not only enhances the ranking of candidates but also showcases the wider range of applications for topic modelling techniques in different information retrieval tasks. The impressive performance of their method in the TREC evaluations indicates the advantages and efficiency of utilising LDA for expert finding.

Overall, the research conducted by Momtazi and Naumann highlights the clear benefits of using LDA for topic modelling in expert finding. Discovering hidden themes and leveraging them to connect search queries and potential matches represents a significant advancement compared to conventional approaches that rely solely on keywords. Their work lays a strong groundwork for future exploration and application of LDA in similar domains.

Enhancing Data Lake Management Systems with LDA Approach

The main focus of the paper "Enhancing Data Lake Management Systems with LDA Approach" [6] is to handle, process, analyse, and present large datasets more effectively by utilising the Latent Dirichlet Allocation (LDA) model for data lake management. This method tackles the problems caused by data lakes' lack of predefined schemas, which can result in inefficiencies and increase the chance that they will turn into data swamps.

The LDA model's output topics are evaluated for quality using the coherence metric. It measures the degree of semantic similarity between high-scoring words in a topic. The CUmass coherence metric, which creates topic representations by examining word

co-occurrences, is specifically used in this paper. Normalised Pointwise Mutual Information (NPMI) and cosine similarity are used to calculate scores. Better performance is indicated by higher coherence ratings, which represent more semantically consistent themes.

Alpha α and Beta β: Within LDA, α and β are hyperparameters affecting the terms within topics and topic distribution.

The α parameter regulates the document-topic distribution. Documents with a lower alpha value are more likely to be dominated by a small number of themes, despite those with a higher alpha value are more likely to contain a variety of topics. The granularity of topic distribution across texts can be balanced by adjusting the alpha.

The β parameter regulates the distribution of topic words. Topics with a higher beta value are more likely to be described by a variety of words, whereas those with a lower beta value are more likely to be characterised by a small number of key words. The specificity of word distribution among subjects can be balanced by adjusting beta.

The researchers emphasise that choosing the right values for alpha and beta is essential to maximising the LDA model's performance. To find the best combination of these characteristics, they conducted a hyper-parameter grid search, which produced the highest topic coherence scores. This optimisation improves data management and avoids problems such as data swamps by guaranteeing that the model generates topics from the data lake that are coherent and meaningful.

The research provides a fresh viewpoint on data management by including LDA, which enhances data accessibility and organisation. The study makes the case that this strategy maximises the use of data lakes in contemporary businesses by addressing the difficulties in managing diverse data and offering a scalable means of deriving useful insights.

The study concludes by showing how LDA may improve data lake management and provide a reliable method for classifying and organising data. Future study on integrating labelled documents to enhance prediction capabilities and insights is made possible by the findings. The suggested method offers useful applications in information retrieval, decision support systems, and information retrieval, all of which are crucial for data-driven businesses to make well-informed decisions.

A new weighting scheme for document ranking based on the modified word-embedding method

In the evaluation section, we will utilise the following publication to assess the proposed algorithm DataBlend. Therefore, we will provide a comprehensive explanation of the algorithm.

The research paper by Edalatfard et al. [10] introduces a novel method for document ranking. This method, called Truncated Average Weighted Frequency-Inverse Document Frequency (TAW-TFIDF), leverages TF-IDF scores to weigh word embeddings, combining these scores to form a document vector. The TAW-TFIDF method has demonstrated superior performance compared to current strategies for ranking documents based on semantic similarity like Average of Weighting and TF-IDF[10, page 259].

The used dataset

The authors used the dataset from the 2019 Text REtrieval Conference (TREC), which consists of 3,213,835 records. To enhance computational efficiency, a smaller subset of 20,000 records was randomly selected from the original dataset for querying and analysis [10, page 262].

The TAW-TFIDF method involves the following key steps:

TF-IDF Weighting: Calculating TF-IDF scores for words in each document. To have a deeper comprehension of the tf-idf algorithm, a brief elaboration follows.

TF-IDF is a statistical measure that examines the significance of a word in a document compared to a collection of documents. It is commonly used in evaluating word importance in a corpus. This is a commonly used method in the field of information retrieval and text mining.

TF is a metric that quantifies the frequency of a term within a document. Here's a straightforward method for calculating TF:

$$\text{TF}(t, d) = \frac{\text{Number of times term t appears in document } d}{\text{Total number of terms in document } d} \tag{3.3}$$

IDF discovers how important a word is by looking at how often it appears in documents. All terms are thought to be equally important when figuring TF. Words like "is," "of," and "that," on the other hand, may be used a lot but not mean much. So, to make the rare terms stand out more, we need to make the common ones less common by

$$\text{IDF}(t, D) = \log\left(\frac{\text{Total Number of documents N}}{Number of documents with term}\right) \tag{3.4}$$

The TF-IDF score for a term in a document is the product of its TF and IDF scores:

$$\text{TF-IDF}(t, d, D) = \text{TF}(t, d) \times \text{IDF}(t, D) \tag{3.5}$$

Truncation: Selecting the top predefined k top words with the highest TF-IDF scores from each document.

Word Embedding: Finding the weighted average of the word embedding vectors for the chosen k words and using that to generate text vectors.

The proposed approach was evaluated using metrics such as average precision, normalised discounted cumulative gain (NDCG), and mean average precision to evaluate its efficacy. In order to achieve these results, it was crucial to prioritise the retrieved documents according to their relevance. This ranking was conducted by experts using a standard methodology in cases where there is no established 'gold standard' or known correct order of results. From this, the precision could be calculated.

This approach guarantees an extensive review of the results by placing them within the context of expert assessment, which is essential for validating the efficiency of the proposed method for ranking documents.

result During the evaluation, the TAW-TFIDF method demonstrated better results compared to other algorithms like traditional TF-IDF and Average of Weighting (AW) in terms of average precision and NDCG metrics. More precisely:

TAW-TFIDF outperformed the baseline methods in terms of AP scores, suggesting that it was more successful in accurately retrieving relevant documents. Based on the results, it is clear that TAW-TFIDF outperformed other methods in ranking relevant documents higher in the results list. This is evident from the higher NDCG scores. In the comparative analysis presented in the paper, TAW-TFIDF consistently outperformed the traditional TF-IDF and AW-TFIDF across multiple queries. The significant enhancement in AP and NDCG metrics showcases the efficacy of the TAW-TFIDF approach in accurately capturing semantic similarity and efficiently ranking documents.

Data Lake Architectures

The paper "On Data Lake Architectures and Metadata Management" by Sawadogo and Darmont[22] (2020) offers a thorough examination of different strategies for designing and handling data lakes, with a specific emphasis on data lake architectures and metadata management. The authors attempted to provide a clear understanding of data lakes and to offer a systematic analysis of data lake architectures and metadata management practices. They have proposed new typologies and classifications to enhance the understanding and implementation of data lakes.

When examining data lake architectures, the authors make a clear distinction between pond and zone architectures. They come up with a new way to group designs into groups based on their function (like data ingestion, storage, processing, and access) and the maturity of the data (raw, refined, etc.). This new system is called Functional Maturity. Zone architectures categorise data into various zones based on their level of improvement. These zones include transient loading, raw data, trusted data, refined data, discovery sandbox, consumption, and governance zones. Various types of data are treated separately in pond architectures, including raw data, analogue data, application data, textual data, and archival data. Each type of data is managed in its own designated 'pond'. Hybrid architectures bring together elements from zone and pond architectures, effectively tackling the limitations of each by integrating data management based on functionality and data refinement levels.

The authors emphasise the importance of metadata in ensuring that data lakes do not turn into data swamps. They split metadata into three groups: global, operational, and business. Within each group, there are also intra-object and inter-object metadata. Graph-based models (provenance-centered, similarity-centered, and composition-centered) and data vault models are looked at as ways to model metadata. These models make the metadata structure flexible and able to change over time. The text explores different tools and techniques used in generating metadata, including Apache Flume for data provenance, Apache Tika for MIME type detection, and Apache Atlas for advanced metadata generation.

The paper comes to the conclusion that a complete information system is necessary for a data lake to work well. There are six important features that need to be considered for this system: semantic enrichment, data indexing, link generation, data polymorphism, data versioning, and usage tracking. The authors analyse various cutting-edge metadata systems and emphasise the importance of continued integration and development in this field.

3.2 Comparative Studies on Similar Issues

The following publications, which use this ranking in their system, highlight the significance of ranking tables according to their semantic similarity effectively within data lakes.

Gen-T: Table Reclamation in Data Lakes

This paper is about integrating data from large repositories efficiently, which fits with the goal of my system, which is to rank and compare documents based on how similar their content is. Gen-T's methodology for table discovery and integration is consistent with my utilisation of numerical representations and complex similarity metrics. It has also been shown to be scalable and efficient, which gives me useful information the base of DataBlend.

The publication by Fan et al. "Gen-T: Table Reclamation in Data Lakes"[11] recently published in 2024, presents an innovative system that tackles the challenging issue of table reclamation in data lakes. The focus is on finding a collection of tables within a data lake that, when combined, can accurately replicate a specific source table. Gen-T takes a different approach compared to traditional methods. Instead of solely focusing on query discovery, it places importance on reclaiming data from existing tables and dealing with incomplete and inconsistent data.

In order to accomplish this, Gen-T utilises a two-step procedure: Table Discovery and Table Integration.

During the Table Discovery phase, Gen-T starts by retrieving potential tables from the data lake that have similar values to the source table. They utilised data-driven table discovery methods to complete this task. The system guarantees that there is minimal duplication among candidates by utilising set similarity algorithms to detect pertinent tables. After conducting extensive research, a groundbreaking methodology known as matrix traversal is employed to further enhance the selection of potential tables. In this approach, tuples from the candidate tables are aligned with those in the source table, and then represented in a matrix form. Table integration is simulated by applying logical OR operations on these matrices. This process is essential for eliminating tables that may contain inaccurate data.

During the Table Integration phase, the tables that are being considered are initially preprocessed to align with the schema and key columns of the source table. As part of this, important columns must be projected and chosen, duplicates must be removed, and subsumed tuples must be controlled. After completing the necessary preparations, the tables are labelled to indicate any shared null values for integration. The integration process involves utilising outer union and a set of unary operations, such as projection, selection, complementation, and subsumption, to merge the tables. The objective is

to enhance the resemblance between the resulting table and the Source Table while reducing the occurrence of incorrect values.

Introducing a novel similarity measure known as Error-Aware Instance Similarity (EIS) to assess the level of resemblance between a reclaimed table and the source table. This measure takes into consideration errors and penalises mismatched values, resulting in a more precise evaluation of similarity.

Through the experiments conducted with real data lake tables, it becomes evident that Gen-T surpasses existing methods in terms of both accuracy and efficiency. Gen-T underwent extensive testing on various benchmarks, including real-world data lakes containing a substantial number of tables, reaching up to 15,000. The findings indicated that Gen-T demonstrated superior precision and recall compared to baseline methods when reclaiming source tables, effectively addressing both nullified and erroneous data. In addition, Gen-T has demonstrated its scalability by efficiently processing large tables and extensive data lakes.

Overall, Gen-T provides a strong solution for Table Reclamation in data lakes, utilising advanced techniques in table discovery and integration to tackle the challenges posed by incomplete and inconsistent data. The unique approach and proven effectiveness of this work make it a valuable addition to the field of data integration.

Integrating Dataset

The paper by R. J. Miller et al. [14] published in 2022 introduces ALITE (Adaptive Lightweight Integration of Tables), a scalable approach for integrating tables discovered in data lakes. The team focused on tackling the difficulties of integrating tables with different structures, missing data, and specific join patterns through the use of Full Disjunction (FD) semantics.

When it comes to finding common columns, ALITE begins by performing schema matching. This process entails assigning integration IDs to columns, ensuring that columns meant to be integrated are given the same ID. They use TURL (Table Understanding through Representation Learning) embeddings to represent columns as numerical vectors. These embeddings are useful for generating a similarity matrix that can cluster columns with similar content, even if the column headers are different.

The integration process requires the use of various important operators. The outer union operator is used to merge tables by including all columns from different tables in the integrated table, even if it means adding null columns where necessary. Subsumption then eliminates tuples that are subsumed by other tuples with fewer null values, thus reducing redundancy.

The ALITE FD algorithm consists of multiple steps. At first, any missing null values are replaced with unique labelled nulls to prevent any errors in tuple integration. Next, the tables are combined through an outer-union operation to generate a preliminary integrated table that encompasses all potential tuples. The complementation step iteratively combines tuples by identifying pairs of tuples that can complement each other, meaning that one tuple has non-null values in columns where the other tuple has nulls. Once the completion process is finished, the null values are returned to their original state. Then, subsumption is used to complete the integration, guaranteeing that the resulting tuples are maximally integrated.

Through the experiments conducted using real data lake tables, it has been demonstrated that ALITE exceeds existing integration algorithms in terms of both speed and accuracy. The empirical evaluation showcased benchmarks derived from real-world data scenarios, illustrating ALITE's efficacy in managing incomplete data and intricate join patterns. Based on the findings, it is evident that ALITE is capable of effectively incorporating extensive tables, ensuring both precision and efficiency even when null values are present.

Data Lake Architectures

The paper "On Data Lake Architectures and Metadata Management" by Sawadogo and Darmont[22] (2020) offers a thorough examination of different strategies for designing and handling data lakes, with a specific emphasis on data lake architectures and metadata management. The authors attempted to provide a clear understanding of data lakes and to offer a systematic analysis of data lake architectures and metadata management practices. They have proposed new typologies and classifications to enhance the understanding and implementation of data lakes.

When examining data lake architectures, the authors make a clear distinction between pond and zone architectures. They come up with a new way to group designs into groups based on their function (like data ingestion, storage, processing, and access) and the maturity of the data (raw, refined, etc.). This new system is called Functional Maturity. Zone architectures categorise data into various zones based on their level of improvement. These zones include transient loading, raw data, trusted data, refined data, discovery sandbox, consumption, and governance zones. Various types of data are treated separately in pond architectures, including raw data, analogue data, application data, textual data, and archival data. Each type of data is managed in its own designated 'pond'. Hybrid architectures bring together elements from zone and pond architectures, effectively tackling the limitations of each by integrating data management based on functionality and data refinement levels.

The authors emphasise the importance of metadata in ensuring that data lakes do not turn into data swamps. They split metadata into three groups: global, operational, and business. Within each group, there are also intra-object and inter-object metadata. Graph-based models (provenance-centered, similarity-centered, and composition-centered) and data vault models are looked at as ways to model metadata. These models make the metadata structure flexible and able to change over time. The text explores different tools and techniques used in generating metadata, including Apache Flume for data provenance, Apache Tika for MIME type detection, and Apache Atlas for advanced metadata generation.

The paper comes to the conclusion that a complete information system is necessary for a data lake to work well. There are six important features that need to be considered for this system: semantic enrichment, data indexing, link generation, data polymorphism, data versioning, and usage tracking. The authors analyse various cutting-edge metadata systems and emphasise the importance of continued integration and development in this field.

4 Method

4.1 The DataBlend

The proposed DataBlend presents a new approach to rank tables by their similarity, utilising the Lambda architecture to effectively process, analyse, integrate, and retrieve data. DataBlend consists of three primary layers: Batch Layer, Serving Layer, and Speed Layer. Every layer plays a crucial role in the process of ranking and retrieving documents.

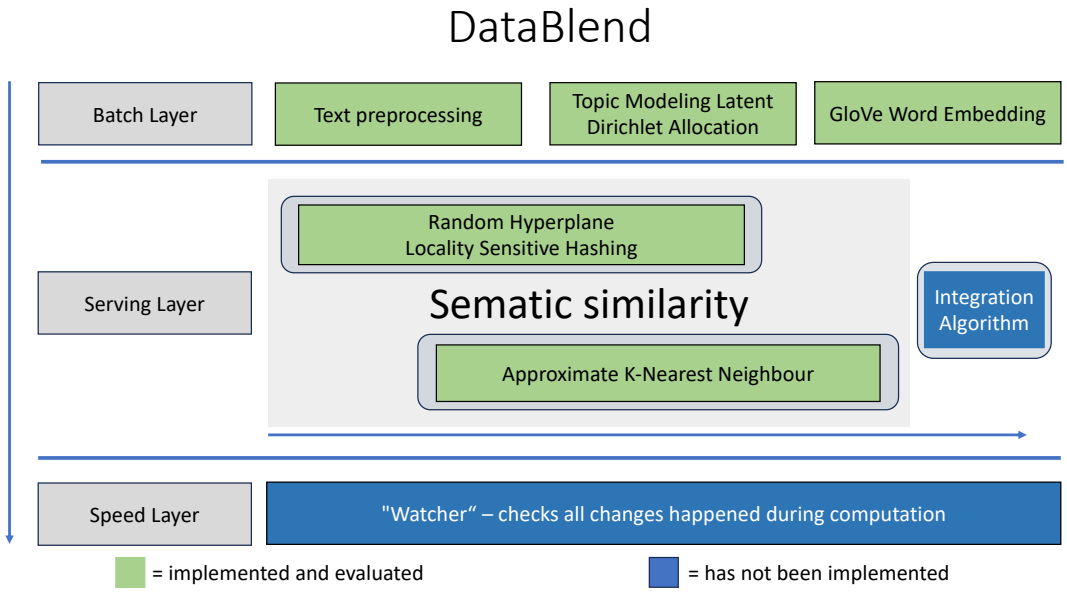


Figure 4.1: Overview of the DataBlend Framework. The layers are arranged from top to bottom, and within each layer, the elements are organized from left to right.

Figure (4.1) shows the overall framework we propose. In the evaluation section of this thesis, we focus on evaluating the algorithm prior to the integration phase, meaning elements coloured in green. We have focused our efforts on optimising the semantic document ranking code, leaving the integration algorithm implementation and the speed layer as a framework for future work. In general, this modular framework allows for the improvement of specific sub-algorithms within the system.

Batch Layer

The Batch Layer involves the execution of preprocessing operations on the table files. This involves text preparation, wherein stop words are eliminated and only words containing a minimum of two alphabetic characters are retained. After preprocessing, the Latent Dirichlet Allocation (LDA) algorithm is employed to perform topic modelling and extract the top 10 keywords that most accurately characterise each table file. It is presumed that each table file contains only one topic. Afterwards, a representative vector is computed for each table file by analysing the distribution of these top 10 terms. The vectors of the keywords are merged by taking a weighted average, where the weights are adjusted to reflect the relative significance of each term.

Serving Layer

The Serving Layer applies Locality Sensitive Hashing (LSH) using Random Hyperplane LSH to the representative vectors provided from the Batch Layer. This technique employs a mapping process to transform the vectors into a spatial arrangement where vectors that are similar in nature are positioned in close proximity to one another. The approximate K-Nearest Neighbour (KNN) search technique is subsequently employed to efficiently locate and recover documents that have resemblance to the query vector, incorporating the Locality Sensitive Hashing (LSH) output for optimal retrieval.

Speed Layer

The Speed Layer guarantees the system's synchronisation with the most recent data modifications through continual tracking of real-time updates. The purpose of this layer is to verify if any modifications have been made to the data lake after the query call, guaranteeing the utilisation of the most up-to-date data during calculation. This systematic method guarantees the system's ability to manage extensive data processing, while also delivering fast and precise document retrieval based on semantic similarity.

4.2 Set Up Overview

The design of the DataBlend and its surrounding environment has been intentionally chosen to allow for easy extension and integration with existing applications.

Docker is a open-source platform that simplifies the process of deploying, scaling, and managing applications within containers. These containers are designed to package an application and its dependencies, ensuring a consistent performance across various environments. Docker streamlines development and deployment, making it perfect for incorporating tools like MinIO into data processing systems. In addition, it is simple to close the container and decrease the storage space used by the application.

MinIO, a high-performance object system, supports the data lake design. The system offers an enhanced environment for efficient data processing, particularly for storing large amounts of objects.

Figure 4.2: MnIO, Inc. [18]

Figure 4.3: Docker, Inc. [8]

Advantages of the use of MinIo
MinIO's API seamlessly integrates with the Amazon Web Services (AWS) S3 infrastructure, providing a smooth and effortless integration with S3 applications. This compatibility makes it easier to work on future projects with minimal adjustments, taking advantage of the extensive use of S3 in the business sector.

Throughout the practical aspect of this thesis, MinIO showed outstanding efficiency in handling a large number of files within a single bucket. The performance remained consistently high without any noticeable degradation. In contrast, storing the same number of CSV files as raw data in a standard folder under macOS 13.5.1 significantly impacted system performance, particularly in terms of memory storage.

MinIO offers a convenient architecture for storing the representative vectors calculated by the DataBlend Algorithm for each table file. It allows for easy attachment and updating of these vectors as needed. Additionally, MinIO is commonly used in setting up data lakes within Apache Spark environments, making it a suitable choice for the data storage requirements of this project.

MinIO's data storage solution is highly reliable, scalable, and efficient, which greatly improves the performance and manageability of large data sets. It perfectly aligns with the project's objectives. The integration with Docker makes deployment and management much simpler, guaranteeing consistent and reliable performance across various environments.

4.3 The structure of the DataBlend repository

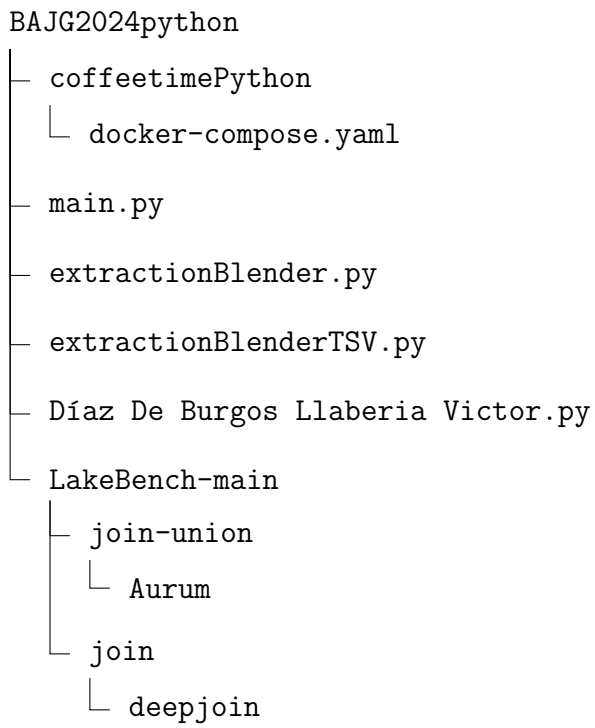


Figure 4.4: Directory Structure of DataBlend Repository (Key Elements Only) [12]

The repository's folder structure is shown in Figure (4.3). The coffeetimePython folder contains all the necessary information for the Docker and MinIO programmes, which are required for DataBlend setup. The ports for MinIO must be defined in the docker-compose.yaml file. This needs to be done for every device that works with the DataBlend because the application determines which ports are available and which ports are already closed by the device for other applications. Both "9001:9000" and "9093:9090" are the current port specifications.

To start the Docker container from within the project folder, use the command "docker-compose up -d", then stop it with "docker-compose stop". On these particular ports, you can access the MinIO user interface at "http://127.0.0.1:9093/login". Both minioadmin and password are "minioadmin". The DataBlend is within the main.py.

We have implemented the extractionBlender.py and extractionBlenderTSV.py to convert the datasets that came as tsv files and json files into table files, specifically csv files. The data from the source file will be changed and put into a MinIO bucket that already exists on the local server.

Further algorithms we tried to evaluate like the "Aurum" and "DeepJoin" are located in the LakeBench-Main but we did not manage to implement and evaluate them.

5 Evaluation

5.1 Evaluation Methodology

Set Based and Position Based metrics

Below is the evaluation table that was utilised to evaluate the performance of the DataBlend semantic document ranking algorithm and its competing algorithms for each query call.

	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
12	Query:	Star Wars' Story of Clone Wars Expands in 'Rebels'.csv														
13	Nr.	Results, Name CSV	TP	FP	Cumulative TP	Cumulative FP	Precision@ K	Cumulative Precision	Average Precision	Relevance Score	Descending Relevance Score	Discounted Gain	Discounted Cumulative Gain (DCG)	Ideal Discounted Gain	Ideal Discounted Cumulative Gain (IDCG):	Normalized Discounted Cumulative Gain (NDCG):
14	1	Star Wars' Story of Clone Wars Expands in 'Rebels'.csv	1	0	1	0	1	1	1	3	3	3	3	3	3	1
15	2	Chewbacca.csv	1	0	2	0	1	2	1	3	3	3	4,416508275	4,416508275	11,41650828	1
16	3	Tim Abell.csv	1	0	3	0	1	3	1	2	2	1,5	12,91650828	0	11,41650828	1,131388684
17	4	Best Current Baseball Players by Uniform Number 11-20.csv	0	1	3	1	0,75	3,75	0,9375	0	0	0	12,91650828	0	11,41650828	1,131388684
18	5	Doug Baldwin on rebel flag: What does Southern heritage real	0	1	3	2	0,6	4,35	0,87	0	0	0	12,91650828	0	11,41650828	1,131388684
19	6	Facebook to simplify privacy controls Wednesday.csv	0	1	3	3	0,5	4,85	0,808333333	0	0	0	12,91650828	0	11,41650828	1,131388684
20	7	Backgrounders & Fact Sheets.csv	0	1	3	4	0,428571429	5,278571429	0,754081633	0	0	0	12,91650828	0	11,41650828	1,131388684
21	8	What does Ace mean?.csv	0	1	3	5	0,375	5,653571429	0,706696429	0	0	0	12,91650828	0	11,41650828	1,131388684
22	9	Viewpoints.csv	0	1	3	6	0,333333333	5,986904762	0,665211640	0	0	0	12,91650828	0	11,41650828	1,131388684
23	**10**	**Become a Texas State Trooper.csv**	**0**	**1**	**3**	**7**	**0,3**	**6,286904762**	**0,628690476**	0	0	0	12,91650828	0	11,41650828	1,131388684
24	11	Dunkin' Brands Group Inc (NASDAQ:DNKN).csv	0	1	3	8	0,272727273	6,559632035	0,596330185	0	0	0	12,91650828	0	11,41650828	1,131388684
25	12	George Washington's false teeth not wooden.csv	0	1	3	9	0,25	6,809632035	0,567469336	0	0	0	12,91650828	0	11,41650828	1,131388684
26	13	Nuggets Tie NBA Record For Biggest Margin of Victory at 58.cs	0	1	3	10	0,230769231	7,040401265	0,541569328	0	0	0	12,91650828	0	11,41650828	1,131388684
27	14	What is the name of the Gecko in the Geico commercial?.csv	0	1	3	11	0,214285714	7,25468698	0,518191927	0	0	0	12,91650828	0	11,41650828	1,131388684
28	15	What Do NFC and AFC Stand For?.csv	0	1	3	12	0,2	7,45468698	0,496979132	0	0	0	12,91650828	0	11,41650828	1,131388684
29	16	storytelling.csv	0	1	3	13	0,1875	7,64218698	0,477636686	0	0	0	12,91650828	0	11,41650828	1,131388684
30	17	Registered Agents.csv	0	1	3	14	0,176470588	7,818657568	0,459921033	0	0	0	12,91650828	0	11,41650828	1,131388684
31	18	Case Number, What Does It Mean?.csv	0	1	3	15	0,166666667	7,985324235	0,443629124	0	0	0	12,91650828	0	11,41650828	1,131388684
32	19	Best Online Reputation Management Services 2018.csv	0	1	3	16	0,157894737	8,143218971	0,428590472	0	0	0	12,91650828	0	11,41650828	1,131388684
33	**20**	**Professional corporation.csv**	**0**	**1**	**3**	**17**	**0,15**	**8,293218971**	**0,414660949**	0	0	0	12,91650828	0	11,41650828	1,131388684
34	21	Hearing aid batteries.csv	0	1	3	18	0,142857143	8,436076114	0,40171791	0	0	0	12,91650828	0	11,41650828	1,131388684
35	22	escrow agent.csv	0	1	3	19	0,136363636	8,572439751	0,389656352	0	0	0	12,91650828	0	11,41650828	1,131388684
36	23	Sarbanes-Oxley Act (SOX).csv	0	1	3	20	0,130434783	8,702874533	0,378385849	0	0	0	12,91650828	0	11,41650828	1,131388684
37	24	oing from Sole Proprietor to S Corp.csv	0	1	3	21	0,125	8,827874533	0,367828106	0	0	0	12,91650828	0	11,41650828	1,131388684
38	25	My wife does not have a social security number.csv	0	1	3	22	0,12	8,947874533	0,357914981	0	0	0	12,91650828	0	11,41650828	1,131388684
39	26	What Are the Qualifications of Becoming a Pediatrician?.csv	0	1	3	23	0,115384615	9,063259149	0,34858689	0	0	0	12,91650828	0	11,41650828	1,131388684
40	27	Bin bag.csv	0	1	3	24	0,111111111	9,17437026	0,339791491	0	0	0	12,91650828	0	11,41650828	1,131388684
41	28	Anatomy of the Hip.csv	0	1	3	25	0,107142857	9,281513117	0,331482611	0	0	0	12,91650828	0	11,41650828	1,131388684
42	29	What is an Apostille?.csv	0	1	3	26	0,103448276	9,384961393	0,323619358	0	0	0	12,91650828	0	11,41650828	1,131388684
43	30	Problems Eating with Dentures.csv	0	1	3	27	0,1	9,484961393	0,31616539	0	0	0	12,91650828	0	11,41650828	1,131388684

Figure 5.1: Example Evaluation Table of
Set Based and Positions Based Evaluation Metrics

The query table's name is displayed at C12 in (5.1). The results are displayed in ascending order from 1 to 30 under C12, most similar to least similar. The cells that require adjustment are shown by the blue and green markings. The specific method by which this is accomplished is outlined as follows. For each result, we determine whether it is a correct result for the query table. If it is a correct result, we insert a "1" into the "TP" column of the result row, indicating a true positive. If it is an incorrect result, we insert a "1" into the "FP" column of the result row, as described in equation (2.8).

By considering the allocations to True Positives and False Positives, we can compute the Precision@K using the formula (2.9). Additionally, we can determine the average precision by taking the average of (2.8). The Precision@K is calculated by dividing the

cumulative true positive (column "F") by the sum of true positive and false positive, which is equal to the rank (column "B").

$$\text{Precision@K column "H"[14-42]} = \frac{\text{column "F"[14-42]}}{\text{column "B"[14-42]}} \tag{5.1}$$

H, B, F are the column names with the focus on the rows 14 to 42

The average precision is calculated by dividing the cumulative precision (column "I") by the sum of true positive and false positive (column "B"). The cumulative precision is calculated by adding the cumulative precision at the previous rank (cumulative precision at rank $(k-1)$, column "I") to the precision@k (column "H") of the present rank.

$$\text{Average Precision column "J"[14-42]} = \frac{\text{column "I"[14-42]}}{\text{column "H"[14-42]}} \tag{5.2}$$

J, I, H are the column names with the focus on the rows 14 to 42

For the position-based metrics, the relevance score and the descending relevance score column, both indicated in green, are crucial. We have assigned a relevance score between 0 and 3 to each result, where 0 indicates that it is not important, a false positive and 1 indicates that it is less relevant. 3 indicates that it is highly relevant. The Discounted Cumulative Gain, as defined in equation (2.10), is computed by

$$\text{DCG column "N"[14-42]}_k = \sum_{i=1}^{k} \frac{2^{\text{column "K"[14-42]}} - 1}{\log_2(\text{column "B"[14-42]} + 1)} \tag{5.3}$$

N, K are the column names with the focus on the rows 14 to 42

and the Ideal Discounted Normalised Cumulative Gain (2.11) in column "L" is the relevance score sorted in descending order. Out of these two columns the Normalised Discounted Cumulative Gain (2.12) is calculated as follows:

$$\text{NDCG column "Q"}_p = \frac{\text{column "N"}_p}{\text{column "P"}_p} \tag{5.4}$$

N, K are the column names with the focus on the rows 14 to 42

5.2 Datasets

For the evaluation we used two datasets. These two datasets **The dataset TREC**

Large-scale deep learning-based information retrieval is the main topic of the Microsoft-organized TREC 2019 Deep Learning Track [3]. Document ranking and passage ranking are its two primary functions. The dataset has a large number of training and test queries, with more than 3 million documents and approximately 9 million passages. The objective is to create and assess machine learning models—in particular, deep learning techniques—for efficient document and passage ranking. We randomly selected 17260 files out of the TREC Repository. Important to notice is that in almost every table file all the information is stored within a single cell. A whole text and therefore all the information to retrieve can be in just one cell. The file for the MinIO bucket folders can be found in the repository in the folder coffeetimePython and then minio data. Additionally the raw dataset we used in´s stored in the repository in the folder "trectables".

Common Web Tables

The WDC Web Table Corpus 2015 [21] by the University of Mannheim contains tables that have been extracted from web pages and stored in JSON format, along with relevant data and contextual information. These tables come in different sizes and consist of an extensive variety of subjects. Compareed to the TREC dataset, information is distributed over the whole table file. The entire corpus is 165 GB, while the relational corpus accounts for 69 GB. By utilising extractionBlender.py, we employed a random selection process to extract the content of JSON files and convert them into CSV files. Each CSV file was given a name based on its corresponding table. The file for the MinIO bucket folders can be found in the repository in the folder coffeetimePython and then minio data. For completetion the raw files are located within the BAJG2024python under "commmonwebcrawlerhuge". The files have been extracted from json files by the extractionBlender.py

LDA implementation within DataBlend

The Latent Dirichlet Allocation (LDA) part of the DataBlednd method starts by examining the word distribution in each file to determine the top 10 words that most aptly capture the primary subject matter of that file. Based on the results of Víctor Díaz Burgos Llaberia's master thesis, it was decided to concentrate on one topic per file [9]. This conclusion was continuously confirmed by additional test runs carried out within my data lake using the standard web crawl dataset from the University of Mannheim. It

became clear that, in most cases, the initial topic that was found included sufficient cohesive words to adequately describe a particular subject area. In our LDA implementation, we make use of the following imports to make the allocation possible:

- from sklearn.decomposition import LatentDirichletAllocation
- from sklearn.feature extraction.text import CountVectorizer

In our implementation we also processed the table, removed stop words, numbers and words with only one character. The following pseudocode is an abstract version of the real implementation.

Algorithm 1 New LDA Term Distribution from CSV

Require: bucket_name, csvName, numKeywords
Ensure: topic_words_and_weights

```
Initialize result list
if csvName is a valid CSV file then
    Load data from CSV
    Preprocess and split text into chunks if necessary
    if no valid text found then
        return empty result list
    end if
    Create document-term matrix
    Fit LDA model
    Extract top keywords and weights
    if fewer than numKeywords found then
        Pad with empty strings and zeros
    end if
    Store keywords and weights in result list any exception
    Log error
end if
return result list =0
```

Word Embedding in DataBlend

For the calculation of our embeddings we did not use the pretained model as Díaz [9] did in his work. The current implementation usese the pretrained model GloVe by Stanford University [23]. This origniates to a problem with the Genism package, which enables the use of the Word2Vec in python.

Algorithm 2 Calculate Embeddings

Require: all_keywords, all_weights, glove_model
Ensure: embeddings_float, new_weights_float
1: {Calculate embedding-weight pairs using a helper function}
2: embedding_weight_pairs ← embeddings_word2vec(all_keywords, all_weights, glove_model)
3: {Extract embeddings from the pairs}
4: embeddings ← Create an array of embeddings from embedding_weight_pairs
5: {Extract weights from the pairs}
6: new_weights ← Create an array of weights from embedding_weight_pairs
7: {Convert embeddings to float type array}
8: embeddings_float ← Convert embeddings to array of type float
9: {Convert new_weights to float type array}
10: new_weights_float ← Convert new_weights to array of type float
11: **return** embeddings_float, new_weights_float =0

Calculation of the representative vector

As was previously noted, a single descriptive topic is computed for each CSV file. A predetermined set of keywords is represented by this single subject vector. Ten terms that the Latent Dirichlent Allocation extracted from a file best characterise it, according to our estimations.
The corresponding distribution for each of these ten keywords may be seen in the attached CSV file. Two calculations, as shown below, can be used to condense the ten keywords into a single representative vector.

When $X = [x_1, x_2, ...x_n]$ has k distinct values, the **weighted average of vectors is calculated**

$X = [x_1, x_2, ...x_n]$

$$\mathbf{x}_{\text{weighted sum}} = \left(\sum_{i=1}^{n} w_i x_{i1}, \sum_{i=1}^{n} w_i x_{i2}, \ldots, \sum_{i=1}^{n} w_i x_{id} \right) \tag{5.5}$$

where:

x_{ij} : $j-th$ of the $i-th$ vector
w_i : weight of the $i-th$ vector
d : dimension of the vector

For the calculation within our implementation we provide as follows the corresponding pseudocode. In advance to this point the LDA has already allocated the keywords and the weights of the current focused file.

Algorithm 3 Calculate Representative Vector

Require: all_keywords, all_weights, glove_model
Ensure: weighted_embeddings

```
{Calculate embeddings and weights
all_embeddings, embeddings_weights ← calculateEmbeddings(all_keywords, all_weights, glove_model)
if all_embeddings.size == 0 or embeddings_weights.size == 0 then
    {Input arrays are empty, skipping computation.}
    weighted_embeddings ← {array of zeros with shape (1, 300)}
else
    {Normalize weights to sum to 1}
    weighted_embeddings ← average(all_embeddings, axis=0, weights=embeddings_weights)
end if
return weightedembeddings =0
```

5.3 The DataBlend Output with tsne and Silhouete scor

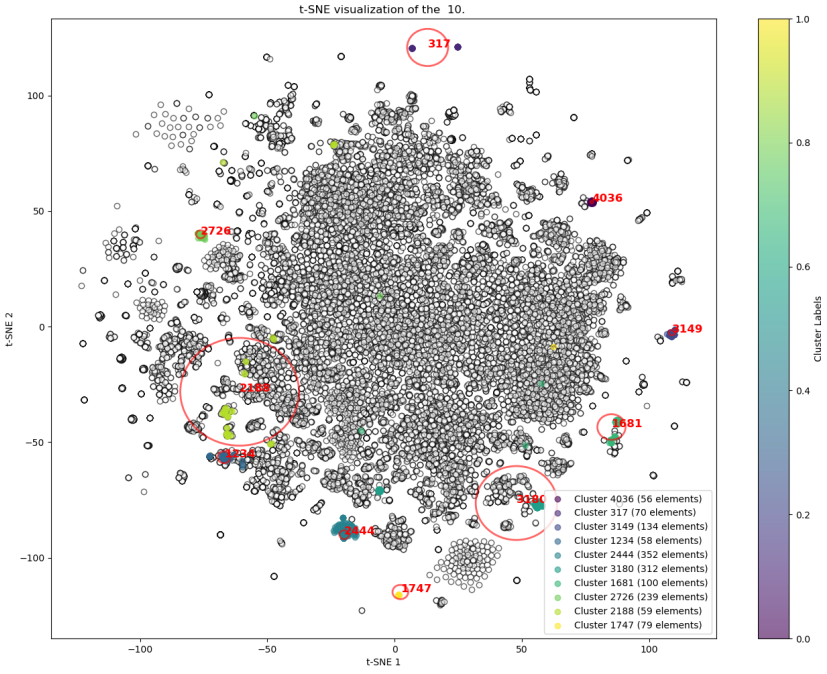


Figure 5.2: T-SNE evaluation of the result of the DataBlend Algorithm, Dataize: 29879 files, 16384 hashbuckets, Keywords 10, Topic 1,

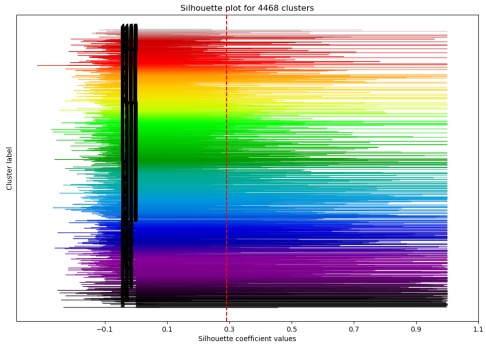


Figure 5.3: T-SNE evaluation of the result of the DataBlend Algorithm, 18100 keywords, 1024 hashtables, Keywords 20, Topic 1,

Mean Average Diagrams

NDCG Comparison

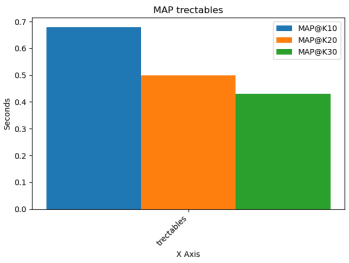
The findings illustrated in these pictures point to an important finding: the DataBlend technique outperforms the TAW-TFIDF strategy in terms of Normalised Discounted Cumulative Gain (NDCG), even in the case where the dataset contains missing values. This shows that DataBlend manages missing data well and keeps up strong performance.

It's crucial to remember, nevertheless, that a complete and accurate assessment of these techniques is made impossible by the absence of some code components. This restriction makes it more difficult for us to comprehend the underlying mechanics and the full scope of DataBlend's benefits over TAW-TFIDF. More research and the addition of the missing code are necessary to obtain thorough insights.

In conclusion, while initial results point to DataBlend's improved NDCG performance, further validation is required due to the inadequate dataset analysis in order to definitively corroborate these findings.

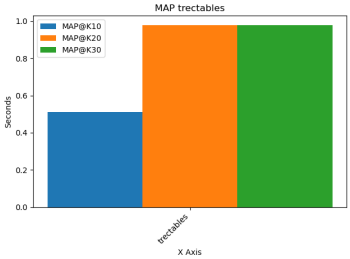
All relevant images and detailed results will be available in the repository for reference. Due to the impending deadline, this chapter concludes here. Regrettably, the incomplete analysis necessitates further exploration beyond the scope of this document.

In summary, while preliminary findings suggest DataBlend's enhanced performance in terms of NDCG and runtime efficiency, the incomplete dataset analysis and missing code components require additional validation to confirm these results conclusively.



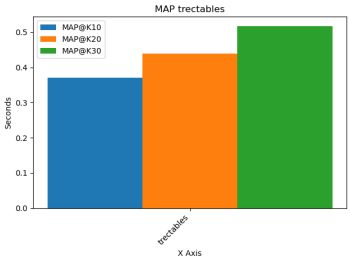
textwidth

Figure 5.4: MAP for the commonwebtables



[b]0.3

Figure 5.5: MAP for TREC tables



[b]0.3

Figure 5.6: Caption for Image 3

Figure 5.7:

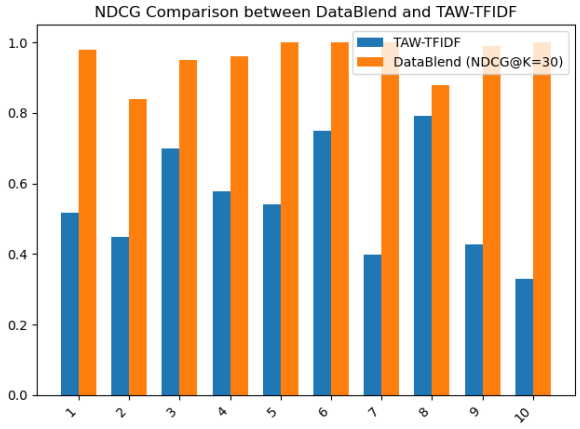


Figure 5.8: Caption

6 Discussion

6.1 Limitations

Evaluation Metrics

We used the allocation to results where experts determined whether the findings were positive (True Positive) or negative (False Positive)() in order to compare approaches using precision (2.8) and normalised discounted cumulative gain (2.12). We, the writers, were the experts in this case. We made advantage of the ChatGPT-4 big language model to minimise any potential bias and guarantee neutrality.

ChatGPT-4o compared the algorithms' output to our DataBlend for analysis. This approach reduced bias and enhanced objectivity in our analysis, notwithstanding the fact that we acknowledge our incomplete understanding of the particular dataset.

Furthermore, we did not include tables that should have come up in response to our query but did not because our study does not focus on False Negatives. This is because there isn't a "golden rule" for these kinds of evaluations.

With NDCG(2.12), a comparable problem arises. The DCG(2.10) and IDCG(2.11) assessments employ relevance ratings. 0 represents not relevant, 1 represents least relevant, 2 represents more relevant, and 3 represents very relevant. This is how we established the relevant range. These relevance scores were assigned by ChatGPT-4 in an effort to lessen bias and enhance objectivity and consistency in decision-making.

Overall, by enlisting specialists, our review satisfies academic requirements. We do, however, recognise the constraints brought about by our incomplete understanding of the dataset and the absence of a well-defined criterion for the relevance ratings of the top 30 results.

MetaData

At DataBlend, we strongly advise storing the representative vector for every file in the meta data. It is a useful method for managing and organising your data. The enormous size of the 300 dimensional vector in the reference indicates that the data has not been implemented in the meta data because it exceeds the capacity of the MinIO storage system. There can be variations due to the variation in MinIO versions. Each element

of the vector is important when determining similarity, so any reductions or changes to the vector are useless.

6.2 Contribution

Our thesis provides a thorough examination of the latest advancements in data lakes, specifically concentrating on the existing methods for ranking document files based on semantic similarity.

We have introduced DataBlend, a technology specifically designed to effectively blend data within a dynamic Data Lake. This thesis largely addresses the implementation of semantic similarity ranking. Utilising the Lambda architecture allows for the effective management of substantial amounts of data. The approach employs Latent Dirichlet Allocation for extracting data, and then calculates a representative embedding vector from the extracted keywords. The vectors are stored according to their positions relative to randomly chosen hyperplanes within the vector space. The approximate k-nearest neighbour approach provides a computationally fast solution for retrieving the most similar CSV tables to a query table.

The implementation and evaluation of the integration component and the speed layer were not conducted in this study. These components are essential for the concept of integration in a highly dynamic data lake, as changes may happen while the initial layers are still processing. Utilising the speed layer is anticipated to greatly improve the outcomes.

When choosing the elements of DataBlend for the purpose of evaluating semantic similarity, our main objective was to attain a high level of precision while also guaranteeing computing efficiency in order to provide fast outcomes.

DataBlend has been proven to surpass existing methods, such as Díaz's LDA and word embedding approach, in terms of accuracy and runtime, according to thorough comparisons. Nevertheless, although the semantic ranking of CSV files is effective, additional comparisons with other systems are required to bolster our conclusions.

Bibliography

[1] Latent dirichlent allocation, 2024. URL https://de.wikipedia.org/wiki/Latent_Dirichlet_Allocation. Accessed: 2024-04-31.

[2] Dirichlet distribution, 2024. URL https://en.wikipedia.org/wiki/Dirichlet_distribution. Accessed: 2024-06-11.

[3] N. C. Bhaskar Mitra. Trec 2019 deep learning track. https://microsoft.github.io/msmarco/TREC-Deep-Learning-2019, 2019. Accessed: 2024-05-09.

[4] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, 2003. URL http://jmlr.org/papers/v3/blei03a.html.

[5] R. Campos, V. Mangaravite, A. Pasquali, A. M. Jorge, C. Nunes, and A. Jatowt. *YAKE! Collection-Independent Automatic Keyword Extractor*, pages 806–810. Springer International Publishing, 2018. ISBN 9783319769417. doi: 10.1007/978-3-319-76941-7_80.

[6] M. Cherradi and A. El Haddadi. Enhancing data lake management systems with lda approach. *Journal of Data Science and Intelligent Systems*, 2024. doi: 10.47852/bonviewJDSIS42022312. URL http://ojs.bonviewpress.com/index.php/jdsis/article/view/2312.

[7] J. Dixon. James dixon's blog - pentaho, hadoop, and data lakes, 2010. URL https://jamesdixon.wordpress.com/2010/10/14/pentaho-hadoop-and-data-lakes/. Accessed: 2024-03-17.

[8] I. Docker. Docker watermark. https://www.docker.com/company/newsroom/media-resources/. Accessed: 2024-06-11.

[9] V. Díaz De Burgos Llaberia. Enhancing table discovery and similarity evaluation in data lakes. Master's thesis, Utrecht University, 2023. URL https://studenttheses.uu.nl/handle/20.500.12932/44314.

[10] M. Edalatfard, M. M. Zanjireh, and M. Bahaghighat. A new weighting scheme for document ranking based on the modified word-embedding method. *Engineering and Applied Science Research*, 51:259–266, 2024. doi: 10.14456/EASR.2024.25.

[11] G. Fan, R. Shraga, and R. J. Miller. Gen-t: Table reclamation in data lakes. *CoRR*, abs/2403.14128, 2024. doi: 10.48550/ARXIV.2403.14128.

[12] J. Gesk. Bajg2024python, 2024. URL https://github.com/JohannesGe98/BAJG2024python.git. Accessed: 2024-06-12.

[13] F. Inc. fasttext, 2022. URL https://fasttext.cc/docs/en/crawl-vectors.html. Accessed: 2024-06-12.

[14] A. Khatiwada, R. Shraga, W. Gatterbauer, and R. J. Miller. Integrating data lake tables. *Proceedings of the VLDB Endowment*, 16(4):932–945, Dec. 2022. ISSN 2150-8097. doi: 10.14778/3574245.3574274.

[15] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to information retrieval*. Cambridge University Press, 2008. ISBN 978-0-521-86571-5. doi: 10.1017/CBO9780511809071. URL https://nlp.stanford.edu/IR-book/pdf/irbookprint.pdf.

[16] N. Marz and J. Warren. *Big Data: Principles and Best Practices of Scalable Realtime Data Systems*. Simon and Schuster, Shelter Island, NY, 2015.

[17] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. In Y. Bengio and Y. LeCun, editors, *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, 2013. URL http://arxiv.org/abs/1301.3781.

[18] I. MinIO. Minio watermark. https://min.io/logo. Accessed: 2024-06-11.

[19] S. Momtazi and F. Naumann. Topic modeling for expert finding using latent dirichlet allocation. *WIREs Data Mining and Knowledge Discovery*, 3(5):346–353, Aug. 2013. ISSN 1942-4795. doi: 10.1002/widm.1102.

[20] F. Nargesian, E. Zhu, R. J. Miller, K. Q. Pu, and P. C. Arocena. Data lake management: Challenges and opportunities. *Proc. VLDB Endow.*, 12(12):1986–1989, 2019. doi: 10.14778/3352063.3352116. URL http://www.vldb.org/pvldb/vol12/p1986-nargesian.pdf.

[21] U. of Mannheim. Wdc web table corpus 2015, 2024. URL http://webdatacommons.org/webtables/2015/downloadInstructions.html. Accessed: 2024-04-14.

[22] P. N. Sawadogo and J. Darmont. On data lake architectures and metadata management. *CoRR*, abs/2107.11152, 2021. doi: 10.48550/arxiv.2107.11152. URL https://arxiv.org/abs/2107.11152.

[23] S. University. Glove: Global vectors for word representation, 2024. URL https://nlp.stanford.edu/projects/glove/. Accessed: 2024-05-03.

[24] M. Zaharia, A. Ghodsi, R. Xin, and M. Armbrust. Lakehouse: A new generation of open platforms that unify data warehousing and advanced analytics. In *11th Conference on Innovative Data Systems Research, CIDR 2021, Virtual*

Event, January 11-15, 2021, Online Proceedings. www.cidrdb.org, 2021. URL http://cidrdb.org/cidr2021/papers/cidr2021_paper17.pdf.

7 Declaration

Declaration of authorship

I hereby declare that I have prepared this thesis using no sources and aids other than those indicated. This thesis has not been submitted in its present or any similar form to any department or equivalent institution of any domestic or foreign university.

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich diese Arbeit unter Verwendung keiner anderen als der angegebenen Quellen und Hilfsmittel angefertigt habe. Die vorliegende Arbeit wurde weder in ihrer jetzigen noch in einer ähnlichen Form bei einem Fachbereich oder einer entsprechenden Institution einer in- oder ausländischen Hochschule eingereicht.

Karlsruhe, 12.06.2024

Ort, Datum Unterschrift