Improving Employee Education Using a Knowledge Graph of Learning Content to Automatically Recommend Relevant Courses

Master Thesis

by

Enzo Hilzinger

Information Systems Engineering and Management M. Sc.

Institute of Applied Informatics and Formal Description Methods (AIFB)

KIT Department of Economics and Management

Advisor:	Prof. Dr. Andreas Oberweis
Second Advisor:	Prof. Dr. Thomas Schuster
Submitted:	March 31, 2024

Abstract

The need for ongoing and targeted employee education presents a major challenge and decisive factor for organization's success in today's rapidly changing market and technology environments. This thesis aims to solve the issue of finding targeted and skill-oriented trainings in a multitude of different training providers by developing a hybrid recommendation system based on content and a knowledge graph of interconnected semantic information on trainings and their respective contents. This knowledge graph is created based on textual representations of the trainings, from which contained entities and topics are extracted. The extracted information is then ontologically linked together to form the knowledge graph. This graph is combined with a content-based recommendation system that relies on similarities among the textual representations of the trainings to recommend content that is both individualized and targeted. These recommendations are derived using node similarity in the knowledge graph and cosine similarity of the training description embeddings, providing more contextualized recommendations than by merely relying on content-based methods. The recommendation system itself is developed based on requirements collected via structured questionnaires from prospective users, indicating a need for such a system and the ability to gain personalized and sensible training recommendations to improve overall employee education.

Contents

List of Abbreviations

AAR Attendance After Recommendation.

AI Artificial Intelligence.

API Application Programming Interface.

AWS Amazon Web Services.

BERT Bidirectional Encoder Representation for Transformers.

CI/CD Continuous Integration/Continuous Delivery.

CNN Convolutional Neural Network.

CRM Customer Relationship Management.

CSV Comma-Separated Value.

CTR Click Through Rate.

ERP Enterprise Resource Planning.

GCP Google Cloud Platform.

GPT Generative Pretrained Transformer.

IaaS Infrastructure-as-a-Service.

ICT Information and Communication Technology.

JSON JavaScript Object Notation.

KNN K-Nearest Neighbors.

LDA Linear Discriminant Analysis.

LLM Large Language Model.

MAE Mean Average Error.

ML Machine Learning.

NER Named Entity Recognition.

NLP Natural Language Processing.

NLTK Natural Language Toolkit.

NMF Non-Negative Matrix Factorization.

OWL Web Ontology Language.

PaaS Platform-as-a-Service.

PCA Principal Component Analysis.

RDF Resource Description Framework.

RDFS Resource Description Framework Schema.

RMSE Root Mean Square Error.

SaaS Software-as-a-Service.

SPARQL SPARQL Protocol and RDF Query Language.

SQL Structured Query Language.

SVM Support Vector Machine.

TCO Total Cost of Ownership.

TF-IDF Term Frequency-Inverse Document Frequency.

TTL Turtle.

UI User Interface.

W3C World Wide Web Consortium.

List of Figures

List of Tables

Listings

1 Introduction

1.1 Background & Motivation

The need for an effective and accessible, intra-company learning environment has become a prime factor in enabling rapid adaption to the ever-changing needs of today's economy, not only for companies in the Information and Communication Technology (ICT) sector, but for all sectors as continuous education is vital for any employee. It is, therefore, of great importance to enable learning opportunities on all levels of an organization [1] and for all employee groups to solidify continuous upskilling. This need for usable learning offerings often clashes with a multitude of course providers and platforms that are available within companies, where it is often complex to find offerings that fit both the individual's training aspirations and interests as well as the company's overall strategic development goals. According to [2], video tutorials are and will continue to be an important factor in the corporate learning strategies of 89% of companies in the next three years, further solidifying the existence and continuous adoption of learning platforms that provide such videos in most companies. This also leads to the continued issue of having to recommend content that is both relevant and also in line with the overall goals. As current job skills are rapidly changing, it is also no longer sufficient to re-evaluate employee training needs in yearly dialogues with managers, but a more fast-paces solution needs to be crafted. This helps organizations to quickly adapt to changing environments by relying on an automated mechanism to monitor and propose training opportunities, rather than having to define these in a tedious, manual process. This manual process is prone to both human error and a significant delay between the identification of a training need and the subsequent closing of corresponding skill gaps in the organization's workforce.

1.2 Goal and Objective

To address the issue of recommending both the right and also interesting content in a structured manner, this thesis aims to develop a proof-of-concept implementation for a recommendation system for employee training, paving the way to providing custom-tailored and fitting training recommendations for increased employee satisfaction and continuous education. It is initially necessary to provide a basic understanding of associated technologies and background information, such as various concepts related to text processing and recommendation systems. A comprehensive overview of natural language processing techniques for analyzing training descriptions and the current state of the art of recommendation systems are introduced to solidify this foundational knowledge. Afterwards, a training classifier is developed, which provides the basis for categorizing training descriptions into distinct classes to enable topic-based recommendations in conjunction

with custom entity extraction to provide for a broad meta data basis of the trainings. This basis can then be integrated into the recommendation system using a knowledge graph to enable a targeted and personalized approach for providing accurate trainings to employees based on their profile, skill level, and personal development aspirations. To better understand the user's pain points and wishes, a structured survey is created to support the extraction of user needs, providing the starting point for defining requirements that enable the subsequent implementation of the system. To allow for easy and user-friendly usage of the system, a frontend application is then created to gain access to the overall recommendation system.

1.3 Contextual Setting

The project is carried out in a department specializing on vocational training, offering various dual study courses to students within the organization. This contextual setting provides an environment in which training and continuous education are a vital part of the day-to-day business, highlighting the need for targeted and personalized recommendations from an organizational point of view. Currently there is no such system readily available. Instead, there are various training providers that offer their services independently and without any interconnectivity or combined platform, resulting in a scattered landscape of training providers in which it is not trivial to find relevant content amidst the vast offerings. This is especially relevant as most students and employees rely on self-paced online courses as their main source of continuing education. It is, therefore, essential to solve the issue of these scattered environments by creating a solution which enables targeted and individualized recommendations of training content across system borders while also taking into account the individual's apsirations and interests to foster continued and motivated education among employees.

2 Methodology

To provide a solid foundation for a semantic layer, knowledge graphs and associated concepts, such as ontologies and query languages, are discussed because these are used as the basis for creating a semantic, combined layer of training data. This is accompanied by delving into the concept of embeddings as a crucial approach for automated, machine-based handling of textual data. As most recommendation systems that are based on text rely on natural language processing, the associated concepts and theoretical foundations for extracting entities and resolving ambiguities in unstructured text data are discussed. Following, it is also important to gain a solid understanding of recommendation systems, thus a thorough explanation of common approaches is carried out, especially comparing collaborative filtering, content-based, and hybrid approaches to provide a thorough foundation of widely used methods. Commonly used evaluation methods are presented to provide a starting point for also evaluating the created recommendation system based on broadly used and widely accepted metrics.

Subsequently, data is collected for training both an entity extractor and a course classificator. For the classificator, an evaluation of different text classification concepts is carried out to determine the best fitting approach while also providing a sound basis for later course classification. To enable this classification, data is synthetically created to solve the problem of data sparsity in the domain of training descriptions. Following on the text classification and entity extraction, an ontology for representing the knowledge gained is derived, which serves as the base vocabulary for integrating the semantic data in a graph that is both concise and well-defined. This graph is subsequently created using the extracted information and the previously created ontology.

Following the creation of the semantic layer, the recommendation system itself is developed. To achieve a targeted and problem-oriented implementation, requirements are derived from a structured questionnaire, which is filled out by potential users. This approach is supplemented by the definition of target groups in conjunction with their respective needs to enable a derivation of requirements, both functional and non-functional. To utilize the most beneficial recommendation approach, it is crucial that the presented algorithms are evaluated for the use case at hand. This evaluation provides the basis for selecting the most suitable approach for the following implementation based on well-defined criteria that enable a logical evaluation of the algorithms. Building on the selected recommendation approach and the derived requirements, the system is implemented in a microservice-oriented architecture and subsequently evaluated in terms of recommendation quality and fulfillment of the defined requirements. This evaluation is then used as a basis for future improvement and continued monitoring of the system's ability to support employees in finding trainings fitting to their specific development aspirations.

3 Theoretical Foundations

3.1 Knowledge Graphs

Knowledge graphs are a graph-based representation of knowledge that is available in the real world, using formal semantics to allow efficient and unambiguous machine-based processing, analysis, and information extraction especially in the fields of AI and also as a foundation for recommendation systems [3]. Another benefit of knowledge graphs is the ability to unify data formats and, therefore, information so that heterogeneous data can be represented in a single system without the need for further integration of different sources or data formats [4]. These knowledge graphs consist of a set of nodes and edges, where the nodes represent entities (such as individuals, technologies, or other knowledge objects) and the edges represent relations (such as "belongs to" or "is a") among those entities [5]. Since a knowledge graph consists of entities and their corresponding relations, the direction of the relation is also important [3] because a relation is not necessarily always bidirectional. The basic unit of a knowledge graph is a so-called triple, consisting of subject, predicate, and object (e.g., *JavaScript, is a, Programming Language*), similar to how natural language is constructed [3].

Types of relations commonly found in knowledge graphs include synonyms, hyponyms, and hypernyms [6], for which an example is displayed in table 1.

Entity	**Synonym**	**Hyponym**	**Hypernym**
JavaScript	JS	-	Programming Language
Programming Language	Coding Language	JavaScript	Computer Language

Table 1: Common relations in a knowledge graph.

From these defined relations, a derivation of triples can be made:

- *(JavaScript, is, JS)*
- *(JavaScript, is a, Programming Language)*
- *(Programming Language, is, Coding Language)*
- *(Programming Language, is a, Computer Language)*

The predicates employed signify the relationship between the subject and object entity, defining the synonymous relation as a *is* relation and the hypernym as a *is a* relation, defining the subject in that relation as being a subclass of the assigned object. These triples are visualized accordingly in the example knowledge graph in figure 1. This example also shows the importance of directed edges because the triple *(JavaScript, is a,*

Programming Language) is a true statement, whereas the reverse triple *(Programming Language, is a, JavaScript)* is nonsensical and would include false information. Another distinction can be made into relations that are attributive, such as the triple *(JavaScript, executed by, Interpreter)*. These relations describe specific attributes among entities and are useful for further detailing the characteristics of such entities. The existence of synonyms also highlights the need for entity disambiguation as *JS* and *JavaScript* might be subsumed under one entity, whereas an entity like *Python* needs to be differentiated because it might be either a programming language or an animal.

To obtain new insights or logical conclusions from the existing information in a knowledge graph, reasoning and inference are commonly used. Inference can be utilized to fill in missing parts of a triple; it is for example possible to determine the corresponding object by providing a subject and predicate [7]. Reasoning can be subdivided into three categories [8]:

- **Logic rules**, which relies mainly on rules and ontologies.
- **Distributed representation** enables more advanced reasoning by projecting the knowledge (entities, relations, and their attributes) into a continuous vector space.
- **Neural network**, which uses feature capturing abilities to provide a more accurate knowledge reasoning by also automatically learning feature representations.

Handling ambiguous queries is also a task that is often needed to be accomplished in knowledge graphs. Various methods exist for approaching this issue, such as the usage of entity context information or its attributes and comparing that to the result candidate using embeddings or using the whole entity relationship and subgraph embeddings to improve precision and recall of such disambiguation [9].

Another essential aspect in the creation and maintenance of knowledge graphs is to maintain up-to-date data and to ensure proper updating of the graph content when new or changed data needs to be incorporated.

One common approach for storing knowledge graphs is to employ a graph database, such as Neo4J, which enables storing of data in a graph structure, providing streamlined query options for the stored data [10]. Using such a database also provides additional advantages, such as the ability to set the directions of nodes, labeling of relations, and having multiple labels per node [10]. Another feasible approach for storing knowledge graphs is the usage of a triple store, such as Apache Jena or RDFLib in Python.

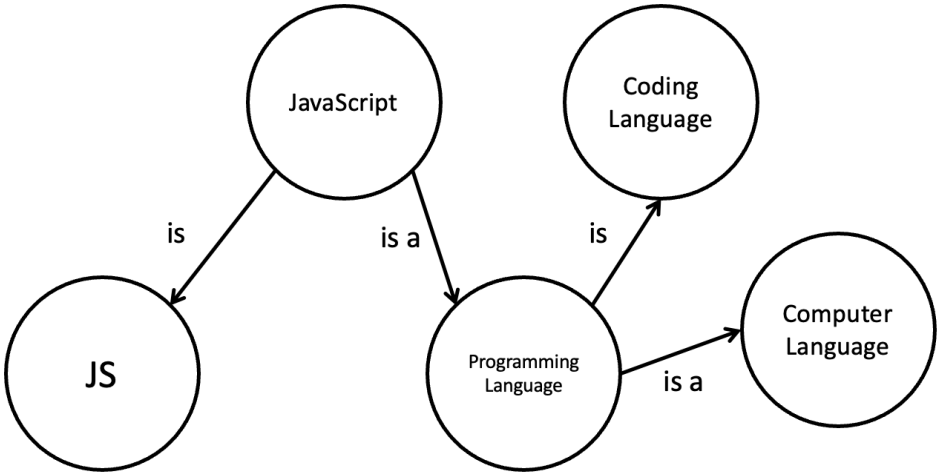


Figure 1: Exemplary knowledge graph.

3.1.1 Ontologies

The term *ontology* describes a human- and machine-understandable digital asset, which describes relations between entities and terms for concepts belonging to a predefined domain [11]. These are an essential part of knowledge graphs as they are a vital part in constructing relationships among real-world entities in any knowledge graph.

One commonly used ontology is Web Ontology Language (OWL), provided by the World Wide Web Consortium (W3C), which includes a variety of classes and properties for expressing relations among entities [12]. OWL enables standardized and structured creation of knowledge graphs in form of triples by using its pre-defined assets. It also provides deductive reasoning using logical inference to provide information, which is not explicitly stated in a given knowledge base [11]. The usage of OWL in conjunction with custom entities enables reusability and interoperability of knowledge graphs by relying on centrally provided properties and classes. Other commonly used pre-defined ontologies include Resource Description Framework (RDF) and Resource Description Framework Schema (RDFS) [13].

3.1.2 SPARQL

SPARQL Protocol and RDF Query Language (SPARQL) provides the standard query language for RDF, allowing the expression of questions for knowledge graphs in a well-defined, machine understandable format, which is partially inspired by Structured Query Language (SQL) [14]. SPARQL queries heavily rely on the triple patterns, which were introduced for knowledge graphs in section 3.1 [13]. This enables easy querying of data contained in such a graph by adhering to the same syntax for querying that is also used

to store the data in.

An example query written in SPARQL, which selects all entities that are a programming language from the example knowledge graph, is displayed in listing 1.

```
SELECT ?s
    WHERE {
        ?s a ex:programming_language.
    }
```

Listing 1: Example SPARQL query.

It is evident that some concepts are borrowed from traditional SQL, such as the existence of the `SELECT` statement or the inclusion of a `WHERE`-clause, while the formulation of the condition itself is nowhere similar to standard SQL. These SPARQL queries are then used to retrieve corresponding information from the knowledge graph using pre-defined queries or to provide easy access to its contents by allowing also for more sophisticated querying and, thus, information retrieval.

3.2 Embeddings

Representing textual data for analysis using machines proves to be difficult due to the unstructured data of natural language. This issue can be overcome by employing embeddings to transform natural language into a vector-based representation, providing a function that maps between words in a given vocabulary and a low-dimensional vector space [15]. Such embedding functions provide a vital role in language processing because their key capability of transforming textual data into a vector space provides the prerequisite for analyzing such data using algorithmic or machine learning techniques.

The resulting word vectors, which are created using such an embedding function, capture the semantic meaning among the words, hence words with similar meanings or those appearing in similar contexts will also have a closer proximity in their vector representations. One example to signify this potential proximity is the relation among the three words laptop, desktop, and car. Laptop and desktop herein would have a closer relation and a more similar vector representation to capture this semantic similarity accordingly.

To differentiate among commonly used methods of word embeddings, [16] distinguish traditional, static, and contextualized approaches, where contextualized approaches represent the most modern methods and include recently developed approaches, such as GPT or Bidirectional Encoder Representation for Transformers (BERT) models. This differentiation is based mainly on the evolvement of the used algorithms or approaches. Drawbacks of such modern models include the high computational and data efforts that these require for their training, which can be mitigated by using static algorithms, such as Word2Vec or FastText, which require less resources but, therefore, sarifice accuracy [16]. When selecting a model, is is important to consider both the required accuracy and the available computational power in relation to the data that is expected to be embedded because a better performing model might be a preferable choice when dealing with large amounts of frequently changing data.

Embedding functions are commonly trained on large corpora of text to gradually and iteratively improve the vector mapping function based on the surrounding contexts of words that appear. These are often based on neural network language models [15], requiring large amounts of computational power to derive an accurate and functional model. This approach is, therefore, considered a unsupervised learning methods because relations and corresponding vectors are derived by the model itself without external assistance [16].

As the embeddings result in a high-dimensional vector representation of the words, it is necessary to take measures to decrease complexity of these generated vectors. Dealing with high-dimensional vectors increases the required computational power while also leading to more noise in the data, thus attributes that do not contribute to the overall meaning lead to cluttering of the data. Cluster identification and, consequently, the extraction

of similar words also proves to become more difficult in higher dimensions. Commonly, dimensionality reduction technologies, such as Principal Component Analysis (PCA), are applied to reduce the dimensions of the extracted vectors and thus enable faster and less resource-intensive processing of the data.

One example of a two-dimensional representation of such word embeddings is depicted in figure 2. This representation was created using a pre-trained embedding model on which

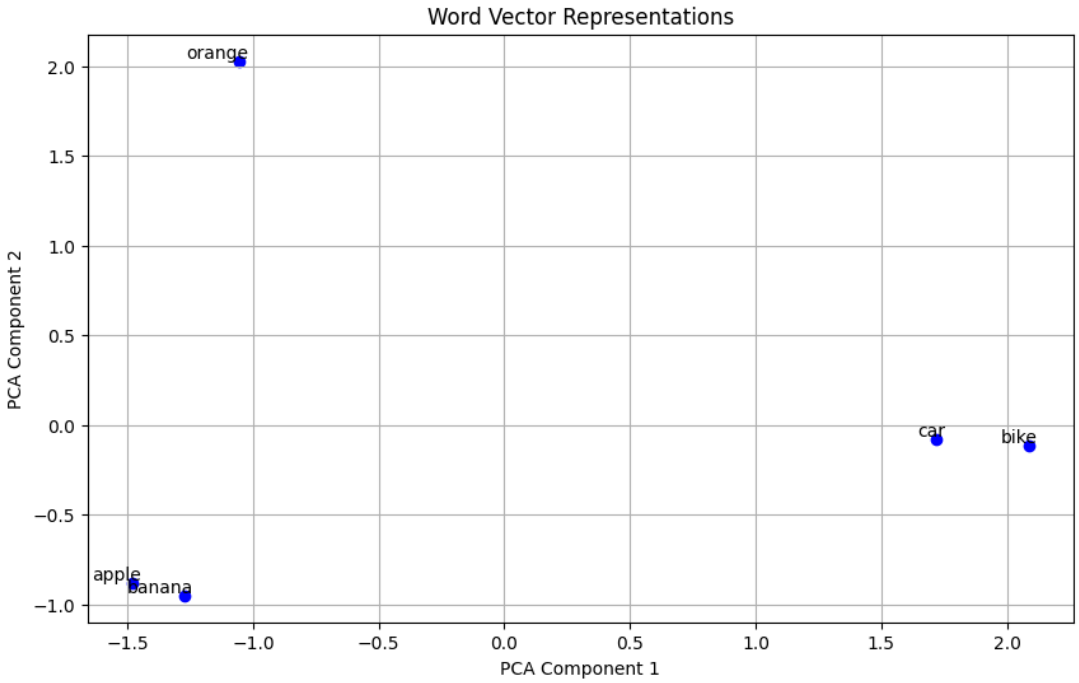


Figure 2: Example word embedding.

dimensionality reduction using PCA is applied to reduce the multi-dimensional vectors to a two-dimensional representation for better visualization. This reduction is done solely for demonstration purposes because the reduction to merely two dimensions proves invaluable in a real scenario. The proximity of similar words like "apple" and "banana" or "car" and "bike" demonstrate the application of the embedding model to correctly infer similarities among words in a text.

Limitations of word embeddings include the inability to distinguish different meanings of words and the required large amount of training data to initially train such a function because these need to learn the corresponding word vectors by tuning a model accordingly. To circumvent the issue of having to provision excessive amounts of computational power, pre-trained models can be used to solve most commonly occuring embedding tasks, enabling a faster and more goal-oriented workflow.

Embeddings can be stored in vector databases that allow for sophisticated querying and retrieval of stored data based on similarity measures. These vector databases provide numerous advantages when compared to traditional, relational database management

systems [17]:

- Sophisticated similarity search, providing a dedicated method for retrieving items in the high-dimensional vector space based on their calculated similarity. Traditional database systems have shortcomings here as they only allow for data querying based on pattern matching or a preset list of criteria.
- Improved performance in comparison to traditional methods because these databases are created for the prime purpose of enabling highly efficient vector operations. This enables the integration with large-scale Artificial Intelligence (AI) systems by providing a sound foundation for delivering query results.
- Support for unstructured data based on embedding this data into a vector representation, enabling the handling of this data, which is unfit for a traditional database schema.

The most common approach to calculating similarities in a vector space and, therefore, a vector database is cosine similarity [18], which allows for quick distance calculation between vector representations of words and is also the method of choice for implementing such a similarity search in the following.

3.3 Natural Language Processing

What is nowadays coined as Natural Language Processing (NLP) is an interdisciplinary field consisting of elements of linguistics and computer science to enable automated processing and analysis of large amounts of unstructured, textual data representations [19]. NLP is one of the major subfields of Machine Learning (ML). Common applications of NLP include Named Entity Recognition (NER), text classification, sentiment analysis, machine translation, and word embeddings [20]. These technologies can, therefore, aid in understanding and using natural, spoken language to enable the development of more sophisticated and robust systems that extract useful information out of textual data, relying on natural language to derive information. In the context of this project, NLP provides a means of extracting information from course and training descriptions, which is then further enhanced to enable linking of data and the subsequent generation of a knowledge graph. To enable such processing, both entity extraction and text classification need to be employed to retrieve relevant data from the provided textual training descriptions.

3.3.1 Named Entity Recognition

Deriving meaningful data from textual data usually involves recognizing and tagging entities within the text, referred to as Named Entity Recognition, which is a subfield of NLP. An entity can be any information that occurs as part of the text, including but not limited to persons, dates, numbers, places, quantities, or organizations [21], representing information that can be assigned to any fixed, preselected category.

An example of such an extraction is depicted in figure 3, highlighting the recognition of products and organizations in the analyzed text snippet. NER can be accomplished using either rule-based methods, machine learning, or deep learning approaches depending on the complexity and use case of the specific problem [22]. Rule-based approaches tend to involve a high manual effort to construct these rules and patterns and thus will not be considered any further, especially as machine learning nowadays provides the predominant toolkit to achieve clean and robust entity extraction.

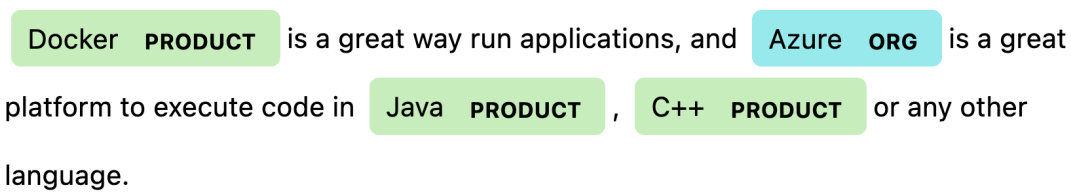


Figure 3: NER depiction.

Use cases for NER include rumour detection, sentiment analysis, topic detection, and event detection from unstructured data, such as text [23]. NER is also commonly applied

for information extraction, providing named entities that represent required information in a given text or for enabling question-answering systems, such as chatbots, in which the user's question is mapped to entities, which are then used to retrieve relevant information.

One of the most commonly used tools for extracting such entities from texts is the free and open-source Python library SpaCy [24]. This library is also commonly used for NLP tasks because it also enables tokenization, text classification, and lemmatization, which will be further covered in section 3.3.2 whilst also providing pre-trained pipelines for various languages [25]. Other commonly used tools include the Natural Language Toolkit (NLTK) or services provided by common cloud computing vendors, such as Google Cloud Platform (GCP) or Amazon Web Services (AWS).

Challenges for NER are directly related to the nature of the simple extraction, which is carried out by such approaches. One major challenge is the disambiguation of entities, posing a problem when an extracted entity might be assigned to more than one corresponding class [26]. This can lead to problems especially when dealing with ambiguous terms, such as the entity "Apple", which might be both an entity of type "fruit" or an entity of type "organization" when referring to the company. Solving this ambiguity proves to be quite challenging when using only entity extraction because it does not consider any contextual information and, therefore, does not consider any intrinsics of the textual data.

Another challenge for entity extraction is the presence of domain-specific terms and the handling of these. An example is the entity "Python", which again may be an entity of type "animal" or an entity of type "programming language", depending on the contextual domain. This poses a similar challenge to the disambiguation explained before, requiring additional context or domain information to provide a satisfactory extraction and assignment to the desired entity type.

3.3.2 Text Classification

Text classification problems generally rely on a multitude of different algorithms, which are used to solve these. As text is unstructured data, these classification problems are non-trivial and, therefore, are commonly solved using supervised learning techniques, such as k-nearest neighbors or support vector machines. For different use cases, it is possible to differentiate the text classification systems by the scope they are using, for example it may be sufficient to classify texts on the document level in cases in which the class for a complete document is to be determined, but there may also be cases needing a more detailed classification of paragraphs or sentences [27]. A text classification problem can be deconstructed in four general phases which are carried out sequentially as depicted in figure 4 [27].

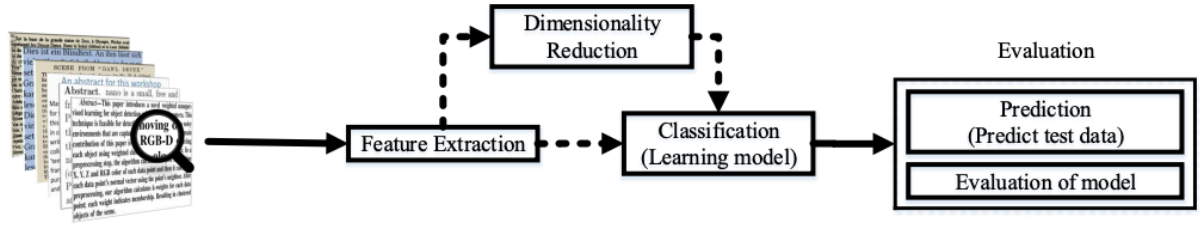


Figure 4: Text classification process according to [27].

During the feature extraction phase, the unstructured text data is converted into a structured representation to be able to use it in a mathematical classification model because textual data is unsuitable to be analyzed by any machine. This phase consists of stopword removal, stemming, and lemmatization before common feature extraction mechanisms, such as Term Frequency-Inverse Document Frequency (TF-IDF), are applied [27]. Stopword removal eliminates all stop words from the document because most words do not contribute to the overall meaning of a text corpus [28]. Stemming algorithms are used to reduce words with the same root to their common form by removing derivational and inflectional suffixes [29]. Lemmatization identifies the basic form of any inflected word to further simplify the text corpus to be classified [30]. In general, it is not necessary to execute both the lemmatization and stemming step because both reduce words to their common form, the difference being that stemming simply removes suffixes, whereas lemmatization results in the words being reduced to their dictionary form. An illustrative example can be made using the words "alumnus", "alumnae", and "alumni", which results in "alumn" after stemming. This common form does, however, not provide the correct contextual information. Lemmatization can be especially useful for words whose root form is not necessarily included in the concrete form, such as the word "better", which may be lemmatized to "good".

Due to natural language texts containing a multitude of unique words, preprocessing can be slowed down due to resulting high requirements in runtime and memory complexity [27]. This phenomenom can be mitigated using either inexpensive, faster algorithms, or dimensionality reduction algorithms, such as Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), or Non-Negative Matrix Factorization (NMF) to achieve less complex data that decreases aforementioned complexity issues [27]. It is, therefore, essential for maintaining a reliable text classification system to integrate an appropriate preprocessing stage before classifying based on the simplified data.

The classification of textual data can be achieved using a variety of possible algorithms of which a selected overview of commonly used approaches is depicted in table 2.

As the choice for a classification algorithm is heavily dependent on the problem and data

Algorithm	**Advantages**	**Disadvantages**
Naïve Bayes	Easy implementation	Limited by data scarcity
Logistic Regression	Fast training possible	Limited to two-class classification problems
Support Vector Machine (SVM)	Best for smaller data sets	Lacking transparency in the results
K-Nearest Neighbors (KNN)	Robust for large datasets, easy handling of multi-class problems	Difficult determination of K, high computational and storage effort, performance depends on data
Random Forest	Useful for large datasets	Requires high amount of computational power

Table 2: Overview of classification algorithms according to [31][27].

set at hand, it is necessary to conduct an in-depth analysis of each approach to determine the most fitting candidate.

Evaluation of text classification algorithms relies on different metrics commonly used in machine learning. One of the predominant metrics is accuracy, which is defined as follows:

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

Accuracy, therefore, reflects the ability of the system to match the predictions with the reality that is to be modeled [32]. To obtain this metric, the model needs to predict the outcomes for test data which has known labels, enabling the comparison of the predicted labels with the data's real labels [32]. It needs to be noted, however, that relying solely on accuracy may tend to ignore the confidence of a model's prediction. This proves to be a problem because the model might provide a confidence that is just above fifty percent whilst reaching accuracy scores that are satisfactory, therefore hiding the model's low confidence [33]. To overcome this issue, accuracy is commonly used in conjunction with further metrics, such as precision and recall [27]. Precision is defined in the following way [34]:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

Recall is defined as follows [34]:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

To simplify evaluation of a classification algorithm, these commonly used metrics can be visualized in a so-called confusion matrix. This matrix combines precision and recall and aids with analyzing the performance of the model.

3.4 Recommendation Systems

The challenge of recommending content to any users of a system is one that has multiple applications, such as recommending jobs [35] or items, for example products in online retail or multimedia content, to end users. This helps users to find more relevant information in massive amounts of often unstructured data (such as texts or images on websites), thus leading to a decrease in manual effort and an increase in user satisfaction when interacting with systems which use such recommendation systems in an appropriate and goal-oriented way. One major issue with most recommendation systems currently in use is that they mostly sacrifice quality for a faster result, causing a reduction in user satisfaction and, consequently, the user's reluctance to interact with the system [36]. Due to this, it is also of importance to develop a system which balances recommendation quality and performance to achieve an optimal user experience and user retention.

In general, recommendation systems rely on a set of users $U = \{u_1, u_2, .., u_m\}$ with a length of m and a set of items $I = \{i_1, i_2, .., i_n\}$ with a length of n, on which either a prediction, a recommendation, or a combination of both can be carried out [37]. A multitude of different approaches exist for such recommendation systems, the most prominent ones being collaborative filtering, content-based methods, or a combination of these in so-called hybrid methods, which will be discussed in the following sections.

3.4.1 Collaborative Filtering

Collaborative filtering is considered to be the predominant approach in recommendation systems [37]. It is commonly applied when recommending movies or products on online platforms that rely on a large user base [38]. In collaborative filtering, recommendations for a user are derived by finding similar user profiles and, therefore, recommending items based on the similarity to other users' items. For this, users U and items I form a matrix $U \times M$ of dimension $m \times n$ to capture user-item interactions that are represented as the user u's respective rating r on a scale from one to five (e.g. a five star rating [39]) for an item i in the following way:

$$r_{ui} = \begin{cases} r & \text{if the user has provided a rating, where } r \in [1, 5] \\ 0 & \text{otherwise} \end{cases}$$

An example for capturing such user-item interactions is depicted in table 3, highlighting the ratings of different users for a set of items.

[37] define memory-based and model-based collaborative filtering algorithms as two major algorithm classes suited for approaching this problem.

In memory-based algorithms, a set of users (the neighborhood H) with similar preferences

User	Item A	Item B	Item C	Item D
1	4	5	0	3
2	2	1	4	0
3	0	5	2	1

Table 3: Example matrix for collaborative filtering.

to the target user is derived from the whole user-item data set, of which recommendations are extracted according to the combination of the items that this neighborhood of the target user has also rated highly [37]. This is commonly achieved using KNN or user-based collaborative filtering [37].

Model-based approaches develop a probabilistic model to predict the target user's rating for a given item and derive the relevant recommendation by evaluating these predicted ratings. Models commonly used are convolutional neural networks (CNN) [40] or perceptrons in combination with advanced optimization techniques, such as hyperparameter tuning and gradient descent [38], leading to an increased training effort for employing such a model.

One major issue of the collaborative filtering approach is the cold start problem, indicating that recommendations are especially difficult to do correctly if there is little to no initial rating data available. Thus, a differentiation is made in *new item cold start problems* and *new user cold start problems* [41], implying that the issue persists when new items or new users are added to the system and, consequently, little information is available about the respective data. Therefore, the approach might be unsuitable for a problem for which no or little initial data exists, as low-quality recommendations tend to worsen the user's experience and will not prove to be of value to the end user, leading to users not actively using such a system and countering the intended purpose of improving user engagement [37], which also leads to a decrease in system performance because less new ratings are likely to be included if less users actively take advantage of the recommendations that are provided. It is, therefore, crucial to also employ techniques to mitigate the cold start problem, especially when first introducing a recommendation system based on collaborative filtering that does rely on novel data which has no previous information attached to it.

The cold start problem can be overcome by enhancing the items to be recommended using auxiliary data, such as meta information that is derived from the textual representation in case of course descriptions, or from analyzing images in case of visual data. This contextual data can be a description of the item, a category, or the price range [41]. By gathering this additional information, it is possible to find similarities between new and existing items using their textual vector representations, thus also enabling a recommendation for previously unknown items by leveraging the calculated similarity scores [41].

A similar approach to alleviate the cold start problem by including contextual data for recommending suitable items has been proposed by [42], highlighting the importance of combining collaborative filtering with such data to increase the value of recommendations that are made.

Another approach to alleviate the cold start problem is proposed by [43], in which a combination of item content information and item clustering is used to derive a decision tree, which then associates the newly entered items with the existing ones, thus also providing an approach to incorporating previously unseen items in a recommendation system. This approach, however, requires further effort to construct and maintain the associated decision tree accordingly, thus increasing the overhead of developing and maintaining a recommendation system based on this enhanced technique.

3.4.2 Content-Based Methods

In contrast to collaborative filtering, content-based methods rely on knowledge about the items to be recommended. This knowledge can be extracted from unstructured representations, such as textual item descriptions, emphasizing the need for correct entity extraction and classification. The extracted information can then be used in conjunction with a user profile to determine whether to recommend an item or not [44]. The major difference to collaborative filtering lies in the fact that content-based methods do not rely on the availability of defined user ratings to propose suitable items to a new user, but rather on attributes of the items to be recommended and the target user's personal preferences. Based on this distinction, content-based methods require a good knowledge basis for providing suitable and tailored recommendations to potential users. This knowledge base can be provided in the form of a knowledge graph, improving the understanding of the items to be recommended by leveraging semantic information about them [45].

Similar to collaborative filtering, content-based methods also suffer from the new user cold-start problem when little to no initial data exists. To overcome this issue, multiple approaches exist. It is proposed to use a set of interlinked data to supplement the existing items and users with additional semantic data, providing more relevant recommendations while also ensuring effectiveness [46].

Additionally, content-based methods often suffer from other challenges, such as limited content analysis, sparsity of ratings, reciprocity, and the improvement of the system's accuracy [47]. These issues need to be addressed in a content-based approach in order to not decrease the recommendation quality due to limited information or other aforementioned issues.

3.4.3 Hybrid Approaches

As both collaborative filtering and content-based methods each have various disadvantages, such as the cold-start problem and the reliance on knowledge about the items to be recommended, hybrid approaches have been developed to overcome common issues of these methods. A hybrid recommendation system leverages a combined approach to creating recommendations, thus mitigating the presented issues that are commonly associated with either approach that was introduced before.

To achieve a higher accuracy and thus improved recommendations, [48] develop a hybrid recommendation system which combines collaborative filtering with content-based methods to overcome the cold-start problem by augmenting the data with additional contextual information. Additionally, hybrid methods aim to solve the problem of items having no or too little ratings, which would otherwise result in these items not being recommended if no ratings exist [36]. Another approach for developing hybrid recommendation systems is the combination of an alternating least-squares collaborative filtering algorithm with deep learning, both enhancing the overall performance and quality of the recommendations and solving the associated cold start problem [49].

3.4.4 Evaluation Methods

In order to validate the effectiveness of a created recommendation system, it is of crucial importance to execute a thorough validation of the produced recommendations. According to [50], this evaluation can be carried out as an online or offline evaluation, while [51] propose more comprehensive user studies as a third evaluation method for recommendation systems.

Offline evaluation mainly focuses on analyzing the recommendation quality based on a static data set, which is then split into test and training data to enable a validation of the proposed output using different metrics [50]. According to [52], these include common metrics, such as Root Mean Square Error (RMSE), Mean Average Error (MAE), precision, recall, and F-score. [53] also provide a more detailed overview of evaluation metrics, providing coverage and serendipity as additional approaches. [54] define serendipity as the system's ability to provide surprising and unexpected results that are still valuable and relevant to the user. Another definition of serendipity is that such items need to be novel, of relevance, and also not expected by the user [55]. This serendipity is, therefore, of essential importance when evaluating recommendation systems because this will lead to users being more engaged and interested in using the system when they feel that they receive well-defined recommendations that they would not consider otherwise. Additional non-accuracy metrics are diversity and novelty [56], providing further evaluation possibilities that lie beyond the traditional aforementioned measures. An overview of possible

metrics with their respective definition is depicted in table 4.

Metric	Definition
RMSE	$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$
MAE	$\frac{1}{n}\sum_{i=1}^{n}\lvert y_i - \hat{y}_i\rvert$
Precision	$\frac{TP}{TP+FP}$
Recall	$\frac{TP}{TP+FN}$
F1-Score	$2 \cdot \frac{\text{Precision}\cdot\text{Recall}}{\text{Precision}+\text{Recall}}$

Table 4: Overview of numeric evaluation metrics.

As not all metrics are commonly used when evaluating a recommendation system, a subset of these needs to be defined for real-world scenarios to enable targeted and precise evaluation.

Online evaluation is carried out as an experiment containing real users who interact with the system under test, providing a more holistic and realistic view on the system's usability and applicability for the given use case [50]. Consequently, online evaluation tends to be more complex because users need to act naturally while monitoring their interactions and corresponding measures need to be defined to derive valuable insights from these interactions. During an online evaluation, users are free to interact with the system in any way they deem appropriate or necessary, which provides more room for exploration and realistic insights [57]. Online evaluation provides a number of benefits when employing such an approach for the evaluation of a recommendation system [50]:

- Better reliability of results, as real user feedback is collected.
- Users can state their opinion openly and provide any additional information they deem necessary.
- Real performance data can be collected, as users interact in a live setting.

Drawbacks of online evaluations include the difficult reproducability, a higher total effort that is required, and difficult scalability of the evaluation [50]. Consequently, online evaluation provides a better insight into the real application of such a system while also enabling the collection of real user feedback when comparing it to offline evaluation.

User studies are characterized by having a relatively small number of human test subjects who perform a variety of predefined, fixed tasks in interaction with the recommendation system to be tested [58]. The goal of such a study is to gain insights into how users interact with the system while also incorporating live data (for example performance metrics) and the perceived value for users [51]. The main advantage lies in the study's ability to derive insights from interactions with the system, mimicking the later usage of it in a

practical real-world scenario. Disadvantages of a user study consist of the high amount of required preparation to define tasks and the close observations and measurements, which are needed to accurately capture the user interactions in a data-driven way [51]. Additionally, according to the Hawthorne effect it might prove to be difficult to have test users act naturally when they are aware that they participate in a field study [59].

4 Semantic Layer

To provide a foundation for the recommendation system and to avoid the cold start problem as described in 3.4.1, it is first necessary to create a knowledge graph for providing semantic metadata. This knowledge graph needs to be developed based on existing training data, which can then be combined with openly available linked data to provide complementary and additional information. To gain a basis for creating the knowledge graph that connects trainings, topics, and related entities, such as technologies and skills, it is required to develop an approach for extracting these topics and entities from existing training descriptions. Therefore, it is necessary to obtain a data set of training descriptions, which can then be used to train a classification model for automatically extracting course topics and technologies or skills that can be acquired from participating in any such course. Building on these classes and extracted entities, a knowledge graph can be derived similar to the approach of [10] by combining the extracted information and linking existing courses according to the derived data as depicted in figure 5. This approach enables parallelization of tasks, thus reaching better efficiency of the overall graph creation. This is especially important because the created graph does not remain static but is subject to constant change due to the changing underlying data because courses might be added or removed in the respective training platforms.

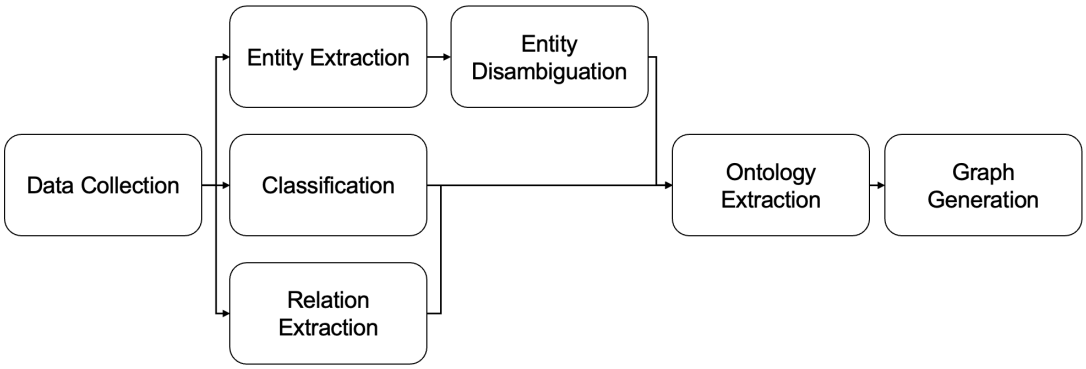


Figure 5: Knowledge graph creation.

Starting with the data collection, an initial set of data is required as a foundation for extracting knowledge. This data is represented as unstructured textual descriptions of training courses, which provide information about used technologies and other entities. These entities then need to be extracted accordingly in order to be able to include them as nodes in a knowledge graph, providing a need for entity disambiguation to resolve ambiguities between entities after the extraction phase.

A classification of the course descriptions is carried out parallel to the entity extraction, allowing for additional insights and, therefore, additional knowledge, which can then be integrated in the graph. Relations between those entities are extracted to form the

basis for creating triples for the knowledge graph. Following the phase of information retrieval, the extracted information is then processed to generate an ontology that is suitable for representing said knowledge. This ontology is subsequently used to populate the knowledge graph with meaningful data.

4.1 Data Collection

As a foundation for the implementation of the classification and entity extraction algorithms, it is necessary to have a data set on which these algorithms can be trained and subsequently evaluated. In order to obtain a diverse and varied data set for optimum results, a combination of publicly available data from existing employee training providers and synthetic data is used. The concept of synthetic data supports the creation of a sufficiently large data set by combining actual human-made data with generated data, which will be explained in the subsequent sections.

4.1.1 Course Data

Available data includes textual course descriptions obtained from the currently used providers of employee video trainings. These are Pluralsight as well as LinkedIn Learning, both of which provide a multitude of training content for different target groups. Available courses are extracted from both sites and saved accordingly for a subsequent usage when constructing the knowledge graph based on this available data.

In order to retrieve course data, it is necessary to either have programmatic access via an Application Programming Interface (API) or to create the data set manually by selecting relevant data and copying it into a structured file format, such as a Comma-Separated Value (CSV) file. As neither PluralSight nor LinkedIn Learning provide open access to their contents via an automatically callable API, the manual approach is chosen for creating an initial data set of real-world courses on which the system can be trained. This manually obtained dataset will be referred to as dataset D_1, as this is later used as a foundation for creating the semantic layer. As a result of the manual extraction, a total of 34 courses and corresponding descriptions are derived from the learning platform providers. Consequently, this data set is too small for useful classification or NER training, which requires a solution to increase the data size without relying on manual data extraction.

4.1.2 Synthetic Data

With the increased adoption and performance of Large Language Models (LLMs), it is possible to not only rely on curated or manually created data sets for NLP tasks, but it is now also feasible to automatically create data sets, simplifying and improving many

classical NLP problems with regards to the obtaining of corresponding data [60]. This automated creation of data sets proves to be of use, especially when creating or obtaining a usable data set is complex or involves a high amount of manual efforts [61]. One major issue with synthetic data creation is that any bias of the underlying LLM is, consequently, also included in the generated synthetic dataset [62]. To produce a varied and realistic data set using a LLM, it is necessary to reach a sufficient amount of topical diversity and a close alignment with the stylistic characteristics of corresponding human-created data [60], strengthening the need for accurate and precise prompting. To achieve such diversity, it is possible to augment existing data with synthetic data using rather simple prompts to generate examples that are semantically similar [61] or by asking the model to do a rewrite of the input data resulting in an adaption of the writing style [60].

As course data for employee trainings is not sufficiently publicly available or easily retrievable, it is deemed appropriate to employ such a synthetic data generation approach to create a sufficiently large data set for subsequent classification and named entity extraction. This data set is created using OpenAI's ChatGPT in its current publicly available version 3.5, based on prompts that accurately reflect the intent of creating diverse and realistic descriptions.

The prompts used to obtain this data are structured in the following pattern:

Create an amount of N training descriptions for the label L. Focus on providing varied and diverse results and incorporate a wide range of technologies and concepts.

By prompting for each label individually, topical diversity is ensured and further encouraged by also adding the focus on variety and diversity. Additionally, example descriptions are included when interacting with ChatGPT to further strengthen the quality of the resulting data set.

4.2 Course Classification

To enable a structured clustering and corresponding labeling of the currently available courses, a clustering approach needs to be developed and subsequently implemented. Therefore, target classes that the course descriptions belong to have to be defined before a corresponding classification model can be selected, trained, and validated. To define meaningful classes for this task, it is first required to analyze the target audience which will be using the recommendation system. This target audience primarily consists of software engineers and technical consultants because these constitute the majority of the workforce of most software companies. These roles typically have a background in business information technology, computer science, or business administration and, therefore, need tailored training recommendations based on their roles and backgrounds while also taking

into account future developments and trends in the ICT sector. This justifies a focus of target classes on technical and closely related topics to best cater to the requirements of this group. Additionally, an emphasis on modularization and extensibility needs to be included to allow for easy adaptation of the classification to varying use cases and scenarios.

4.2.1 Definition of Target Classes

As any classification model needs a set of target classes, these have to be developed before starting model training in order to be able to provide a proper classification. These can be derived from the data set's main topical areas and the targeted skills to be obtained through these trainings. Therefore, the data set is to be analyzed accordingly. In the following, these classes are presented including their labels, and an explanation of each class is given.

Mobile Development includes courses and trainings that contain relevant learning in the areas of native application development for Android and iOS as well as those that cover hybrid app development in frameworks, such as Flutter, Ionic, or React Native. These courses also cover other aspects of app development, such as deployment to app stores or integration of mobile-friendly features, and serverless backends, such as Google Firebase.

Frontend Development covers all content about developing web-based user interfaces and applications. This includes HTML and CSS as well as web development frameworks, such as SAP UI5, React, Angular, and Vue.js.

Backend Development contains languages, such as ABAP, C++, Java, and C#, whilst also accomodating a variety of backend frameworks including but not limited to Spring Boot, Node.js, Express, and Django. These courses are usually closely related because they focus on API development or server-side scripting and programming.

Soft Skills do not have a defined technology or framework that enables their classification. These trainings rather focus on providing employees with different skill sets and methodologies, such as resilience and adaptability or presentation skills; therefore, little to no overlap exists with the other defined target classes.

IT Security covers relevant trainings on topics related to IT security, such as securing web applications, developing resilient systems and maintaining data center security using firewalls and other defensive measures.

Management and Business covers trainings related to topics such as strategic leadership, people management, and negotiations, but also includes contents covering various areas of economics, such as finance and accounting or sales.

Cloud Engineering defines content related to developing, deploying, and maintaining cloud-based software. This typically includes container orchestration technologies, such as Kubernetes and containerization tools, e.g. Docker, but also concepts, such as serverless computing or Continuous Integration/Continuous Delivery (CI/CD). Additionally, cloud platforms, ecosystems, and major vendors of Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS), and Infrastructure-as-a-Service (IaaS) fall into this category.

Product-specific Training includes content that can be directly linked to a specific product or piece of software that is addressed and further explained within the training. These can include enterprise software, such as Enterprise Resource Planning (ERP) or Customer Relationship Management (CRM) solutions, but may also cover consumer-facing products, such as word processors or presentation software.

Machine Learning covers trainings on various topics of ML and Artificial Intelligence (AI), ranging from trainings related to the implementation of common algorithms, such as SVM or KNN, to more advanced concepts, such as deep learning and reinforcement learning.

Software Architecture describes content related to designing software in a maintainable and structured way to support modularity and encapsulation while also incorporating common design patterns and best practices. The focus hereby lies more on abstract concepts than on concrete implementation examples in order to differentiate from the above mentioned classical development topics.

Naturally, there exists some overlap between these classes, such as when a course covers full-stack development and includes both backend and frontend development. In order to overcome this issue, it is necessary to also consider the relevant entities that occur in the course descriptions, enabling a linking of such closely related courses even if they are classified in different classes. This second step is further described in section 4.3. These defined classes are subsequently also used as entities for the knowledge graph, providing the first set of these, which are then linked with their related courses. After defining and explaining the target classes for course classification, a corresponding classification model needs to be selected and trained accordingly to realize the subsequent implementation of this model.

4.2.2 Data Preparation

To achieve optimum results when classifying texts, it is first necessary to preprocess the data set accordingly. Therefore, a preprocessing pipeline is developed which simplifies further analysis of the data at hand when dealing with the classification. As discussed in chapter 3.3.2, it is initially necessary to remove stopwords and apply a feature extraction mechanism. Therefore, stopwords are first removed from the data set by applying a

custom-built function that iterates over the data set, checking if any stopwords occur and removing them once they are identified. The result of this stopword removal process results in the description

iOS App Development with Swift and SwiftUI Master the art of iOS app development with Swift and SwiftUI, covering essential concepts like UI design, Core Data, and integrating with RESTful APIs.

being shortened to

iOS App Development Swift SwiftUI Master art iOS app development Swift SwiftUI, covering essential concepts like UI design, Core Data, integrating RESTful APIs.,

highlighting the impact of stopword removal on text simplification. It can be clearly seen that the removed words do not impact the overall meaning and content of the given description, thus enabling a better performing algorithm by removing these. Additionally, capitalization can be removed to further simplify the given text.

After simplifying the text, TF-IDF is then applied as a formal feature extraction mechanism. This results in a pre-processed data set, which is then split into training and test data with their according labels that correspond to the target classes defined in section 4.2.1. This split is carried out on the whole available labeled data set, where twenty percent of the data are used for testing and eighty percent are used for training the models subsequently. This split concludes the data model preprocessing because stopwords are removed, TF-IDF is applied and a corresponding train-test-split is carried out to gain a basis for model training with the labeled data.

4.2.3 Model Evaluation

As discussed in chapter 3.3.2, a multitude of algorithms may be applied for solving the problem of classifying texts. In the following, a practical criteria-based evaluation of these is carried out based on the data set obtained in chapter 4.1. Before obtaining concrete measures from each of the approaches, those that do not fulfill the requirement of robust multi-class prediction are removed from the test because these do not provide the necessary functionality that is required. Therefore, logistic regression is not considered further because it only supports two-class classification by default. It is, therefore, necessary to implement each approach in Python and collect the necessary metrics for a stringent and goal-oriented comparison, which then serves as the basis for deciding which model to use. To achieve a comparable outcome for an informed choice of the model to use, each model is trained using the same preprocessed data set that was obtained before.

As precision, recall, and F1-score are determined on an individual class level, it is necessary to aggregate these in order to be able to compare the overall model performance instead

of comparing each model on a class predicition basis. To calculate an aggregated value, micro-averaging, macro-averaging, or weighted average can be used. For the problem at hand, the most suitable approach is weighted averaging because this takes into account the class imbalances and proves to give more importance to the performance of the prediction of larger classes, which is beneficial because these tend to be the ones that also have a higher impact on user satisfaction.

The formula for calculating weighted-average precision is defined as

$$Weighted - P = \frac{1}{N} \sum_i \left(\frac{TP_i}{TP_i + FP_i} \right) \cdot \frac{(TP_i + FN_i)}{\sum_i (TP_i + FN_i)}$$

Weighted-average recall is, consequently, defined as

$$Weighted - R = \frac{1}{N} \sum_i \left(\frac{TP_i}{TP_i + FN_i} \right) \cdot \frac{(TP_i + FP_i)}{\sum_i (TP_i + FP_i)}$$

and weighted-average F1-score is defined as

$$Weighted - F1 = \frac{2 \cdot Weighted - P \cdot Weighted - R}{Weighted - P + Weighted - R}$$

In the above formulae, TP is a true positive and FP a false positive, consequently, TN is a true negative whereas FN is a false negative. For each model, the accuracy, $Weighted{-}P$, $Weighted - R$, and $Weighted - F1$ is subsequently obtained as shown in table 5.

Model	Accuracy	$Weighted - P$	$Weighted - R$	$Weighted - F1$
Naive Bayes	82%	78%	82%	79%
SVM	95%	95%	95%	95%
KNN	77%	77%	77%	75%
Random Forest	90%	90%	90%	89%

Table 5: Classification model evaluation result.

The results clearly show that SVM outperforms each of the other candidates in every category, delivering the best performance in each of the metrics. Random Forest also proves to be a viable option, even if it is not performing as well as SVM. For the purpose of classifying training descriptions as a basis for the knowledge graph generation, SVM will be the model of choice to achieve a robust and accurate multi-class classification result in each possible scenario.

To analyze the potential downsides of the models, a runtime analysis is carried out to compare the training runs for each individual model to determine if there is any trade-off associated with the evaluation results that were achieved. This runtime analysis is carried out 1000 times for each of the four classification algorithms and the values are plotted correspondingly in figure 6 . The runtime is scaled logarithmically because the Random

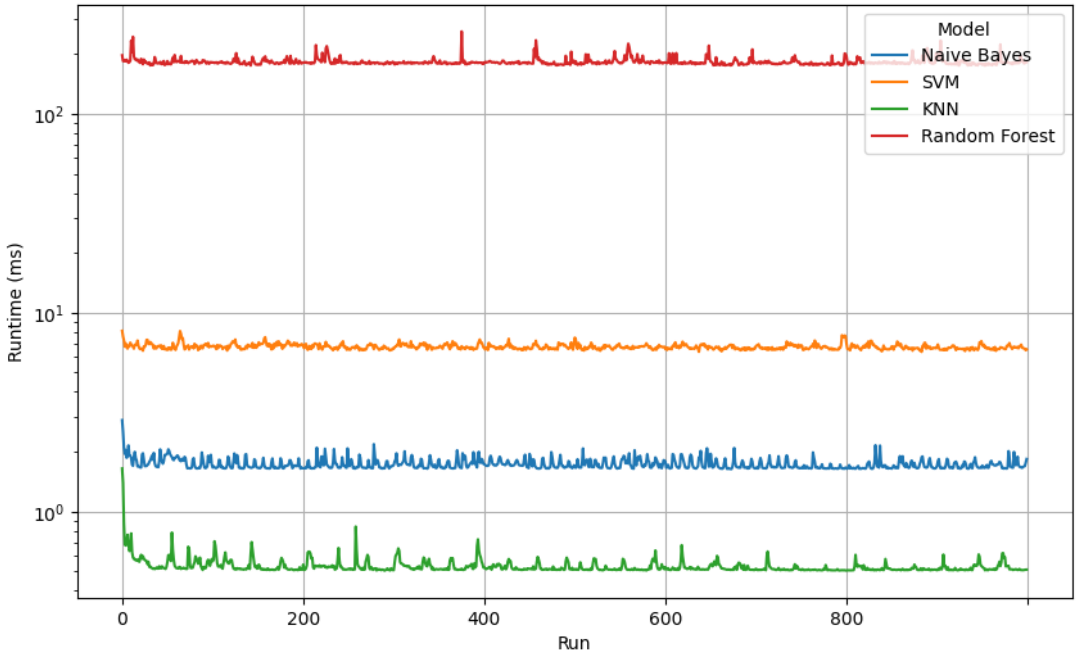


Figure 6: Classification model runtimes.

Forest model performs exceptionally worse than all of the other models. Interestingly, this model is also the one that performs second best in the overall evaluation but this performance comes at the cost of execution time. Similarly, SVM performs second-worst in the overall runtime analysis but delivers the best overall performance in terms of metrics, implying that this result also comes with a trade-off in performance, which is, nevertheless, not as impactful as the performance loss of the Random Forest model. KNN performs best in terms of execution time but is, consequently, also the worst in terms of overall metrics, similar to how the Naive Bayes model performs second best in overall runtime but also second-worst in overall metrics. The mean runtimes derived from the executed runs are displayed in table 6.

Model	Naive Bayes	SVM	KNN	Random Forest
Mean Runtime	1.73 ms	6.74 ms	0.53 ms	182.71 ms

Table 6: Mean model runtimes.

Another analysis to be carried out is the model's runtime with increasing data set size because the data set might grow indefinitely in a productive environment. In order to be able to properly evaluate the corresponding behavior, the trainings are carried out individually for each model with differently sized training data sets. The result of the runtime evaluation with an exponentially growing dataset is shown in figure 7. The result of this analysis clearly shows that all algorithms increase in runtime with an increase in data, but the increase happens in an exponential manner. The random forest model still performs worst as expected, followed by the SVM model, which remains on the second-

worst position but is still the one achieving the best overall performance with regards to the evaluation results. KNN and Naive Bayes perform considerably better in terms of runtime with a growing dataset, but a trade-off needs to be made when focusing on their overall classification performance. Additionally, the impact on performance due to a growing dataset is considerable less in a real-world scenario than in this experiment because a total size of $x * 2^n$ is rather improbable. In the experiment setting $x = 684$, therefore, the total size of the largest data set is 350,208 descriptions with an execution time for the SVM classifier of 42.1 seconds, which serves as an upper bound estimate for a worst-case scenario. This upper bound is feasible as it is not realistic to have that many training descriptions.

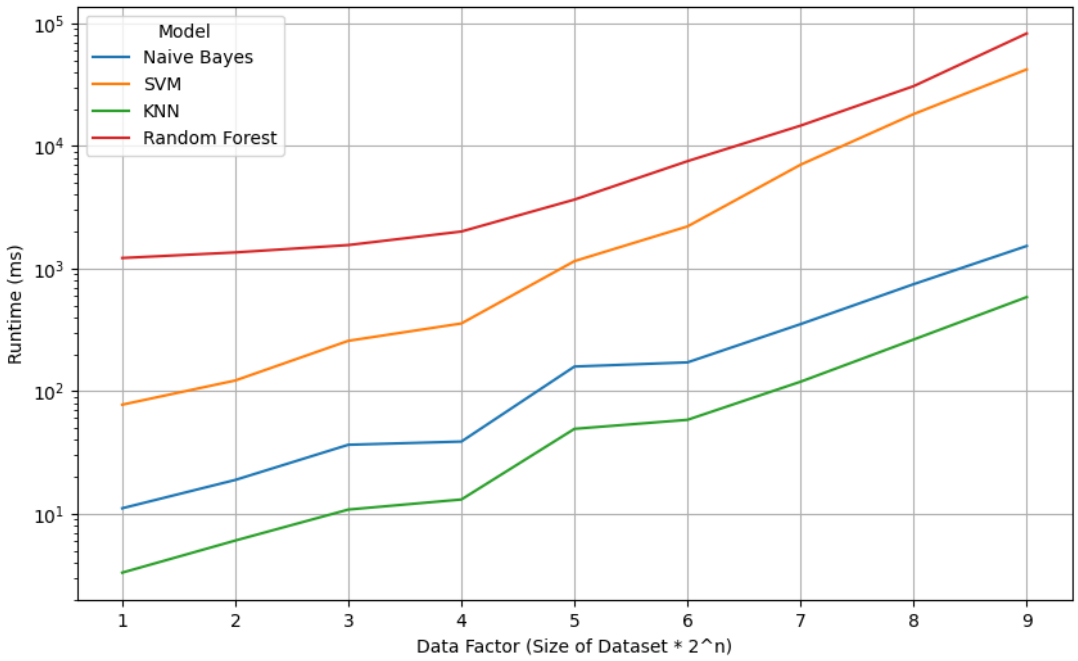


Figure 7: Execution time in relation to dataset size.

The result of the classification of each target class of the SVM model is visualized in the confusion matrix in figure 8, further highlighting the robust classification results that the model delivers. It can also be seen that there exists some class imbalance towards mobile development and cloud engineering, further proving that weighted averaging is the best fit for determining the aggregated precision, recall, and F1-scores across all class predictions. Consequently, even if the SVM model does not provide the best overall execution time, its total performance metrics make it the preferred choice for integrating it into the system as a multi-class classification model.

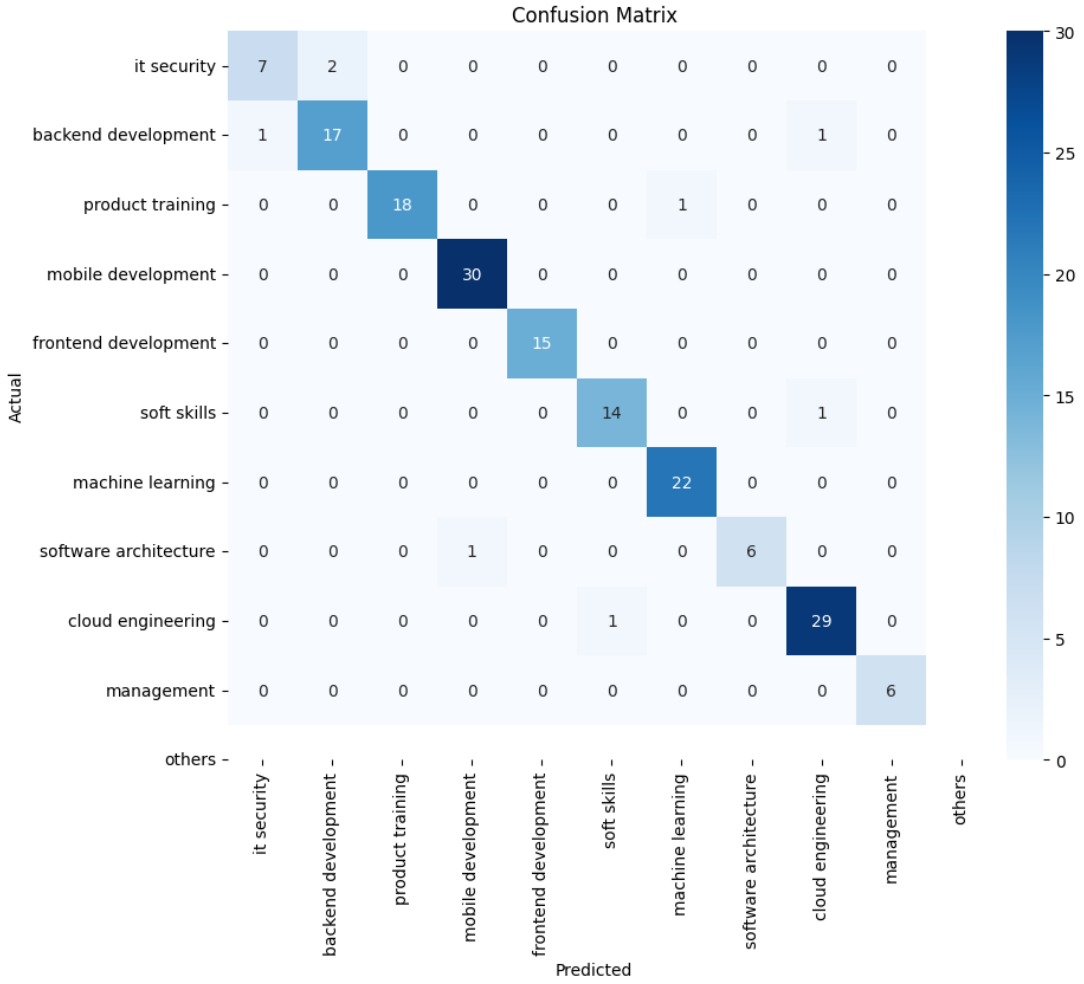


Figure 8: Confusion matrix for trained SVM model.

4.3 Named Entity Extraction

To enhance the knowledge base not just with the trainings' associated class, it is also necessary to extract further information, such as technologies, languages, frameworks, or other entities, from the textual descriptions in the data set. These entities need to be defined properly based on the use case at hand while focusing on generalization of the approach to also incorporate different training descriptions and to enable broad adaptability of the system. After defining the required custom entities, it is also necessary to evaluate whether to train a custom model for extracting these entities or if there exists an approach for achieving a satisfying result without the need for customized training.

4.3.1 Definition of Custom Entities

As most pre-configured NER models contain mostly generalized entities and the context here lies in a specialized domain, the need for custom defined entities arises. These entities need to be defined in accordance with the overall context of the expected data of

the application, emphasizing the need for a correct as well as comprehensive list. It is, therefore, necessary to define these entities in a way that is both expressive but also not unnecessarily bloated to avoid over-specialization of the entities. As most of the expected descriptions belong to technical topics related to software engineering, these provide an ideal starting point for defining custom labels. Mobile development, backend development, and frontend development all rely on *programming languages* (e.g., Java, C++), *frameworks* (e.g., React, SpringBoot), and *technologies* (e.g., iOS, Android), making these the first custom entities. Softskill and management-related trainings provide various inputs regarding professional skills (e.g., negotiation and presentation skills) or methodologies (e.g., SCRUM, KANBAN) that are vital to any employee, resulting in the custom entities *skill* and *methodology*. Product-related trainings also define the need for a *product* entity to extract products, such as concrete ERP or CRM systems.

For the example description without stopwords

iOS tech App Development Swift lang SwiftUI Master art iOS tech app development Swift lang SwiftUI framework , covering essential concepts like UI design, Core Data, integrating RESTful tech APIs.,

relevant entities include Swift (Language), SwiftUI (Framework), iOS (Technology), and RESTful (Technology). These should be extracted automatically by an NER algorithm in order to be able to extract semantic metadata from the textual course descriptions.

As this list of custom entities may be subject to change over time, an emphasis on adaptability of the entity extraction needs to be made to be able to re-use the extractor even if there is a change in the custom entities.

4.3.2 Entity Extractor

In order to evaluate whether a custom extractor needs to be trained, it is first necessary to understand which capabilites a pre-trained model brings. The commonly used spaCy models only recognize entities generic types, such as "Person", "Product", "Quantity", and "Time" among others, making such a pre-trained model without any customization unsuitable for the use case of extracting the required custom entities that are defined in section 4.3.1 [63].

To circumvent the issue of not being able to use a pre-trained model, an evaluation of two approaches is carried out. These are a custom-trained model to extract the defined entities by providing labeled training data to a NER model and the usage of an LLM in conjunction with precise prompting to achieve the same extraction results. To show the results based on real data, the course description depicted in table 7 is used as a basic example for the following NER approaches. When applying the pretrained "en_core_web_trf" model to

the example data, it recognizes "Docker" and "Kubernetes" as entities of type "Person" while also recognizing "first" as being of type "Ordinal", highlighting that it does not provide domain-specific knowledge for solving the issue at hand.

Title	**Description**
Docker and Kubernetes: The Big Picture	Docker and Kubernetes are transforming the application landscape – and for good reason. This course is the perfect way to get yourself – and your teams – up to speed and ready to take your first steps.

Table 7: Example data for entity extraction [64].

A custom entity extractor is trained based on the data set that is created in section 4.1.2 to enable targeted entity extraction. This requires manually labeling the included entities according to their occurence in the data set and subsequently creating a training data set to be used with Python's spaCy library to facilitate the creation of this custom extractor. To achieve fast and error-free labeling, the data set is loaded into Label Studio where the annotations for the training data set can be created in a visual environment. After completion of the manual labeling, the training data is then exported as a JSON format, which is then converted into the required training format for spaCy using a custom-built function. The converted file is used for training a spaCy model with this input to obtain a custom NER model, which is then applied to the dataset D_1. When applying the custom entity extractor to the example data from table 7, only "Kubernetes" is extracted as an entity of type "framework". Docker is not recognized at all, although it is also a technology that should be extracted accordingly. The result of the custom entity extraction proves to be not as reliable as a pretrained model, due to the rather small amount of training data and the fact that not all possible entities occur in a significantly large amount of texts. Additionally, a set of custom entities like these are merely found in sufficiently well-trained models, making it challenging to productively rely on any traditional entity extraction approach.

To enable a better result for custom NER, a LLM is analyzed for this use case. For this analysis, OpenAI's GPT-4 is selected because it provides the most up-to-date model [65]. As this is a general-purpose model for text generation as well as a wide array of other NLP tasks [66], it is necessary to first develop an accurate prompt for achieving the desired outcome of correct and accurate entity extraction. This technique of so-called *prompt engineering* describes the process of creating and adapting input prompts for a LLM in order to elicit a desired result which is of sufficient quality [67]. A simple prompt for extracting named entities from any input text might look like this:

Out of the following text, extract all named entities that occur: <TEXT>.

This prompt is then supplied with the corresponding text to be analyzed. When supplied

with the example description of table 7, the resulting entities are *Docker* and *Kubernetes*, both of which represent technologies. This poses the problem that the entities are extracted, but no mapping to the defined custom entity types declared in section 4.3.1 can be inferred. Instead, GPT-4 extracts any entity deemed appropriate without ensuring consistency in the result. To circumvent this issue, the prompt needs to be extended to also include the possible entity types that should be extracted. The inclusion of these entities results in the following revised prompt:

Out of the following text, extract named entities for programming language, framework, technology, skill, methodology, and product: <TEXT>.

The resulting data of this prompt is displayed in table 8, showing that this result is now classifying the entities accordingly in the defined categories while still providing a verbose output which cannot be processed automatically because it includes a lot of additional words and, therefore, information that is not needed for further analysis. Additionally, short answers are preferred because no sentences should occur as values for the entity types to also improve automated further steps.

Entity Type	**Occurrences**
Programming Language	None
Framework	None mentioned directly, but Kubernetes could be considered in a wider sense as a framework for orchestrating containerized applications.
Technology	Docker, Kubernetes
Skill	None
Methodology	None mentioned directly. However, using Docker and Kubernetes implies a methodological approach to development, deployment, and scaling of applications, although not named here.
Product	Docker, Kubernetes (Both are technologies but also considered as products offered by their respective companies).

Table 8: Result of the refined prompt.

To alleviate these issues, the prompt is again extended to adhere to a JSON format when outputting data and to keep to simple and short values when obtaining the named entities from the input text:

[...] The skills to be extracted should be either soft-skills or skills related to the mentioned technologies, try to find generalized terms and use only single words for the skills.

To also find skills that are implicitly stated, the LLM is leveraged by prompting for extracted skills that relate to mentioned soft skills or technologies whilst trying to generalize as far as possible. The result of this final revision of the prompt is displayed in listing 2.

This shows a clear representation of the required NER, providing thorough extraction of encountered entities in their according category. The result also includes a selection

```
{
  "technologies": [
    "Docker",
    "Kubernetes"
  ],
  "skills": [
    {
      "name": "Containerization",
      "description": "The use and management of containers
         to deploy applications"
    },
    {
      "name": "Orchestration",
      "description": "Managing containers across multiple
         environments with tools like Kubernetes"
    }
  ]
}
```

Listing 2: Entity extraction result data.

of skills that were inferred from the input text, which can be used as additional meta information when working with the data later on.

From the above examples and explanations, it is clearly evident that the usage of a LLM outperforms the custom-built extractor as well as the pre-trained model due to its more robust extraction result and inferrence of skills from the contextual information of the text. It may occur, however, that a sufficiently trained custom extractor achieves similar performance for entity extraction at the expense of the flexibility and adaptability that the LLM provides, further highlighting the superiority of the LLM approach which does not require manual training or adaptation to changed data. Additionally, the approach does not suffer from the disambiguation problem when entities may be assigned to more than one class because it also considers contextual information included in the training description. The usage of this approach also ensures easy adaptation to changing entities because these only need to be changed in the prompt itself without the need for retraining or reconfiguration of any underlying system. This ensures future-proofing of the system in case the included entities need to be changed or the whole system is adapted to a different use case. The only downside of the LLM approach lies in the fact that it does rely on a fee-based business model, therefore requiring financial investment when using it as compared to a custom trained model, which can be implemented for free. For a custom-trained model, costs may occur for training and provisioning of the system, which

then also increases the total cost basis. Subsequently, the prompt-based approach is used as the method of choice for extracting named entities from the training course descriptions despite the fees because it proves to yield superior results in relation to the required efforts.

4.4 Ontology Extraction

As the extracted entities and derived classes itself provide only meta information about the trainings, it is necessary to infuse these with an ontology to enable proper consolidation into a knowledge graph. A custom ontology is used in conjunction with pre-defined ones likes OWL, RDF, and RDFS targeted creation of the graph while also avoiding a re-definition of commonly used entities in the graph. The creation of this custom ontology provides the advantage that a more flexible and compact schema can be employed, fostering an easier and targeted creation of the corresponding knowledge graph. To create this custom ontology, the data and target structure needs to be studied. As the main use of the knowledge graph is the combination of training courses with their respective entities and topics, corresponding predicates can be derived. This is reflected in an *includes* predicate to allow for the connection between training and entities. Additionally, a *has_topic* predicate reflects the connection between the training and its corresponding extracted class, providing a more refined and domain-specific predicate than just relying on pre-defined ontology terms that might be fitting but represent a trade-off in their specificity.

To supplement these custom predicates, standard RDF predicates are employed for the *type* of an entity, allowing for the linking between the entity, such as "Python" and its corresponding entity class, in this case "programming language". This reliance on pre-defined ontologies also allows for easier future integration of external data sources, enabling integration of the knowledge graph with openly available linked data.

4.5 Knowledge Graph Generation

To generate the initial knowledge graph, the existing training descriptions of data set D_1, which was defined in section 4.1.1, are used to populate it. This data is then later enhanced using additional data extracted from the training systems, but for the initial proof-of-concept only the data set D_1 is used. The general, theoretical process of obtaining such a graph is described in section 4 and the corresponding steps that are required for automated knowledge graph creation for this project are described in sequential order in the following.

It is to be noted that the classification and entity extraction may be parallelized because both only rely on the textual input and have no interdependencies, thus allowing for parallel execution.

Classification

Initially, the courses are assigned to a class using the SVM model that was evaluated and trained in section 4.2.3. The class with the highest confidence is chosen as the target class, which is required because the model will always output confidence values for all of the included classes as described below. This results in a distribution of courses across classes as depicted in figure 9.

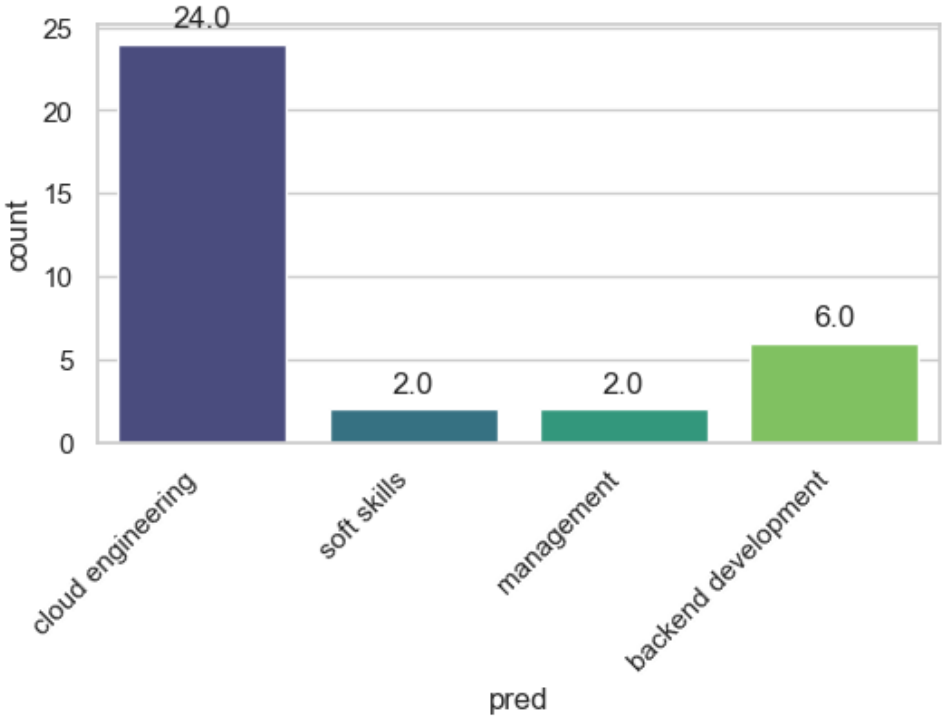


Figure 9: Distribution of prediction results.

The confidence and standard deviation for the classification results on D_1 are shown in table 9. The average confidence scores are not as high as one might expect, which relates to the nature of the multi-class classification that is carried out. As the SVM classifier predicts the probabilities for each of the available classes and then chooses the one with the highest probability as the final assigned class, the confidence may not reach very high scores due to this distribution of confidence values across a total of ten possible classes.

Class	**Average Confidence**	**Standard Deviation**
Cloud Engineering	63%	9%
Soft Skills	43%	4%
Management	46%	5%
Backend Development	57%	2%

Table 9: Result of D_1 classification.

The classes are subsequently used as the main topic the training belongs to, making each of them one node in the knowledge graph together with the trainings itself, which form another set of nodes for graph creation.

Entity Extraction

In order to not only rely on topical areas and trainings themselves, it is also necessary to incorporate the entities that exist within each description. To achieve this, the entity extractor based on GPT-4 is applied on all training descriptions, resulting in a set of entities that appear in one or more descriptions each as well as the mapping of which entities are included in each training description. This results in a one-to-many cardinality describing the entity class-training relationship.

To solve the issue of ambiguity in the extracted entities, the set of entities for each defined category is submitted to GPT-4 with the following prompt:

From the following list, subsumize terms that are the same under the most general/well-known term and return it as a json structure.

This yields a key-value structure having the most well-known term for ambiguities as the key and the remaining terms as values. When no ambiguity is found, the values are empty for the respective term and only a key containing the input value is returned. These synonyms are subsequently linked together in the knowledge graph using an *is* predicate.

An overview of the relations between the extracted entities, their classes, and the classes that are assigned to the training descriptions is displayed in figure 10.

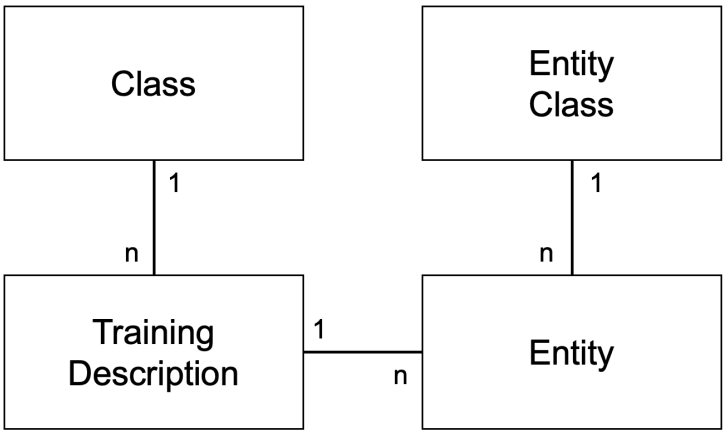


Figure 10: Relations in the extracted data.

Graph Creation

In order to create a knowledge graph based on the extracted information, it is required to generate triples adhering to the schema of *subject, predicate, and object* that represent the knowledge as explained in section 3.1. These triples are created for each entry of the extracted information, starting with the predicted classes of the training descriptions. For

each description, a corresponding class to which the description belongs is extracted using the text classification model developed in section 4.2.3. Because the relationship between training and extracted class is one to one, a set of triples in the format

(<COURSE>, has_topic, <CLASS>)

is created. Similarly, triples are created for the relations between extracted entities and the training descriptions. These triples have the pattern

(<COURSE>, includes, <EXTRACTED_ENTITY>)

and represent the connections between e.g., the used technologies or specific tools that are covered by the corresponding training. Another set of derived triples represents the relationships between the extracted entities that are included in the course descriptions and their recognized class (e.g., *Programming Language*), resulting in a third set of triples in the pattern of

(<EXTRACTED_ENTITY>, is_of_type, <ENTITY_CLASS>),

which then links the entities to their corresponding classes.

For the synonyms that were defined using the entity disambiguation, triples of the following form are created:

(<EXTRACTED_ENTITY_SYNONYM>, is, <ENTITY_COMMON_NAME>)

An overview of the used entities for these two steps can be found in appendix A.3. These triples are created and serialized accordingly, resulting in an RDF graph that can be represented in Turtle (TTL) syntax as demonstrated in listing 3.

```
ex:Developing_Java_Apps_with_Docker a ex:TRAINING ;
   rdfs:label <Developing_Java_Apps_with_Docker>;
   ex:CONTAINS ex:building_java_applications_with_docker ,
        ex:configure_properties_and_variables_for_applications ,
        ex:develop_java_applications_using_docker ,
   ex:HAS_CLASS ex:backend_development ;
   ex:HAS_ID <65e60da3ad22dd34dca6f793>.
```

Listing 3: Knowledge graph excerpt.

This excerpt indicates that the entity `ns1:Developing_Java_Apps_with_Docker` is of type `ns1:TRAINING` and has a corresponding label. The entity also contains a variety of

other entities, such as skills and programming languages. The entity is further classified as being of class backend development and has a unique identifier connected to it.

The created triples can then be easily stored and retrieved for querying within any triple store or, using a transformation into an appropriate format, in any graph database.

5 System Development

As a basis for providing knowledge about the data at hand has been developed, it is crucial to subsequently embed this semantic layer in an accessible and easy-to-use recommendation system. Various approaches for developing such a system are discussed in section 3.4, which are now to be compared and rated for their respective usefulness. To enable a targeted comparison of collaborative filtering, content-based methods, and hybrid approaches, these are to be compared accordingly. Finally, the most suitable approach needs to be implemented and correspondingly evaluated.

5.1 Requirements Elicitation

Requirements elicitation is needed to properly develop a system that is suitable for serving its intended purpose, but it is mostly difficult to obtain correct and complete requirements [68]. To get a holistic understanding of requirements, it is not sufficient to simply ask prospective users which features and functionalities they deem necessary, but a more comprehensive and systematic approach is needed [68]. Gaining user input and obtaining a set of requirements that is suitable for the system is also essential to avoid a failed or non-fitting system implementation [69]. A requirement in the context of software product development is characterized as a capability that is used to solve a specific problem or gain a specific outcome [70]. A further distinction in the case of software product development is made into stakeholder needs as the topmost priority, which are supported by the system's features and the corresponding requirements for each feature [69], resulting in a pyramid-like structure for these three key aspects. Features should be described in simple and understandable language to enable clear communication and subsequent validation of those features with the system's stakeholders [69]. Stakeholders in this project consist of employees who wish to receive training recommendations in different roles that are to be determined, where their needs also provide the basis for subsequent feature definitions and thus corresponding requirements for the implementation, following a top-down approach to elicit requirements starting with the stakeholder needs that the system aims to solve.

These requirements are distinguished into functional and non-functional requirements and need to be raised in close conjunction with stakeholders to avoid aforementioned failures due to non-involvement of users. Non-functional requirements further describe the quality attributes of a system under development, including criteria, such as performance, reliability, or usability that are crucial for an implementation's success [71], describing how well a system performs. Functional requirements describe the essential characteristics and features the system should fulfill so that it can fulfill its desired purpose [72], describing what a system does in terms of functionality and features. In order to obtain accurate

requirements, a combination of questionnaires for stakeholders and an analysis of user groups is used. The combined result of these analyses is then used to extract and define meaningful requirements, which form the basis of the system development and its later validation.

5.1.1 Structured Survey

To best understand individual user needs and requirements, it is necessary to gather these information in a structured way from a wide variety of stakeholders. To achieve such broad information retrieval, structured questionnaires provide an ideal way of obtaining data from all involved stakeholders because they allow for a larger sample size and automated evaluation when compared to traditional interviews. In the context of developing a training recommendation system, it is crucial to involve a large sample size of potential users in the system's development; thus interviews are not able to provide enough variety in their respondents due to their high manual effort. The survey is divided into several parts, which are listed and explained in detail in table 10. These questions aim to provide a holistic view of the current training process as well as desired system features and important aspects of respondents, enabling the creation of a usable and helpful solution.

Section	**Purpose**	**Collected Information**
Personal Information	Gain an understanding of the recipient, segment results by characteristics, such as work experience and role	Role, years of work experience, department
Current Process	Understand challenges and current process of training assignment and identification	Identification of training needs, challenges with the process
Desired Features	Collect a list of features users need	Features based on pre-defined criteria
Content Preferences	Evaluate usefulness of different trainings, collect preferred topics	Favourite training delivery methods, topics based on defined classes
System Usability	Determine necessary functionalities of a recommendation system	Importance of ease of use, novelty, custom parameters, individual goal setting, course evaluation
User Involvement	Understand the user's desired level of interaction	Ability to influence recommendations, desired level of support for using the system, showstoppers
Comments	Additional free-text comments	Optional comments and wishes

Table 10: Survey contents and structure.

This survey is subsequently sent to potential participants for information gathering, resulting in a total of N=47 responses.

The main challenges that respondents currently face when trying to find suitable trainings are aggregated in figure 11. These responses clearly indicate that one major challenge for employees searching for training courses currently is a lack of awareness of trainings,

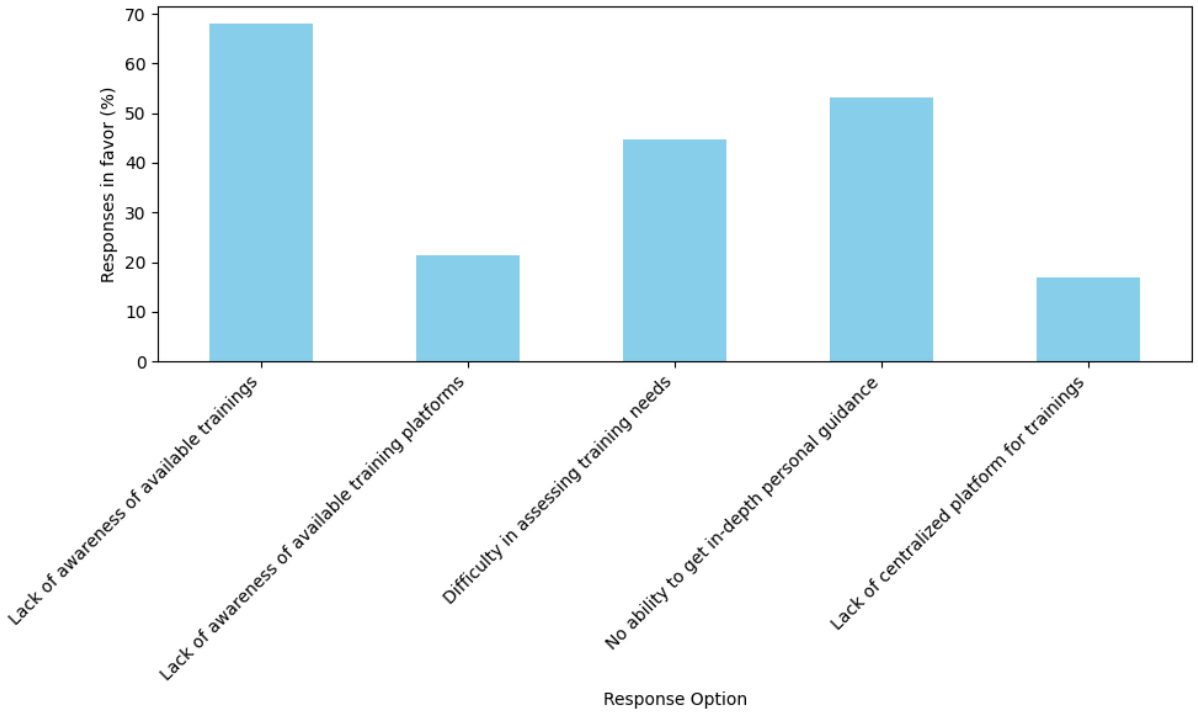


Figure 11: Main current challenges for respondents.

followed by the absence of personal guidance, and a difficulty in assessing the training needs individually. Most respondents do not have any difficulty regarding training platforms, showing that the main issues are related to awareness, assessment of needs, and proper guidance and that there exist less issues concerning the usability or accessibility of platforms.

Regarding the desired features, the responses show clear preferences for some of the proposed system functions. For 87% of respondents, ease of use is either important or very important, highlighting the necessity to develop the recommendation system in an accessible and easy-to-use way to gain and sustain users, while also providing suitable recommendations. This is further supported by eighty percent of the respondents who find it important or very important to get interesting and novel recommendations, clearly indicating that it is insufficient to simply provide basic recommendations based on previously attended courses but to really emphasize the ability to propose unexpected and novel training contents to prospective users. For more than eighty percent, it is also important or very important to be able to include custom parameters (e.g., known or desired skills), indicating a need for individualized and customized adaptation of the recommendations to be made by defining these custom parameters individually, providing users with sufficient abilities to accomplish this. For 55% of respondents, the definition of individual learning goals is either neutral or not so important, showing that this feature is not necessarily of great use for a majority of potential users. Rating and evaluation of courses are deemed

important or very important by 51% of respondents, making it also a feature to be considered. 68% of respondents wish to be able to also contribute recommendations to the system, making it further a necessity to consider the integration of such a feature in the developed system.

Looking at the training contents that should be addressed, a clear dominance of ML can be seen, followed by backend development and cloud engineering. As most respondents are working in a technically oriented role, this dominance of practical and foundational software engineering topics is to be expected, showing a clear need for precise and tailored content related to these topics. The complete result of the desired training topics, as extracted from the survey responses, is displayed in figure 12. It is also evident that no content class is completely irrelevant, proofing that these classes are suitable and covering all relevant topics. No answer has provided an additional comment because there is no "other" option provided for this question.

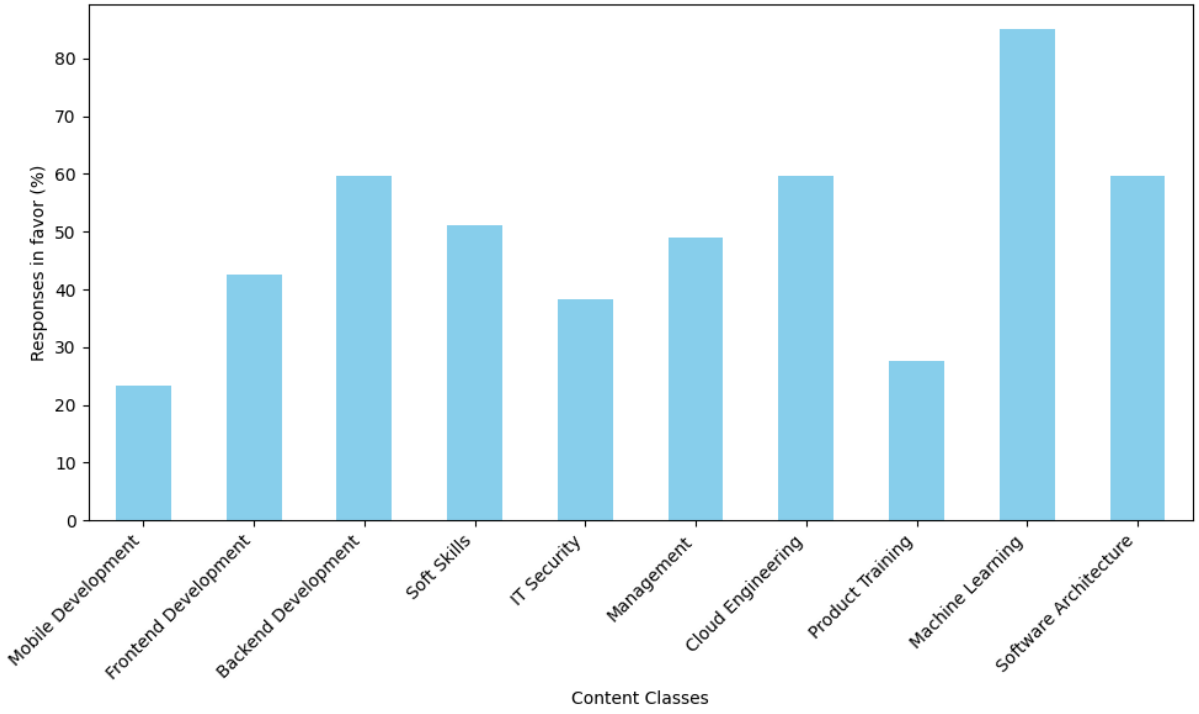


Figure 12: Desired training topics extracted from the survey.

For potential showstoppers that might prevent users from effectively using the recommendation system, various free-text answers were provided by respondents. These mainly concern accessibility, complexity, and usability but also low-quality recommendations, highlighting a need for a well-structured user interface and again the desire for high-quality and targeted creation of recommendations. The detailed responses for the survey can be found in appendix A.2.

5.1.2 Target Groups

To be able to develop a system that is human-centered and fulfills the needs of its potential users, it is helpful to model and define the target user groups. While [73] propose to employ personas for creating a software system that is beneficial for the users, these will not be created, but instead a focus on the target user groups is preferred. The groups who represent potential users of the recommendation system can be characterized by their technical proficiency, work experience, and seniority level of their role. These required attributes can be partially derived from the conducted survey because the professional work experience, functional area, and position are included in the set of questions. As a first characteristic, a differentiation based on the overall proficiency can be made. This includes users who have little to no previous work experience, providing a more detailed need for generalized training recommendations, whereas for senior roles, a more granular and specific recommendation approach needs to be done to target skill gaps effectively without repeating what is already known. Additionally, user groups differ based on their role's focus. As various roles, such as human resources, development, consulting, or marketing, are included in the survey respondents and the overall organization, an intuitively usable system that combines relevant trainings for all of these groups needs to be implemented. Furthermore, depending on the responsibilities of the role, different training focuses need to be included. A senior manager, for example, may have a more extensive need for soft skill focused trainings, such as conflict resolution or employee conversations, whereas an individual contributor, such as a developer, may focus more on gaining hard skills like programming techniques. Nevertheless, there exists an overlap between these focus areas, so this also provides a need for a wide and diverse variety of training contents.

5.1.3 Synthesis of Requirements

After collecting information regarding current issues and prospective functionalities using the survey and further detailing them with the help of the analysis of potential user groups, corresponding requirements are derived, which are subsequently leveraged to facilitate the implementation of the system. To achieve a structured derivation of requirements, the distinction into stakeholder needs, features, and requirements introduced in section 5.1 is used.

Stakeholder needs can be extracted using the survey responses presented in section 5.1.1. The key need, which stakeholders require, is a platform to solve the lack of awareness of available trainings, while providing in-depth personal guidance in conjunction with an assessment of training needs. Additionally, the system needs to support a wide variety of contents, as defined in section 4.2.1 to best support a broad range of potential user backgrounds. The system also needs to be easy to use, should provide novel and interesting

recommendations and needs to enable users to integrate custom parameters in order to fine-tune the recommendations that are made. It should also provide an ability to rate the courses accordingly. The combined stakeholder needs are displayed correspondingly in table 11, resulting in a total of five major needs covering all of the above mentioned aspects.

Identifier	**Stakeholder Need (SN)**
SN1	Provide accessible, easy-to-use personal learning guidance with interesting, varied, and novel recommendations
SN2	Increase awareness of available trainings while also incorporating a multitude of different training formats
SN3	Support assessment of training needs
SN4	Provide a subsequent option for training ratings
SN5	Enable integration of custom parameters and skills to be learned to allow fine-tuning and adaptation of recommendations

Table 11: Stakeholder needs.

System features are defined to further detail each of the needs, providing a common understanding of the desired system capabilities. These need to be defined to closely model each of the stakeholder needs, while also detailing a way in which the corresponding need can be fulfilled, without introducing technical specifications yet. These features are shown in table 12 and are subsequently used to derive functional and non-functional requirements, which are used as the basis for the system implementation.

Stakeholder Need	**Feature Description**
SN1	Easy and understandable user interface, intelligent recommendations based on user history and role, incorporate adjacent topics to provide more than just obvious recommendations
SN2	Incorporate multiple training platforms, provide recommendations from a comprehensive training catalogue
SN3	Include guidance on trending topics, ability to define an individual goal by means of a skill assessment tool
SN4	Include a rating function to evaluate trainings after taking them to provide feedback on effectiveness and relevance
SN5	Provide a selection of custom skills, labels, and goals to select from. Enable users to see how these influence training recommendations

Table 12: Derived system features.

Functional requirements are derived from the system features in order to describe what the system does in terms of function and what value it delivers to its users. These functional requirements are introduced and explained in the following:

- **F1 Recommendation Engine:** The system needs to be able to recommend relevant trainings to its users by combining preferences, historical data, and user-defined interests. These recommendations should also take into account adjacent fields of interest, instead of merely providing the most obvious trainings to keep users engaged and learning. The recommendations should also focus on incorporating a variety of different aspects and provide not only trainings, but also skills to be obtained to foster employee education across different areas.

- **F2 Inclusion of Multiple Training Sources:** The system should be able to work with any training provider, so that it can adapt to changes of the underlying training platforms without the need for adaptation or reconfiguration of the system's core characteristics. The system should work with any training platform that provides textual descriptions for their courses. This is of relevance because not only technical trainings need to be included, but also a wide variety of other training content to also accommodate for different seniority and roles in the user groups. These other trainings may reside in a different training system, further highlighting the need for this adaptability.

- **F3 Definition of Custom Parameters:** Users need to be able to select custom parameters, such as desired skills to be learned or topics of interest, from a pool of potential pre-configured parameters, so that the system can incorporate these user interests when generating a recommendation. These pre-configured parameters should be integrated into the system automatically without the need for user intervention or any manual effort. This also stems from the defined variety of user groups so that different user groups can provide individual focuses for their personalized training needs.

- **F4 Rating Function:** The system should provide a functionality for users to rate courses after completing them to inform future course participants whether the course meets the expectations or not. There needs to be a rating function to evaluate both the effectiveness of the training and the relevance of the content for the user.

- **F5 Recommendation Reasoning:** The system should provide reasons for the recommendations that are made so that the user can understand the rationale behind the recommendations. These should include the steps taken to achieve the provided recommendation, so that it is clear how any choice is made.

- **F6 User Profiles:** Users should be able to create a profile, which then stores all of their training data and preferences accordingly. This profile should be created with the help of a setup assistant that guides the users in an easy-to-understand way through their initial registration.

- **F7 User Feedback:** Users should be able to provide their own recommendations that they deem fitting when accessing the system so that human input can also be taken into account when generating recommendations.

Non-functional requirements further detail how the system reaches its desired goals. These are commonly more difficult to validate because they do not rely on implementable functionalities, but rather provide information about factors that describe the overall usability and, consequently, the associated user experience of a system:

- **NF1 Usability:** The system needs to be developed in such a way that it can be used intuitively by users a multitude of backgrounds. Ease of use needs to be ensured to attract and retain users while fostering long-term learning. This requirement stems from the focus of responses on a usable system because many respondents also included subpar usability as a major showstopper when interacting with such a system.
- **NF2 Performance:** The generation of recommendations should be completed in a reasonable time so that users are not discouraged from using the system due to inconsistent or undesirable latency, catering to the need of an easy-to-use and accessible system. Exhaustive latencies will also lead to a decrease in user satisfaction and push users away from continuing the interaction with the system, making this requirement a vital part in retaining users.
- **NF3 Reliability:** Recommendations that are made should be accurate and relevant based on the stored user preferences and historical data as well as the output of the recommendation engine itself. These should be reliable based on factual data to create interesting and varied outputs, which keep users engaged in using the system and, therefore, support continued education.

These requirements aim to provide a complete coverage of potential system features. The derivation of these from the user needs that are identified using the questionnaire ensures the development of a system that solves the user's challenges, so that the following implementation is of value for the users.

5.2 Selection of Recommendation Approach

To determine the best possible fit of the recommendation system types introduced in section 3.4, a thorough evaluation based on the defined requirements is carried out. Therefore, these approaches need to be compared on various criteria.

When comparing collaborative filtering with content-based recommendation systems, the following aspects need to be considered. A collaborative filtering approach requires user ratings of the items to be recommended, which are represented by the trainings that are included in the recommendation system. As obtaining such ratings in a reliable and sufficiently large amount proves to be difficult, content-based recommendation outperforms collaborative filtering in this regard due to the fact that content-based methods do not rely on user ratings. These ratings are also not easily obtainable and a reliance on only ratings for providing recommendations might lead to the problem of users not rating enough courses to derive meaningful recommendations. Content-based filtering handles novel or niche items better because there is no reliance on user ratings or previous interactions of any kind, which is usually the case with newly introduced systems. Recommendations are also provided in a more transparent and understandable way because the recommendation is based on a specific set of features and their similarity, which can be clearly understood by evaluating the executed calculations. Content-based methods also allow for more control over the individual preferences and variables, allowing for more control on the user's side by introducing profile preferences and certain skills to be included, which can simply be entered as plain text into such a system. On the downside, the quality of a recommendation is limited by the availability and quality of the relevant item information, while also being prone to overspecialization or a reduction of diverse recommendations, which may result in a loss of quality of the recommendations and can, therefore, decrease the overall user satisfaction.

To counter any issue with the two classical approaches, such as the cold start problems or sparse data in general, a hybrid method is further studied. This aims to combine a common recommendation method with some other form of infusing knowledge or meta information into the recommendations. Therefore, an approach combining traditional recommendation approaches with a knowledge graph as a semantic basis provides three main advantages [3]:

- **Good data representation:** Classical recommendation systems do not rely on a semantic data basis for providing their recommendations, thus making them prone to data sparsity issues resulting in less accurate recommendations.
- **Mitigation of the cold start problem:** Without initial data, recommendations cannot be made accurately, therefore a knowledge base can alleviate this issue by providing a defined and well-structured foundation.

- **Explainable and understandable results:** The reasoning for each recommendation can be traced through the underlying knowledge graph, allowing for better insights into and understanding of the recommendation process.

The combination of a traditional recommendation system with a knowledge graph not only improves the overall items that are recommended to the user, but also helps to provide a more explainable and, therefore, understandable result [45], also improving the acceptance of recommendations on the user's side by providing reasons for recommendations that are made. An additional benefit of the combination of a recommendation system with linked data in the form of a knowledge graph is the solving of the apparent cold-start problem when initiating the system for new users as shown by [46]. Consequently, a higher user satisfaction is reached by providing clear and understandable results while also focusing on delivering new and undiscovered recommendations.

As such a combined approach is most appropriate in the given setting with sparse initial information and no existing user-item-ratings, a decision for a recommendation approach, which is either collaborative filtering or content-based, needs to be made and is then combined with the created knowledge graph.

Due to the combined approach with the knowledge-based system implemented in the knowledge graph, the availability of item information is provided using that semantic layer, solving the problem that content-based recommendation systems mostly face. Additionally, the proposed hybrid approach also enables the recommendation of a more diverse set of items to prospective users by ensuring topical variety using the underlying semantics.

Consequently, a combination of a content-based recommendation system with the knowledge provided by the knowledge graph yields the best results for recommending trainings in a sparse dataset with no initial ratings or user-item interactions available, providing an advantage over collaborative filtering because this approach relies on existing user-item interactions to function reliably. It also provides the benefit of not having to have any user-item interactions, resulting in less strain on the users as they do not need to provide any information in the form of course ratings.

5.3 Implementation

To implement a robust, scalable, and resilient system, it is required to encapsulate and loosely couple the various components as efficiently as possible. To achieve such desired encapsulation and loose coupling, each part of the system is developed as a separate independent service instance, fostering a microservice-oriented architecture which allows for easy scaling and updating of individual services without the need for scaling the complete system when there are changes in the load for individual components. This additionally enables a more lightweight system than a monolithic approach would result in because each service can be implemented using the best-fitting technology stack and, therefore, dependencies can be kept at a minimum level. The overall architecture is depicted on a high level in figure 13, detailed explanations of the individual services are provided in the subsequent sections. Service interdependencies are kept to a minimum

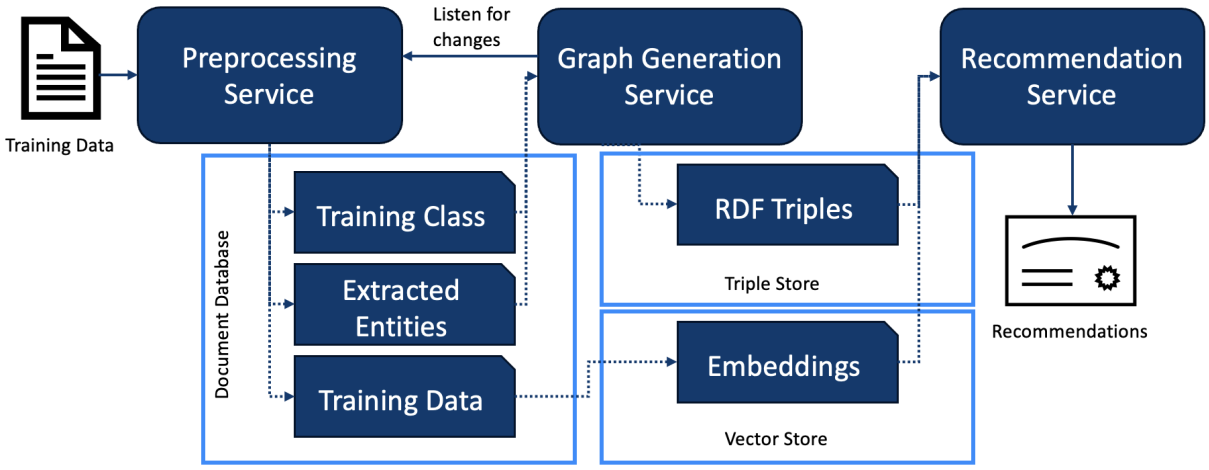


Figure 13: System architecture.

as to not unnecessarily increase the communication overhead, resulting in an improved overall maintainability and less need for well-defined interfaces between the services. The implementation itself is done in Python because this language of choice integrates well with the text classifier that is also created using Python and a wide variety of possibilities exists to easily integrate backing services, such as vector databases or document stores, into the application. Additionally, the services are developed in a way that they can easily be deployed using Docker images to prepare for a live system roll out, which would then need to be as portable as possible. This means a strong separation of configuration information, such as credentials for backing services, and application logic, as well as a stateless implementation of the services to enable easy containerization.

5.3.1 Data Preprocessing Service

To achieve a centralized and unified data preprocessing that is compatible with any textual training data representation, a service is developed which combines the entity extraction, course classification, and generation of embeddings that are subsequently stored in a vector database. This service relies on the SVM model developed in section 4.2.3 for classification and also incorporates the approach for entity extraction proposed in section 4.3.2. When receiving a new training, the attributes `title` and `description` are extracted for analysis and passed to the two models accordingly. To conceptualize the sequence of actions performed when adding new courses via this service, a sequence diagram, as shown in figure 14 is created.

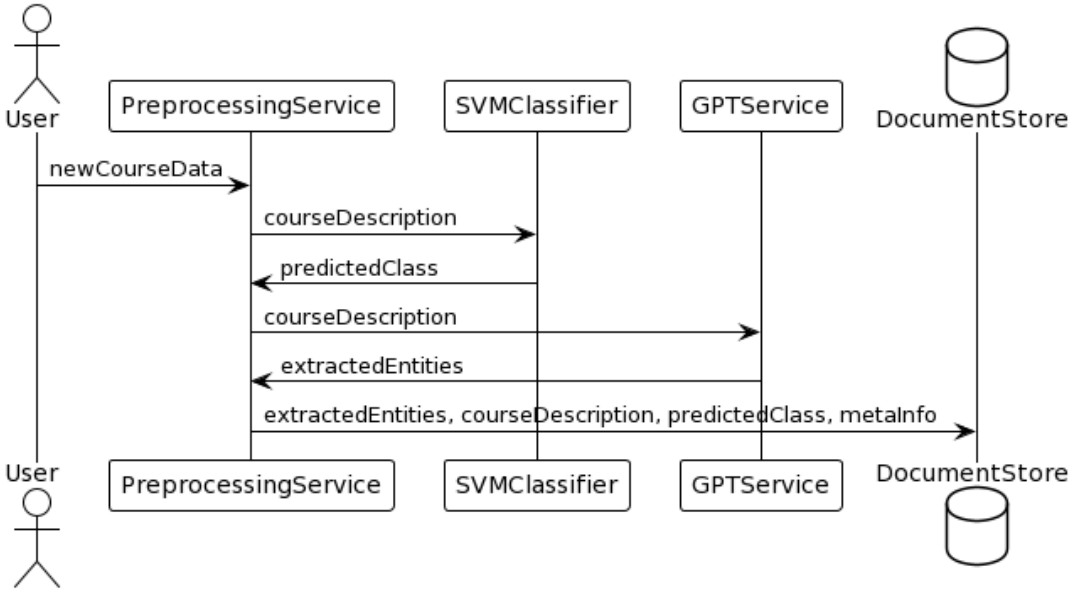


Figure 14: Sequence diagram of preprocessing service.

The results of any prediction made are stored into a document-oriented database in JavaScript Object Notation (JSON) format for easy retrieval and flexible schema adaptation because a document database does not rely on any predefined schema, but instead supports flexible, semi-structured data. This also adds the benefit of easy schema evolution when more metadata needs to be stored alongside the document. An exemplary preprocessed document, which results from this service is depicted in appendix A.4. Added metadata includes information, such as the timestamp when the document was inserted, to enable a retrieval of documents that were added recently. Further data includes the predicted class as well as the associated confidence of the prediction. Entities that were extracted are stored accordingly in arrays if there are any, resulting in empty arrays in case there are no matches for a given entity. The original course's description and title are also stored because they need to be passed on to a vector database, which enables storing of the corresponding embeddings to enable content-based recommendations.

Due to the service's ability to analyze any textual data for training courses, this implementation aims to complement the functional requirement **F2** by not making any assumptions on the data source, but instead providing a universal interface to adapt to any potential provider.

5.3.2 Graph Generation Service

To facilitate automated generation and updating of the knowledge graph, it is required to adapt its contents based on the data that is extracted using the data preprocessing service. To facilitate such automated creation, an automated reaction to any new data that is analyzed by the preprocessing service is required. This reaction is triggered via an event, which is fired when new data is stored, leading to the automated creation of relevant triples to be incorporated into the knowledge graph, which then fosters continuous and automated adaption to changes in the knowledge base.

These triples are created in the schema that is defined in section 4.5 and are subsequently stored in a collection in the document store, which also holds the analyzed training data. From there, these triples can easily be retrieved to build a knowledge graph either in-memory or using any triple store that is available. This provides the advantage of easy adaptation of the knowledge base to any implementation, further increasing the loose coupling between the services while ensuring conformity with a common standard because the triples are serialized and stored in TTL syntax.

5.3.3 User Profile Service

To fulfill functional requirements **F3 Definition of Custom Parameters** and **F6 User Profiles**, a service to manage and store user profiles is implemented. These should store information as a set of custom defined preferences and a set of previously attended courses for the user. These pre-defined preferences are an individual subset of the entities and topics that are extracted from the available course data using the data preprocessing service created in section 5.3.1, allowing for automated creation of these selection options. Another benefit of using the extracted attributes is that these can then automatically be mapped to nodes in the knowledge graph for providing a seed that serves as an entry point for analysing relevant topics and related trainings, which can be passed to the recommendation engine accordingly. An exemplary user profile in JSON format is depicted in listing 4.

This shows the user's history reflected in an array of course descriptions and titles and the skills and trainings as their corresponding node from the knowledge graph. Furthermore, any custom parameters, such as desired learnings or any skills the user wants to obtain,

```
{
    "user_id": "43127e04-194e-4461-807a-811876d66ed1",
        "history": ["Developing Node.js Apps with Docker"],
        "skill_nodes": [
                "http://example.com#installing_docker"
        ],
    "training_nodes": ["http://example.com#
       Cloud_Computing_The_Big_Picture],
    "custom_parameters": ["Docker", "Presentation
       Competencies"]
}
```

Listing 4: Exemplary user profile.

are incldued in an array as well. Additionally, the user's identifier is included to link the recommendation to the user.

These user profiles are stored accordingly in a collection in the document database for easy retrieval and integration into the recommendation service, which also provides an easy means of updating and modifying profile-related data.

5.3.4 Recommendation Engine

To develop a robust and targeted recommendation engine fulfilling functional requirement **F1**, a hybrid approach, as outlined in section 5.2, is developed. This method requires the combination of a traditional content-based recommendation system with a knowledge graph to provide additional semantic information. First, a content-based recommendation system is developed using embeddings of the training descriptions that are stored in a vector database. These embeddings can then be used to generate recommendations based on previously viewed courses or a user profile by calculating the similarity between the vector representations of the user profile or user history and the other known course descriptions that are stored in this database in their corresponding vector representation. The overall architecture of this approach is depicted in figure 15.

To improve the generated recommendations, this content-based recommendation system is combined with the knowledge graph developed in section 4.5. The aim of this combination is a reduction in duplicate recommendations and the integration of a wider variety of topics and, therefore, trainings because this contextual information is not apparent when using only vector representations and similarity scores to derive recommendations. Additionally, this integration aims to fulfill the functional requirement **F5** so that recommendations can also be explained and the proper reasoning for each recommendation can be provided to

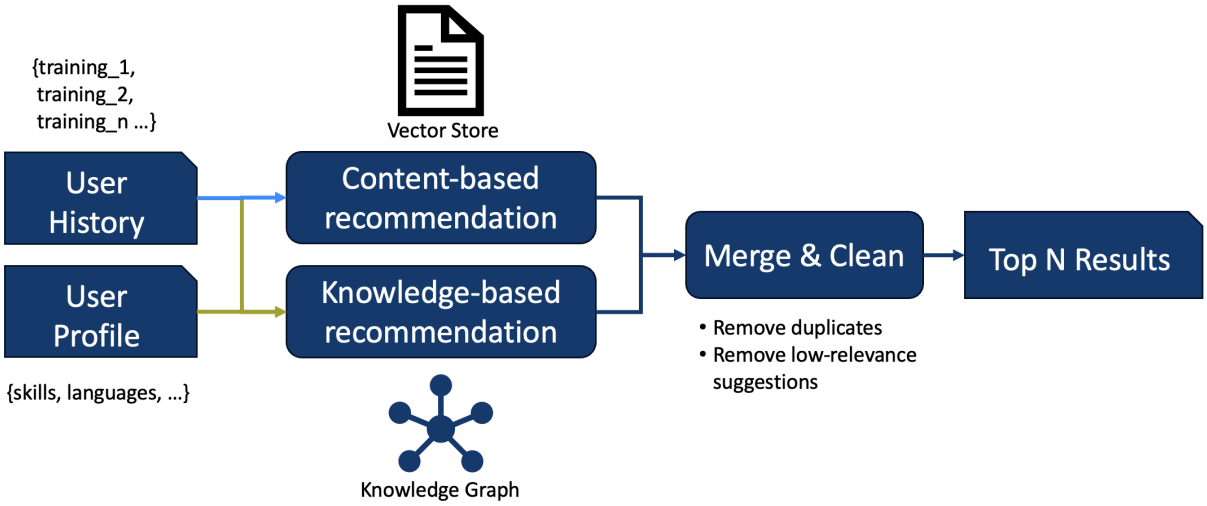


Figure 15: Recommendation engine approach.

the user receiving it.

The overall approach to generating recommendations can be modeled as follows:

- U as the set of users in the system.
- For each user u in U, let C_u be the set of completed trainings for this user which is represented as nodes in the knowledge graph.
- For each user u in U, let S_u be the set of skills the user knows or is interested in, which are also represented by corresponding nodes in the knowledge graph.
- For each user u in U, let P_u be the profile containing interests and desired learning opportunities.
- For each user u in U, let T_u be the set of trainings to be recommended for this user.
- For each user u in U, let K_u be the set of skills to be recommended for this user.

We wish to find T_u and K_u such that:

$$T_u = \{t \mid t \text{ is a training recommendation for user } u\}$$

subject to the following constraint. For each user u in U, the recommended trainings T_u should match the skills S_u and take into account while also excluding the completed trainings C_u:

$$T_u = \{t \mid t \in T, \exists s \in S_u \text{ such that } t \text{ matches } s \text{ and } t \notin C_u\}$$

$$R_u = \{(t, s) \mid (t, s) \in T_u \times S_u\}$$

The corresponding algorithm to find such new recommendations for skills and trainings to be learned can be defined and implemented as follows:

1. Provide C_u, P_u, and S_u, provide n as the amount of trainings to be recommended.
2. Derive a set R_{cb} of recommendations using the content-based recommendation from the vector representations of the trainings using the vectorized representations of C_u, P_u, and S_u, calculating the similarity as vector distances between the input and output.
3. Derive a set R_{kb} of recommendations using the knowledge-based recommendations from the knowledge graph by sequentially inputting the items to the similarity calculation, calculating the similarity as the cosine distance between the nodes in the knowledge graph.
4. Sort the resulting set by similarity and cosine distance to determine the most fitting items, combining the result into the set T_u.
5. Remove elements so that $\{t \mid t \in T, t \notin C_u\}$.

The implementation of this service subsequently applies the approach defined above by translating it into Python, enabling the integration of the algorithm into the service implementation. Here, the knowledge graph is created in-memory to improve performance. For the cosine similarity, it is transformed into a regular directed graph which is then used to calculate the distances accordingly. The request data for generating recommendations is sent to the service in a JSON format as shown in listing 5.

```
{
        "history": ["Developing Node.js Apps with Docker"],
        "num_of_recs_content_based": 3,
        "num_of_recs_knowledge_based": 3,
        "nodes": [
                "http://example.com#installing_docker"
        ]
}
```

Listing 5: Payload for the recommendation engine.

The course history is provided as an array of course titles and optionally descriptions to enable the content-based recommendation to leverage the provided textual data for the vector similarity search. Subsequently, the selected nodes that contain the user's previous knowledge or interests are provided via the nodes array to enable the knowledge-based recommendations to be created.

These values are extracted by the backend service, which then retrieves content-based recommendations using the pre-filled vector database, resulting in a list of recommendations that are sorted by their corresponding vector distance. To determine the final result of this content-based recommendation, the top results, as indicated by the parameter contained in line three, are selected and added to the result set after eliminating any duplicates that are already contained in either the user history or the nodes section of the request data.

For deriving the knowledge-based recommendations, the number of results is indicated by the parameter defined in line four. The underlying calculation retrieves the combined similarities for all nodes in comparison to the input nodes from the "nodes" attribute and orders them descending by similarity. From this resulting list, the first results are then included in the result set, as determined by the corresponding parameter. Finally, any duplicates between these two recommendations are removed and filled with the next candidates from either result, depending on where the duplicate occurred. The overall response of the service then includes the recommendations, their corresponding entity type and the overall similarity scores that lead to them being recommended. This resulting structure is then returned as a JSON response to the caller, enabling easy integration of the service into any application (e.g., the frontend for the users).

5.3.5 Frontend

To develop a streamlined and focused User Interface (UI), it is of great importance to only include necessary and relevant information and avoid cluttering of the application frontend. Because this is the user's single touch point with the recommendation system, this provides an additional necessity to focus on streamlining and simplifying the interface to enable an abstraction of the underlying system complexity, resulting in a minimalist and reduced graphical design of the UI. The implemented UI hereby represents a proof-of-concept, which can be further enhanced to allow for the integration of additional features and a more concise displaying of the required information. The implementation at hand focuses on providing a demonstration of the system's capabilities while also laying the foundation for early testing and obtaining feedback from prospective users.

First, the application for inserting course data into the recommendation engine is implemented. This consists of two input fields, which contain the course title and description, respectively. These are then used as an input for the preprocessing service as defined in section 5.3.1. The corresponding user interface is depicted in figure 16.

Upon submitting the data, it is sent to the preprocessing service for classification and entity extraction. In addition, to output the extracted information, the result is displayed to the user to provide visual feedback. This resulting output is depicted correspondingly in figure 17. It can also be seen that the system stores data upon preprocessing because

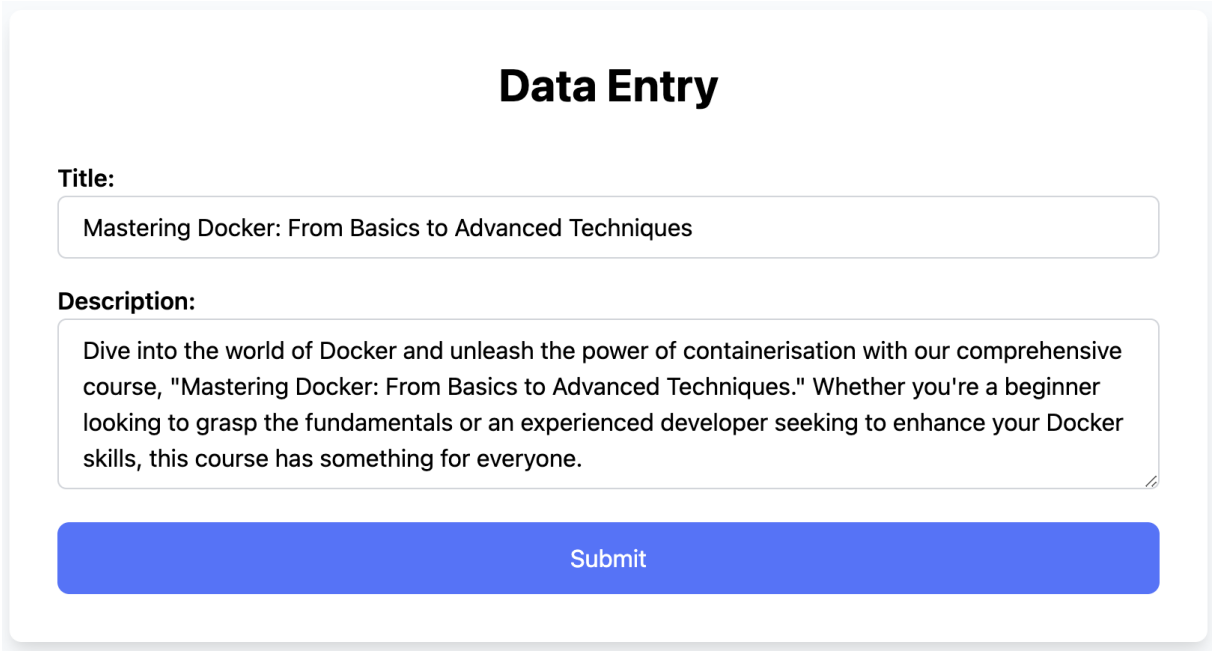


Figure 16: Course data entry.

it also indicates whether the data is retrieved from storage or if it was newly analyzed by the "Cached File" attribute in the response.

Secondly, the user interface for inputting profile data (e.g., the course history and knowledge) is created as depicted in figure 18. This input data is equivalent to the required data for creating recommendations as defined for the recommendation service in section 5.3.4.

This allows the user to input the course history, currently as a string that is comma-separated. This input is what is then used for the content-based recommendation by the recommendation engine. As this is provided as a plain string, it is converted into the required array structure before sending the data to the backend service by splitting the textual input at each comma, producing an array of descriptions. Additionally, the known or desired skills, technologies, and other entities can be selected using a multi-selection field, creating an accessible and understandable input for this data. These entities are fetched automatically based on the available entities of type skill, product, technology, framework, programming language, or methodology, which are available in the knowledge graph. Therefore, a SPARQL query is executed in the backend API that retrieves all of these entities correspondingly for displaying them, removing potential duplicates from the result set before rendering these to the user. As these entities are nodes in the knowledge graph, they need to be converted into a more readable format by removing the namespace prefix of the knowledge graph and replacing the underscores with spaces.

Upon submitting this data to the recommendation system, a list of recommendations is

Figure 17: Training data preprocessing result.

created, which is obtained using the service defined in section 5.3.4 by providing it with the necessary input parameters. The resulting recommendations are presented to the user in the form of a list as depicted in figure 19. These results are tagged according to their entity type, as defined above, and contain supplementary text as descriptions which has been omitted for brevity. This provides the user with recommendations based on their input data, enabling rapid consumption and retrieval of relevant recommendations while also focusing on usability and ease of use.

5.4 System Evaluation

Evaluating a recommendation system can be done in a comprehensive way including both on- and offline evaluation as well as user studies, as discussed in chapter 3.4.4 to assess the quality of the developed system using a set of distinct research methods and

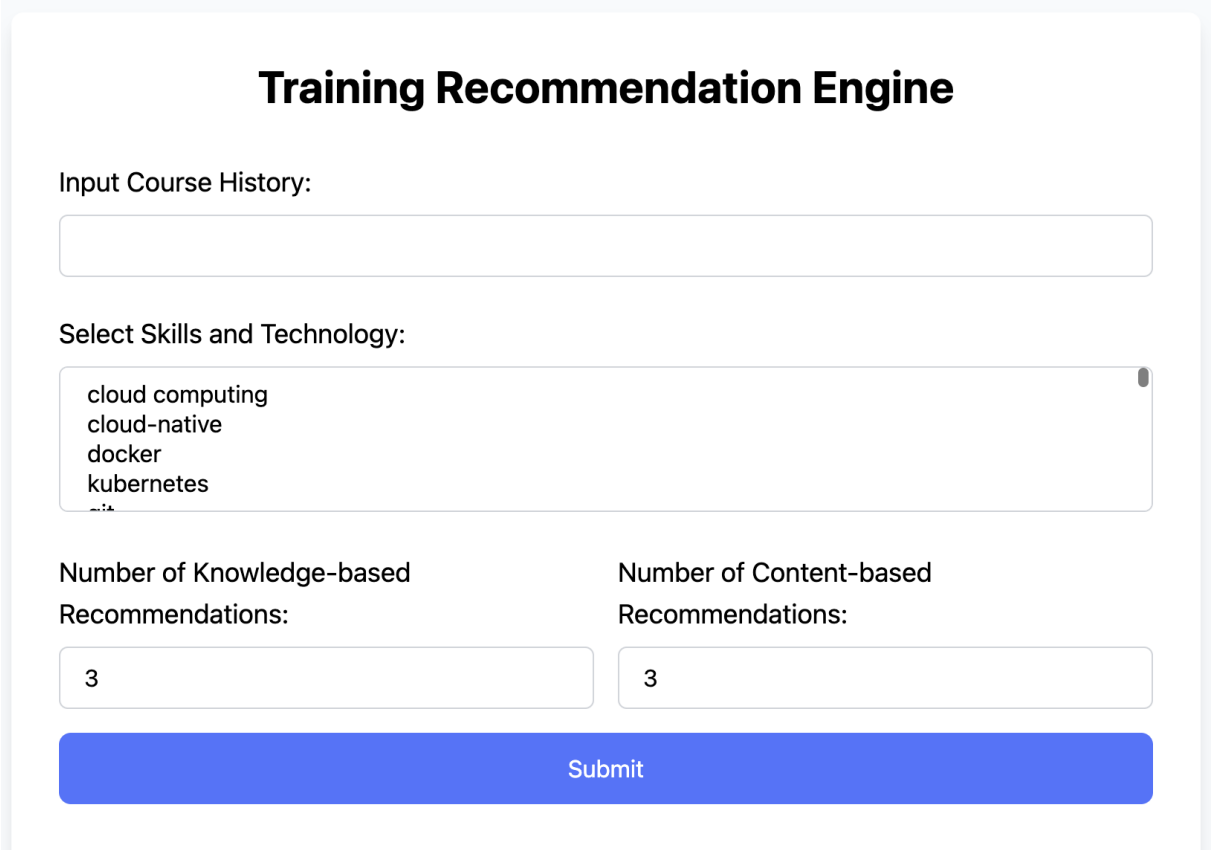


Figure 18: Recommendation creation form.

methodologies [74]. The evaluation is carried out to ensure fitness of the system for the use case of improving employee learning as well as for ensuring continuous improvement of the system to enable adequate adaption to changes in the underlying information and user needs. It is, therefore, necessary to evaluate both the user satisfaction and the executed system implementation according to defined metrics, such as the overall fulfillment of requirements. At the heart of any evaluation lies the overall goal, which then outlines the evaluation's core characteristics. In the project setting at hand, this main goal is defined as *the system's ability to increase employee learning activities by providing suitable and interesting training recommendations to an employee.* To validate this goal with users, the evaluation to be conducted is an online evaluation as described in chapter 3.4.4, focusing on the perceived user satisfaction when actively interacting with the system and discovering insights into how potential users leverage the system in their daily lives. This proves more valuable in comparison to offline evaluation, as the final adoption of the system in a user's daily routine largely depends on the real usability and, therefore, online evaluation can provide the best insights.

As a detailed online evaluation involves a large amount of participants, only a small, adapted version is carried out due to the lack of sufficient observation capabilities and a

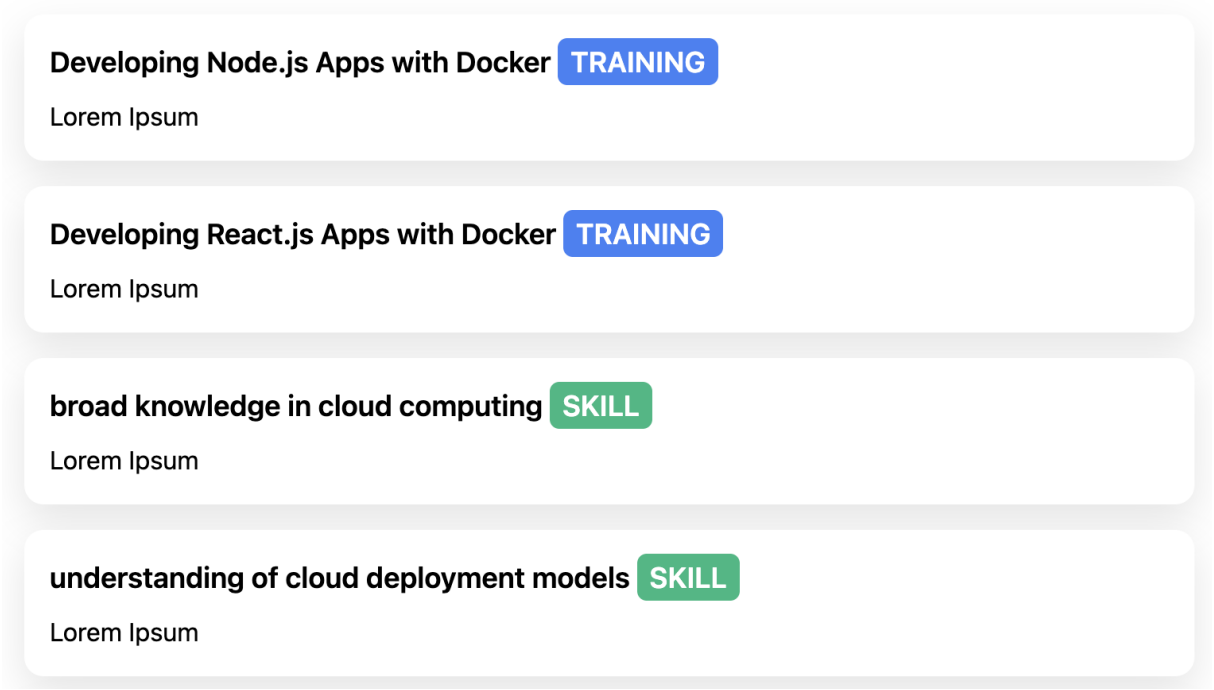


Figure 19: Recommendation user interface.

limited time scope. Further studying of the user behavior will be done by also relying on the herein defined evaluation characteristics to gain a more detailed understanding of user needs and their overall satisfaction with the system at hand in the future. The presented approach for online evaluation is thus scalable independently of the number of participants and can easily be reused to further study the effectiveness and to uncover any potential issues in the system.

To execute a targeted online evaluation, metrics that are to be covered when the user interacts with the system need to be defined. For this evaluation, these will include the Click Through Rate (CTR) [75] and the Attendance After Recommendation (AAR) to capture both the user's clicks on the proposed trainings and the subsequent attendance to that training. CTR is defined as the fraction between the clicked and the total displayed items, providing insight into how relevant the recommendations are for the user [75]. AAR is the fraction of trainings that are also attended to after they are recommended to a user, providing a stronger indicator for the relevance than obtaining only the CTR. These two metrics are then combined with a questionnaire to derive additional user feedback from within the evaluation. In this questionnaire, participating users are asked to rate the recommendation system based on pre-defined criteria, as shown in table 13.

The online evaluation is done using two participants to gather results initially and to demonstrate the feasibility of the approach. It needs to be noted, however, that no derivation of the AAR can be done because the attendance of a user to the course may

Number	Question	Answer Options
1	Do you find the system easy to use?	Yes/No
2	Are the recommendations suited to your needs?	Yes/No
3	How likely are you to recommend the system to a colleague?	[1,5]
4	How would you rate the system's overall performance?	[1,5]
5	Are the recommended trainings presented in an understandable way?	Yes/No

Table 13: Online evaluation questionnaire.

occur only in the future. The resulting data from this evaluation is displayed in table 14. Additionally, user feedback, which was provided orally, includes the wish to also use the approach that was developed for recommending other items, such as job profiles or study courses, to users based on their history, desired learning opportunities, or skills. Furthermore, test subjects also hinted at the possible inclusion of entities to be excluded from the result set, highlighting the need for not only including data, but also excluding some information to not be recommended.

Aspect to be evaluated	Respondent 1	Respondent 2
Question 1	Yes	Yes
Question 2	Yes	Yes
Question 3	4	5
Question 4	5	5
Question 5	Yes	Yes
CTR	33%	42%
AAR	Not evaluated	Not evaluated

Table 14: Online evaluation result.

This results do not provide a statistically significant sample size, but are, nevertheless, able to demonstrate that the proposed approach is both easy to execute and also retrieves data that is usable for further studying the system performance. The CTR values here show a need for improving the presentation of the results, which should be further investigated. According to user feedback, some of the training descriptions tend to be too long and could be reduced in length. For employing this analysis in a productive setting, it is especially useful to easily collect the CTR and AAR metrics from any user interacting with the system, which will then provide a more detailed and exhaustive basis for subsequent evaluation when more users interact with the recommendation system.

In addition to the online evaluation, the requirements derived in section 5.1.3 are evaluated in the following to quantify their successful implementation.

The functional requirement **F1** is successfully completed, as demonstrated by the recommendation engine that is developed. This engine does not only recommend relevant trainings, but also other relevant entities, extending on the base requirement that defined

skills as the second entity to be recommended. Additionally, the engine also combines selected entities, training history, and free-text user preferences to enable fine-grained control over the outputs. The integration of different and novel aspects is ensured by the underlying semantic layer in form of a knowledge graph, providing relationships between entities to not only recommend similar items based on their content similarity, but also across different topics based on their graph similarity.

The inclusion of a wide variety of training sources, as defined in **F2**, is ensured by the system's ability to ingest textual data from any source, creating and adapting the knowledge graph automatically. This ensures maximum flexibility when adapting to different providers, paving the way for a wide-spread integration and subsequent roll-out of the solution in combination with heterogeneous providers. Consequently, this requirement is fulfilled by the overall text-based interface in combination with an adaptable entity extractor based on an LLM and re-configurable course classificator.

Functional requirement **F3** is fulfilled by the system's ability to let the user choose parameters based on the knowledge graph's integrated nodes based on the extracted entities. Consequently, it is very easy for users to select and include custom parameters for their recommendations from the list of pre-extracted and thus known nodes. There is no user intervention or manual effort associated with this selection and extraction of entities because this is carried out during the preprocessing step of adding new trainings to the recommendation system. The fulfillment of this requirement also ensures wide usability for users with varying backgrounds because they can choose any entity as a parameter that they deem necessary for their learning.

The rating function defined in **F4** has not been implemented yet as further investigation is needed on whether this data can be obtained from the underlying source systems. It might be beneficial to incorporate the existing ratings from the training providers because users will then see a wider variety of provided feedback and can, moreover, integrate their feedback and rating into the source system. A duplication of this rating functionality in the recommendation system does not seem feasible because this would lead to duplicated data which is not reflected in the original training platforms.

The system can, in its current state, provide reasoning in form of distance scores for the provided recommendations, regardless of the recommendation type. These scores are presented to the user as a "match probability" to their overall interests, serving as an indicator on why the respective item is recommended and its relevance. To provide for more accurate reasoning, the knowledge graph could be leveraged to further provide the paths that the similarity calculation relies on, which is currently not integrated but might be added at a later stage depending on the user adoption. As the basis is created, this integration is realizable with reduced effort compared to any new implementation of such a system. Consequently, **F5** is partially fulfilled.

User profiles can currently be created using the recommendation engine itself, which ingests profile data to create recommendations. These are not persisted at the moment, which still is subject to subsequent addition to the system. As the system is in a proof-of-concept state, persistence of profile data is not yet needed for realizing an initial roll-out. This also adds the benefit of reduced complexity when handling such information because there is no requirement for implementing a change process that allows users to adapt the profile when needs change. Therefore, requirement **F6** is partially fulfilled, with only the persistent layer remaining subject to further inclusion.

User feedback that provides fine-tuning for recommendations can currently not be integrated in the system because the knowledge graph only relies on pre-extracted entities and no approach is available for combining this data with user's recommendations. If and how this feature is implementable remains subject to further studies in order to be able to combine the created knowledge graph with user-inserted recommendations. It might be feasible to include a weighting of recommendations based on similar users, which would then lead to the integration of a collaborative filtering approach into the currently content- and knowledge-based hybrid approach. The added complexity of this additional layer for recommendation might, consequently, also provide an increase in maintenance and a decrease in performance. This might then counteract the overall goal of providing a usable and well-performing system. Therefore, **F7** is not fulfilled by this implementation.

Performance is evaluated by running a total of 1000 recommendation requests against the service and logging the required time correspondingly. This results in an accurate indicator for whether the performance of the implementation is sufficient, providing the response times, as depicted in figure 20. Additionally, the median response time is at 104.5 milliseconds, the 95th percentile at 117.0 milliseconds, and the 99th percentile at 173.2 milliseconds respectively, showing strong performance and low latency in the overall recommendations. This shows that the recommendation engine is capable of producing recommendations in acceptable time without impacting user satisfaction. This is also further analyzed using the online evaluation, which includes a question to address the performance of the system so that it may be rated based on the subjective feeling of the users as well. **NF1** is fulfilled based on the presented results. The non-functional requirement of usability depicted in **NF2** is covered by the development of an easy-to-use and streamlined user interface, highlighting accessbility and understandability for all potential users. Whether this current implementation is sufficient for all users remains to be studied. This will be covered by further analyzing the user interactions using the developed online evaluation approach, which also specifically covers the aspect of ease of use. Lastly, reliability as defined in **NF3** is ensured by providing recommendations based on the facts that the user provides, thus leading to an achievement of this non-functional requirement because the system does not make any assumptions on how it

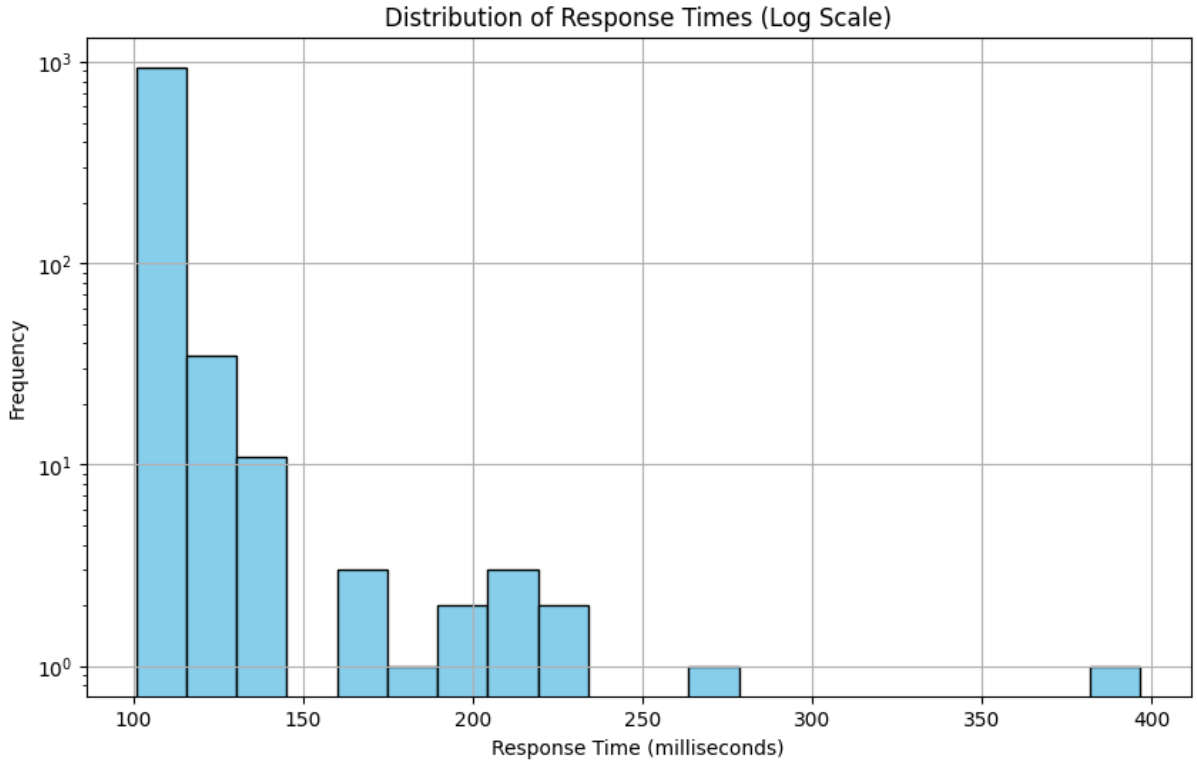


Figure 20: Execution times for creating recommendations.

should recommend items and is, instead, based solely on user-provided data.

The overall evaluation proves the successful design and implementation of the training recommendation system. The not yet implemented requirements remain subject to further investigation, but given the current state of the system it seems feasible to integrate it into the daily routines of finding relevant trainings. Totally, two requirements are currently not fulfilled which will then be subject to further research on whether they need to be integrated at a later point in time.

6 Conclusion and Outlook

This thesis developed an approach for combining a traditional, content-based recommendation system that relies on textual descriptions of trainings with a knowledge-based recommendation system for advanced providing of targeted recommendations. Employing this system greatly decreases the manual effort that is associated with deriving training needs and finding corresponding training material, leading to an improval of the accessibility of trainings and a faster adaption to changing skill requirements. To successfully integrate the system into the organization, a roll-out strategy needs to be developed, which then includes enablement sessions for fostering the adoption of the system. This is further a result from the requirements survey, in which participants also highlighted the wish for proper documentation and onboarding so that the system can be used effectively. It moreover remains subject to evaluation whether the rollout requires approval by works councils because personal data, such as skills or a detailed history of attended trainings, is processed during the recommendation phase. As this data is provided by the user without any automatic replication mechanism, the hurdles could be easily overcome by provisioning the system as an opt-in solution, which requires users to actively input this information.

Due to the modular architecture of the system, included entities and classes can easily be added or changed, providing great flexibility for adapting the system to other use cases. These may include the recommendation of study courses to students based on their interests and previous knowledge or the recommendation of job postings to prospective applicants based on their previous experience and professional aspirations.

From an architectural perspective, the developed system provides a micro-service based, extensible architecture, which can easily be adapted to different use cases. The employed technologies rely on common open-source tools, such as widely used Python libraries, with the only exception being the usage of the proprietary GPT-4 as an entity extractor. This can easily be exchanged to an open-source LLM, allowing for quick adaption to different scenarios and decreasing the reliance on proprietary technology by keeping the usage of such to a required minimum. The possible usage of an open-source LLM as a replacement for GPT-4 as an entity extractor in the current implementation can be further evaluated in subsequent work because this might provide the ability to reduce vendor dependency and enable organizations to regain control over all aspects of the recommendation system. This would also prove to be beneficial because there are no license costs associated with the usage of an open-source model, greatly benefiting the Total Cost of Ownership (TCO) of the overall system. Furthermore, this work provides a proof-of-concept for employing an LLM as a viable and improved alternative to common entity extraction approaches, resulting in a more robust and less error-prone NER. Another contribution that is made

is the practical application of synthetic data for training an NER model. This synthetic data is created using an LLM, proving that the usage of such a model can lead to a simplified data collection and creation process, reducing the manual effort that is required in sparse data conditions by enabling automated training data creation that is already labeled correspondingly. Improvement is still needed for the handling of deletes and updates of any included data because the current approach only allows for substitution of data so that changes always result in a deletion and then a re-insertion of the relevant information. Additionally, performance needs to be closely monitored when scaling the user base because no reliable data for the scalability of the overall system exists in this proof-of-concept.

From an economic perspective, the TCO of the provisioned system needs to be further studied because the current approach relies heavily on an externally provided GPT-4 model and also requires computation capacities to create the word embeddings for storing the training descriptions in the vector database. It is also to be noted that the current implementation does not provide any fault-tolerance mechanism or auto-recovery in case of individual service failure, highlighting the need for a more resilient approach when using the system in a productive scenario. Mitigating these prevalent issues may moreover result in an increase of the overall costs of running and maintaining the system because auto scaling using a container orchestration platform results in higher operational costs due to increased hardware requirements.

The effectiveness of the system on improving the overall knowledge that is obtained by employees remains subject to further studying because this needs to be monitored after its roll-out. To derive data that is usable, there needs to be a sufficient delay between the initial introduction of the system and the evaluation of the long-term effectiveness to allow for widespread diffusion and integration of the system in the employees' training routines. This data will then be raised by leveraging the already developed online evaluation, enabling rapid and structured data collection for evaluating the long-term effects of the system.

To wrap it up, this thesis provides a viable proof-of-concept for improving employee education while focusing on individual needs instead of providing a one-size-fits-all approach.

A Appendix

A.1 Requirements Survey

Personal Information:

1. **How many years of professional work experience (incl. vocational training or studies) do you have?**

   ☐ <3 Years

   ☐ 3-5 Years

   ☐ 6-9 Years

   ☐ 10-15 Years

   ☐ >15 Years

2. **What is your department/functional area?**

   ☐ HR

   ☐ Development

   ☐ Consulting

   ☐ Sales

   ☐ Marketing

   ☐ Finance & Administration

   ☐ Other (please specify): _____

3. **What is your position?**

   ☐ Executive

   ☐ Team Leader

   ☐ Individual Contributor

   ☐ Student

   ☐ Other (please specify): _____

Current Training Process:

1. **How are employee training needs currently identified in your department? (Check all that apply)**

   ☐ Individual self-assessment

   ☐ Managerial assessment

- ☐ Department-wide allocation
- ☐ Other (please specify): _____

2. **What challenges do you face in the current training identification and recommendation process? (Check all that apply)**

   - ☐ Lack of awareness about available trainings
   - ☐ Lack of awareness about available training platforms
   - ☐ Difficulty in assessing training needs
   - ☐ No ability to get in-depth personal guidance
   - ☐ Lack of centralized system for trainings
   - ☐ Other (please specify): _____

Desired System Features:

1. **What features would you like to see in a training recommendation system? (Check all that apply)**

   - ☐ Personalized training recommendations
   - ☐ Skill gap analysis
   - ☐ Training path recommendations
   - ☐ Integrated performance tracking
   - ☐ Other (please specify): _____

Training Content Preferences:

1. **What types of training content do you find most valuable? (Check all that apply)**

   - ☐ Instructor-led workshops and seminars
   - ☐ On-demand video courses
   - ☐ Live online courses
   - ☐ Hands-on exercises
   - ☐ On-the-job training
   - ☐ Certifications
   - ☐ Soft skills development

☐ Other (please specify): _____

2. **For which topics would you like to get training recommendations? (Check all that apply)**

   ☐ Mobile Development

   ☐ Frontend Development

   ☐ Backend Development

   ☐ Softskills

   ☐ IT Security

   ☐ Management and Business

   ☐ Cloud Engineering

   ☐ Product-specific Training

   ☐ Machine Learning

   ☐ Software Architecture

   ☐ Other (please specify): _____

System Usability and Integration:

1. **How important is ease of use for the training recommendation system?**

   ☐ Very Important

   ☐ Important

   ☐ Neutral

   ☐ Not So Important

   ☐ Not Important at All

2. **How important are novel and interesting recommendations?**

   ☐ Very Important

   ☐ Important

   ☐ Neutral

   ☐ Not So Important

   ☐ Not Important at All

3. **How important is the ability to include custom parameters, such as desired or known skills?**

- ☐ Very Important
- ☐ Important
- ☐ Neutral
- ☐ Not So Important
- ☐ Not Important at All

4. **How important is the ability to define individual learning goals?**

   - ☐ Very Important
   - ☐ Important
   - ☐ Neutral
   - ☐ Not So Important
   - ☐ Not Important at All

5. **How important is the ability to directly rate and evaluate taken courses that were recommended?**

   - ☐ Very Important
   - ☐ Important
   - ☐ Neutral
   - ☐ Not So Important
   - ☐ Not Important at All

User Involvement:

1. **Would you like to be able to contribute training recommendations to the system?**

   - ☐ Yes
   - ☐ No

2. **What training or support do you think would be necessary for users to effectively use such a recommendation system?**

   - ☐ Onboarding sessions
   - ☐ User guides/documentation
   - ☐ Helpdesk support
   - ☐ Trainings/workshops

☐ Other (please specify): _____

3. **What would stop you from using such a recommendation system?**

_____

_____

_____

_____

_____

Additional Comments:

1. **Do you have any additional comments, suggestions, or specific requirements for the employee training recommendation system?**

_____

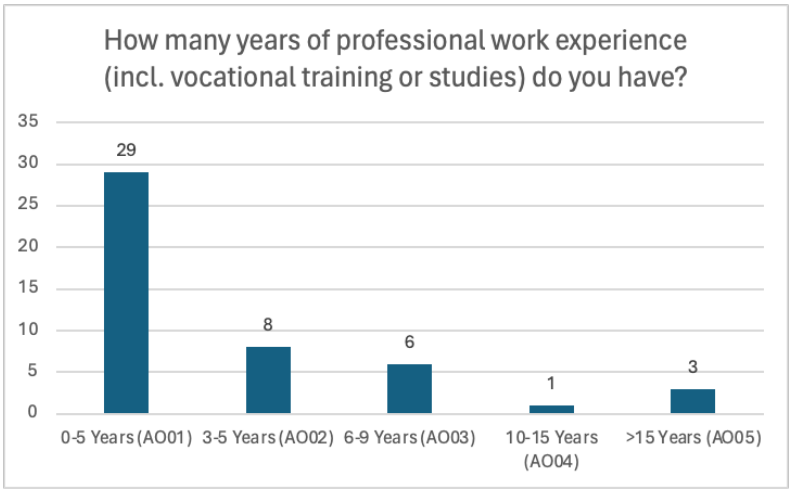
_____

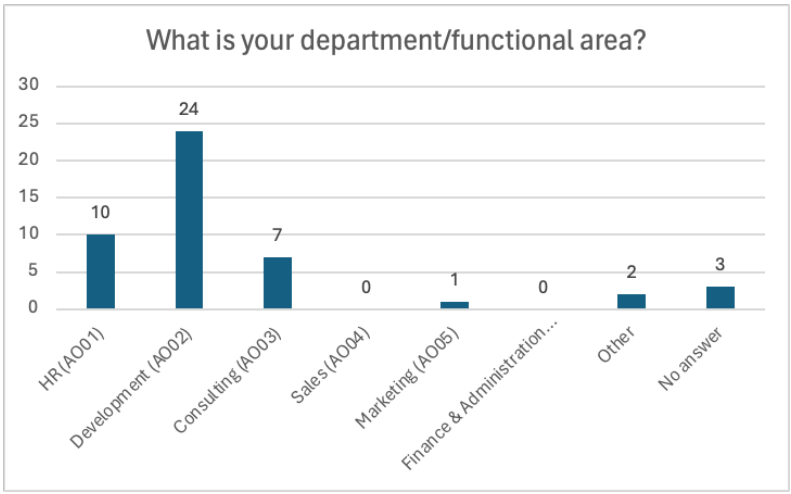
_____

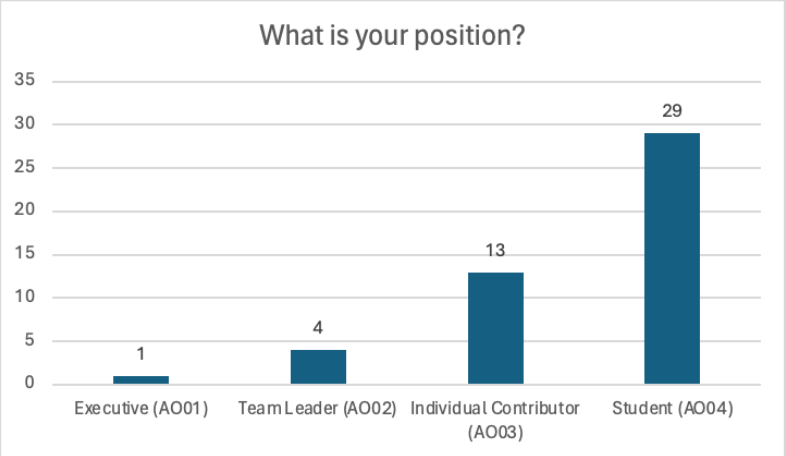
_____

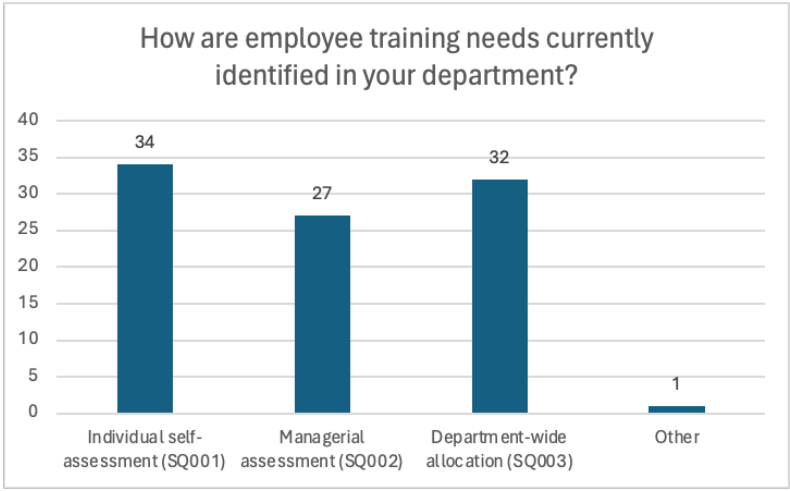
_____

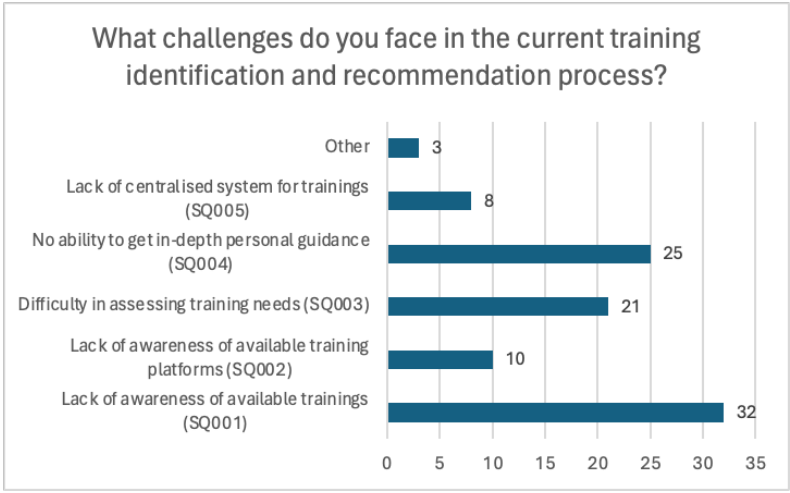
A.2 Survey Responses

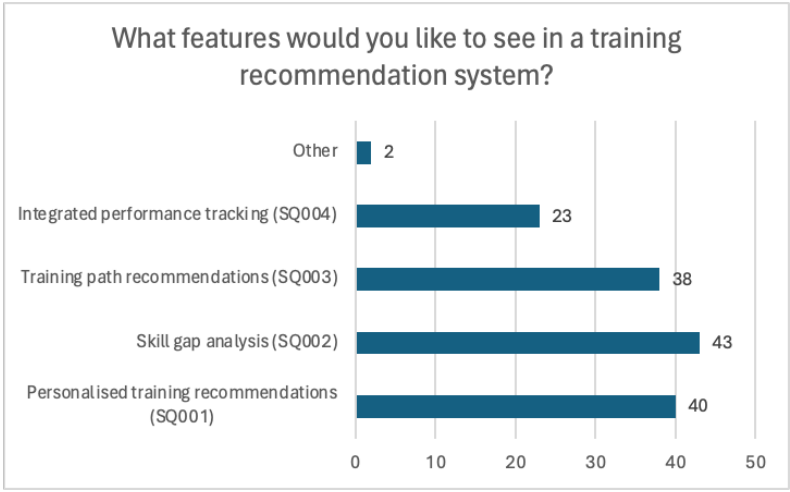


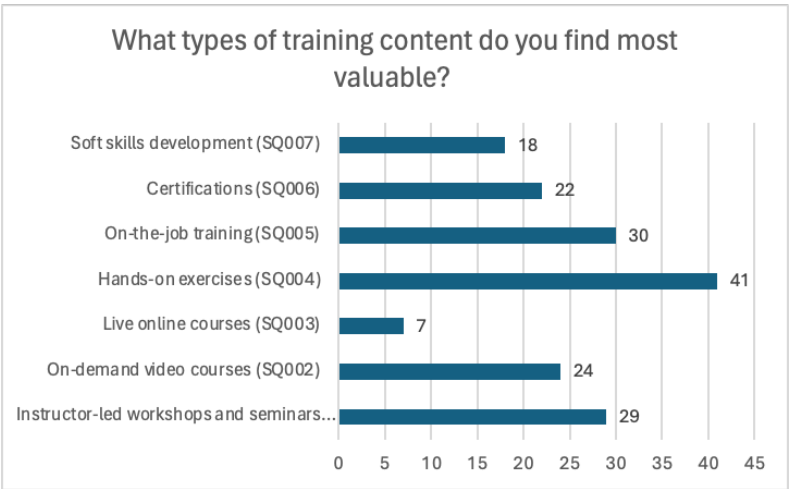




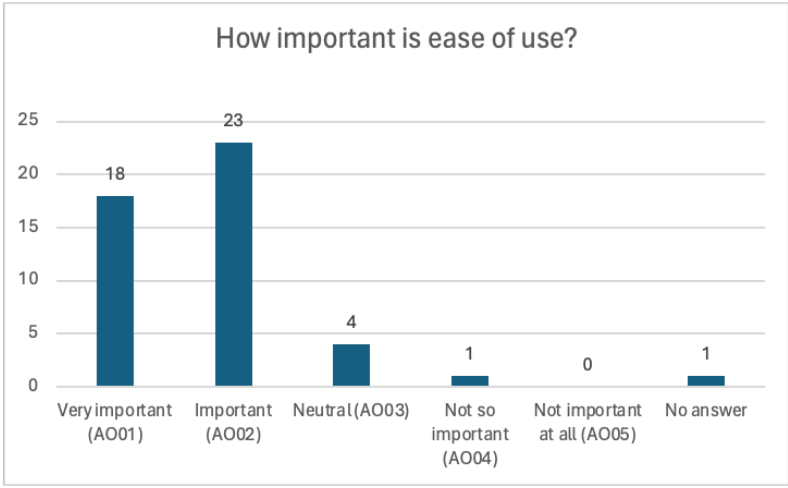

How are employee training needs currently identified in your department?
40
35
30
25
20
15
10
5
0
34
27
32
1
Individual self-assessment (SQ001)
Managerial assessment (SQ002)
Department-wide allocation (SQ003)
Other

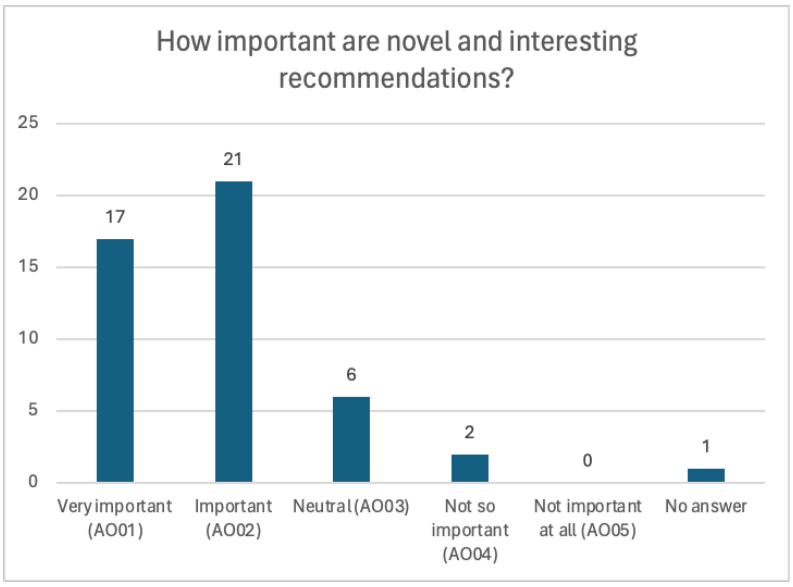

What challenges do you face in the current training identification and recommendation process?
Other
3
Lack of centralised system for trainings (SQ005)
8
No ability to get in-depth personal guidance (SQ004)
25
Difficulty in assessing training needs (SQ003)
21
Lack of awareness of available training platforms (SQ002)
10
Lack of awareness of available trainings (SQ001)
32
0 5 10 15 20 25 30 35

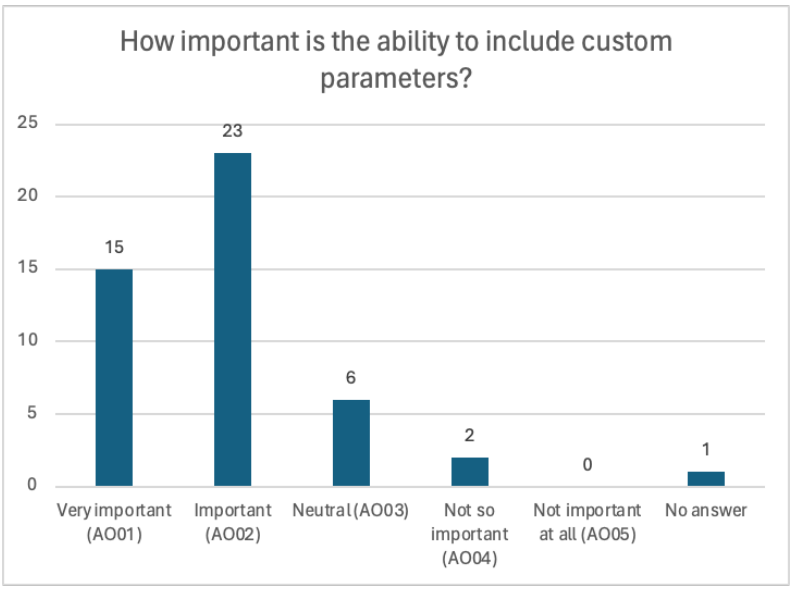

What features would you like to see in a training recommendation system?
Other
2
Integrated performance tracking (SQ004)
23
Training path recommendations (SQ003)
38
Skill gap analysis (SQ002)
43
Personalised training recommendations (SQ001)
40
0 10 20 30 40 50

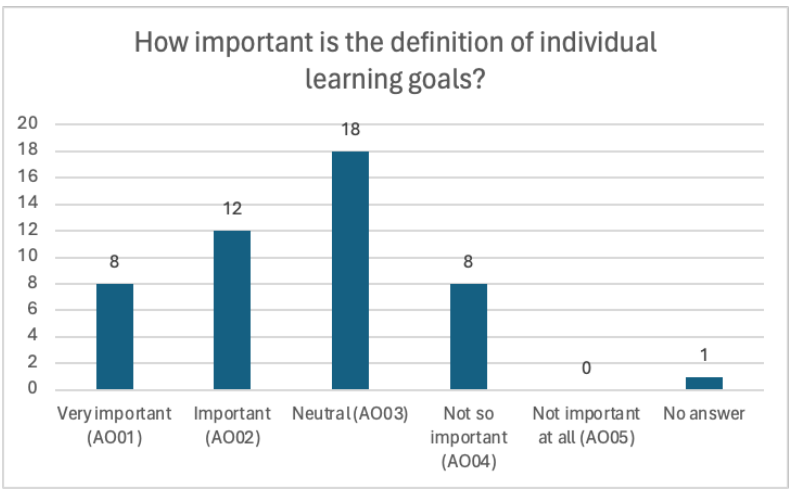

What types of training content do you find most valuable?
Soft skills development (SQ007)
18
Certifications (SQ006)
22
On-the-job training (SQ005)
30
Hands-on exercises (SQ004)
41
Live online courses (SQ003)
7
On-demand video courses (SQ002)
24
Instructor-led workshops and seminars...
29
0 5 10 15 20 25 30 35 40 45

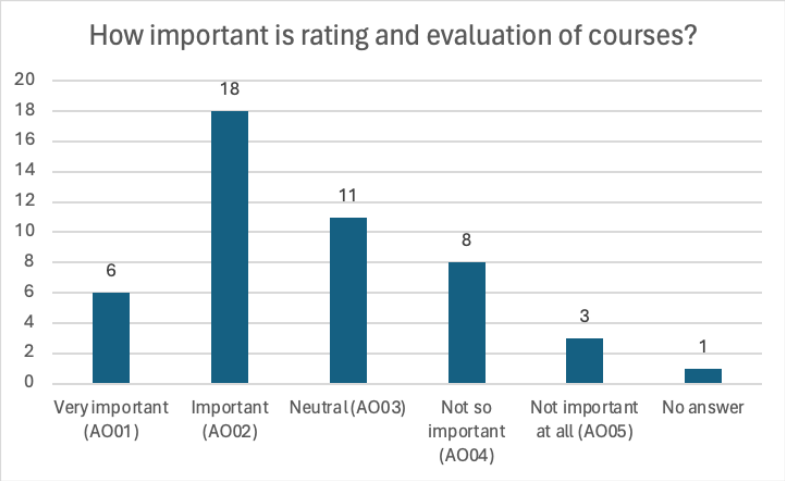

For which topics would you like to get training recommendations?
Other
3
Software Architecture (SQ010)
28
Machine Learning (SQ009)
40
Product-specific Training (SQ008)
13
Cloud Engineering (SQ007)
28
Management and Business (SQ006)
23
IT Security (SQ005)
18
Soft Skills (SQ004)
24
Backend Development (SQ003)
28
Frontend Development (SQ002)
20
Mobile Development (SQ001)
11
0 5 10 15 20 25 30 35 40 45

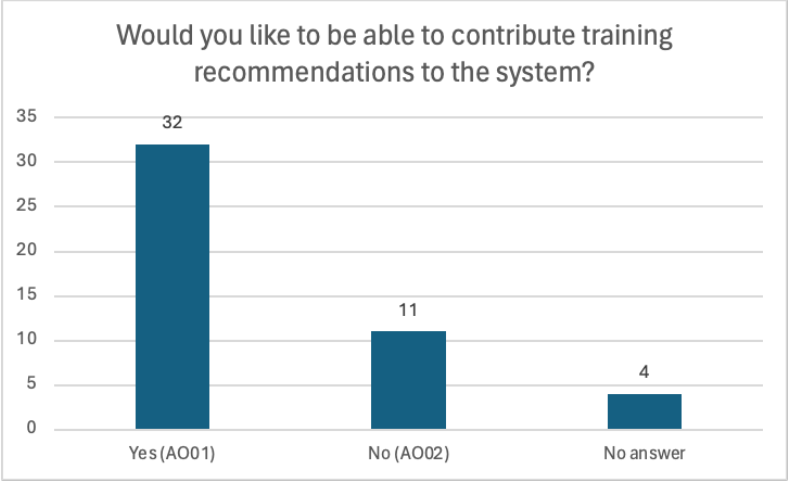

How important is ease of use?
25
20
15
10
5
0
18
23
4
1
0
1
Very important (AO01)
Important (AO02)
Neutral (AO03)
Not so important (AO04)
Not important at all (AO05)
No answer

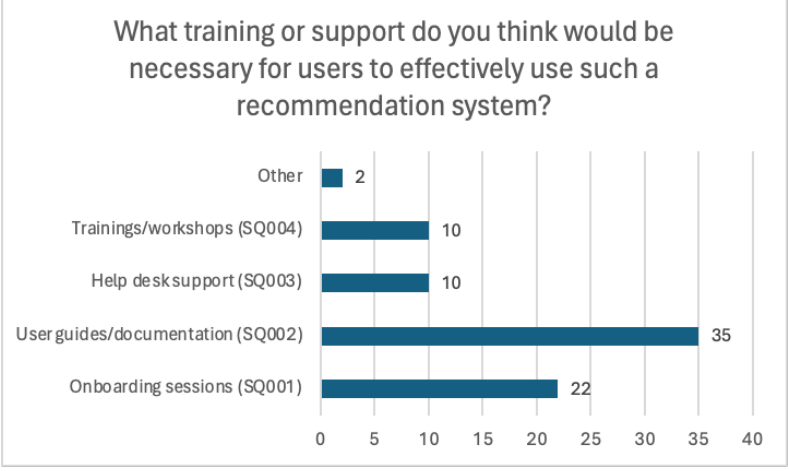

How important are novel and interesting recommendations?
25
20
15
10
5
0
17
21
6
2
0
1
Very important (AO01)
Important (AO02)
Neutral (AO03)
Not so important (AO04)
Not important at all (AO05)
No answer


How important is the ability to include custom parameters?
25
20
15
10
5
0
15
23
6
2
0
1
Very important (AO01)
Important (AO02)
Neutral (AO03)
Not so important (AO04)
Not important at all (AO05)
No answer


How important is the definition of individual learning goals?
20
18
16
14
12
10
8
6
4
2
0
8
12
18
8
0
1
Very important (AO01)
Important (AO02)
Neutral (AO03)
Not so important (AO04)
Not important at all (AO05)
No answer


How important is rating and evaluation of courses?
20
18
16
14
12
10
8
6
4
2
0
18
11
8
6
3
1
Very important (AO01)
Important (AO02)
Neutral (AO03)
Not so important (AO04)
Not important at all (AO05)
No answer


Would you like to be able to contribute training recommendations to the system?
35
30
25
20
15
10
5
0
32
11
4
Yes (AO01)
No (AO02)
No answer


What training or support do you think would be necessary for users to effectively use such a recommendation system?
Other
2
Trainings/workshops (SQ004)
10
Help desk support (SQ003)
10
User guides/documentation (SQ002)
35
Onboarding sessions (SQ001)
22
0
5
10
15
20
25
30
35
40

What might stop you from using such a recommendation system?
Lack of time and knowledge
Extensive onboarding,too complicated,wrong recommendations
Irrelevant recommendations, low Usability, low Usefulness
bad trainings
Wenn es nicht leicht und intuitiv zu benutzen ist. Man sollte jederzeit, wenn man auch mal nur 20-30 Minuten hat, schnell auf begonnene Schulungen zugreifen können. Wenn man mit recommendations überflutet wird, zu viel ist auch nicht gut.
If it is too complex
If it's not accurate or if it takes too long to understand how to navigate
Complexity: The system might be too complex to easily become familiar with and effectively use. Lack of time: The user might not have the time to utilize the system or engage with new training.
Bad performance of the system
Accessibility of the training, Uncertainty if recommended training meets tracking KPIs
repetitive recommendations
Concerns about the privacy and security of individual performance data or personal information may lead to hesitancy among team members to participate in such a system. A recommendation system that does not allow customization based on the specific needs and preferences.
There are already enough trainings

Do you have any additional comments, suggestions, or specific requirements for the employee training recommendation system?
Regularly update the training content and recommendations to keep them relevant. Ensure robust data security measures to protect sensitive employee information and comply with relevant data protection regulations Ensure that the recommendation system accommodates different learning styles such as visual, auditory, and kinesthetic

Question	**Answers in "other" category**
What is your department/functional area?	VT, Board Office
How are employee training needs currently identified in your department?	Mandatory Trainings
What challenges do you face in the current training identification and recommendation process?	Motivation, Keine. Schulungen werden uns vom Manager/Dezi vorgeschlagen, auch selbst, findet man ziemlich leicht Schulungen zu aktuellen Themen, Lack of nothing
What features would you like to see in a training recommendation system?	Interactive Exercises, Recommendations for joint training within the team/department
For which topics would you like to get training recommendations?	Data Engineering, GenAI, AI, Product Development, AI
What training or support do you think would be necessary for users to effectively use such a recommendation system?	should be self-explaining, None, in the best case

A.3 Knowledge Graph Entities

Entities of type "class"	**Entities of type "content"**
Mobile Development	Programming Language
Frontend Development	Framework
Backend Development	Technology
Soft Skills	Skill
IT Security	Methodology
Management and Business	Product
Cloud Engineering	
Product Training	
Machine Learning	
Software Architecture	

A.4 Preprocessed Document


```
{
    "_id": {
        "$oid": "65e6141cad22dd34dca6f7a5"
    },
    "title": "Cloud Computing Fundamentals Cloud Concepts",
    "description": "In this course, you will learn the basics of
       cloud computing focusing on the core features of cloud
       technologies, including cloud deployment models, networking
       concepts, storage types, and cloud design.",
    "from_cache": false,
    "created_at": {
        "$date": {
            "$numberLong": "1709577244478"
        }
    },
    "pred": "cloud engineering",
    "confidence": {
        "$numberDouble": "0.8220365708975127"
    },
    "programming_languages": [],
    "frameworks": [],
    "technologies": [
        "cloud computing"
    ],
    "skills": [
        "understanding of cloud deployment models",
        "knowledge of networking concepts",
        "familiarity with different storage types",
        "cloud design skills"
    ],
    "methodologies": [],
    "products": []
}
```

B Bibliography

[1] B. J. Ngereja and B. Hussein, "Employee learning in the digitalization context: An evaluation from team members' and project managers' perspectives", *Procedia Computer Science*, International Conference on ENTERprise Information Systems / ProjMAN - International Conference on Project MANagement / HCist - International Conference on Health and Social Care Information Systems and Technologies 2021, vol. 196, pp. 902–909, Jan. 2022, ISSN: 1877-0509. DOI: 10.1016/j.procs.2021.12.091. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050921023140 (visited on 01/04/2024).

[2] A.-S. Turulski, *Umfrage zur Bedeutung von E-Learning Anwendungen in Unternehmen 2022*, 2023. [Online]. Available: https://de.statista.com/statistik/daten/studie/203748/umfrage/bedeutung-von-e-learning-anwendungen-in-unternehmen/ (visited on 10/06/2023).

[3] C. Peng, F. Xia, M. Naseriparsa, and F. Osborne, "Knowledge Graphs: Opportunities and Challenges", en, *Artificial Intelligence Review*, vol. 56, no. 11, pp. 13 071–13 102, Nov. 2023, ISSN: 0269-2821, 1573-7462. DOI: 10.1007/s10462-023-10465-9. [Online]. Available: https://link.springer.com/10.1007/s10462-023-10465-9 (visited on 02/14/2024).

[4] M. Krötzsch and V. Thost, "Ontologies for Knowledge Graphs: Breaking the Rules", en, in *The Semantic Web – ISWC 2016*, P. Groth, E. Simperl, A. Gray, *et al.*, Eds., vol. 9981, Cham: Springer International Publishing, 2016, pp. 376–392, ISBN: 9783319465227 9783319465234. DOI: 10.1007/978-3-319-46523-4_23. [Online]. Available: https://link.springer.com/10.1007/978-3-319-46523-4_23 (visited on 02/18/2024).

[5] A. Bordes, J. Weston, R. Collobert, and Y. Bengio, "Learning Structured Embeddings of Knowledge Bases", *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 25, no. 1, pp. 301–306, Aug. 2011, ISSN: 2374-3468, 2159-5399. DOI: 10.1609/aaai.v25i1.7917. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/7917 (visited on 02/17/2024).

[6] Ta Duy Cong Chien, "DETECTION OF SEMANTIC RELATIONS BASED ON KNOWLEDGE GRAPH", *Journal of Science and Technology - IUH*, vol. 52, no. 04, Feb. 2022, ISSN: 2525-2267, 2525-2267. DOI: 10.46242/jstiuh.v52i05.4110. [Online]. Available: https://jst.iuh.edu.vn/index.php/jst-iuh/article/view/4110 (visited on 02/17/2024).

[7] J. Wang, W. Wang, F. Meng, L. Zhou, and S. Guo, "A Review of Knowledge Reasoning Based on Neural Network", in *2023 IEEE 12th International Conference on Communication Systems and Network Technologies (CSNT)*, Bhopal, India:

IEEE, Apr. 2023, pp. 580–584, ISBN: 9781665462617. DOI: 10.1109/CSNT57126.2023.10134585. [Online]. Available: https://ieeexplore.ieee.org/document/10134585/ (visited on 02/17/2024).

[8] X. Chen, S. Jia, and Y. Xiang, "A review: Knowledge reasoning over knowledge graph", en, *Expert Systems with Applications*, vol. 141, p. 112 948, Mar. 2020, ISSN: 09574174. DOI: 10.1016/j.eswa.2019.112948. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0957417419306669 (visited on 02/17/2024).

[9] J. Ma, D. Li, Y. Chen, Y. Qiao, H. Zhu, and X. Zhang, "A Knowledge Graph Entity Disambiguation Method Based on Entity-Relationship Embedding and Graph Structure Embedding", en, *Computational Intelligence and Neuroscience*, vol. 2021, A. N. Belkacem, Ed., pp. 1–11, Sep. 2021, ISSN: 1687-5273, 1687-5265. DOI: 10.1155/2021/2878189. [Online]. Available: https://www.hindawi.com/journals/cin/2021/2878189/ (visited on 03/24/2024).

[10] D. Yuan, K. Zhou, and C. Yang, "Architecture and Application of Traffic Safety Management Knowledge Graph Based on Neo4j", en, *Sustainability*, vol. 15, no. 12, p. 9786, Jun. 2023, ISSN: 2071-1050. DOI: 10.3390/su15129786. [Online]. Available: https://www.mdpi.com/2071-1050/15/12/9786 (visited on 02/17/2024).

[11] M. C. Silva, P. Eugénio, D. Faria, and C. Pesquita, "Ontologies and Knowledge Graphs in Oncology Research", en, *Cancers*, vol. 14, no. 8, p. 1906, Apr. 2022, ISSN: 2072-6694. DOI: 10.3390/cancers14081906. [Online]. Available: https://www.mdpi.com/2072-6694/14/8/1906 (visited on 02/17/2024).

[12] *OWL 2 Web Ontology Language Document Overview (Second Edition)*. [Online]. Available: https://www.w3.org/TR/owl2-overview/ (visited on 02/17/2024).

[13] B. DuCharme, *Learning SPARQL: querying and updating with SPARQL 1.1*, eng, 2nd ed. Sebastopol, CA: O'Reilly Media, 2013, OCLC: 853679890, ISBN: 9781449371562.

[14] A. Hogan, "SPARQL Query Language", en, in *The Web of Data*, Cham: Springer International Publishing, 2020, pp. 323–448, ISBN: 9783030515799 9783030515805. DOI: 10.1007/978-3-030-51580-5_6. [Online]. Available: http://link.springer.com/10.1007/978-3-030-51580-5_6 (visited on 02/29/2024).

[15] R. P. Lebret, "Word Embeddings for Natural Language Processing", en, Sep. 2016. DOI: 10.5075/EPFL-THESIS-7148. [Online]. Available: http://infoscience.epfl.ch/record/221430 (visited on 03/13/2024).

[16] A. Neelima and S. Mehrotra, "A Comprehensive Review on Word Embedding Techniques", in *2023 International Conference on Intelligent Systems for Communication, IoT and Security (ICISCoIS)*, Coimbatore, India: IEEE, Feb. 2023, pp. 538–

543, ISBN: 9798350335835. DOI: 10.1109/ICISCoIS56541.2023.10100347. [Online]. Available: https://ieeexplore.ieee.org/document/10100347/ (visited on 03/13/2024).

[17] Y. Han, C. Liu, and P. Wang, *A Comprehensive Survey on Vector Database: Storage and Retrieval Technique, Challenge*, arXiv:2310.11703 [cs], Oct. 2023. [Online]. Available: http://arxiv.org/abs/2310.11703 (visited on 03/19/2024).

[18] G. Sidorov, A. Gelbukh, H. Gómez-Adorno, and D. Pinto, "Soft Similarity and Soft Cosine Measure: Similarity of Features in Vector Space Model", *Computación y Sistemas*, vol. 18, no. 3, Sep. 2014, ISSN: 1405-5546. DOI: 10.13053/cys-18-3-2043. [Online]. Available: http://cys.cic.ipn.mx/ojs/index.php/CyS/article/view/2043 (visited on 03/19/2024).

[19] R. Németh and J. Koltai, "Natural language processing: The integration of a new methodological paradigm into sociology", *Intersections*, vol. 9, no. 1, pp. 5–22, Apr. 2023, ISSN: 2416-089X. DOI: 10.17356/ieejsp.v9i1.871. [Online]. Available: https://intersections.tk.hu/index.php/intersections/article/view/871 (visited on 03/19/2024).

[20] Abhay A. Dande and Dr. M. A. Pund#, "A Review Study on Applications of Natural Language Processing", en, *International Journal of Scientific Research in Science, Engineering and Technology*, pp. 122–126, Mar. 2023, ISSN: 2394-4099, 2395-1990. DOI: 10.32628/IJSRSET2310214. [Online]. Available: https://ijsrset.com/IJSRSET2310214 (visited on 01/04/2024).

[21] M. Madan, A. Rani, and N. Bhateja, "A Systematic Study of Various Approaches and Problem Areas of Named Entity Recognition", en, in *Proceedings of International Conference on Recent Innovations in Computing*, Y. Singh, C. Verma, I. Zoltán, J. K. Chhabra, and P. K. Singh, Eds., vol. 1011, Series Title: Lecture Notes in Electrical Engineering, Singapore: Springer Nature Singapore, 2023, pp. 545–558, ISBN: 978-981-9906-00-0 978-981-9906-01-7. DOI: 10.1007/978-981-99-0601-7_42. [Online]. Available: https://link.springer.com/10.1007/978-981-99-0601-7_42 (visited on 01/21/2024).

[22] X. Liu, H. Chen, and W. Xia, "Overview of Named Entity Recognition", en, *Journal of Contemporary Educational Research*, vol. 6, no. 5, pp. 65–68, May 2022, ISSN: 2208-8474, 2208-8466. DOI: 10.26689/jcer.v6i5.3958. [Online]. Available: http://ojs.bbwpublisher.com/index.php/JCER/article/view/3958 (visited on 01/04/2024).

[23] A. Sharma, S. Shreya, and S. Terni, "Named Entity Recognition in Social Media Data", *International Journal for Research in Applied Science and Engineering Technology*, vol. 10, no. 8, pp. 268–288, Aug. 2022, ISSN: 23219653. DOI: 10.22214/

ijraset.2022.46181. [Online]. Available: https://www.ijraset.com/best-journal/named-entity-recognition-in-social-media-data-765 (visited on 02/18/2024).

[24] M. Kumar, K. K. Chaturvedi, A. Sharma, *et al.*, "An Algorithm for Automatic Text Annotation for Named Entity Recognition using spaCy Framework", In Review, preprint, May 2023. DOI: 10.21203/rs.3.rs-2930333/v1. [Online]. Available: https://www.researchsquare.com/article/rs-2930333/v1 (visited on 01/21/2024).

[25] S. Algamdi, A. Albanyan, S. K. Shah, and Z. Tariq, "Twitter Accounts Suggestion: Pipeline Technique SpaCy Entity Recognition", en, in *2022 IEEE International Conference on Big Data (Big Data)*, Osaka, Japan: IEEE, Dec. 2022, pp. 5121–5125, ISBN: 978-1-66548-045-1. DOI: 10.1109/BigData55660.2022.10020570. [Online]. Available: https://ieeexplore.ieee.org/document/10020570/ (visited on 01/21/2024).

[26] A. Hassaine, J. Al Otaibi, and A. Jaoua, "Named Entity Disambiguation using Hierarchical Text Categorization", en, in *Qatar Foundation Annual Research Conference Proceedings Volume 2016 Issue 1*, Qatar National Convention Center (QNCC), Doha, Qatar, Hamad bin Khalifa University Press (HBKU Press), 2016. DOI: 10.5339/qfarc.2016.ICTPP3064. [Online]. Available: https://www.qscience.com/content/papers/10.5339/qfarc.2016.ICTPP3064 (visited on 03/14/2024).

[27] Kowsari, Jafari Meimandi, Heidarysafa, Mendu, Barnes, and Brown, "Text Classification Algorithms: A Survey", en, *Information*, vol. 10, no. 4, p. 150, Apr. 2019, ISSN: 2078-2489. DOI: 10.3390/info10040150. [Online]. Available: https://www.mdpi.com/2078-2489/10/4/150 (visited on 01/04/2024).

[28] H. P. Luhn, "A Statistical Approach to Mechanized Encoding and Searching of Literary Information", en, *IBM Journal of Research and Development*, vol. 1, no. 4, pp. 309–317, Oct. 1957, ISSN: 0018-8646, 0018-8646. DOI: 10.1147/rd.14.0309. [Online]. Available: http://ieeexplore.ieee.org/document/5392697/ (visited on 01/04/2024).

[29] J. B. Lovins, "Development of a stemming algorithm", *Mech. Transl. Comput. Linguistics*, 1968. [Online]. Available: https://www.semanticscholar.org/paper/Development-of-a-stemming-algorithm-Lovins/6b3853f08c482fe1bfbe39d656d50a8c73976f3c (visited on 01/04/2024).

[30] T. Korenius, J. Laurikkala, K. Järvelin, and M. Juhola, "Stemming and lemmatization in the clustering of finnish text documents", en, in *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, Washington D.C. USA: ACM, Nov. 2004, pp. 625–633, ISBN: 978-1-58113-874-0. DOI:

10.1145/1031171.1031285. [Online]. Available: https://dl.acm.org/doi/10.1145/1031171.1031285 (visited on 01/04/2024).

[31] S. U. Hassan, J. Ahamed, and K. Ahmad, "Analytics of machine learning-based algorithms for text classification", *Sustainable Operations and Computers*, vol. 3, pp. 238–248, Jan. 2022, ISSN: 2666-4127. DOI: 10.1016/j.susoc.2022.03.001. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666412722000101 (visited on 01/21/2024).

[32] "Accuracy", en, in *Encyclopedia of Machine Learning*, C. Sammut and G. I. Webb, Eds., Boston, MA: Springer US, 2011, pp. 9–10, ISBN: 9780387307688 9780387301648. DOI: 10.1007/978-0-387-30164-8_3. [Online]. Available: https://link.springer.com/10.1007/978-0-387-30164-8_3 (visited on 01/27/2024).

[33] J. Huang and C. Ling, "Using AUC and accuracy in evaluating learning algorithms", *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 3, pp. 299–310, Mar. 2005, Conference Name: IEEE Transactions on Knowledge and Data Engineering, ISSN: 1558-2191. DOI: 10.1109/TKDE.2005.50. [Online]. Available: https://ieeexplore.ieee.org/document/1388242 (visited on 01/21/2024).

[34] K. M. Ting, "Precision and Recall", en, in *Encyclopedia of Machine Learning*, C. Sammut and G. I. Webb, Eds., Boston, MA: Springer US, 2011, pp. 781–781, ISBN: 9780387307688 9780387301648. DOI: 10.1007/978-0-387-30164-8_652. [Online]. Available: https://link.springer.com/10.1007/978-0-387-30164-8_652 (visited on 01/27/2024).

[35] S. Kumari, "Job Recommendation System Using NLP", en, *International Journal for Research in Applied Science and Engineering Technology*, vol. 11, no. 5, pp. 2721–2728, May 2023, ISSN: 23219653. DOI: 10.22214/ijraset.2023.52183. [Online]. Available: https://www.ijraset.com/best-journal/job-recommedation-system-using-nlp-867 (visited on 01/04/2024).

[36] L. Li, Z. Zhang, and S. Zhang, "Hybrid Algorithm Based on Content and Collaborative Filtering in Recommendation System Optimization and Simulation", en, *Scientific Programming*, vol. 2021, Y.-Z. Jiang, Ed., pp. 1–11, May 2021, ISSN: 1875-919X, 1058-9244. DOI: 10.1155/2021/7427409. [Online]. Available: https://www.hindawi.com/journals/sp/2021/7427409/ (visited on 01/27/2024).

[37] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms", en, in *Proceedings of the 10th international conference on World Wide Web*, Hong Kong Hong Kong: ACM, Apr. 2001, pp. 285–295, ISBN: 978-1-58113-348-6. DOI: 10.1145/371920.372071. [Online]. Available: https://dl.acm.org/doi/10.1145/371920.372071 (visited on 01/04/2024).

[38] A. Chakraborty, "Perceptron Collaborative Filtering", en, *International Journal For Science Technology And Engineering*, vol. 11, no. 2, pp. 437–447, Feb. 2023, ISSN: 2321-9653. DOI: 10.22214/ijraset.2023.49044. [Online]. Available: https://typeset.io/papers/perceptron-collaborative-filtering-3ufhudns (visited on 01/04/2024).

[39] P. Nagaraj, V. Muneeswaran, B. Mohith Kumar, K. Rama Krishna Rao, M. S. Nagaraju, and M. V. H. Kumar, "Comparative Analysis of Different Approaches to the Music Recommendation System", en, in *2023 International Conference on Computer Communication and Informatics (ICCCI)*, Coimbatore, India: IEEE, Jan. 2023, pp. 1–7, ISBN: 9798350348217. DOI: 10.1109/ICCCI56745.2023.10128645. [Online]. Available: https://ieeexplore.ieee.org/document/10128645/ (visited on 01/04/2024).

[40] J. Kumari, "A Movie Recommendation System Based on A Convolutional Neural Network", en, *International Journal for Research in Applied Science and Engineering Technology*, vol. 9, no. 12, pp. 1445–1455, Dec. 2021, ISSN: 23219653. DOI: 10.22214/ijraset.2021.39538. [Online]. Available: https://www.ijraset.com/best-journal/movie-recommendation-system-based-on-a-convolutional-neural-network (visited on 01/04/2024).

[41] R. Patel and P. Thakkar, "Addressing Item Cold Start Problem in Collaborative Filtering-Based Recommender Systems Using Auxiliary Information", en, in *IOT with Smart Systems*, J. Choudrie, P. Mahalle, T. Perumal, and A. Joshi, Eds., ser. Smart Innovation, Systems and Technologies, Singapore: Springer Nature, 2023, pp. 133–142, ISBN: 978-981-19357-5-6. DOI: 10.1007/978-981-19-3575-6_16.

[42] Y. Sun, N. J. Yuan, X. Xie, K. McDonald, and R. Zhang, "Collaborative Intent Prediction with Real-Time Contextual Data", en, *ACM Transactions on Information Systems*, vol. 35, no. 4, pp. 1–33, Oct. 2017, ISSN: 1046-8188, 1558-2868. DOI: 10.1145/3041659. [Online]. Available: https://dl.acm.org/doi/10.1145/3041659 (visited on 01/04/2024).

[43] D. Sun, Z. Luo, and F. Zhang, "A novel approach for collaborative filtering to alleviate the new item cold-start problem", en, in *2011 11th International Symposium on Communications & Information Technologies (ISCIT)*, Hangzhou, China: IEEE, Oct. 2011, pp. 402–406, ISBN: 978-1-4577-1295-1 978-1-4577-1294-4 978-1-4577-1293-7. DOI: 10.1109/ISCIT.2011.6089959. [Online]. Available: http://ieeexplore.ieee.org/document/6089959/ (visited on 01/04/2024).

[44] M. J. Pazzani and D. Billsus, "Content-Based Recommendation Systems", en, in *The Adaptive Web*, P. Brusilovsky, A. Kobsa, and W. Nejdl, Eds., vol. 4321, Series Title: Lecture Notes in Computer Science, Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 325–341, ISBN: 978-3-540-72078-2. DOI: 10.1007/978-3-540-72079-

9_10. [Online]. Available: http://link.springer.com/10.1007/978-3-540-72079-9_10 (visited on 01/04/2024).

[45] Q. Guo, F. Zhuang, C. Qin, *et al.*, *A Survey on Knowledge Graph-Based Recommender Systems*, arXiv:2003.00911 [cs, stat], Feb. 2020. [Online]. Available: http://arxiv.org/abs/2003.00911 (visited on 02/17/2024).

[46] H. Zitouni, "On Solving Cold Start Problem in Recommender Systems Using Web of Data", in *2022 4th International Conference on Pattern Analysis and Intelligent Systems (PAIS)*, Oum El Bouaghi, Algeria: IEEE, Oct. 2022, pp. 1–8, ISBN: 9781665461610. DOI: 10.1109/PAIS56586.2022.9946899. [Online]. Available: https://ieeexplore.ieee.org/document/9946899/ (visited on 02/18/2024).

[47] S. A. Thorat, G. Ashwini, and M. Seema, "Survey on Collaborative and Content-based Recommendation Systems", in *2023 5th International Conference on Smart Systems and Inventive Technology (ICSSIT)*, Tirunelveli, India: IEEE, Jan. 2023, pp. 1541–1548, ISBN: 9781665474672. DOI: 10.1109/ICSSIT55814.2023.10061072. [Online]. Available: https://ieeexplore.ieee.org/document/10061072/ (visited on 03/24/2024).

[48] R. K. Rajasekaran and A. I. Iliev, "Hybrid Recommendation System", en, in *Advances in Information and Communication*, K. Arai, Ed., ser. Lecture Notes in Networks and Systems, Cham: Springer Nature Switzerland, 2023, pp. 1–11, ISBN: 978-3-031-28076-4. DOI: 10.1007/978-3-031-28076-4_1.

[49] P. K. Biswas and S. Liu, "A hybrid recommender system for recommending smartphones to prospective customers", en, *Expert Systems with Applications*, vol. 208, p. 118 058, Dec. 2022, ISSN: 09574174. DOI: 10.1016/j.eswa.2022.118058. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0957417422012659 (visited on 03/12/2024).

[50] K. Najmani, L. Ajallouda, E. H. Benlahmar, N. Sael, and A. Zellou, "Offline and Online Evaluation for Recommender Systems", en, in *2022 International Conference on Intelligent Systems and Computer Vision (ISCV)*, Fez, Morocco: IEEE, May 2022, pp. 1–5, ISBN: 978-1-66549-558-5. DOI: 10.1109/ISCV54655.2022.9806059. [Online]. Available: https://ieeexplore.ieee.org/document/9806059/ (visited on 01/04/2024).

[51] E. Zangerle and C. Bauer, "Evaluating Recommender Systems: Survey and Framework", en, *ACM Computing Surveys*, vol. 55, no. 8, pp. 1–38, Aug. 2023, ISSN: 0360-0300, 1557-7341. DOI: 10.1145/3556536. [Online]. Available: https://dl.acm.org/doi/10.1145/3556536 (visited on 01/04/2024).

[52] Y. Liu, "Research on Deep Learning-Based Algorithm and Model for Personalized Recommendation of Resources", en, vol. 4, no. 2, 2023.

[53] M. P. Shareef, L. Rose Jimson, and B. R. Jose, "Recommendation Systems : A Comparative Analysis of Classical and Deep Learning Approaches", in *2021 8th International Conference on Smart Computing and Communications (ICSCC)*, Jul. 2021, pp. 269–274. DOI: 10.1109/ICSCC51209.2021.9528193. [Online]. Available: https://ieeexplore.ieee.org/document/9528193 (visited on 01/04/2024).

[54] R. Boldi, A. Lokhandwala, E. Annatone, Y. Schechter, A. Lavrenenko, and C. Sigrist, *Improving Recommendation System Serendipity Through Lexicase Selection*, en, arXiv:2305.11044 [cs], May 2023. [Online]. Available: http://arxiv.org/abs/2305.11044 (visited on 01/04/2024).

[55] D. Kotkov, J. Veijalainen, and S. Wang, "Challenges of Serendipity in Recommender Systems:" in *Proceedings of the 12th International Conference on Web Information Systems and Technologies*, Rome, Italy: SCITEPRESS - Science, 2016, pp. 251–256, ISBN: 9789897581861. DOI: 10.5220/0005879802510256. [Online]. Available: http://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0005879802510256 (visited on 03/24/2024).

[56] M. Kaminskas and D. Bridge, "Diversity, Serendipity, Novelty, and Coverage: A Survey and Empirical Analysis of Beyond-Accuracy Objectives in Recommender Systems", en, *ACM Transactions on Interactive Intelligent Systems*, vol. 7, no. 1, pp. 1–42, Mar. 2017, ISSN: 2160-6455, 2160-6463. DOI: 10.1145/2926720. [Online]. Available: https://dl.acm.org/doi/10.1145/2926720 (visited on 01/10/2024).

[57] R. Kohavi, A. Deng, B. Frasca, T. Walker, Y. Xu, and N. Pohlmann, "Online controlled experiments at large scale", en, in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, Chicago Illinois USA: ACM, Aug. 2013, pp. 1168–1176, ISBN: 978-1-4503-2174-7. DOI: 10.1145/2487575.2488217. [Online]. Available: https://dl.acm.org/doi/10.1145/2487575.2488217 (visited on 01/10/2024).

[58] A. Gunawardana, G. Shani, and S. Yogev, "Evaluating Recommender Systems", en, in *Recommender Systems Handbook*, F. Ricci, L. Rokach, and B. Shapira, Eds., New York, NY: Springer US, 2022, pp. 547–601, ISBN: 978-1-07-162196-7 978-1-07-162197-4. DOI: 10.1007/978-1-0716-2197-4_15. [Online]. Available: https://link.springer.com/10.1007/978-1-0716-2197-4_15 (visited on 01/10/2024).

[59] H. A. Landsberger, *Hawthorne revisited: management and the worker : its critics, and developments in human relations in industry* (Cornell studies in industrial and labor relations), eng. Ithaca, N.Y.: Cornell University, 1958, OCLC: 491395.

[60] V. Veselovsky, M. H. Ribeiro, A. Arora, M. Josifoski, A. Anderson, and R. West, "Generating Faithful Synthetic Data with Large Language Models: A Case Study in Computational Social Science", 2023, Publisher: arXiv Version Number: 1. DOI:

10.48550/ARXIV.2305.15041. [Online]. Available: https://arxiv.org/abs/2305.15041 (visited on 01/10/2024).

[61] A. G. Møller, J. A. Dalsgaard, A. Pera, and L. M. Aiello, "Is a prompt and a few samples all you need? Using GPT-4 for data augmentation in low-resource classification tasks", 2023, Publisher: arXiv Version Number: 1. DOI: 10.48550/ARXIV.2304.13861. [Online]. Available: https://arxiv.org/abs/2304.13861 (visited on 01/10/2024).

[62] M. Josifoski, M. Sakota, M. Peyrard, and R. West, "Exploiting Asymmetry for Synthetic Training Data Generation: SynthIE and the Case of Information Extraction", 2023, Publisher: arXiv Version Number: 2. DOI: 10.48550/ARXIV.2303.04132. [Online]. Available: https://arxiv.org/abs/2303.04132 (visited on 01/10/2024).

[63] *English · spaCy Models Documentation*, en. [Online]. Available: https://spacy.io/models/en (visited on 02/18/2024).

[64] N. Poulton, *Docker and Kubernetes: The Big Picture*, en, 2019. [Online]. Available: https://www.pluralsight.com/courses/docker-kubernetes-big-picture (visited on 03/25/2024).

[65] J. A. Baktash and M. Dawodi, "Gpt-4: A Review on Advancements and Opportunities in Natural Language Processing", 2023. DOI: 10.48550/ARXIV.2305.03195. [Online]. Available: https://arxiv.org/abs/2305.03195 (visited on 02/18/2024).

[66] A. Othman, "Demystifying GPT and GPT-3: How they can support innovators to develop new digital accessibility solutions and assistive technologies?", en, *Nafath*, vol. 7, no. 22, Apr. 2023. DOI: 10.54455/MCN2204. [Online]. Available: https://nafath.mada.org.qa/nafath-article/MCN2204 (visited on 02/18/2024).

[67] S. Ekin, "Prompt Engineering For ChatGPT: A Quick Guide To Techniques, Tips, And Best Practices", preprint, May 2023. DOI: 10.36227/techrxiv.22683919.v2. [Online]. Available: https://www.techrxiv.org/doi/full/10.36227/techrxiv.22683919.v2 (visited on 02/18/2024).

[68] G. B. Davis, "Strategies for information requirements determination", *IBM systems journal*, vol. 21, no. 1, pp. 4–30, 1982.

[69] D. Leffingwell and D. Widrig, *Managing software requirements: a unified approach* (The Addison-Wesley object technology series). Reading, MA: Addison-Wesley, 2000, ISBN: 9780201615937.

[70] R. H. Thayer, M. Dorfman, and S. C. Bailin, Eds., *Software requirements engineering*, 2nd ed. Los Alamitos, Calif: IEEE Computer Society Press, 1997, ISBN: 9780818677380.

[71] H. T. Kanwal, F. Arif, and A. M. Zaidi, “"Software requirement engineering", a new leave towards the silver bullet”, in *2015 Science and Information Conference (SAI)*, London, United Kingdom: IEEE, Jul. 2015, pp. 189–198, ISBN: 9781479985470. DOI: 10.1109/SAI.2015.7237144. [Online]. Available: http://ieeexplore.ieee.org/document/7237144/ (visited on 02/29/2024).

[72] S. Saroja and S. Haseena, “Functional and Non-Functional Requirements in Agile Software Development”, en, in *Agile Software Development*, S. Hooda, V. M. Sood, Y. Singh, S. Dalal, and M. Sood, Eds., 1st ed., Wiley, Feb. 2023, pp. 71–86, ISBN: 9781119896395 9781119896838. DOI: 10.1002/9781119896838.ch5. [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1002/9781119896838.ch5 (visited on 02/29/2024).

[73] A. R. Djamarullah and W. A. Kusuma, “Elicitation of Needs Using User Personas to Improve Software User Experience”, *Ultimatics : Jurnal Teknik Informatika*, pp. 28–35, Jul. 2022, ISSN: 2581-186X, 2085-4552. DOI: 10.31937/ti.v14i1.2633. [Online]. Available: https://ejournals.umn.ac.id/index.php/TI/article/view/2633 (visited on 03/14/2024).

[74] E. Stern, *Evaluation Research Methods*. London, 2005. DOI: 10.4135/9781446261606. [Online]. Available: https://sk.sagepub.com/navigator/evaluation-research-methods (visited on 01/10/2024).

[75] L. Peska and P. Vojtas, “Off-line vs. On-line Evaluation of Recommender Systems in Small E-commerce”, en, in *Proceedings of the 31st ACM Conference on Hypertext and Social Media*, Virtual Event USA: ACM, Jul. 2020, pp. 291–300, ISBN: 9781450370981. DOI: 10.1145/3372923.3404781. [Online]. Available: https://dl.acm.org/doi/10.1145/3372923.3404781 (visited on 03/24/2024).

Assertion

I sincerely affirm to have composed this thesis work autonomously, to have indicated completely and accurately all aids and sources used and to have marked anything taken from other works, with or without changes. Furthermore, I affirm to have observed the constitution of the KIT for the safeguarding of good scientific practice, as amended.

Karlsruhe, July 4, 2024 Enzo Hilzinger