An In-Depth Exploration of the High-Quality Entity Linking for Information Retrieval: MMEAD

by

Luyun Lin

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2023

Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

Chapter 3 is based on the co-authored work published in [29]. I declare that I am responsible for the code contribution, conducting of experiments, and paper writing.

Abstract

Entity linking has emerged significantly during the digital information explosion, aiming to provide context and meaning to huge amounts of unstructured data. While traditional information retrieval primarily relied on keyword-based searches, it often yielded contextually insufficient and ambiguous results. Entity linking fundamentally involves connecting distinct mentions to specific entities in a knowledge base. This process not only resolves ambiguities but also enriches the contextual understanding of the data by associating named entities with their corresponding entries in the knowledge graph.

To truly harness the potential of entity linking, we must explore this concept within specific contexts and scenarios, which necessitates the availability of robust benchmark datasets that reflect the complexities of real-world information. MS MARCO serves as a key benchmark and resource for the development of deep learning models in the domain of information retrieval. It offers a distinctive chance to observe entity linking in practice and to test its effectiveness across a variety of situations. In this background, we introduce and emphasize the MS MARCO Entity Annotations and Disambiguations (MMEAD), a framework uniquely designed to bridge MS MARCO collections with state-of-the-art entity linkers. Grounded in the solid foundation of Wikipedia knowledge graphs, MMEAD prioritizes user-friendliness, precision, extensibility, and comprehensive metadata provision. Its data representation through the intuitive JSONL files ensures a seamless entity-linking experience.

Addressing the challenges in information retrieval, the research utilizes MMEAD to explore expansion via entity-linked terms, fine-tuning both sparse and dense retrieval techniques. This entity expansion approach to data augmentation aims to align more closely with the real user intentions.

Furthermore, this work integrates the strengths of MMEAD with powerful systems such as Faiss and DuckDB, dissolving the barriers between structured and unstructured data searches into a unified comprehensive search framework, providing enhanced data categorization, entity frequency assessments, and more, pointing toward a transformative shift in data retrieval and management systems. It also demonstrates the merging of MMEAD with Wikidata capitalizing on the strengths of open-linked data to offer a rich, synthesized view of global information.

Acknowledgements

I want to express my gratitude to my supervisor Prof.Jimmy Lin for his invaluable guidance throughout my research. His insights have broadened my perspective, deepening my understanding beyond my specific work. From before I even began my master's program, Prof. Lin provided hands-on guidance that shaped my career path in NLP research. His consistent support has not only polished my research skills but also influenced my personal growth. I'm truly honored to have had the chance to work under his supervision.

I would also like to thank the readers of my thesis, Professor Charles Clarke and Professor Tamer Özsu, for dedicating their valuable time to review my work.

Then, I want to express my gratitude for the opportunity to be a part of RSVP.ai. It was here that I was first led to the world of Natural Language Processing (NLP). My experience at the company not only deepened my knowledge but also provided me with invaluable hands-on experience on NLP projects. I'm especially thankful for my amazing colleagues. Their continuous encouragement, guidance, and support played an instrumental role during my time at the company. Their friendship made every challenge seem achievable and turned our workplace into a nurturing environment for growth and learning.

Dedication

This is dedicated to my families for their unconditional love.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Entity linking has become increasingly important in today's digital environment. Given the explosive growth of information on the internet and the complexity of unstructured data, the need to connect entities to their corresponding real-world meaning has become more crucial. In the past, information retrieval was primarily keyword-based, with search engines returning results based solely on the presence of specific words or phrases. This approach often led to results that lacked context and were ambiguous. Entity linking, at its core, refers to connecting distinct pieces of text or data points to specific entities in a knowledge base, offering a structured and meaningful context to unstructured information. This process involves identifying named entities within a text, such as people, organizations, locations, and concepts, and linking them to their corresponding entries in a knowledge graph or database. This not only helps in disambiguating a term's meaning but also facilitates a richer understanding of the context in which it is used.

The rise and success of deep learning in various NLP fields have led to the emergence of deep learning-based entity linking strategies [64, 68]. In contrast, traditional machine learning methodologies face challenges in two primary areas: first, achieving high-performance features requires extensive amounts of detailed and labor-intensive feature engineering; second, the reliance on selected knowledge base and domain knowledge during feature design makes it difficult to generalize the trained model across others [68]. Recent trends highlight the growth in deep learning-oriented entity linking techniques, reducing the need for manual feature creation. Deep learning has become the leading framework for state-of-the-art entity linking methods, excelling in autonomously identifying key features and adapting across domains [64, 67, 68].

Entity linking is pivotal in addressing challenges associated with information retrieval [22,

16, 86]. One common challenge in information retrieval is the "vocabulary mismatch" problem [20], which occurs when users phrase searches differently than the terminology in relevant passages. In this scenario, entity linking acts as a bridge by identifying and unifying named entities in the text. One of the effective strategies to counteract this problem is expansion. Through the selection and addition of candidate terms, expansion refines queries or passages, ensuring they align more closely with the user's intended search, thereby yielding more accurate results.

On the other hand, while neural-driven entity linking methods offer enhanced performance, they also come with specific challenges [64, 68]. A notable concern is their considerable computational power requirement. These methods often need advanced hardware, which might not always be available or financially feasible. The complexity and depth of neural models, especially those like Transformers [80], require robust hardware setup, often involving specialized GPUs. Additionally, many of these studies continue to depend on external sources [4, 47, 53, 57]. Such dependence can reduce the flexibility and adaptability of systems when these resources are unavailable or constrained. Furthermore, the complexity inherent in these models can impact computational speed. Neural operations, especially within deep learning processes, consume significant processing time, potentially slowing down real-time applications and development processes and affecting workflow efficiency.

To explore the potential of entity linking and leverage its power to enhance data comprehension effectively, it is crucial to engage with the right datasets and experimental frameworks. These should be capable of addressing the complexities of real-world data, serving as benchmarks for both development and evaluation. Contextual exploration and validation of entity linking demand not only advanced algorithms but also extensive, annotated datasets that can harness the full capabilities of these advanced entity linking systems. Nonetheless, acquiring datasets that are robust, comprehensive, and accurately annotated has always been a challenge. A robust dataset not only provides the materials that enable a deeper dive into the complexities of the task, but also provides efficient analytical operations and fast data access.

The richness of data sources such as the MS MARCO collections [50] offers a prime opportunity to observe entity linking in action and to challenge its capabilities in diverse scenarios. The MS MARCO datasets have emerged as benchmarks for evaluating deep learning techniques in Information Retrieval, serving not only for benchmarking purposes but also as valuable tools for zero-shot and few-shot learning across a range of retrieval tasks. However, neural IR models based on MS MARCO's text often struggle with complex concepts in the real world [63]. Entity Linking, which plays a pivotal role in associating text with such concepts, can pinpoint references within the text and map them to entries in a knowledge graph. Nonetheless, one of the challenges to adopting neuro-symbolic in IR

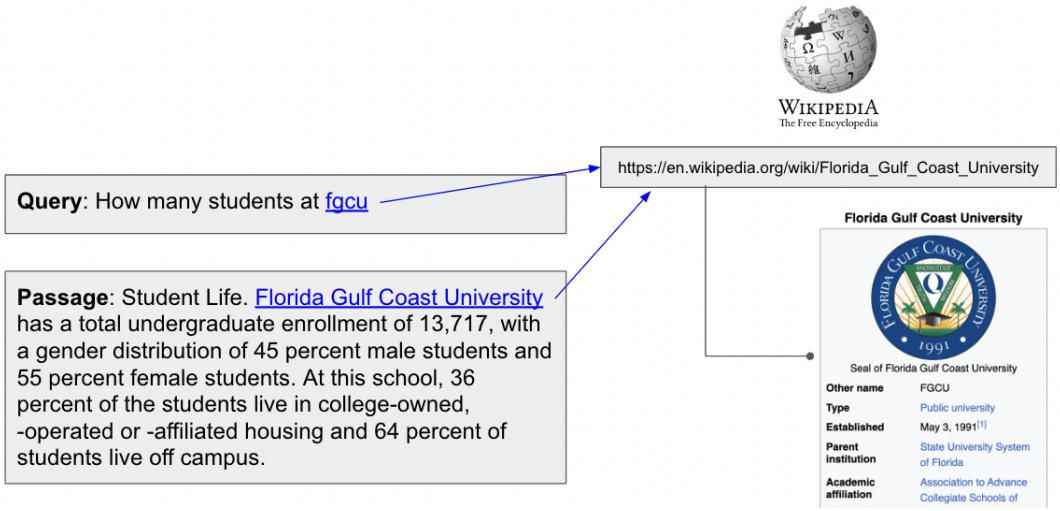

Figure 1.1: MMEAD entity connection found in the MS MARCO v1 query and passage.

is the complexity and computational challenge of annotating large datasets with entities. To meet these needs and advance both the research and practical applications of entity linking, here the MS MARCO Entity Annotations and Disambiguations (MMEAD) [29] comes into play. It bridges the gap between advanced entity linking approaches and the practical challenges of real-world information retrieval by integrating the MS MARCO collections (versions 1 and 2) [50] with cutting-edge entity linkers, namely REL [79] and BLINK [36]. MMEAD aims to address this by offering accessible entity annotations for the MS MARCO collections. The example Figure 1.1 illustrates how the entity links in MMEAD can bridge the gap between a query and the corresponding passages in the MS MARCO dataset.

MMEAD is constructed using knowledge graphs derived from Wikipedia dumps and prioritizes user accessibility, high-quality entity links, extensibility, and the inclusion of valuable metadata. Essentially, it presents data in user-friendly JSONL files that follow the MS MARCO v2 collections format. This design ensures seamless entity link identification and guarantees precision by integrating with top-tier entity link algorithms and systems. Through this thesis, I will delve deeper into MMEAD's objectives, design principles, and its potential to tackle real-world challenges.

By utilizing the high-quality entity linking dataset MMEAD, my primary focus is on expansion through entity-linked terms. In this context, the information retrieval system

augments queries or passages by incorporating terms associated with entities found in the text. I explore this approach in both sparse and dense retrieval contexts.

The integration of structured and unstructured data has always been a challenge in the world of data management. Traditional two-stage search methods can struggle to extract cohesive information from mixed data sources.

With MMEAD, it does not just search for keywords or strings but actively understands the mentions and entities. This deep understanding boosts both search accuracy and contextual relevance. Building upon the functionality of the MMEAD entity, showcases have been developed using the strengths of Faiss (Facebook AI Similarity Search) [27] and DuckDB [56]. Faiss specializes in efficient similarity search and clustering of dense vectors, making it a prime choice for managing and retrieving unstructured data. DuckDB, on the other hand, is an analytical data management system optimized for complex queries on large-scale structured datasets. Integrating these systems facilitates a revolutionary approach. Instead of separating searches between structured and unstructured data, we now have a seamless, one-fit-all search method. The search happens concurrently, utilizing the strengths of each component. The result is a smooth, efficient, and highly accurate retrieval system.

Beyond simple searches, We demonstrate how to use MMEAD to enhance interactive search applications. The system can categorize passages by entity requests, aligning them with user intentions. Additionally, it can produce entity frequency tables, offering a detailed view of specific data occurrences, thus enhancing analysis precision. The blend of MMEAD with vector search is not just an upgrade but an insight approach for next-generation data systems that promise intelligence, connectivity, and intuitive insights. Furthermore, the integration of Wikidata with MMEAD maps demographic patterns, digital divide, and illustrates the capability of open-linked data to merge diverse datasets into a singular, insightful visualization that can enhance our understanding of spatial narratives and inform a more equitable digital representation.

1.1 Contributions

The main contributions of this work are as follows:

- Introduce a comprehensive resource MMEAD, offering reusable entity links for the MS MARCO document and passage ranking collections. This resource is enhanced by state-of-the-art entity linking tools to ensure precise annotation. It's readily available

as a Python package on PyPI. While MMEAD aims to support neuro-symbolic IR research and enhance neural retrieval models, it also tackles the challenges posed by the computational demands of advanced neural-empowered entity linking methods.

- Demonstrate the usage of MMEAD significantly enhances retrieval effectiveness for both sparse and dense retrieval, with notable improvements observed for hard queries that were under-performed with the text-only method.

- Demonstrate the capabilities of MMEAD within a vector search database. By integrating MMEAD with tools like Faiss and DuckDB, we've established a dynamic search that combines structured and unstructured data. This application not only retrieves information rapidly but also categorizes and evaluates the significance of specific segments. As a result, it emphasizes key concepts and crucial details within the searched information.

- Showcase the application of data in geographical contexts by plotting all MMEAD entities with available geographical data from the MS MARCO v2 passage collection on a static map.

1.2 Thesis Organization

This thesis is structured as follows:

Chapter 2 provides the foundational knowledge by introducing essential background information, general terms, necessary techniques and concepts.

In Chapter 3, we delve into MMEAD, discussing its objectives, motivations, and foundational structure.

Chapter 4 takes a practical turn, presenting experiments conducted on MMEAD for both sparse and dense information retrieval, accompanied by results and observations.

Chapter 5 showcases the integration capabilities of MMEAD with vector search and traditional database systems, alongside with an illustrative example of geographical context application.

Finally, Chapter 6 wraps up the thesis, reflecting on potential future research and development directions.

Chapter 2

Background

In this section, I will introduce the basic background, general terms, key concepts, and techniques required to understand the workflow.

2.1 Knowledge Graph

Knowledge graph is a powerful tool for integrating information, providing a structured and dynamic representation. It is a structured collection of organized data that can be accessed for particular details. A Knowledge graph is a Knowledge base that is typically constructed using three distinct and infinite sets: E (Entities), P (Predicates), and L (Literals). Entities represent distinct objects, concepts, or predicates that can be uniquely identified, either physically or abstractly [59]. A predicate acts as a relational link or property that connects entities or binds an entity to a value. A literal is an atomic, indivisible value that represents basic pieces of information. Unlike entities, literals serve as the foundational data points, describing attributes or properties of entities [59, 8, 17]. Knowledge graphs are key to the Semantic Web, enabling the interlinking of diverse datasets to form a vast, global knowledge graph. This term gained popularity with Google's semantic search initiative, which focused on understanding and connecting things, not just keywords [1]. Knowledge graphs are characterized by their descriptive nature of real-world entities, schema-defined classifications, the capacity for linking any entities, and their coverage across multiple domains. Moreover, knowledge graphs can assist with advanced features such as question answering [10, 44], information retrieval [14], etc.

[1] https://blog.google/products/search/introducing-knowledge-graph-things-not/

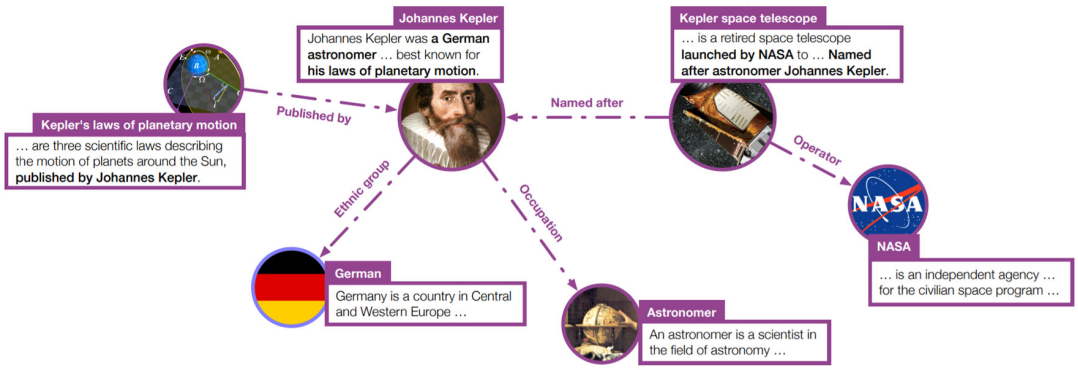

Figure 2.1: Example Knowledge graph with entity descriptions.

Some popular knowledge graphs include DBpedia [3], which extracts structured content from Wikipedia; YAGO [73, 74], which integrates information from various sources like Wikipedia and WordNet [48]; and Wikidata [81], a collaboratively edited knowledge graph hosted by the Wikimedia. Figure 2.1 shown an example of a KG with entity descriptions sourced from Wikidata5m[83].

2.2 Entity Linking

Entity Linking refers to the process of extracting the mentioned terms from a text and connecting them to the corresponding entries in a knowledge base. The primary objective of entity linking is to disambiguate textual mentions and accurately associate them with their appropriate identifiers in a reference source. Such associations enhance the semantic richness of textual data and establish a bridge between unstructured content and structured databases.

Its applications span a diverse range of domains. For instance, in information extraction, entity linking clarifies ambiguities related to named entities, thereby improving the quality of the data extracted [31]. In information retrieval, it enhances traditional keyword-focused searches by accurately understanding the specific semantic meaning of user queries [65, 87, 60]. In content analysis, especially in platforms such as recommendation systems [92, 49, 13], entity linking helps in providing content aligned with a user's interest in particular entities. Moreover, question answering systems rely on entity linking to better understand user queries, enabling more contextually relevant responses [97, 69].

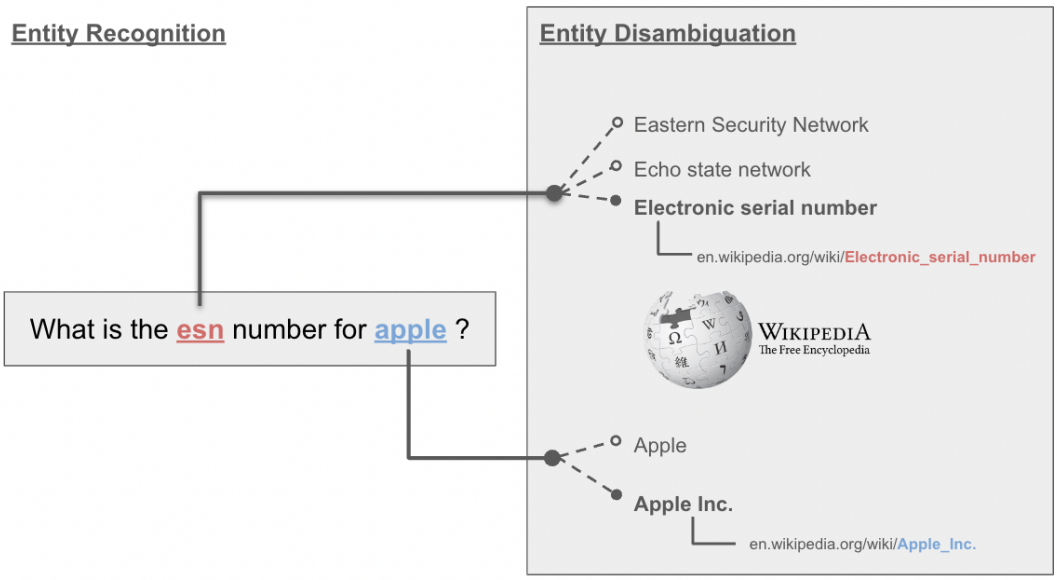

Figure 2.2: Entity Linking process, showcasing steps of Entity Recognition to identify entity candidates followed by Entity Disambiguation to accurately link each entity to its unique identifier in Wikipedia.

There are two main challenges associated with entity linking: Entity Recognition and Entity Disambiguation. Entity Recognition is the process of identifying and classifying entities present in text, often categorize them into predefined categories such as persons, organizations, locations, dates, and more. This task primarily focuses on identifying spans in sentences that correspond to these unique entities [45]. On the other hand, entity disambiguation seeks to link the recognized entities in unstructured content to their appropriate identifiers in a structured database. This task focuses on determining the correct and unique reference for each entity based on its contextual information, given that many entities can have multiple potential references. The challenge is in accurately aligning the recognized entity with its intended representation within the knowledge base, ensuring a match between the information extracted from the text and the existing database [15, 75]. As shown in Figure 2.2, the Entity Recognition process identifies *esn* and *apple* as entities. Subsequently, the Entity Disambiguation process determines the correct references for these entities, where *esn* refers to *Electronic Serial Number* and *apple* is linked to *Apple Inc.*

2.3 Information Retrieval

Information Retrieval (IR) involves extracting relevant data from large collections based on specific user queries, matching user queries to relevant pieces of data sourced from documents or passages in large pre-build databases [71]. The primary IR models include the Boolean, Vector Space, Probabilistic, and Inference Network models. The Boolean model utilizes Boolean algebraic operations such as AND, OR, and NOT to define relations between terms and documents, though it lacks the capability to rank the retrieved documents [34, 37]. In the Vector Space model, both documents and queries are represented as vectors within a multidimensional space, and they are ranked based on their cosine similarity, incorporating a TF-IDF weighting scheme [70]. The Probabilistic model ranks documents based on the probability of their relevance to a given query, using binary vectors to represent documents and queries [19]. The Inference Network model [78] treats document retrieval as an inference process within a network and scores documents based on term strength [23, 62, 93].

BM25 and Dense Passage Retrieval (DPR) are currently among the popular techniques in the field of information retrieval [26, 77, 46], widely recognized for their effectiveness in handling retrieval tasks.

2.3.1 BM25

BM25 is a popular and widely used algorithm in the field of information retrieval. It stands as a more advanced evolution of the TF-IDF (Term Frequency-Inverse Document Frequency) approach [58]. BM25, part of the family of probabilistic retrieval models, is designed to rank documents based on their relevance to a given search query. At its core, BM25 calculates the relevance score of a document to a search query using two main components: term frequency (TF) and inverse document frequency (IDF). The term frequency component measures how often a query term appears in a document, assuming that the more frequent a term is in a document, the more relevant the document is to that term. The inverse document frequency component assesses the importance of the term across the entire collection, with rarer terms being given more weight as they are considered more significant [28, 61, 58]. This algorithm is particularly effective in large-scale text databases, where it can efficiently sift through vast amounts of data to find the most relevant documents based on the user's query.

2.3.2 Dense Passage Retrieval

Dense Passage Retrieval (DPR) is an advanced method in the field of information retrieval, aiming to enhance the effectiveness and accuracy of extracting relevant information within large collections of text. In contrast to traditional bag-of-word search methods, it utilizes deep learning models, especially transformers, to understand and match the context and semantics of queries with passages. The core concept of DPR is to represent both the query and the passages in a high-dimensional vector space using neural network models. These models are trained to project the queries and the passages into a space where the semantic similarity between them can be measured effectively. The key advantage of DPR is its ability to capture the underlying meaning of the text, rather than relying solely on keyword matches. This allows the model to compare and rank passages based on their semantic similarity to the query, effectively addressing the "vocabulary mismatch" [20] issue commonly encountered in sparse retrieval techniques [30, 94, 39, 5, 88, 95].

2.4 MS MARCO

MS MARCO, MAchine Reading COmprehension dataset [50], is a series of datasets in large-scale specifically constructed for the development and evaluation of machine reading comprehension systems. This dataset has been influential in the field of natural language processing (NLP) and has played a pivotal role in advancing the state of the art in machine understanding of text. The uniqueness of MS MARCO lies in its composition, which is based on real-world data. The dataset itself is composed of anonymized questions sampled from Bing's search queries along with multiple candidate answers sourced from real documents. These documents include a variety of sources like web pages, providing a rich and diverse set of language usage and topics. One of the key aspects of MS MARCO is that it includes not only the questions and answers but also the human-generated answers. The structure of MS MARCO supports deep learning models to be trained in a way that mirrors the unpredictability and variety of questions posed by real-world users, a significant departure from datasets composed of synthetic questions.

The dataset contains 1M anonymized questions with human-generated answers and 8.8M passages extracted from web documents. One of the most impactful tasks of this dataset is passage ranking, created by combining all passages from the MS MARCO dataset and pairing them with relevant questions and passage identifiers. These passage and question collections for an ad-hoc retrieval task at TREC. [2]

[2] https://trec.nist.gov

The following is the example of the passage, query and relevant judgment *qrel*, which labels each query alongside its corresponding relevant passages.

what is paula deen's brother

Query 1048585

Over the last decade, *Costa Rica* has evolved from being a mere eco-tourism destination and emerged as a country of choice for foreigners, particularly from *United States* and *Canada*. These seek quality healthcare services and surgeries at a much lower price than their home countries.

Passage 48 - MS MARCO Passage collection v1

300674 0 7067032 1 125705 0 7067056 1 94798 0 7067181 1 ...

Qrel Relevence Judgement

The release of MS MARCO has had a substantial impact on AI research. It has served as a benchmark for numerous machine reading comprehension competitions and shared tasks, helping to push the boundaries of what deep learning models can achieve in terms of language understanding. The tasks derived from MS MARCO test a system's ability to handle ambiguity, reasoning, and the synthesis of information across different parts that are crucial for a wide range of applications.

2.4.1 MS MARCO Chameleons

MS MARCO Chameleons [2] consist of challenging queries from the original MS MARCO passage dataset. These queries are determined by state-of-the-art rankers, BM25, DeepCT [11], DocT5Query[51], RepBERT [96], ANCE [89] and TCT-ColBERT [42]. These rankers show poor effectiveness in finding relevant matches for these queries, so the testing focuses on the bottom 50% of the worst-performing queries from the subsets of Veiled Chameleon (Hard), Pygmy Chameleon (Harder), and Lesser Chameleon (Hardest), which represent increasing levels of difficulty.

- Veiled Chameleon: It comprises 3,119 queries that were consistently among the bottom 50% of the worst-performing queries across at least 4 rankers.
- Pygmy Chameleon: It comprises 2,473 queries that were consistently among the bottom 50% of the worst-performing queries across at least 5 rankers.
- Lesser Chameleon: It comprises 1,693 queries that were consistently among the bottom 50% of the worst-performing queries across at least 6 rankers.

2.5 DuckDB

DuckDB [56] is a high-performance analytical database system, designed for speed, dependability, portability, and user-friendliness. Beyond the foundational SQL features, DuckDB offers an enriched SQL language variant that supports a range of complex data structures and file formats.

Efficiency: High efficiency for OnLine Analytical Processing (OLAP) [6, 55] workloads while maintaining reasonable OnLine Transaction Processing (OLTP) [55] performance. This balance makes it suitable for scenarios that entail concurrent data modifications and visualization-driven OLAP queries.

Stability: System stability is crucial, as a crash in the embedded database should never lead to the host system's failure; instead, queries should be abort-able and resource contention should be managed gracefully.

Transferring: Efficient data transfer between the database and application, facilitated by their shared process and address space, there's a significant potential for optimization.

Portability: The database is both embeddable and portable. It operates seamlessly in the host environment without the need for external library dependencies or process state modifications while handling the signals.

2.6 Vector Database

Vector database have become crucial due to the increasing need to digitally represent complex data like text, images, and videos. This need is increasingly in areas such as recommendation engines, search engines, and question answering systems. The data is

represented using numerical vectors, which are cost-effective in terms of storage and comparison. However, its high dimensionality and sparsity require custom solutions for effective storage, retrieval, and conducting mathematical tasks over these vectors. [76]

There are two categories for vector database: specialized and one-size-fits-all.

Specialized systems are optimized for contemporary computing infrastructures, leveraging both CPUs and GPUs to ensure peak efficiency. These systems support a wide range of query types, from vector similarity searches with diverse similarity functions to attribute filtering and multi-vector query processing. They often feature various indexing options and a flexible interface for integrating new indexes. With an emphasis on high-dimensional vectors, these systems guarantee scalability and consistent availability across multiple systems [82, 85].

On the other hand, the "one-size-fits-all" offers a general-purpose solution that aims to cater to a wide range of use cases related to vector embeddings. These databases are designed with the belief that a single set of features and tools can effectively address diverse needs. For instance, a customer on an e-commerce platform might search for a dress not only based on visual similarity but also using structured attributes like price. [84]

Chapter 3

MMEAD

MMEAD [29], stands for MS MARCO Entity Annotations and Disambiguations. It is a Python-based resource package that offers entity links for the MS MARCO datasets. MMEAD provides a standardized format to store and share links for documents and passages across both MS MARCO collections (v1 and v2) [50]. These entity links are generated using state-of-the-art entity linking tools, namely REL [79] and BLINK [36], leveraging knowledge graphs from Wikipedia. MMEAD aims to simplify the process of loading link data and entity embeddings, making it easy for users to work with.

The design principles of MMEAD are derived from these goals:

- Easy-to-use: Interacting with linked entities should be straightforward. Ideally, only a handful of code lines should be necessary to access data, its textual occurrences, and representations. MMEAD data is made publicly accessible in the user-friendly JSONL format, which aligns with the MS MARCO v2 collections [50]. Each JSON line contains entity links for a document or passage, identifiable via unique identifiers. These lines have distinct JSON fields for each section's entities, body, header, and title. For every entity, *entity_id*, *start_pos*, *end_pos*, *entity*, and *details* fields are available. Notably, the *details* field is a JSON object offering linker-specific details, like the entity type from the NER module and the confidence level.

- High-quality: A high quality entity links for the MS MARCO collections [50] enable the application of models and reasoning over the entities. MMEAD offers entity links generated by leading entity linking systems. This includes links from REL for both MS MARCO v1 and v2 passages and documents and from BLINK [36] for MS MARCO v1 passages. Given the high precision of these systems and their

dependency on reputable knowledge bases, the accuracy of detected mentions and their corresponding entities is assured.

- Extensibility: The MMEAD framework promises effortless integration of the entity data with other entity linking systems. Following the MMEAD entity link format, it allows the MMEAD Python library to work directly with any system sharing links similarly. Notably, REL [79] provides comprehensive guidelines for updating newer knowledge base versions, enable the easy releases of links to newer versions.

- Useful metadata: with supplementary data that beneficial for experiments, this implies linking entities to their corresponding identifiers and associated latent representations. Specifically, Wikipedia2Vec embeddings [90] are released with both 300d and 500d feature vectors, a mapping of entities to their identifiers is provided alongside.

3.1 Example

Start from an example of MMEAD data, here is a text span from MS MARCO V1 PASSAGE [50] collection, there are a few mentions in the text that can be linked to Wikipedia.

> Over the last decade, *Costa Rica* has evolved from being a mere eco-tourism destination and emerged as a country of choice for foreigners, particularly from *United States* and *Canada*. These seek quality healthcare services and surgeries at a much lower price than their home countries.

In the JSON form output from MMEAD [29], the above entities will be represent as:


```
{
   "passage":[
      {
         "entity_id":5551,
         "start_pos":22,
         "end_pos":32,
         "entity":"Costa Rica",
         "details":{
            "tag":"LOC",
            "md_score":0.9983808696269989
         }
      },
      {
```

```
            "entity_id":3434750,
            "start_pos":156,
            "end_pos":169,
            "entity":"United States",
            "details":{
               "tag":"LOC",
               "md_score":0.9943509995937347
            }
         },
         {
            "entity_id":5042916,
            "start_pos":174,
            "end_pos":180,
            "entity":"Canada",
            "details":{
               "tag":"LOC",
               "md_score":0.9999330043792725
            }
         }
      ],
      "pid":48
}
```


where:

- *pid*: identifier of passage in the collection
- *passage*: Linked entities in list
- An entity is presented as:
  - *entity_id*: internal entity identifier which can be mapped to corresponding wikipedia identifier
  - *start_pos* the start position of mention found in text
  - *end_pos* the end position of mention found in text
  - *entity* Entity found in the text
  - *details* Linker specific information, confidence etc

3.2 Entity Links

In this section, I will describe the systems used to generate entity annotations for the MS MARCO collections [50] in the context of MMEAD.

3.2.1 Radboud Entity Linker

The Radboud Entity Linker (REL) [79] is an open-source toolkit designed for entity linking. Built on state-of-the-art methods and packages from advanced natural language processing research, REL efficiently maps text mentions to their corresponding entities in knowledge bases. Its architecture follows conventional entity linking pipelines and consists of three components: (i) Mentions Detection, (ii) Candidates Selection, and (iii) Entity Disambiguation.

Mentions Detection

The objective of mention detection is to identify text spans, or "mentions," that may be linked to specific entities. To achieve this, a Named Entity Recognition (NER) tool is used, particularly the state-of-the-art NER tool called Flair [1], which is built on contextualized word embeddings. By allowing the replacement of NER tools, users can replace Flair with other NER tools like spaCy[1] or dictionary-based methods that best suit their needs.

Candidates Selection

REL select up to 7 top candidate entities from the mentions provided in the previous step. The first 4 out of these top 7 are ranked by $P(e|m)$ as $min(1, P_{Wiki}(e|m) + P_{YAGO}(e|m))$ for given entity e and mention m, where $P_{Wiki}(e|m)$ is estimated by summing the hyperlink counts from Wikipedia and CrossWikis [72]. On the other hand, $P_{YAGO}(e|m)$ is the uniform probability derived from YAGO dictionary[73]. The remaining 3 entities are ranked based on context similarity, described by $e^T \sum_{w \in c} w$, where c is 50-words context surrounding mention m, and both w and e are entity and word embedding vectors provided by Wikipedia2Vec [90].

[1] https://spacy.io

Entity Disambiguation

The final step involves entity disambiguation. REL associates mentions with their corresponding entities in the Wikipedia knowledge graph. REL's approach to entity disambiguation is based on the Ment-norm method [35]. A two-layer neural network is utilized to combine $P(e|m)$ with the max-marginal likelihood of an entity related to a document.

3.2.2 BLINK

BLINK [36], which stands for Bi-directional Linking of Nodes using Kernels, is one of the state-of-the-art entity linking systems designed to map textual mentions in documents to distinct entities within large knowledge bases. BLINK offers efficient and precise entity identification, and annotating text in alignment with organized knowledge sources. The system employs a dual-stage approach to entity linking, grounded in refined BERT [12] architectures. In the first stage, BLINK performs retrieval in a dense space, steered by a bi-encoder that separately embeds the mentioned context and respective entity descriptions. In the subsequent phase, every potential candidate undergoes detailed examination via a cross-encoder, merging the mention and entity textual content. In comprehensive entity linking benchmarks, BLINK's performance is comparable to that of REL in terms of effectiveness.

3.3 DuckDB

The entity annotations, which comprise metadata and other descriptive information related to various entities, are securely housed within a DuckDB [56] database. This specialized storage system is highly optimized for analytical tasks, ensuring that operations performed on the data are executed with remarkable efficiency. As a result of this optimization, users experience smooth and efficient access to the entities contained within the database. DuckDB's design is particularly adept at handling complex queries and large volumes of data, this setup is especially beneficial for environments where rapid retrieval and manipulation of annotation data are crucial to the workflows, such as in machine learning projects, research databases where annotations are integral to categorization and search functionality.

3.4 How to use

MMEAD data can be easily accessed via a few lines of Python code.

- Installation

```
$ pip install mmead
```

- Resource Loading

```
>>> from mmead import get_links
>>> links = get_links('v1', 'passage', linker='rel')
```

- Get entity links with specified doc/passage id

```
>>> links.load_links_from_docid(123)
{"passage":[{"entity_id":"7954681",...}
```

Benefiting from DuckDB as a backbone of MMEAD, entity linking data can be directly accessed via SQL queries.

```
>>> from mmead import load_links
>>> cursor = load_links(
...     'msmarco_v1_passage_links',
...     verbose=False
... )
>>> cursor.execute("""
...     SELECT pid
...     FROM msmarco_v1_passage_links_rel
...     WHERE entity='Nijmegen'
... """)
>>> cursor.fetchall()
[(771129,), (1273612,), (1418035,), ... ]
```

Chapter 4

Information Retrieval with MMEAD

In this section, I present a series of experiments that demonstrate the effectiveness and potential of MMEAD [29] in enhancing neural retrieval models. These experiments integrate MMEAD annotations into well-established information retrieval models. Regarding sparse retrieval, I built upon the work of Early Stage Sparse Retrieval with Entity Linking [65, 66] incorporating with MMEAD for entity expansion.

4.1 Datasets

4.1.1 MS MARCO

In this study, I utilized the MS MARCO passage collection v1 for the information retrieval experiments. Specifically, I employed the development set, referred to as *dev*, and filtered for entries that had relevance judgments available in the *qrel* file. This file consists of a total of 6,980 queries.

4.1.2 MS MARCO Chameleons

Further experiments are conducted on the obstinate query sets of the MS MARCO Chameleons [2], include all the subsets of Veiled Chameleon (Hard), Pygmy Chameleon (Harder), and Lesser Chameleon (Hardest).

4.2 Methods

4.2.1 Sparse

To showcase the effectiveness of MMEAD's [29] entity expansion in sparse retrieval, I followed a previous work, Early Stage Sparse Retrieval with Entity Linking [65, 66], carried out experiments that enhance existing sparse retrieval models using MMEAD annotations.

For sparse entity expansion retrieval, a state-of-the-art open-source sparse passage retriever Anserini, is used here. Anserini's [91] sparse retrieval represents the traditional bag-of-words representations and keyword-based matching techniques. It utilizes classic term-frequency-inverse-document-frequency (TF-IDF) weightings and BM25 ranking algorithms to score and rank documents based on their relevance to a query. These techniques have been the backbone of text-based search for decades, proving effective in many scenarios. Built on the robust foundation of the Lucene backend, Anserini's implementation provides a reliable and efficient alternative for those seeking traditional IR methods without the complications of advanced neural models. In the context of sparse retrieval, Anserini efficiently indexes, ranks, and retrieves documents based on term frequency-inverse document frequency (TF-IDF) and other classic retrieval models. This ensures rapid and precise retrieval from large-scale text collections. Given its strong performance and extensive stability, Anserini's sparse retrieval serves as a robust baseline for many information retrieval tasks and provides an essential contrast to emerging dense retrieval model.

No Expansion

For the base comparison, I adopted BM25, utilizing its implementation as used in Anserini with hyper-parameters were specifically set at k1 = 0.82 and b = 0.68. These parameters have been demonstrated to yield optimal results for the MS MARCO dataset [50]. During the indexing process of MS MARCO, standard procedures were employed. Notably, neither the queries nor the passages underwent any form of expansion during this baseline testing.

Expansion With Entity Text

In this approach, both passages and queries are expanded using the textual representations of the annotated entities, as sourced from MMEAD [29]. By augmenting these passages and queries with the context of these entities, I expected a more comprehensive understanding and potentially more relevant for retrieval tasks. Following this expansion process, I

applied the BM25 retrieval model. It's important to note that the BM25 model was run using the identical configuration and hyper-parameter settings as previously detailed in the baseline setting. This consistency ensures that any observed performance variations can be attributed to the entity expansion process rather than changes in the model's parameters.

Query Expansion:

what is prime rate in canada Canada

Passage Expansion:

What is the Prime Rate? In Canada, the prime rate is a guideline interest rate used by banks on loans for their most creditworthy, best, or prime clients. The prime rate rises and falls with the ebb and flow of the Canadian economy, influenced significantly by the overnight rate, which is set by the Bank of Canada. Bank of Canada Canada

Expansion With Entity Hash

Instead of directly using the textual representation of entities for expansion, an alternative method to expand passages and queries is to incorporate the MD5 hash of the entity text [65], sourced from MMEAD [29], into the passages and queries. The choice to employ MD5 hashing was influenced by two main factors. First, it offers a standardized representation for multi-word terms, ensuring uniformity. Second, it effectively minimizes the chance of inaccurate matches between queries and unrelated passages. After establishing these expansions, we then implemented the BM25 model, adhering to the hyper-parameter configurations described in our baseline.

Query Expansion:

what is prime rate in canada 445d337b5cd5de476f99333df6b0c2a7

Passage Expansion:

What is the Prime Rate? In Canada, the prime rate is a guideline interest rate used by banks on loans for their most creditworthy, best, or prime clients. The prime rate rises and falls with the ebb and flow of the Canadian economy, influenced significantly by the overnight rate, which is set by the Bank of Canada. 73bb9596e36cd23969cbf72c16d0a0df 445d337b5cd5de476f99333df6b0c2a7

Fusion

In the domain of information retrieval (IR), the term *Fusion* refers to the process of integrating results from various retrieval systems or algorithms. This aims to establish a

unified, coherent ranking set of results. On one hand, it aims to enhance retrieval effectiveness by capitalizing on the distinct strengths found within various retrieval methods. On the other hand, it aims to provide consistent and reliable performance across a wide spectrum of topics and diverse query formulations. By employing run combinations, researchers can unleash the collective potential of multiple systems, ensuring that the final output is not only comprehensive but also optimally aligned with user intent and needs.

The second series of experiments are conducted using Reciprocal Rank Fusion (RRF) [9], an established method in information retrieval, widely recognized for its effectiveness in fusing results from multiple retrieval systems. Unlike traditional score-based aggregation methods that often requires complex normalization due to divergent scoring metrics across systems, RRF leverages the stability of rank positions. For a given result, the RRF score is derived from the summation of the reciprocals of its rank across the assorted result lists. Subsequent to the computation, items are ranked based on their cumulative RRF score to produce the final consolidated ranking. RRF is computed as the formula:

$$RRF(d \in D) = \sum_{r \in R} \frac{1}{k + r(d)} \tag{4.1}$$

Here, k is a hyperparameter that can be fine-tuned. However, I chose to use a default value of k = 60 across all configurations.

4.2.2 Dense

For dense retrieval, I utilize Tevatron [21] - a cutting-edge efficient and flexible toolkit designed to synergize the best of dense retrieval methods. Dense retrieval is gaining popularity in recent studies. It focuses on understanding the deeper meaning in searches and benefits from using advanced, pre-trained language models. The need for adaptable solutions has grown clear in light of challenges such as CPU memory limitations when dealing with large corpora and increasing accelerator (GPU/TPU) demands with growing model sizes. Tevatron offered a comprehensive solution to these challenges. At its core, Tevatron integrates leading open-source packages: datasets for management, transformers for modeling, and FAISS [27] for retrieval. It's also adaptable, supporting the widely-used Pytorch framework [54]. Overall, Tevatron emerges as a solution, providing a cutting-edge platform equipped with state-of-the-art models, ensuring researchers have the flexibility to dive into varied research domains across multiple datasets.

No Expansion

Consistent with the approach taken in the sparse experiments, I set a baseline where both the queries and the passages remained untouched, without any form of expansion. For this baseline setup, I used specific hyper-parameters: a batch size of *8*, a learning rate set to *5e-6*, a maximum query length of *16*, and a maximum passage length of *128*. After training this configuration over three epochs, I was able to replicate the baseline performance from Tevatron, ensuring its reliability and serving as a foundation against which other configurations can be compared.

Expansion With Entity Text

1. **Query and Passage Expansion**

   In parallel with the sparse experiments, this method subjects both passages and queries to an expansion process. This expansion incorporates the textual representations of the entities sourced from MMEAD [29] annotations. The fundamental idea is that by augmenting the passages and queries with this additional entity context, they would not only have more detailed information but also be more relevant when searching or retrieving information. Anticipating that this enriched context would amplify their relevance, I proceeded to employ the Tevatron [21] model. To ensure a controlled experiment, I maintained consistency by using the identical configuration and hyper-parameter settings that were outlined in our baseline scenario. Maintaining this consistency is essential, as it ensures that any performance changes can be attributed directly to the effects of entity expansion rather than variations in the model's configurations.

   - Query

what is prime rate in canada Canada

   - Passage

What is the Prime Rate? In Canada, the prime rate is a guideline interest rate used by banks on loans for their most creditworthy, best, or prime clients. The prime rate rises and falls with the ebb and flow of the Canadian economy, influenced significantly by the overnight rate, which is set by the Bank of Canada. Bank of Canada Canada

2. **Passage Expansion**

   A notable experimental approach involves solely focusing on passage expansion rather than query expansion. Passage expansion is often privileged over query expansion due to its ability to uphold the user's original intent, eliminating the modification of their initial query and thereby reducing the potential for misinterpretation [98, 52]. Notably, passage expansion can be executed offline during the indexing phase, thereby optimizing real-time query processing, a critical advantage when deploying large language models. This method of expansion not only retains but also augments the contextual richness of a passage, thereby amplifying its relevance to an extended spectrum of associated queries. Using this method often leads to more consistent and trustworthy search results. This boosts user trust and interest while avoiding the unpredictability and extra computing effort that comes with expanding queries.

   - Query

what is prime rate in canada Canada

   - Passage

What is the Prime Rate? In Canada, the prime rate is a guideline interest rate used by banks on loans for their most creditworthy, best, or prime clients. The prime rate rises and falls with the ebb and flow of the Canadian economy, influenced significantly by the overnight rate, which is set by the Bank of Canada. Bank of Canada Canada

Expansion With Wikipedia

Instead of using hash entity expansion, I incorporated the *background* context from Wikipedia into the expansion. In the context of large language models (LLMs), hash entity expansion is deemed sub-optimal. The primary reason is the inherent loss of semantic and contextual information when entities are converted into hash values, rendering them semantically meaningless to the pretrained model. These hash functions, being non-reversible, further isolate original data, making it inaccessible for the LLM to produce meaningful content. Moreover, unless LLMs are specifically trained on datasets with prevalent hash entity expansion, they lack the foundational understanding to process these hashes.

1. **Query and Passage Expansion**

   Both passages and queries to an expansion process, not only with the entity text, but also the background context from Wikipedia are added to each labeled entity. By incorporating supplementary context, not just a broadening of vocabulary, but also heightened the ability to distinguish variations in phrasing and expression. This comprehensive textual expansion acts as a bridge, seamlessly connecting different terminologies, aiming to enhancing the model's capabilities and ensuring a more adaptable response to a wide spectrum of user inputs. Additionally, by offering clearer and more detailed context, the expanded text plays a crucial role in minimizing ambiguities that might arise, thereby ensuring that the model's output is precisely and contextually aligned.

   To do this, I initially annotate both the query and the passage utilizing the MMEAD [29] tool. Subsequent to this annotation, I retrieved its description from Wikipedia for every identified entity. This is achieved by referencing its unique Wikipedia ID and employing the Anserini tool in conjunction with the Wikipedia dump from 2019. Only the first sentence from each Wikipedia document is extracted and subsequently integrated as background context during the expansion process.

   - Query

     what is prime rate in canada Canada

   - Passage

     What is the Prime Rate? In Canada, the prime rate is a guideline interest rate used by banks on loans for their most creditworthy, best, or prime clients. The prime rate rises and falls with the ebb and flow of the Canadian economy, influenced significantly by the overnight rate, which is set by the Bank of Canada. Canada: canada is a country in north america Bank of Canada: the bank of canada (boc;) is a crown corporation and canada's central bank.

2. **Passage Expansion**

   Consistent with the previously mentioned expansion technique, I retained the query in its original form. However, I enriched the passages by incorporating annotations sourced from MMEAD [29]. Additionally, I sourced relevant context from Wikipedia for these passages, using the identical methodology employed during the Query and Passage Expansion processes.

- Query

  what is prime rate in canada

- Passage

  What is the Prime Rate? In Canada, the prime rate is a guideline interest rate used by banks on loans for their most creditworthy, best, or prime clients. The prime rate rises and falls with the ebb and flow of the Canadian economy, influenced significantly by the overnight rate, which is set by the Bank of Canada. Canada: canada is a country in north america Bank of Canada: the bank of canada (boc;) is a crown corporation and canada's central bank.

4.3 Experimental Setup

ALL experiments are conducted using a single machine of, Intel(R) Xeon(R) Gold 6134 CPU @ 3.20GHz, 128G RAM for sparse retrievals on the MS MARCO passage [50], run combinations, evaluations. A single Tesla P40 GPU is employed to conduct the dense retrieval, including training, encoding and evaluations.

Recall

Recall@K is a metric used to show the effectiveness of a retrieval system. Specifically, it measures the proportion of relevant documents that are retrieved in the top K results out of all the relevant documents available. It can be defined mathematically below:

$$Recall@K = \frac{\text{Number of relevant documents in top K results}}{\text{Total number of relevant documents in the dataset}} \tag{4.2}$$

Mean Reciprocal Rank

Mean Reciprocal Rank, is a metric used particularly in scenarios where the system is expected to return a list of ranked items to a query. Given a set of queries, the reciprocal rank for each query is the multiplicative inverse of the rank of the first relevant result. The formula for reciprocal rank is shown below:

$$ReciprocalRank = \frac{1}{\text{Rank of First Relevant Document in results}} \tag{4.3}$$

MRR is the average of the reciprocal ranks of results for a set of queries defined as:

$$ReciprocalRank = \frac{1}{len(Q)} \sum_{n=1}^{len(Q)} \text{Reciprocal Rank}_n (4.4)$$

4.4 Results

4.4.1 Sparse

The experiments were conducted utilizing the MS MARCO v1 passage ranking collection [50] and MS MARCO Chameleons, specifically focusing on queries that have at least one entity annotation. For queries that don't contain any linked entities, the expanded query remains identical to the original one, so it will not make a difference with expansion or not. The result shown in Table 4.1 and Table 4.2.

The 4 columns correspond to varying difficulty levels of the dataset. The *dev* column represents the development set of MS MARCO [50] with 6,980 queries. The *hard* column is the Veiled Chameleon subset of MS MARCO Chameleons [2], which has 3,119 queries. The *harder* category is represented by the Pygmy Chameleon set with 2,473 queries. Lastly, the *hardest* column pertains to the Lesser Chameleon set, comprising 1,693 queries.

The rows, labeled in alphabetical order, represent different experimental setup. The names of these setup clearly state the type of experiment.

Table 4.1: Results on the MS MARCO v1 passage collection, Recall@1000

		R@1000			
		dev	hard	harder	hardest
a.	BM25 – No Expansion	0.9111	0.7855	0.7444	0.6677
b.	BM25 – Entity Text	**0.9183**	**0.8240**	**0.7951**	**0.7298**
c.	BM25 – Entity Hash	0.9105	0.7980	0.7576	0.6848
d.	RRF – No Expansion + Entity Text	**0.9338**	**0.8436**	**0.8124**	**0.7500**
e.	RRF – No Expansion + Entity Hash	0.9250	0.8260	0.7921	0.7205
f.	RRF – Entity Text + Hash	0.9231	0.8260	0.7982	0.7314
g.	RRF – No Expansion + Entity Text + Hash	0.9313	0.8370	0.8043	0.7376

Table 4.1 shows the recall@1000 results of the subset queries with at least one entity. The expansion using *Entity Text* boosted a lot across all difficulty levels compared to the

rest. For RRF, *No Expansion + Entity Text* configuration achieves the highest Recall@1000 for all difficulty levels. The rest of the combination does not seem to improve much or remains unchanged compared with the *Entity Text* Expansion.

Table 4.2 shows the MRR@10 results of the subset queries with at least one entity. In the analysis of the dev set, there is no identification of any further relevant passages. It appears that expansion with *Entity Text* gives the best results when the difficulty level increases. When evaluating the reciprocal rank fusion methods, although they notably enhance the performance of recall, however, when evaluating the MRR@10 metric, there was no noticeable advantage in utilizing RRF compared to solely employing one of the entity expansion methods.

Table 4.2: Results on the MS MARCO v1 passage collection, MRR@10

		MRR@10			
		dev	hard	harder	hardest
a.	BM25 – No Expansion	**0.2413**	0.0373	0.0137	0.0000
b.	BM25 – Entity Text	0.2202	**0.0385**	0.0173	**0.0057**
c.	BM25 – Entity Hash	0.2199	0.0383	**0.0175**	0.0052
d.	RRF – No Expansion + Entity Text	0.2372	0.0385	0.0163	0.0019
e.	RRF – No Expansion + Entity Hash	0.2378	0.0367	0.0152	0.0034
f.	RRF – Entity Text + Hash	0.2218	0.0375	0.0161	0.0053
g.	RRF – No Expansion + Entity Text + Hash	0.2358	0.0391	0.0156	0.0035

In summary, leveraging entities in sparse retrieval enhances the model performance, as evidenced by the provided tables. Whether incorporated as text or hash, entity information typically yields better retrieval metrics than the baseline. For the R@1000 metric, configurations utilizing entity data consistently outperform the base configurations, with the /textitRRF - No Expansion + Entity Text achieving the highest scores across all test scenarios. Similarly, for the MRR@10 metric, the inclusion of entity information, particularly in the *BM25 - Entity Text* configuration, delivers superior results in the harder categories. Overall, the data indicates that incorporating entity information into sparse retrieval techniques can substantially enhance their performance.

4.4.2 Dense

The experiments were conducted similarly to the sparse experiments, utilizing both the MS MARCO v1 passage ranking collection [50] and MS MARCO Chameleons [2] for queries that contained at least one entity annotation. The result shown in Table 4.3 and Table 4.4

The result structure is consistent with the sparse experiments.

Table 4.3: Results on the MS MARCO v1 passage collection, RECALL@1000

		R@1000			
		dev	hard	harder	hardest
a.	Baseline – No Expansion	0.8888	0.7201	0.6714	0.5978
b.	Entity Text – Query & Passage	**0.8899**	**0.7299**	**0.6836**	**0.6165**
c.	Entity Text – Passage	0.8870	0.7223	0.6684	0.5978
d.	Wiki Sentence – Query & Passage	0.8694	0.6887	0.6542	0.5839
e.	Wiki Sentence – Passage	0.8693	0.6892	0.6440	0.5776

Table 4.3 shows the Recall@1000 results from dense experiments for subset queries containing at least one entity. Using *Entity Text* expansion on both passages and queries resulted in the most improvement across all difficulty levels compared to other methods. Methods based on Wikipedia yield scores that are quite similar across datasets, but they don't notably alter the results. In general, strategies that integrate both the *Query and Passage* tend to outperform those focused solely on the passage. Approaches that use Wikipedia sentences, regardless of whether they take the query into account, tend to underperform compared to those that exclusively rely on entity text.

Table 4.4: Results on the MS MARCO v1 passage collection, MRR@10

		MRR@10			
		dev	hard	harder	hardest
a.	Baseline – No Expansion	0.3707	0.0824	0.0503	0.0353
b.	Entity Text – Query & Passage	0.3666	0.0901	0.0561	0.0411
c.	Entity Text – Passage	**0.3744**	0.0890	0.0498	0.0340
d.	Wiki Sentence – Query & Passage	0.3545	**0.0931**	**0.0630**	**0.0480**
e.	Wiki Sentence – Passage	0.3622	0.0913	0.0572	0.0448

Table 4.4 shows the MRR@10 results for the subset queries with at least one entity. For the development set, methods that focus solely on passages perform relatively better. As query complexity increases, techniques that utilize information from both the query and passage prove to be more effective. The benefit is especially apparent when Wiki sentences are expanded for both the query and passage for these more challenging queries.

Overall, in dense retrieval, the *Entity Text* expansion in both query and passage emerges as the best approach for this dataset. It consistently delivers top-tier performance in the

R@1000 metric across all difficulty levels. This dominance suggests that it has a robust capability to recall relevant results within the top 1000 items, regardless of the challenge's complexity. Moreover, under the MRR@10 metric, it remains competitive, securing high rankings, especially in the *hard* and *hardest* datasets, even if it doesn't always clinch the top spot. Its all-around competence and its evident superiority in the R@1000 metric demonstrate its effectiveness when leveraging MMEAD [29] into dense retrieval tasks.

4.4.3 Discussion

Query: What is protected by *hipaa* rules
Query Entity Expansion: Health Insurance Portability and Accountability Act
Target Passage: *HIPAA* (*Health Insurance Portability and Accountability Act* of 1996) is *United States* legislation that provides data privacy and security provisions for safeguarding medical information.
Passage Entity Expansion: United States Health Insurance Portability and Accountability Act

---

Query: what does *cissp* stand for?
Query Entity Expansion: Certified Information Systems Security Professional
Target Passage: *Certified Information Systems Security Professional* (*CISSP*) is an independent information security certification governed by the International Information Systems Security Certification Consortium, also known as *(ISC)²*.
Passage Entity Expansion: (ISC)² ... Certified Information Systems Security Professional

Examples of how MMEAD can help the retrieval results.

Example 4.4.3 shows how MMEAD expanded entities can help with the retrieval results. In the first instance, MMEAD expands the *HIPAA* in the query to its full form. Without this expansion, information retrieval systems struggled to match the query with relevant documents, especially the relevant document using the full form instead of the acronym. The entity expansion ensures that the query and the target passage are semantically aligned, improving the accuracy of the retrieval. Similar to the first example, the second example also expanded the acronym *CISSP*. This is particularly important because

CISSP is a less commonly known term compared to *HIPAA*. Without entity linking, the connection between the query and the relevant passage could be easily missed. The expansion enables the retrieval system to accurately match the query with a passage that defines *CISSP*, even if the passage doesn't directly use the acronym. In the result, both examples successfully recognized the relevant target passages following the expansion, in contrast to previous misses.

Query: What is a *mra* business definition
Query Entity Expansion: Mail retrieval agent
Target Passage: Diffen Health Diagnostics. An *MRA*, or magnetic resonance angiogram, is a type of *MRI* scan that uses MRI's magnetic fields and radio waves to produce pictures of blood vessels inside the body, allowing doctors to locate problems that may cause reduced blood flow.
Passage Entity Expansion: Marketing Research Association Magnetic resonance imaging

---

Query: What are *cra*
Query Entity Expansion: Fair Credit Reporting Act
Target Passage: What is *CRA*? The *Community Reinvestment Act* (CRA) is a law intended to encourage depository institutions to help meet the credit needs of the communities in which they operate, including low- and moderate-income (LMI) neighborhoods, consistent with safe and sound banking operations. (CRA does not encourage the extension of unsafe or unsound credit.)
Passage Entity Expansion: Community Reinvestment Act.

Examples of MMEAD negatively impacts the retrieval results.

Example 4.4.3 illustrate how entity expansion can sometimes negatively impact information retrieval result by causing misalignment between the user's query and the target passage. Here, the first query likely refers to a business-related term *MRA* but the entity expansion mistakenly interprets it as *Mail retrieval agent*. The target passage, on the other hand, discusses a medical imaging technique. The passage entity expansion further confuses the context by introducing *Marketing Research Association*. This mismatch indicates that the entity expansion led the retrieval system the wrong way, aligning the

query with an irrelevant passage due to an incorrect interpretation of the acronym. In the second case, the query incorrectly expanded to *Fair Credit Reporting Act*. The correct expansion, as indicated by the target passage, should be *Community Reinvestment Act*. This misinterpretation by the entity expansion also leads to a rank drop of the relevant information. The user's intent was to know about the *Community Reinvestment Act*, but the expanded query might direct the system towards information related to a different aspect, thus boosting irrelevant information to the top.

Incorrect entity expansion can cause a misalignment between the query's intended meaning and the information provided in the target passages. This demonstrates the importance of accurate context understanding in entity linking for effective information retrieval. These examples also highlight the challenge faced by most entity-linking systems, which are biased towards longer texts. However, real-world user queries tend to be noisy, poorly structured, and short, often lacking helpful context especially when facing the less commonly known terms [7, 18, 25, 38].

Chapter 5

Applications

Through these quantitative evaluations, I have demonstrated MMEAD's [29] ability to leverage entities effectively boosts retrieval effectiveness, especially on standard benchmark datasets. However, the potential of MMEAD doesn't end there. In the following section, I will explore diverse examples that further highlight a range of applications and comprehensive usefulness of MMEAD in enriching search applications.

ALL experiments in this section are conducted using a single machine of, Intel(R) Xeon(R) Gold 6134 CPU @ 3.20GHz with 128G RAM.

5.1 Vector Search in DuckDB

Vector search is a modern technique used in information retrieval where textual data, such as passages or documents, are represented as vectors in a high-dimensional space using embedding models. When a user submits a query, it is converted into a vector representation in the same space. By comparing the similarity between the query vector and passage vectors, the system can identify and rank passages most relevant to the query.

Integrating vector search into a database management system (DBMS) can address the growing need to process and retrieve complex, unstructured, and semantically-rich data in modern applications. Traditional DBMSs, optimized for structured data and exact-match queries, often struggled when encountering with the requirements such as semantic text search and cross-modal search. By leveraging dense embeddings, vector search can capture the underlying semantics of data, enabling more context-aware and meaningful search results. Vector search not only enhances the flexibility and efficiency of the DBMS

in handling modern data challenges but also provides a smooth and easy-to-use connection between managing structural data and the growing world of deep learning data methods.

5.1.1 Architecture

DuckDB Template

DuckDB [56] stands out as a distinctive in-memory analytical database, carefully designed for enhanced analytical performance. Developed in C++, this relational Database Management System (DBMS) has been optimized for Online Analytical Processing (OLAP). It is particularly notable for its ability to efficiently process complex queries that encompass large portions of stored datasets. Beyond its performance capabilities, DuckDB is well-known for its seamless deployment and integration features. Coupled with its extensive SQL language variant, make it a preferred choice for both developers and data analysts. Furthermore, DuckDB extends its flexibility by offering a customizable extension mechanism via the DuckDB extension template[1], a tool designed mainly to assist users in the smooth creation, evaluation, and distribution of custom DuckDB extensions.

FAISS Integration

FAISS (Facebook AI Similarity Search) [27], stands out by its specialized design for efficient similarity search of high-dimensional vectors. As modern applications increasingly rely on vector representations for tasks such as semantic text search or cross-model retrieval, traditional DBMS can struggle with the computational demands of nearest neighbor searches. FAISS offers a solution to the challenge of efficiently finding the most similar vectors in large datasets. By integrating FAISS into a DuckDB, combines the robustness and structured data management capabilities of traditional databases with state-of-the-art vector search capabilities.

5.1.2 Dataset

I utilized the MS MARCO passage collection v1 [50] for the vector search integration showcases. Specifically, I employed the 8.8 million passages only as the knowledge base, and conduct vector searches on top of it.

[1] https://github.com/duckdb/extension-template

5.1.3 Vector Preparation

I first create the embeddings(vectors) of the 8.8 million passages. These vectors are obtained from Pyserini [41], a Python toolkit for information retrieval (IR) that serves as a bridge between the popular Anserini [91] library and Python. With its user-friendly APIs, Pyserini allows for vector encoding in just a single line of code. For example, via:

```
python -m pyserini.encode \
  input   --corpus collections.json \
          --fields text \
          --delimiter "\n" \
          --shard-id 0 \
          --shard-num 1 \
  output  --embeddings test_emb/ \
  encoder --encoder castorini/tct_colbert-v2-hnp-msmarco \
          --fields text \
          --batch 32 \
          --fp16
```

This creats the embeddings for the passages using the model TCT-ColBERTv2 [43].

Then I utilize DuckDB's built-in load function to integrate the Faiss-DuckDB extension. This approach allows us to directly load a pre-compiled, customized extension into DuckDB. By doing so, we can leverage the powerful features of Faiss within the DuckDB environment. This integration enhances our database's capabilities, enabling DuckDB to perform complex vectorized queries and operations more efficiently.

```
>>> LOAD 'faiss.duckdb_extension'
```

After obtaining the vector embbeddings, they can be directly loaded into DuckDB from the result JSON file. Utilizing *read_json_auto* function, DuckDB automatically infers the data types within the JSON file, effectively creating a table view of the contained data. This process simplifies data handling, as it enables the direct application of SQL operations on the data structured within the JSON file, thereby streamlining database management and query execution.

```
>>> CREATE TABLE msmarco AS
    SELECT * FROM read_json_auto('test_emb/embeddings.jsonl',
                                 format='newline_delimited');
```

The next step involves creating a FAISS index within DuckDB. This is achieved by calling a specialized function named *FAISS_CREATE*, which is defined within the custom extension. Once the FAISS index is created, the subsequent action is to populate it with relevant data. This is accomplished by using another customized function, *FAISS_ADD*, which is specifically defined to insert data into the FAISS index. These functions are part of the custom extension and are essential for integrating FAISS indexing capabilities into DuckDB, thereby enabling efficient search and retrieval operations within the database.

```
>>> CALL FAISS_CREATE('index_name', 768, 'IDMap,Flat') ;
>>> CALL FAISS_ADD((SELECT id as docid, vector as vec FROM msmarco),
                   'index_name');
```

Until now, DuckDB has achieved comprehensive support for vector search operations by integrating with FAISS, enabling efficient and effective handling of high-dimensional data searches.

5.1.4 Vector Search Example

In this example, we conduct a passage retrieval experiment, retrieving the top 1000 passages using queries from the TREC 2019 Deep Learning Tracks[2]. We employ the TCT-ColBERTv2 [43] model, the same as used in passage embedding generation for the queries. These pre-encoded queries are also available for download through Pyserini's API.

This step is important as it tests our ability to replicate standard results using vector search in DuckDB.

Result

Before doing the actual search, we examine the query in detail to understand its structure. As shown in the table, *qid* represents the query id and *emb* is the float vector representation of each query.

[2]https://microsoft.github.io/msmarco/TREC-Deep-Learning-2019.html

```
>>> SELECT id, emb from query_embedding limit 10;
```

qid int64	emb float[ ]
19335	[0.110..., 0.192..., 0.004..., -0.029..., 0.151..., 0.189..., 0.200..., -0.072..., -0.126...,
47923	[-0.083..., 0.083..., 0.064..., 0.013..., 0.050..., 0.130..., 0.169..., -0.182..., -0.234...,
87181	[-0.039..., 0.116..., 0.171..., 0.066..., 0.075..., 0.052..., 0.084..., -0.128..., -0.018...,
87452	[0.077..., 0.167..., 0.120..., 0.098..., 0.210..., 0.167..., 0.160..., -0.238..., 0.027...,
104861	[0.166..., 0.196..., 0.168..., 0.234..., 0.337..., 0.260..., 0.108..., -0.101..., 0.196...,
130510	[-0.091..., 0.047..., 0.023..., -0.062..., 0.137..., 0.187..., 0.145..., -0.015..., -0.124...,
131843	[-0.178..., 0.048..., 0.219..., 0.047..., 0.074..., 0.116..., 0.120..., 0.002..., -0.093...,
146187	[0.158..., 0.027..., 0.181..., -0.034..., 0.128..., 0.064..., 0.207..., 0.009..., -0.030...,
148538	[0.157..., -0.082..., 0.022..., 0.031..., 0.139..., -0.051..., 0.302..., 0.012..., -0.069...,
156493	[0.092..., 0.318..., 0.206..., -0.016..., 0.366..., 0.097..., 0.006..., -0.135..., -0.061...,

In this process, the first step involves loading a collection of passages, which is the query dataset we examed in the previous process. This dataset is then subjected to a search operation to identify and retrieve the top 1000 passages using the vector database created in 5.1.3. These passages are selected based on their similarity via inner product. Following the retrieval of these passages, an intermediate table, referred to as *raw_rst*, is created. The purpose of this table is to systematically display the intermediate results of the search operation. This table include the query id as *id* and its raw matching result of score, rank, document id label. The creation of this intermediate table is allowing us to review and analysis of the search results before any further processing or refinement is undertaken.

```
>>> CREATE TABLE raw_rst AS
    SELECT id, NULL AS COL2, NULL AS COL5,
            UNNEST(FAISS_SEARCH('flat', 1000, emb)) AS RAW
            FROM query_embedding;
>>> SELECT * FROM raw_rst limit 5;
>>>
         id       ...                                  RAW

        int64     ...     struct(rank integer, 'label' bigint, distance float)
```

```
19335    ...    {'rank': 0, 'label': 8412682, 'distance': 79.02593}
19335    ...    {'rank': 1, 'label': 342431, 'distance': 79.00702}
19335    ...    {'rank': 2, 'label': 2304005, 'distance': 78.89528}
19335    ...    {'rank': 3, 'label': 527692, 'distance': 78.886}
19335    ...    {'rank': 4, 'label': 1720389, 'distance': 78.85274}

5  row                                                4 columns
```

Then, we store the obtained results into a file, specifically formatting it according to the *trec* (Text REtrieval Conference)[3]. This is a widely recognized format used primarily in information retrieval systems, allowing for efficient and standardized evaluation and comparison of different systems or algorithms. By saving the results in this format, it becomes easier to conduct comparisons and analyses later on.

```
>>> COPY (select id, IfNull(COL2,'Q0'),
          raw.label as docid,
          (raw.rank + 1) as rank,
          raw.distance as distance,
          IfNull(COL5,'Faiss') from raw_rst)
    TO 'query_result.csv' WITH (DELIMITER '-');
```

After we obtained the result file, we use Pyserini's integrated evaluation tools to thoroughly assess the performance of the retrieval result file. This process involves running the result file through Pyserini's evaluation tool, computes various metrics. These metrics provide insights into how well the retrieval system is performing in terms of relevance and accuracy of the returned results.

```
>>> python -m pyserini.eval.trec_eval -c -l 2 -m map dl19-passage \
           query_result.csv
0.4469
>>> python -m pyserini.eval.trec_eval -c -m ndcg_cut.10 dl19-passage \
           query_result.csv
0.7204
```

[3]https://trec.nist.gov/pubs/trec28/trec2019.html

```
>>> python -m pyserini.eval.trec_eval -c -l 2 -m recall.1000 dl19-passage \
            query_result.csv
0.8261
```

All the evaluated results aligned perfectly with the results obtained through Pyserini's two-click reproductions process [40]. This consistent correlation underscores the reliability and accuracy of the results achieved in both evaluations.

5.1.5 Further Search Refinement Example 1

Imagine you are a researcher who would like to delve into the depths to uncover insights about the *Lewis and Clark expedition*, and you are particularly interested in finding information about *Sacagawea's* involvement. As you explore this extensive dataset, your goal is to extract narratives and reveal her contributions. You are not just looking for general information about the expedition; you are on a quest to find those hidden texts that reveal *Sacagawea's* impact.

In this example, one can further refine search results by categorizing vector search results based on entity mentions. We can accomplish this by first retrieving passages using vector search, and then a subsequent processing step uses the MMEAD entity tool to categorize specific entities within these passages:

- Passages that mention the specified entity or entities.
- Passages that do not contain the entity mentions.

This approach ensures that users not only find content relevant to their broad search term but also have the option to focus on the mentioned specific entities they might be particularly interested in. This combination of vector search with entity-based categorization offers a more refined and effective search experience.

Result

In the process of examining the query, we utilize the functionality of DuckDB to directly import data from JSON format. This approach simplifies the process of data manipulation and analysis. By loading JSON data directly into DuckDB, we bypass the need for intermediate data conversion. DuckDB's ability to handle JSON format allows for a seamless integration of complex, nested data structures. Once the JSON data is loaded into

DuckDB, it will automatically inference the data types and transformed into a structured table format. By converting JSON data into a tabular form, we can leverage SQL queries to efficiently extract, filter, and manipulate the data as required for our specific analytical needs. Here query id shown as *qid*, query text as *text* and the query vector representation as *emb*.

```
>>> select * from read_json_auto('../data/emb_queries.json',
                                 format='newline_delimited');
```

qid int64	text varchar	emb double[ ]
0	Lewis and Clark expedition to the Pacific Ocean	[0.047.., 0.184.., -0.043.., 0.284..,

For the process of retrieval, we first load the query data from the JSON file using the same function *read_json_auto*. The next step is to conduct a search for passages that are relevant to the encoded query. The relevance of a passage is determined based on inner product. In this example, we identify the top 1000 passages that are most relevant to the query. To facilitate a clearer examination and analysis of these search results, an intermediate table, named *passage_rst*, is created. We only need the relevant passage id in this example to further refine the search result as passage id can be queried as an indicator in MMEAD.

```
>>> CREATE TABLE passage_rst AS
    SELECT qid, UNNEST(FAISS_SEARCH('index_name', 1000, emb)).label as docid
    from read_json_auto('emb_queries.json', format='newline_delimited');
>>> SELECT * FROM passage_rst limit 5;
>>>
        qid        docid
      int64        int64

          0      7915180
          0      2104864
          0      2104859
          0      8839349
          0      7253743

          5  rows
```

Then we execute a query to retrieve a specific set of passages. These passages are characterized by their explicit inclusion of *Sacagawea* as an identified entity. To accomplish this, we join the intermediate results obtained from previous step with the MMEAD database. After the joining process, we apply a filter specifically targeting the entity term *Sacagawea*. By doing so, we ensure that the final list of passages extracted from the database is highly relevant and focused solely on content that specifically mentioned *Sacagawea*.

```
>>> SELECT DISTINCT passage_rst.docid FROM passage_rst
          LEFT JOIN mmead_v1_pas ON mmead_v1_pas.docid=passage_rst.docid
          WHERE mmead_v1_pas.entity='Sacagawea';
>>>
         docid
         int64

        3044517
        5373614
        5373613
        7244392
        ...

        34 rows
```

In this scenario, we harness the capabilities of DuckDB, Faiss, and MMEAD with minimal SQL code, efficiently retrieving and categorizing relevant data points from a multi-dimensional space. This process involves organizing search results into meaningful categories, showcasing the adaptability of our integrated system in handling complex data operations from search and retrieval in high-dimensional spaces to effective classification and organization of the outcomes.

5.1.6 Further Search Refinement Example 2

Beyond categorizing vector search results, we can also measure how often specific entities, such as individuals, locations, institutions, etc., emerge in the result from vector search.

Suppose a researcher is delving into the popularity of tourist attractions in Paris, seeking insights beyond the general information by uncovering detailed narratives about the city's landmarks, culture, and connections. To achieve this, one can highlight the most talked-about landmarks and cultural aspects, revealing each attraction's relative popularity by quantifying their presence in the data source.

By counting the popularity of the entities, we can obtain a more clearer insight of the major themes, topics, or objects of interest associated with the search query within the returned passages.

Result

Same as the process of examining the query in 5.1.5, we use DuckDB to directly import JSON data, streamlining data manipulation and analysis. Upon loading JSON into DuckDB, it infers data types and automatically structures the data into a table view. In the query table, *qid* denotes the query id, *text* represents the query text, and *emb* indicates the query vector representation.

```
>>> select * from read_json_auto('../data/emb_queries.json',
                                 format='newline_delimited');
```

qid int64	text varchar	emb double[ ]
0	Paris	[-0.029.., -0.012.., 0.172.., 0.147.., 0.104.., 0.0767.., 0.145.., 0.0131..,

Again in this process, we begin by loading and searching for relevant passages using the encoded query data aiming to identify and retrieve the top 1000 passages that are most pertinent to our query. Then we create an intermediate table, named *popularity_rst* to display intermediate results in a clear and organized manner. In this example, we require only the relevant passage ids to enhance the search results. The passage id serves as a key indicator in the MMEAD, allowing for more precise querying and refinement of the results.

```
>>> CREATE TABLE popularity_rst AS
    SELECT qid, UNNEST(FAISS_SEARCH('index_name', 1000, emb)).label as docid
    from read_json_auto('emb_queries.json', format='newline_delimited');
>>> SELECT * FROM popularity_rst limit 5;
>>>
        qid        docid
      int64        int64

          1      5219520
          1      1485616
          1      1047740
```

```
1     4295731
1     3204988

5  rows
```

Following the data retrieval, the next step is refinement. During this stage, we focus on measuring the frequency of entities that are mentioned within the retrieved passages. This is achieved by performing a join operation between the intermediate result table and MMEAD. The join operation is to efficiently correlate the data from these two sources. By doing this, we can gain insights into how frequently specific entities are referenced across the collected data.

```
>>> SELECT mmead_v1_pas.entity, COUNT(mmead_v1_pas.entity)
          FROM passage_rst LEFT JOIN mmead_v1_pas
          ON mmead_v1_pas.docid=passage_rst.docid
          GROUP BY mmead_v1_pas.entity
          ORDER BY COUNT(mmead_v1_pas.entity) DESC;
>>>
        entity                  count(entity)
        varchar                     int64

       Paris                            1699
       France                            668
       United States                     118
       Notre-Dame de Paris                70
       Seine                              69
       ...                               ...

       1074 rows
```

This example illustrates how our integration allows for the easy creation of a mentioned entity frequency table for further analysis, enabling a deeper and more comprehensive understanding of the topic, subjects, or objects of interest that are connected with the search query in the returned text passages. Moreover, this process can be enhanced by applying additional filters to the data found in MMEAD's metadata, allowing for more refined and targeted insights.

5.2 MS MARCO GEO Heatmap

The incorporation of entity links to Wikidata [81] can serve as a gateway to the expansive universe of open-linked data. This interconnection is instrumental in enriching the data ecosystem, enabling a seamless fusion with other databases and resources. By leveraging these links, we unlock the potential to create innovative applications.

5.2.1 Dataset

Here, a practical application of this using the MS MARCO dataset [50]. I utilized the MS MARCO passage collection v2, the 138.4M passages, to construct the heatmap.

5.2.2 Method

By leveraging the identified entities within these passages sourced from MMEAD, we queried Wikidata [81] to obtain geographical coordinates for those entities within the text that correspond to physical locations, example query shown in 5.2.2. Each location entity was matched with its unique identifier on Wikipedia. Upon successfully retrieving the coordinates associated with these identified entities, we then generated a visual representation by plotting each data point onto a global map. The representation through this mapping is an illustrative canvas that allows for the geographical context of the textual information to emerge vividly.

```
PREFIX wdt: <http://www.wikidata.org/prop/direct/>
PREFIX wikibase: <http://wikiba.se/ontology#>
PREFIX wd: <http://www.wikidata.org/entity/>
PREFIX p: <http://www.wikidata.org/prop/>
PREFIX ps: <http://www.wikidata.org/prop/statement/>
PREFIX psv: <http://www.wikidata.org/prop/statement/value/>
SELECT ?coordinate_node ?lon ?lat WHERE {
  values ?ids { entity_ids }
  ?ids ?p ?statement .
  ?statement psv:P625 ?coordinate_node .
  ?coordinate_node wikibase:geoLongitude ?lon.
  ?coordinate_node wikibase:geoLatitude ?lat.
}
```

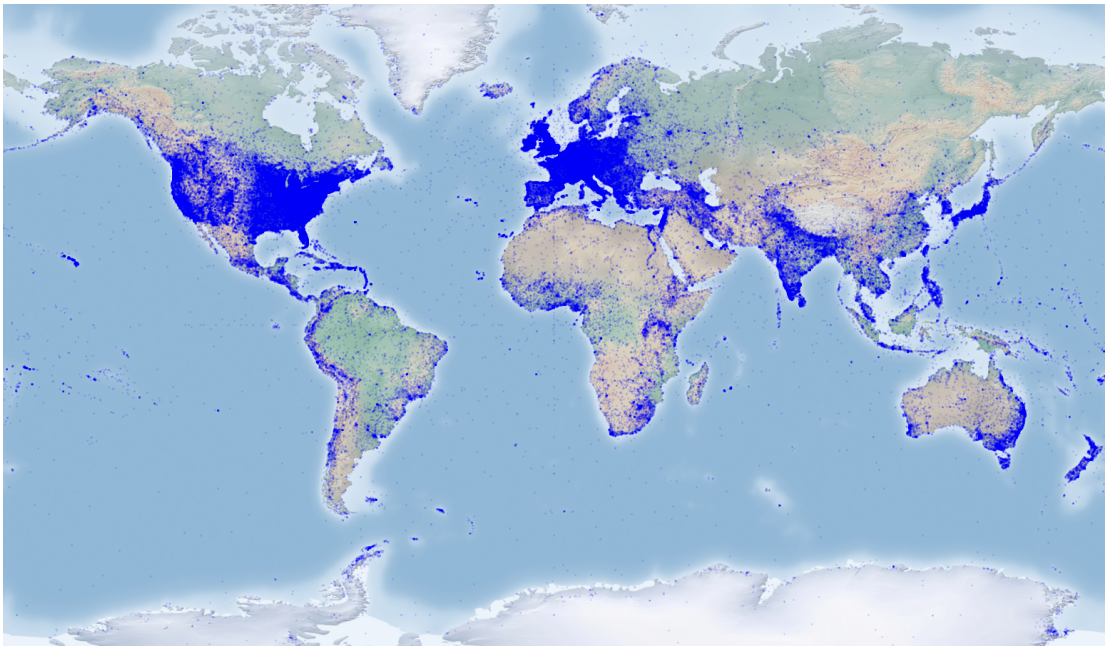

Figure 5.1: Locations of entities found in the MS MARCO v2 passage collection.

5.2.3 Result

An illustrative example of this is presented in Figure 5.1, which displays all the geographical entities culled from the MS MARCO version 2 passages collection [50] comprises 138.4M passages. Every entity is denoted by a semi-transparent blue dot on the map. The geographic distribution of these blue dots tells a story that aligns with real-world demographics. However, this also reveals a disproportionate representation of entities from regions such as North America and Europe, along with other economically advanced areas. Such a pattern likely reflects the digital divide and the data biases in knowledge representation that tend to favor more developed parts of the globe.

This kind of visualization is not only informative but also highlights the utility of open-linked data in providing more detailed insights. By leveraging the connections, we can transform raw data into a geographical narrative, offering a spatial dimension to our understanding of data distributions and biases. Moreover, these visual and data integrative approaches pave the way for more complex analyses and applications, bridging the gap between textual information and geographic contextualization.

Chapter 6

Conclusion and Future Work

In conclusion, entity linking has become a fundamental component of NLP systems. As the volume of digital information continues to grow, the ability to connect words and phrases to real-world entities is crucial for making sense of the vast amount of unstructured data available today. Whether it's for search engines, recommendation systems, or data integration, entity linking plays an important role in unlocking the full potential of the information age.

In this study, I delved into the MMEAD, a high-quality entity linking dataset. By providing entity annotations for the MS MARCO document and passage ranking collections, MMEAD addresses a critical gap in the development of neuro-symbolic IR models. More than just unifying the entity linking data format, this resource is readily available to researchers through its intuitive Python package.

When turning to sparse retrieval, incorporating entity information enhances the performance. Whether using entity data as text or hash, it consistently outperforms the baseline configurations. For the R@1000 metric, configurations utilizing entity data consistently outperform the base configurations. Similarly, for the MRR@10 metric, the inclusion of entity information, especially with the Entity Text configuration, delivers superior results, even in more challenging categories. In the context of dense retrieval, the Entity Text expansion in both queries and passages stands out as the best approach for this dataset. It consistently achieves top-tier performance in the R@1000 metric across all difficulty levels, indicating its robust ability to recall relevant results within the top 1000 items, regardless of complexity. Additionally, under the MRR@10 metric, it competes effectively, securing high rankings. Its all-around competence and evident superiority in these evaluation metrics demonstrate its effectiveness in enhancing information retrieval tasks when leveraging

MMEAD.

Beyond quantitative results, MMEAD has the potential to enhance search applications through a variety of methods. Exploring MMEAD with a vector search database reveals several key insights. Similar to enhanced information retrieval, this dataset provides a foundational knowledge graph, enabling the vector search system to navigate through a web of interconnected data points. Instead of treating data as isolated silos, the integrated system presents them in relation to one another, offering users a comprehensive view. This integration addresses a longstanding challenge in data management: harmonizing structured and unstructured data. The entity links in MMEAD provide structured insights into otherwise unstructured datasets. When combined with database and vector search, there's a seamless fusion of structured logic with the fluidity of unstructured data, addressing a broad spectrum of applications. This results in a dynamic search experience, where users can ask specific questions while also exploring a broad range of topics. Moreover, this exploration underscores the importance of adaptability and scalability in modern data systems. As cross-model needs increase, the underlying search infrastructure must be agile enough to incorporate these changes without sacrificing performance. Integrating MMEAD with a vector search database provides notable enhancements in data retrieval and analysis and sets the stage for the next generation of intelligent, interconnected, and intuitive data systems. Lastly, the fusion of Wikidata with MMEAD exemplifies the capabilities of open-linked data, moving beyond the simple data connection to a richer synthesis of global information. The visual representation not only reflects the underlying demographic patterns but also highlights the digital divide. This insightful visualization showcases the power of integrating diverse data sets to uncover spatial narratives and biases, enhancing our understanding and potentially guiding more equitable information representation in the digital age.

Within the context of MMEAD, there are numerous possibilities to explore. One significant advantage is its seamless integration of annotations from a wide range of linking systems, greatly simplifying the process of diversifying data sources. This enables us to incorporate entity data from multiple sources, enriching the metadata with additional contextual information. Furthermore, we can expand the entity linking dataset beyond MS MARCO by incorporating other essential benchmarks, allowing us to adapt to various scenarios effectively.

6.1 Future Work

For dense retrieval, we have many options yet to explore. For instance, we can use information from large language models, or we can train the model using only data enriched with entities.

In *Question Answering systems*, entity linking enables us to connect questions with relevant knowledge base entries, thereby improving the accuracy and relevance of answers. In *Summarization* tasks, it assists in identifying key entities, enriching the summary with external context. In *Classification*, entity linking aids in categorizing documents with greater precision. Lastly, in *Recommendation Systems*, it enhances personalization by identifying user preferences and interests.

The potential applications are not limited to these areas alone, as entity linking continues to evolve and find new use cases. Having access to this unified format provides us with a potent tool for extracting, enhancing, and establishing connections among entities. This ultimately advances capabilities in information retrieval and knowledge management across various domains. The possibilities are indeed boundless, and we are just scratching the surface of what can be achieved through this innovative resource.

References

[1] Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. Flair: An easy-to-use framework for state-of-the-art nlp. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics (demonstrations)*, pages 54–59, 2019.

[2] Negar Arabzadeh, Bhaskar Mitra, and Ebrahim Bagheri. Ms marco chameleons: challenging the ms marco leaderboard with extremely obstinate queries. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 4426–4435, 2021.

[3] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: A nucleus for a web of open data. In Karl Aberer, Key-Sun Choi, Natasha Noy, Dean Allemang, Kyung-Il Lee, Lyndon Nixon, Jennifer Golbeck, Peter Mika, Diana Maynard, Riichiro Mizoguchi, Guus Schreiber, and Philippe Cudré-Mauroux, editors, *The Semantic Web*, pages 722–735, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[4] Yixin Cao, Lifu Huang, Heng Ji, Xu Chen, and Juanzi Li. Bridge text and knowledge by learning multi-prototype entity mention embedding. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1623–1633, Vancouver, Canada, July 2017. Association for Computational Linguistics.

[5] Wei-Cheng Chang, Felix X. Yu, Yin-Wen Chang, Yiming Yang, and Sanjiv Kumar. Pre-training tasks for embedding-based large-scale retrieval, 2020.

[6] Surajit Chaudhuri and Umeshwar Dayal. An overview of data warehousing and olap technology. *ACM Sigmod record*, 26(1):65–74, 1997.

[7] Lihan Chen, Jiaqing Liang, Chenhao Xie, and Yanghua Xiao. Short text entity linking with fine-grained topics. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, CIKM '18, page 457466, New York, NY, USA, 2018. Association for Computing Machinery.

[8] Xiaojun Chen, Shengbin Jia, and Yang Xiang. A review: Knowledge reasoning over knowledge graph. *Expert Systems with Applications*, 141:112948, 2020.

[9] Gordon V Cormack, Charles LA Clarke, and Stefan Buettcher. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 758–759, 2009.

[10] Wanyun Cui, Yanghua Xiao, Haixun Wang, Yangqiu Song, Seung-won Hwang, and Wei Wang. Kbqa: learning question answering over qa corpora and knowledge bases. *arXiv preprint arXiv:1903.02419*, 2019.

[11] Zhuyun Dai and Jamie Callan. Context-aware sentence/passage term importance estimation for first stage retrieval. *arXiv preprint arXiv:1910.10687*, 2019.

[12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[13] Tommaso Di Noia and Vito Claudio Ostuni. Recommender systems and linked open data. In *Reasoning Web International Summer School*, pages 88–113. Springer, 2015.

[14] Qian Dong, Yiding Liu, Suqi Cheng, Shuaiqiang Wang, Zhicong Cheng, Shuzi Niu, and Dawei Yin. Incorporating explicit knowledge in pre-trained language models for passage re-ranking. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '22, page 14901501, New York, NY, USA, 2022. Association for Computing Machinery.

[15] Mark Dredze, Paul McNamee, Delip Rao, Adam Gerber, Tim Finin, et al. Entity disambiguation for knowledge base population. In *Proceedings of the 23rd International Conference on Computational Linguistics*, 2010.

[16] Faezeh Ensan and Weichang Du. Ad hoc retrieval via entity linking and semantic similarity. *Knowledge and Information Systems*, 58:551–583, 2019.

[17] Dieter Fensel, Umutcan Şimşek, Kevin Angele, Elwin Huaman, Elias Kärle, Oleksandra Panasiuk, Ioan Toma, Jürgen Umbrich, Alexander Wahler, Dieter Fensel, et al. Introduction: what is a knowledge graph? *Knowledge graphs: Methodology, tools and selected use cases*, pages 1–10, 2020.

[18] Paolo Ferragina and Ugo Scaiella. Fast and accurate annotation of short texts with wikipedia pages. *IEEE Software*, 29(1):70–75, 2012.

[19] Norbert Fuhr. Probabilistic models in information retrieval. *The computer journal*, 35(3):243–255, 1992.

[20] G. W. Furnas, T. K. Landauer, L. M. Gomez, and S. T. Dumais. The vocabulary problem in human-system communication. *Commun. ACM*, 30(11):964971, nov 1987.

[21] Luyu Gao, Xueguang Ma, Jimmy J. Lin, and Jamie Callan. Tevatron: An efficient and flexible toolkit for dense retrieval. *ArXiv*, abs/2203.05765, 2022.

[22] Emma J Gerritse, Faegheh Hasibi, and Arjen P de Vries. Entity-aware transformers for entity search. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1455–1465, 2022.

[23] Dr S Gomathi and Dr M Lavanya. A survey on application of information retrieval models using nlp. *Int. J. of Aquatic Science*, 12(3):2129–2138, 2021.

[24] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *The LaTeX Companion*. Addison-Wesley, Reading, Massachusetts, 1994.

[25] Yingjie Gu, Xiaoye Qu, Zhefeng Wang, Baoxing Huai, Nicholas Jing Yuan, and Xiaolin Gui. Read, retrospect, select: An mrc framework to short text entity linking, 2021.

[26] Venkat N. Gudivada, Dhana L. Rao, and Amogh R. Gudivada. Chapter 11 - information retrieval: Concepts, models, and systems. In Venkat N. Gudivada and C.R. Rao, editors, *Computational Analysis and Understanding of Natural Languages: Principles, Methods and Applications*, volume 38 of *Handbook of Statistics*, pages 331–401. Elsevier, 2018.

[27] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547, 2019.

[28] Ammar Ismael Kadhim. Term weighting for feature extraction on twitter: A comparison between bm25 and tf-idf. In *2019 international conference on advanced science and engineering (ICOASE)*, pages 124–128. IEEE, 2019.

[29] Chris Kamphuis, Aileen Lin, Siwen Yang, Jimmy Lin, Arjen P de Vries, and Faegheh Hasibi. Mmead: Ms marco entity annotations and disambiguations. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2817–2825, 2023.

[30] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online, November 2020. Association for Computational Linguistics.

[31] Jun-Tae Kim and Dan I. Moldovan. Acquisition of linguistic patterns for knowledge-based information extraction. *IEEE transactions on knowledge and data engineering*, 7(5):713–724, 1995.

[32] Donald Knuth. *The TEXbook*. Addison-Wesley, Reading, Massachusetts, 1986.

[33] Leslie Lamport. *LATEX — A Document Preparation System*. Addison-Wesley, Reading, Massachusetts, second edition, 1994.

[34] Arash Habibi Lashkari, Fereshteh Mahdavi, and Vahid Ghomi. A boolean model in information retrieval for search engines. In *2009 International Conference on Information Management and Engineering*, pages 385–389. IEEE, 2009.

[35] Phong Le and Ivan Titov. Improving entity linking by modeling latent relations between mentions. *arXiv preprint arXiv:1804.10637*, 2018.

[36] Martin Josifoski Sebastian Riedel Luke Zettlemoyer Ledell Wu, Fabio Petroni. Zero-shot entity linking with dense entity retrieval. In *EMNLP*, 2020.

[37] Joon Ho Lee. Analyzing the effectiveness of extended boolean models in information-retrieval. Technical report, Cornell University, 1995.

[38] Belinda Z. Li, Sewon Min, Srinivasan Iyer, Yashar Mehdad, and Wen tau Yih. Efficient one-pass end-to-end entity linking for questions, 2020.

[39] Yizhi Li, Zhenghao Liu, Chenyan Xiong, and Zhiyuan Liu. More robust dense retrieval with contrastive dual learning. In *Proceedings of the 2021 ACM SIGIR International Conference on Theory of Information Retrieval*, ICTIR '21, page 287296, New York, NY, USA, 2021. Association for Computing Machinery.

[40] Jimmy Lin. Building a culture of reproducibility in academic research, 2022.

[41] Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Nogueira. Pyserini: A python toolkit for reproducible information retrieval research with sparse and dense representations. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2356–2362, 2021.

[42] Sheng-Chieh Lin, Jheng-Hong Yang, and Jimmy Lin. Distilling dense representations for ranking using tightly-coupled teachers. *arXiv preprint arXiv:2010.11386*, 2020.

[43] Sheng-Chieh Lin, Jheng-Hong Yang, and Jimmy Lin. In-batch negatives for knowledge distillation with tightly-coupled teachers for dense retrieval. In Anna Rogers, Iacer Calixto, Ivan Vulić, Naomi Saphra, Nora Kassner, Oana-Maria Camburu, Trapit Bansal, and Vered Shwartz, editors, *Proceedings of the 6th Workshop on Representation Learning for NLP (RepL4NLP-2021)*, pages 163–173, Online, August 2021. Association for Computational Linguistics.

[44] Aiting Liu, Ziqi Huang, Hengtong Lu, Xiaojie Wang, and Caixia Yuan. Bb-kbqa: Bert-based knowledge base question answering. In *China National Conference on Chinese Computational Linguistics*, pages 81–92. Springer, 2019.

[45] Xing Liu, Huiqin Chen, and Wangui Xia. Overview of named entity recognition. *Journal of Contemporary Educational Research*, 6(5):65–68, 2022.

[46] Xueguang Ma, Kai Sun, Ronak Pradeep, Minghan Li, and Jimmy Lin. Another look at dpr: reproduction of training and replication of retrieval. In *European Conference on Information Retrieval*, pages 613–626. Springer, 2022.

[47] Pedro Henrique Martins, Zita Marinho, and André F. T. Martins. Joint learning of named entity recognition and entity linking. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 190–196, Florence, Italy, July 2019. Association for Computational Linguistics.

[48] George A. Miller. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):3941, nov 1995.

[49] Cataldo Musto, Giovanni Semeraro, Pasquale Lops, and Marco de Gemmis. Combining distributional semantics and entity linking for context-aware content-based recommendation. In *User Modeling, Adaptation, and Personalization: 22nd International*

Conference, UMAP 2014, Aalborg, Denmark, July 7-11, 2014. Proceedings 22, pages 381–392. Springer, 2014.

[50] Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. Ms marco: A human-generated machine reading comprehension dataset. 2016.

[51] Rodrigo Nogueira, Jimmy Lin, and AI Epistemic. From doc2query to doctttttquery. *Online preprint*, 6:2, 2019.

[52] Rodrigo Nogueira, Wei Yang, Jimmy Lin, and Kyunghyun Cho. Document expansion by query prediction. *arXiv preprint arXiv:1904.08375*, 2019.

[53] Yasumasa Onoe and Greg Durrett. Fine-grained entity typing for domain independent entity linking. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8576–8583, 2020.

[54] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[55] Hasso Plattner. A common database approach for oltp and olap using an in-memory column database. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, pages 1–2, 2009.

[56] Mark Raasveldt and Hannes Mühleisen. Duckdb: an embeddable analytical database. In *Proceedings of the 2019 International Conference on Management of Data*, pages 1981–1984, 2019.

[57] Priya Radhakrishnan, Partha Talukdar, and Vasudeva Varma. ELDEN: Improved entity linking using densified knowledge graphs. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1844–1853, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[58] Juan Ramos et al. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, pages 29–48. Citeseer, 2003.

[59] Simon Razniewski, Hiba Arnaout, Shrestha Ghosh, and Fabian Suchanek. Completeness, recall, and negation in open-world knowledge bases: A survey, 2023.

[60] Ridho Reinanda, Edgar Meij, and Maarten de Rijke. Mining, ranking and recommending entity aspects. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, page 263272, New York, NY, USA, 2015. Association for Computing Machinery.

[61] Stephen Robertson, Hugo Zaragoza, et al. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389, 2009.

[62] Akram Roshdi and Akram Roohparvar. Information retrieval techniques and applications. *International Journal of Computer Networks and Communications Security*, 3(9):373–377, 2015.

[63] Christopher Sciavolino, Zexuan Zhong, Jinhyuk Lee, and Danqi Chen. Simple entity-centric questions challenge dense retrievers. *arXiv preprint arXiv:2109.08535*, 2021.

[64] Özge Sevgili, Artem Shelmanov, Mikhail Arkhipov, Alexander Panchenko, and Chris Biemann. Neural entity linking: A survey of models based on deep learning. *Semantic Web*, 13(3):527–570, 2022.

[65] Dahlia Shehata, Negar Arabzadeh, and Charles LA Clarke. Early stage sparse retrieval with entity linking. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 4464–4469, 2022.

[66] Shehata, Dahlia. Information retrieval with entity linking. Master's thesis, 2022.

[67] Leixian Shen, Enya Shen, Yuyu Luo, Xiaocong Yang, Xuming Hu, Xiongshuai Zhang, Zhiwei Tai, and Jianmin Wang. Towards natural language interfaces for data visualization: A survey. *IEEE transactions on visualization and computer graphics*, 2022.

[68] Wei Shen, Yuhan Li, Yinan Liu, Jiawei Han, Jianyong Wang, and Xiaojie Yuan. Entity linking meets deep learning: Techniques and solutions, 2021.

[69] Wei Shen, Jianyong Wang, and Jiawei Han. Entity linking with a knowledge base: Issues, techniques, and solutions. *IEEE Transactions on Knowledge and Data Engineering*, 27(2):443–460, 2015.

[70] Vaibhav Kant Singh and Vinay Kumar Singh. Vector space model: an information retrieval system. *Int. J. Adv. Engg. Res. Studies/IV/II/Jan.-March*, 141(143), 2015.

[71] Amit Singhal et al. Modern information retrieval: A brief overview. *IEEE Data Eng. Bull.*, 24(4):35–43, 2001.

[72] Valentin I Spitkovsky and Angel X Chang. A cross-lingual dictionary for english wikipedia concepts. 2012.

[73] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: A core of semantic knowledge. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, page 697706, New York, NY, USA, 2007. Association for Computing Machinery.

[74] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: A large ontology from wikipedia and wordnet. *Web Semant.*, 6(3):203217, sep 2008.

[75] Yaming Sun, Lei Lin, Duyu Tang, Nan Yang, Zhenzhou Ji, and Xiaolong Wang. Modeling mention, context and entity with neural networks for entity disambiguation. In *Twenty-fourth international joint conference on artificial intelligence*, 2015.

[76] Toni Taipalus. Vector database management systems: Fundamental concepts, use-cases, and current challenges. *arXiv preprint arXiv:2309.11322*, 2023.

[77] Nandan Thakur, Nils Reimers, Andreas Rckl, Abhishek Srivastava, and Iryna Gurevych. Beir: A heterogenous benchmark for zero-shot evaluation of information retrieval models, 2021.

[78] Howard Turtle and W Bruce Croft. Evaluation of an inference network-based retrieval model. *ACM Transactions on Information Systems (TOIS)*, 9(3):187–222, 1991.

[79] Johannes M. van Hulst, Faegheh Hasibi, Koen Dercksen, Krisztian Balog, and Arjen P. de Vries. Rel: An entity linker standing on the shoulders of giants. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20. ACM, 2020.

[80] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 60006010, Red Hook, NY, USA, 2017. Curran Associates Inc.

[81] Denny Vrandečić and Markus Krötzsch. Wikidata: A free collaborative knowledgebase. *Commun. ACM*, 57(10):7885, sep 2014.

[82] Jianguo Wang, Xiaomeng Yi, Rentong Guo, Hai Jin, Peng Xu, Shengjun Li, Xiangyu Wang, Xiangzhou Guo, Chengming Li, Xiaohai Xu, et al. Milvus: A purpose-built vector data management system. In *Proceedings of the 2021 International Conference on Management of Data*, pages 2614–2627, 2021.

[83] Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhengyan Zhang, Zhiyuan Liu, Juanzi Li, and Jian Tang. Kepler: A unified model for knowledge embedding and pre-trained language representation. *Transactions of the Association for Computational Linguistics*, 9:176–194, 2021.

[84] Chuangxian Wei, Bin Wu, Sheng Wang, Renjie Lou, Chaoqun Zhan, Feifei Li, and Yuanzhe Cai. Analyticdb-v: A hybrid analytical engine towards query fusion for structured and unstructured data. *Proc. VLDB Endow.*, 13(12):31523165, aug 2020.

[85] Wei Wu, Junlin He, Yu Qiao, Guoheng Fu, Li Liu, and Jin Yu. Hqann: Efficient and robust similarity search for hybrid queries with structured and unstructured constraints. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 4580–4584, 2022.

[86] Chenyan Xiong, Jamie Callan, and Tie-Yan Liu. Bag-of-entities representation for ranking. In *Proceedings of the 2016 ACM International Conference on the Theory of Information Retrieval*, ICTIR '16, page 181184, New York, NY, USA, 2016. Association for Computing Machinery.

[87] Chenyan Xiong, Jamie Callan, and Tie-Yan Liu. Word-entity duet representations for document ranking. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17, page 763772, New York, NY, USA, 2017. Association for Computing Machinery.

[88] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. Approximate nearest neighbor negative contrastive learning for dense text retrieval, 2020.

[89] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. Approximate nearest neighbor negative contrastive learning for dense text retrieval. *arXiv preprint arXiv:2007.00808*, 2020.

[90] Ikuya Yamada, Akari Asai, Jin Sakuma, Hiroyuki Shindo, Hideaki Takeda, Yoshiyasu Takefuji, and Yuji Matsumoto. Wikipedia2vec: An efficient toolkit for learning and visualizing the embeddings of words and entities from wikipedia. *arXiv preprint arXiv:1812.06280*, 2018.

[91] Peilin Yang, Hui Fang, and Jimmy Lin. Anserini: Enabling the use of lucene for information retrieval research. In *Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval*, pages 1253–1256, 2017.

[92] Yi Yang, Ozan Irsoy, and Kazi Shefaet Rahman. Collective entity disambiguation with structured gradient tree boosting. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 777–786, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[93] Binbin Yu. Research on information retrieval model based on ontology. *EURASIP Journal on Wireless Communications and Networking*, 2019(1):1–8, 2019.

[94] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Min Zhang, and Shaoping Ma. Learning to retrieve: How to train a dense retrieval model effectively and efficiently, 2020.

[95] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Min Zhang, and Shaoping Ma. Repbert: Contextualized text embeddings for first-stage retrieval, 2020.

[96] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Min Zhang, and Shaoping Ma. Repbert: Contextualized text embeddings for first-stage retrieval. *arXiv preprint arXiv:2006.15498*, 2020.

[97] Wenzheng Zhang, Wenyue Hua, and Karl Stratos. Entqa: Entity linking as question answering. *arXiv preprint arXiv:2110.02369*, 2021.

[98] Shengyao Zhuang and Guido Zuccon. Fast passage re-ranking with contextualized exact term matching and efficient passage expansion. *arXiv preprint arXiv:2108.08513*, 2021.