Extracting Entity Relationship Diagrams from Unstructured Text - A Hybrid Approach

Vaihunthan Vyramuthu
Department of Electronics and Computer Science
Aalen University
Aalen, Germany
email: vaihunthan@gmail.com

Gregor Grambow
Department of Electronics and Computer Science
Aalen University
Aalen, Germany
email: Gregor.Grambow@hs-aalen.de

***Abstract*—Data models are a crucial part of computer science applications that automate real-world processes. An important model for abstractly capturing data sets and their relations is the Entity-Relationship (ER) model. To create these models, real-world entities and their relations have to be abstracted. However, the transformation from real world descriptions in natural language to standardized ER diagrams can be tedious and error-prone. Recently, Natural Language Processing (NLP) has gained much attention, but this specific area is still mostly handled manually by humans. With this contribution, we aim at automating the time-consuming interpretation of textual database scenarios by describing a hybrid system to capture components of the ER model from German texts using NLP. This hybrid system combines rule-based and model-based approaches. This may lead to a more robust and reliable extraction, as errors in one of the approaches can be compensated by the other. We implemented and tested both approaches, whereas the main extraction is performed by the rule-based variant. The results of the model-based approach are used as a comparison to the rule-based results and can be applied to check the correctness and improve the results. We conducted an evaluation using a set of real-world scenarios describing various entities and their relationships and applied a set of metrics suited to the imbalanced nature of the dataset. Our evaluation shows promising results indicating that a hybrid approach can deliver better results than classical approaches.**

***Index Terms*—Entity-Relationship Model; Natural Language Processing; Named Entity Recognition; POS-Tagging; SpaCy; LSTM; Transformer; Imbalanced Data**

I. Introduction

This publication is an extended version of our previous work from 2025 [1]. In today's data-driven world, large amounts of text are generated that may contain valuable information. Examples of this include police reports from online press portals, social media posts or Amazon reviews. Various analyses are possible that can provide insights about future trends, user ratings and sentiment or other questions [2]. Extracting structured data from unstructured text, however, is a complex task that mostly requires manual processing. Automating this process can save time and resources and improve the efficiency of data analysis [3] [4].

At the same time, the use of database systems for storing and managing data is prevalent. A common approach to database modeling is the use of *Entity Relationship Models* (ERM), followed by conversion to a relational model and finally to *Structured Query Language* (SQL) statements to represent real world data in a database [5].

Fig. 1. Overview from plain text to ERM.

Figure 1 presents a transformation of unstructured text into an ERM. The process includes the extraction of relevant entities and relationships from textual requirements, which are then formalized into an ERM. This conceptual model is subsequently translated into a relational schema, preserving the semantic integrity of the original requirements. Finally, the relational schema is implemented using SQL statements, resulting in a database schema. This workflow exemplifies the systematic progression from human-readable specifications to machine-executable data definitions. The primary focus of this publication is the transition from text to ERM generation, as it

represents the most complex and interpretation-sensitive step in the workflow.

Transforming textual requirements into an ERM serves several important purposes in the database design process. First, it enables a clearer and more intuitive understanding of the conceptual structure of the data. Second, working at the ER level allows the early identification and elimination of redundant entities or relationships. The ultimate goal is to minimize the number of relational tables, which in turn reduces the need for complex and time-consuming JOIN operations during query execution. Thus, the ERM plays a critical role in optimizing both the design quality and performance of the resulting relational database.

Figure 2 and Figure 3 present extracts of an ER schema for an university which governs how professors, students, lectures, and other entities should be connected. The comparison of Figure 2 and Figure 3 highlights the impact of cardinalities in ER diagrams on the compactness of the resulting relational models. In Figure 2, a many-to-many (n:m) relationship exists between students and lectures, which necessitates the introduction of a separate associative table. This results in a minimum of three relational tables, as the relation "hören" requires a composite primary key consisting of both *student-ID* and *lecture-ID*. In contrast, Figure 3 illustrates a one-to-many relationship between professors and lectures. The "lesen" relationship does not require a separate table, as it can be merged into the *lecture* table, which already uses *lecture-ID* as its primary key. This allows for a more compact representation using only two tables. Clearly defined cardinalities thus support schema simplification by enabling merging of relationships into entities.

Fig. 2. Relationship with minimum 3 tables.

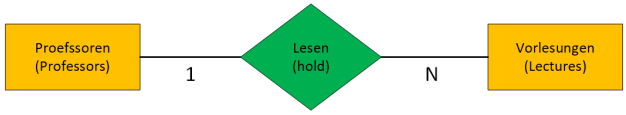

Fig. 3. Relationship with minimum 2 tables.

Manual data extraction involves reading the text and interpreting it correctly. The aim of this paper is to automate data extraction and thereby simplify the process of database modeling and integration. The results of this process are considered successful if the discrepancy between the human and software interpretation of a text with regard to correct ERM generation is as small as possible. This paper proposes a hybrid approach for the identification of entities and attributes, as well as the recording of entity relationships (including cardinalities). A hybrid approach may lead to increased error resistance. Rule-based systems alone are often prone to inaccurate results if the texts do not exactly match the expected patterns. The model-based approach compensates for this with its ability to learn from contexts and recognize variations. The analysis of semantic contexts and complex text structures that go beyond the recording of entities, attributes, and relationships is not part of this work. The implementation of the approach focuses on the processing of texts in German language.

The paper is organized as follows: Section II presents related work, followed by Section III, which contains details on the concept and architecture of the proposed approach. The implementation is presented in Section IV. Section V discusses limitations of the approach, while Section VI presents the evaluation. Finally, the conclusion can be found in Section VII.

II. Related Work

In literature, there exist several approaches that deal with the creation of ER models from natural language. Ghosh et al. [6] propose a method that uses grammatical knowledge patterns as well as lexical and syntactic analyses of request texts to create ERMs. This system assumes that the input text consists only of simple subject-predicate-object sentences for correct information extraction. This sentence structure makes it possible to detect entities (subject), relationships (verb) and attributes or other related entities (object). The approach applies NLP techniques, such as sentence segmentation, word separation (tokenization) or *POS-Tagging*. After that a domain-specific database is used to identify ERM components.

The segmentation in this work is carried out in a way that cases in English, such as "Mr. Mustermann", in which (.) appears after Mr, are not recognized as the end of the sentence. In *POS-Tagging*, the individual words are assigned to the corresponding word types, e.g., noun, verb, pronoun, adjective, preposition etc. The recognition of entities, attributes and relationships is performed using a database and the *Support Vector Machine* (SVM) classifier.

Six tables (word & synonym for each ERM component) are implemented, which contain domain-specific words and synonyms. To capture related entities, synonyms of a word from the table are given the same ID. This prevents redundant SQL tables from being created. Pronouns such as "he", "she" or "it" are already recognized in the *POS-Tagging* phase. In this phase, the pronoun is identified based on the closest previous entity that is present either in the same sentence or in the previous sentences. The technical term for this is *coreference resolution* [7].

In the publication by Kashmira et al. [8], the ERM components are recorded using neural networks. Three main modules are presented, namely the pre-processing module, the machine learning module, and the ER modeling module. In the first step, the pre-processing module is implemented using *NLTK* to pre-process the text. This includes steps such as converting the text into lowercase letters, tokenization into sentences, etc. The machine learning module is implemented using supervised learning. This module is trained with an English dataset where words are categorized into different categories including entity,

sub-entity, attribute, and irrelevant category. Four classifiers are taken into account when training the model: *Random Forest*, *Naive Bayes*, *Decision Table*, and *Sequential minimal optimization* (SMO).

To address the problem of attribute selection for entities in an ER diagram, the proposed model uses a combination of *ontology* and *web mining*. By using an ontology, an attempt is made to filter relevant attributes from the extracted entities. In addition, web mining is used to obtain further information that can be helpful in determining the attributes.

The publication by Habib [9] also follows similar preprocessing steps at the beginning. After the parsing process, the grammatical sentence structure is obtained so that the components of the ERM can be determined based on rules. The words are converted into a parse tree structure to understand how the individual parts of the sentence are related to each other. Using appropriate rules for sentence structures, entities, attributes, cardinalities and relationships can be determined.

There are several other publications that go in a similar direction and examine the topic of automatic ERM generation in the context of NLP in more detail. An example by Omar et al. [10] describes heuristic-based analysis options for ERM generation. In contrast, Omar and Abdulla [11] pursue the approach of training a machine learning model that can extract the entities from the text. Depending on the complexity of the input text and the scope of training, the model achieves precision values of up to 85%. The approach of Btoush and Hammad [12] can also be placed in a similar context. Here, a method is presented which, like [9], defines and applies certain rules for extracting information from texts.

The paper by Li et al. [13] proposed a hybrid approach that combines classical classifiers with large language models for relation extraction tasks. In their approach, a classifier is first used to identify potentially relevant entity pairs before the final relation prediction is performed by an LLM. Such combinations can improve the overall extraction quality and reduce unnecessary processing of unrelated entities. However, despite the hybrid architecture, these approaches still mainly rely on neural components and therefore remain largely non-transparent due to the black-box characteristics of LLMs and deep learning models.

Banjac et al. [14] presented *OmniaDB*, a web-based prototype for the automatic derivation of conceptual database models (CDMs) from heterogeneous source artifacts. Their approach combines the outputs of multiple existing tools that derive CDMs from different types of sources and integrates the resulting models into a unified representation. A major challenge addressed by their work is the integration of automatically generated and potentially unreliable CDMs. Initial results indicate that the integrated model can achieve higher precision and completeness compared to models derived from a single source type alone. Although their work does not directly address the extraction of ER components from unstructured natural language text, it nevertheless demonstrates the broader relevance of automated conceptual database design and highlights the importance of combining complementary extraction approaches.

It can be stated that two basic methodological approaches are used for ERM generation. The ERM components are determined either rule-based or using neural networks or artificial intelligence. Both approaches have advantages and disadvantages. The biggest advantage of rule-based extraction is the more time-efficient implementation as, unlike neural networks, no data pre-processing and training is required. Furthermore, the unambiguous definition of the rules ensures high extraction probability. In neural networks, a residual inaccuracy always remains. In contrast, model-based extraction is not limited to a few rules, but can learn complex and nested sentence structures to determine the ERM components. These sentence structures can contain linguistic variations and ambiguities, which can be recognized more easily by neural networks than by fixed heuristics. The heuristic approach is more suitable for small application areas and cannot maintain its effectiveness in large application areas.

Purely rule-based methods are prone to lack of generalization, while purely model-based approaches often depend on large training datasets and have difficulties in capturing rare or complex linguistic structures. This hybrid approach has the potential to overcome these limitations by combining the strengths of both methods: The rule-based method enables accurate extraction, while the model-based method helps to validate and improve the results, resulting in a more robust and adaptable solution to different text scenarios.

Existing rule-based approaches often focus only on selected elements, such as entities and relationships, while other important aspects are frequently omitted. In contrast, our approach systematically considers all core components of an ER model, including entities, relationships, attributes, and cardinalities. In particular, explicit handling of cardinalities distinguishes our approach from many existing rule-based methods. We conducted an additional literature search, but were unable to identify closely related work that applies such a hybrid strategy in the specific context addressed by our paper.

III. Overview of the architecture

Figure 4 shows a general overview of the individual processes of the proposed approach. The latter is divided into a rule-based and a model-based part. The model-based algorithm can be used either to train new or existing models or to extract the ER components from a text. However, these results are not directly used for the resulting ERM, but can be applied to evaluate the rule-based results. This makes it possible to check whether the final result from the rule-based process may still need to be modified manually.

The input for both parts of the approach is a text that is saved in a *.txt* file. The *output.json* file contains all ER components and relationships found in a structure that can be read by an external ER modeling tool. The artifacts (in the figure: ordinary rectangles or arrow labels) represent the created files or results. These files are required for the subsequent processes.

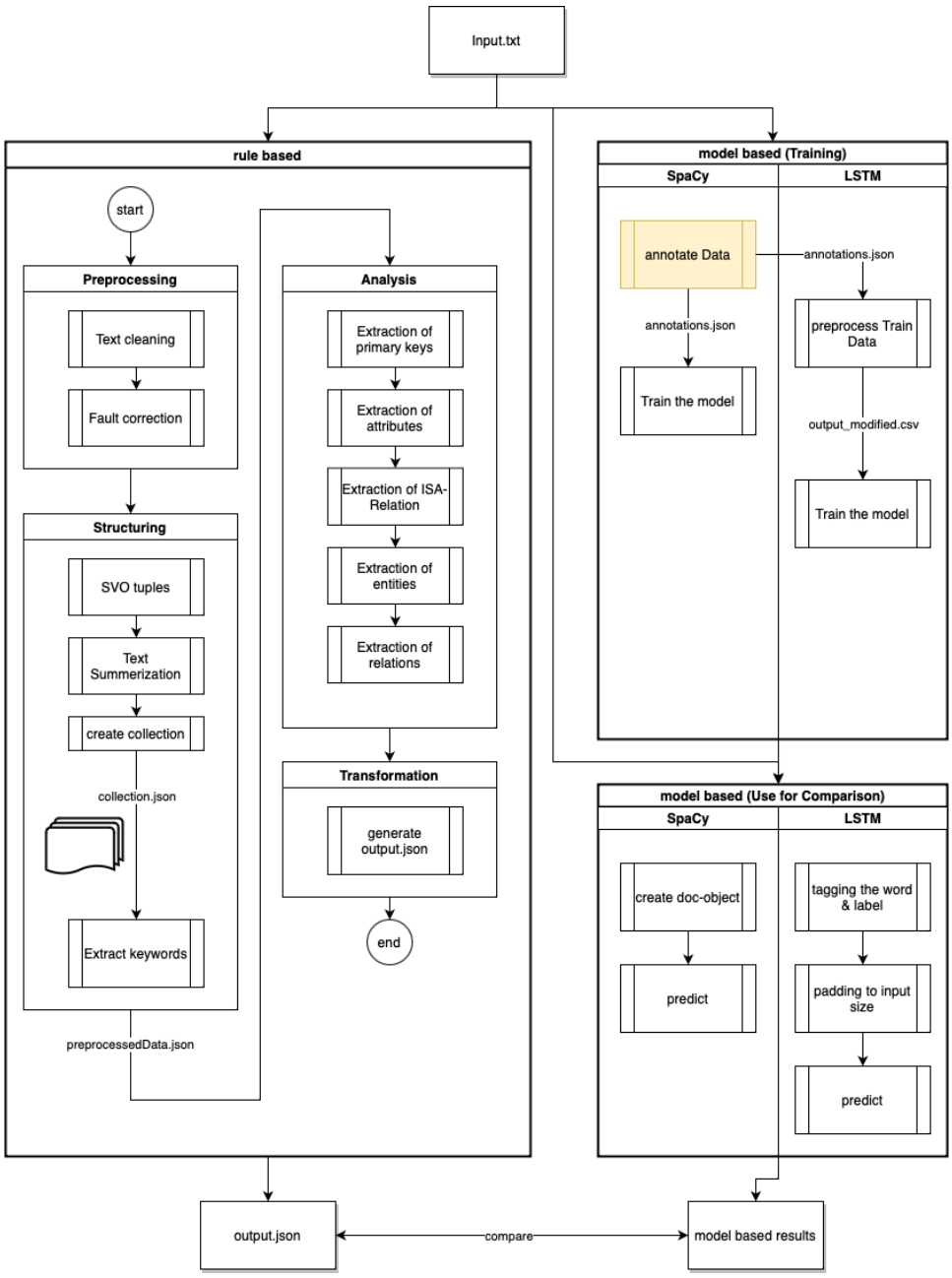

Fig. 4. Overall flow chart of the implementation.

The rule-based approach is made up of the sub-steps pre-processing, structuring, analysis and transformation. During pre-processing, the text data is cleansed and corrected for spelling errors. The structuring phase contains processes that break down the text into more meaningful parts and store them temporarily. In the main block of the analysis, the ER components are extracted one after the other.

In the model-based part, a different process path is carried out depending on the selected model type (*SpaCy*-Transformer [15] or *LSTM*). Due to the individual input requirements of the models, the text data must be converted into the permitted form for both the training case and the use case. The process block highlighted in yellow indicates an annotation process that is performed using an external tool. For the *LSTM* model, the output of the annotation tool must be pre-processed again into a *.csv* file so that it can be used for training. After the training processes, the model can be applied to new text data. For the *SpaCy* model, it is sufficient to convert the text into a doc object. The text for the *LSTM* model, on the other hand, requires additional processing (analogous to the training process), which decodes the words and labels into numbers and scales the input word vector to a specified size (so-called *padding*). This hybrid combination provides an important advantage for continuous optimization. The results

of the model-based approach can be used to dynamically adapt and further develop the rules of the rule-based system. This means that the hybrid system can become increasingly precise and effective over time through feedback and new data sets.

The selection of *SpaCy* as a model and NLP-Tool is based on its robust capabilities and user-friendly implementation. SpaCy is recognized for its comprehensive documentation and strong support from an active developer community. *LSTM* were chosen for their effectiveness in handling sequential data and their ability to capture contextual dependencies over extended text passages. This makes them particularly suitable for tasks that require understanding the relationships within long text sequences of information.

LSTM networks are a specialized form of Recurrent Neural Networks (*RNN*) designed to address the vanishing gradient problem. Traditional *RNN* struggle to capture long-range dependencies in sequences because gradients tend to vanish during backpropagation over time, making learning ineffective for long texts [16].

LSTM introduce a gating mechanism—comprising an input gate, forget gate, and output gate that allows the model to selectively retain or discard information over time [17]. This structure enables *LSTM* to capture dependencies across long sequences, which is particularly beneficial for longer sentences or documents with complex syntax. Unlike *RNN* or *LSTM*, Transformers like *SpaCy* eliminate recurrence entirely and instead rely on a self-attention mechanism, which allows the model to weigh the relevance of all other words in the sequence simultaneously [18].

IV. Implementation

Starting with the rule-based part, it can be noted that it is important not to define too many special rules. Otherwise, the implementation will be too application-specific and errors for other text styles will sometimes occur. The first process in data pre-processing is text cleansing. This process is divided into three steps. Each result of the individual pre-processing and structuring steps is saved in the *preprocessedData.json*. As starting point, the unstructured text is filtered from the *.txt* file.

In this first step, existing white-spaces or empty lines are also taken into account. In the second step of text cleansing, individual sentences from the text are found and saved using the *SpaCy* model (*de_core_news_sm*). In the last step of the text cleansing process, unnecessary sentences that do not provide necessary information for the ER diagram design are removed. In this process, sentences are removed using *RegEx* matches of certain words, such as “database” or “modeling”, are sorted out.

The next process in pre-processing is the error correction of words. The *Levenshtein-similarity* is used here. The Python library *pyspellchecker* checks whether there is an error for each word in a sentence. If this is the case, the word is replaced by the closest one.

In the structuring task block, the main focus is on capturing the subject-verb-object (SVO) sentence structure and some important term frequency analyses, with the help of which the text can be broken down into smaller information-rich parts. While only the *normalized word frequency* (TF) is used for the text summary, the *inverted document frequency* (TF-IDF) helps to capture the most significant keywords from the text.

The TF is used to calculate the frequency of a specific term in a text document. TF measures how often a term appears in a document. The calculation of Term Frequency can be performed in various ways, with the most common method being the counting of the number of occurrences of the term in the text. The formula for the calculation is given by [19]:

$$TF_n(t,d) = \frac{TF_a(t,d)}{max_{t_i \in d}(TF_a(t_i,d))}$$

- n relative frequency
- a absolute frequency
- t term
- d document

The numerator represents the number of occurrences of the term t in document d, and the denominator describes the maximum frequency of any term in d. This normalization prevents distortion in large documents by adjusting the absolute frequency. TF yields a value in the range of 0 to 1, where a value close to 1 indicates a high frequency of the term in the document.

The Document Frequency (DF), on the other hand, calculates the frequency of a term within a corpus of text documents. DF measures how many documents contain a particular term and allows assessing the relevance of the term across the entire collection of documents. For example, this is used to identify stop words, which occur in many documents and therefore are of little significance for identifying key terms. The value of DF also ranges between 0 and 1.

The combination of TF and DF leads to the TF-IDF method (Term Frequency-Inverse Document Frequency), which is calculated by the following formulas [19]:

$$IDF(t) = \log \frac{N}{|d : t \in d| + 1}$$

$$TF\_IDF(t,d) = TF_n(t,d) \times IDF(t)$$

- N total number of documents
- t term
- d document
- n relative frequency

The denominator of $IDF(t)$ represents the number of documents that contain the term t. Thus, the TF-IDF method is used to evaluate the relevance of a term in a document within a corpus.

Figure 5 illustrates Zipf’s law. This law states that the frequency of a word is inversely proportional to its rank. In other words, the more frequently a word appears, the lower its rank, and vice versa. Luhn [20] proposed a similar idea. In his approach, the distribution of important words in

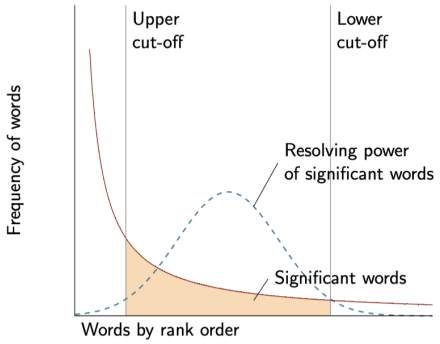

Fig. 5. Relationship between word frequency and rank [19].

documents is bounded by two thresholds that define the range of significance. Words outside these thresholds are either stop words (upper cut-off) or rare proper.

In contrast to SVO generation, these two structuring steps are only used optionally. These results are not used actively in the NLP workflow because it is possible that relevant information may be lost. Another use case for these results is to look for the most essential keywords in the text in order to compare them with the entities and attributes found. It should also be noted that to calculate the TF-IDF for keyword extraction, a document corpus (*collection.json*) must be created in which all existing ER diagram sample texts are stored. The *TfidfVectorizer()* function provided by the *Scikit-learn* library calculates the TF-IDF. No numbers are included in the keywords.

The text summary is implemented chronologically according to the following key points.

- count the number per word in the text (stop words excluded),
- calculate the normalized weight per word used by dividing the respective word count by the maximum word frequency occurring,
- calculate the sentence weight by adding the weight per word,
- search for sentences with the highest weighting.

Individual sentences are scored by adding the normalized TF for each word in the sentence. Depending on the original length of the text, a certain number of sentences is selected in descending order of the evaluation number. For this purpose, a corresponding factor is determined at the beginning with which the number of sentences is calculated. If there are fewer than three sentences in the text, the text is not summarized.

To generate SVO tuples, the sentence must be analyzed using *dependency parsing* and *POS-Tagging* provided in *SpaCy*. Certain commands can be used to analyse the grammatical structure of sentences so that the visualization shown in Figure 6 is displayed. Similarly to the tree structure, the successors or predecessors of a word are addressed with "children" or "parent".

Based on this structure, a generally valid logic can be developed for the extraction of subject, verb, and object, which

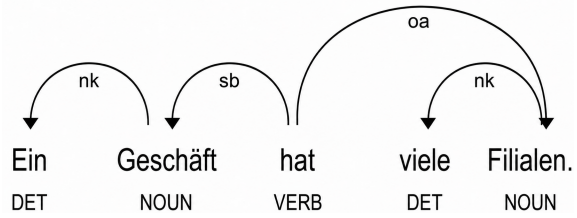

Fig. 6. Visualization of the grammatical sentence structure.

is shown in the structure diagram in Figure 7. The sentences in the text are entered individually into the function. There may be several verbs in each sentence, but each verb must belong to exactly one subject and object. *Lemmatization* can be used for the output of recorded relationships in texts. This involves changing the verb from its inflected form back to its basic form. For example, the German word "angeboten" is "anbieten" after lemmatization.

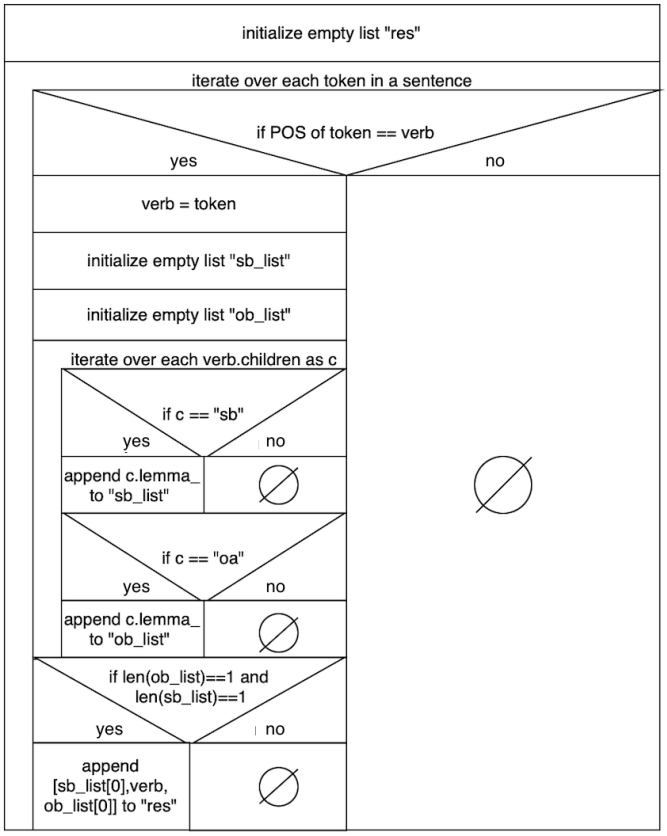

Fig. 7. Structogram for SVO extraction from a record.

A. Primary Key

Although this paper focuses on applications for German texts, English terminology is used throughout this work for reasons of readability and consistency.

The primary keys can be found using a *RegEx* comparison. The words "-id" or "-number" indicate a key candidate (see Figure 8).

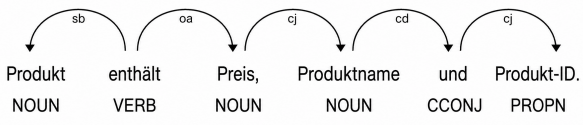

Fig. 8. Example sentence for extracting primary keys

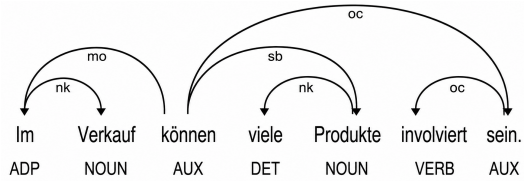

Fig. 9. Example sentence for the extraction of relationships

The words are first converted to lower case so that there is a certain amount of leeway in the comparison. However, the limitation is the hyphen, which must be contained in a primary keyword. The following rules are implemented as Python code:

- Determine all nouns and filter primary key via *RegEx*,
- If the record only contains the word “unique”, then the closest noun should be the primary key with “noun ID”,
- If only the words “ID” or “id” appear, then the nearest noun should be the primary key with “noun ID”.

B. Attribute

Attributes are identified as soon as a sentence contains a list of more than two nouns. The first noun that occurs in the sentence is the entity to which the remaining nouns or attributes belong. This simple rule serves as a first step in the identification process, but can be refined and enhanced through model-based results in the future.

C. ISA-Inheritance

To detect the specialization or generalization of entities, the following rules are followed:

- If a sentence contains the following verbs: [“include”, “consist”, “comprise”, “share”, “include”], then the sentence describes a generalization,
- If the sentence contains “type” as a word (noun), the first noun or entity is the generalization of the following nouns (entities)

It should be noted that the recognition of ISA relationships using rule-based approaches is limited. For example, the order of membership may be different for the entities, or the ISA description may extend over several sentences.

D. Entity

In the *preprocessed.json* under the item in which the SVO tuples are stored, the subject and object in each record can be either an entity or an attribute. These tuple words are therefore compared again with the attributes and primary keys found so far. If the same tuple word is also contained in the list of attributes or primary keys, it is discarded. The final result is a list that contains only the final entities.

E. Relationship

The SVO tuples can again be used for the relationships. If both the subject and the object are contained in the list of final entities, the verb is a relationship between two entities. There are sentences that are not formulated in an ordinary SVO style, but which define further relations between entities. Figure 9 shows an example sentence.

In order to take such sentences into account as well, a check is made to see whether two nouns occur in a sentence in addition to a verb, which are not contained in the attribute, ISA-Inheritance and primary key list. If this is the case, a relationship tuple can be extracted from this sentence again if it has not already been extracted from the SVO tuple.

F. Cardinality

Two min/max cardinalities must be determined for each relationship between two entities. The sentence in Figure 6 shows that the cardinalities can be taken from the determiners of the nouns. Once the SVO tuples have been reassigned to complete sentences with the help of indexing, the corresponding determiners of each entity can be determined. These are then translated into the corresponding min/max value using a comparison. For the correct min/max notation, the cardinalities of the entities in an SVO tuple must be swapped.

When interpreting the adverbs “at most” and “at least”, the following word must also be taken into account, as this defines either the upper (max) or lower limit (min). An example sentence is illustrated in Figure 10.

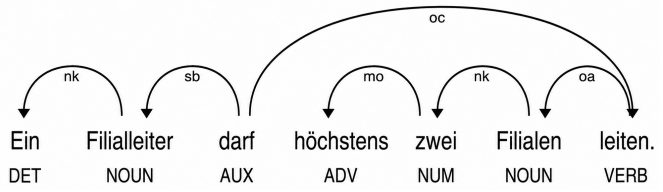

Fig. 10. Example sentence with the adverb “höchstens”

Due to the unlimited possibilities for translating this adverb, no rule-based application is suitable for this. It would make the rules too specialized for one use case.

G. LSTM Architecture

The model-based approach primarily serves as a comparison tool for the ER components found in the rule-based algorithm. Therefore, the hybrid approach is also advantageous for performance reasons. The rule-based methods often deliver faster results as they do not rely on extensive calculations, while the model-based part intervenes where more in-depth analyses are required. This efficient applicability ensures that the goal can be achieved faster without losing accuracy. In the following, only the architecture of the *LSTM* model is examined in detail, as the Transformer model is provided as a pre-trained component by *SpaCy*.

The model architecture, designed using the *Tensorflow Keras* library [21] and the *Sequential API*, is shown in Figure 11. In Python, this is implemented as depicted in Listing 1:

Listing 1. Model architecture code

```
model = keras.Sequential()
model.add(InputLayer((max_len)))
model.add(Embedding(
    input_dim=num_words,
    output_dim=max_len,
    input_length=max_len
))
model.add(SpatialDropout1D(0.1))
model.add(Bidirectional(
    LSTM(
        units=100,
        return_sequences=True,
        recurrent_dropout=0.1
    )
))
```

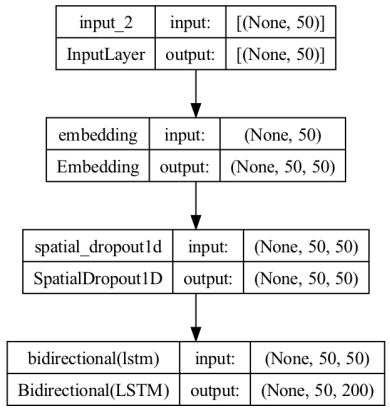

Fig. 11. Architecture of the designed LSTM model

The architecture consists of five layers. The output layer is not depicted in Figure 11. In addition to the input layer, the model includes an embedding layer, a dropout layer, and a bidirectional *LSTM*. The dropout layer prevents overfitting during training by randomly deactivating certain weights in the layer. When the model is applied to test data, the dropout layer is ignored.

The use of a bidirectional *LSTM* allows the model to better capture contextual information from both directions of the input text during training. Furthermore, this increases the number of trainable parameters, which helps reduce the bias error (also referred to as training loss). However, it is important to note that the model should not be overly complex, as this could again lead to overfitting.

The input layer has a size of 88 neurons, corresponding to the number of words. Multiplying this value by the input size of the embedding layer (50 neurons) results in a total of 4400 trainable parameters for this layer.

In NLP, words and phrases are often represented as one-dimensional vectors, where each dimension corresponds to a word in the vocabulary. This method is referred to as *one-hot encoding*. These vectors are high-dimensional, making them inefficient and difficult to work with. Instead of using these sparse high-dimensional vectors, the embedding layer maps each word to a lower-dimensional vector, where each dimension captures specific features of the word [22].

The dropout layer contains no trainable parameters. In contrast, the bidirectional *LSTM* has 120800 trainable parameters. This value is derived as follows:

$$p = 2 \times 4 \times (h \times (h + i) + h) = 120800$$

p	Number of trainable parameters
2	Factor due to bidirectionality
4	Four neural networks (gates) in LSTM
h	100 (number of LSTM cells)
i	50 (embedding dimension) + 100 (recurrent input) = 150 (input dimension)

A fixed input dimension is defined for the input layer. Based on the distribution of average sentence lengths (see Figure 12), a value of 50 is selected as the maximum sentence length or input dimension. The boxplot displayed above the histogram provides an additional statistical summary of the distribution. The central line inside the box represents the median sentence length, while the box itself corresponds to the interquartile range (IQR), containing the middle 50% of all sentence lengths. The whiskers indicate the typical spread of the data outside the quartiles, and the isolated point on the left side represents an outlier. Overall, the boxplot confirms that the sentence lengths are relatively concentrated around the median.

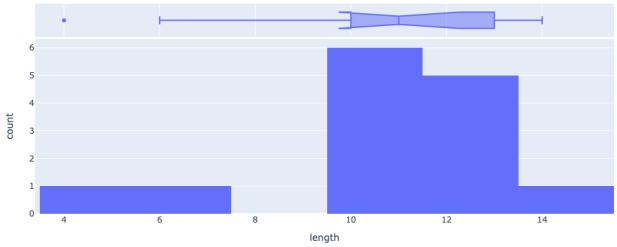

Fig. 12. Distribution of sentence lengths in the small dataset used for testing. The median is at length 11.

This means that the word vector of each sentence must be adjusted to this fixed size. After the tagging process, the words are decoded as numerical values. Due to varying sentence lengths, the sequences need to be expanded to the uniform input size of 50 by padding. These processes are illustrated in Figure 4, both in the model based (training) block and in the model based (use for comparison) block.

For updating the weight values, the Gradient Descent algorithm is used in combination with the *Sparse Categorical Crossentropy* loss function, as this function internally handles one-hot encoding. Moreover, this loss function is particularly suitable for classification problems because, compared to

Mean Squared Error, it enables a significantly faster adjustment of the weight parameters.

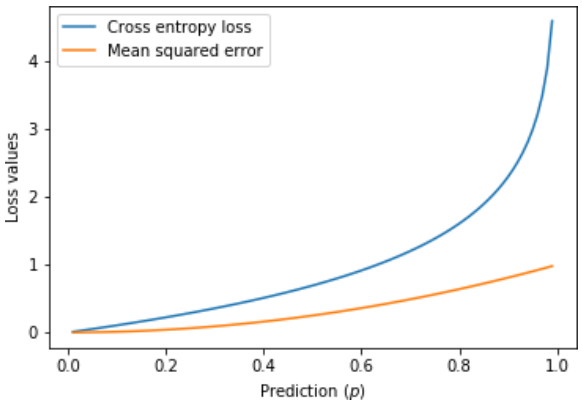

Fig. 13. Comparison of the loss functions Mean Squared Error and Cross Entropy. The Ground Truth is at 0. [23]

The reason lies in the characteristics of the *Categorical Cross Entropy* function: due to the logarithmic transformation of values between 0 and 1, errors result in substantially higher loss values than with the same input using *Mean Squared Error* (see Figure 13). As a result, the *Categorical Cross Entropy* function enables larger step sizes in the gradient descent process, thereby accelerating the training.

V. Limitations

The following aspects were not taken into account in the rule-based algorithm:

- Attributes for relationships,
- special cardinalities ("two", "three", etc.),
- weak entities, relationships, attributes, etc.,
- described ISA-Inheritance across several sentences,
- multi-valued or complex attributes.

In the future, the results of the model-based approach can be used to include more ER components in the results. An extension to the rule-based approach for this would be rather difficult, as a generally valid formulation of the rules is difficult and often tied to a specific use case. Moreover, limitations such as coreference resolution and those listed above can be better addressed by model-based approaches. Current coreference resolution tools are primarily optimized for English, while there is still a lack of robust solutions for German texts. Accurately interpreting the meaning of sentences presents a significant challenge, as words and phrases can change meaning depending on context. Ambiguity and polysemy further complicate the precise analysis of text. For example, the German sentence

"Er sieht den Verkäufer mit dem Fernglas"

("He sees the salesman with the binoculars") is structurally ambiguous, as it is unclear whether the binoculars are associated with the observer's action or with the salesman as an attribute. Furthermore, words often need to be correctly recognized and interpreted despite spelling errors. Techniques such as the Levenshtein distance can be applied in this context.

TABLE I
Word-level challenges for NLP

Word Type	Meaning	Examples
Homonyms	same spelling; different meaning	Schale, Bank
Homophones	same pronunciation; different meaning	mahlen/malen, leeren/lehren
Homographs	same spelling; different pronunciation	Montage (Pl. Montag / Zusammenbau)
Synonyms	different words for the same concept	essen, speisen, konsumieren
Compounds	composition of multiple words	Kapitänsmütze
Abbreviations	shortened forms of words	u.s.w.
Proper Nouns	individual names	Angela Merkel, Apple corp.
Inflections	grammatical adaptation of words	ich singe, ich sang
Anglicisms	words borrowed from English	surfen, boostern

Another major challenge in natural language processing is the handling of lexical ambiguities. Polysemy refers to words that have multiple meanings. In contrast, homonymy describes different words that share a similar spelling or pronunciation but have different meanings. Correctly understanding and assigning such ambiguities in a given context can be very challenging. Table I illustrates various types of words that can cause difficulties in NLP-based interpretation for German language.

This highlights the need for further research and development to improve the processing of German-language input in this context.

VI. Evaluation

The evaluation is based on several German texts that describe specific DB scenarios. This means that they contain specific formulations that describe the ER components. The rule-based algorithm only extracts ER components that correspond to the defined grammatical regularities. In contrast to the model-based approach, emphasis is placed on a qualitatively correct ER extraction instead of a quantitative result set.

Fig. 14. Rule-based extraction of ER components from a German database scenario text.

Figure 14 shows the output of the rule-based approach applied to a German-language text describing a specific database scenario. The extracted ER components—entities, relationships, attributes, cardinalities, and keys—are highlighted based on their semantic roles. The text is rich in domain-specific formulations, making it well-suited to evaluate ER extraction. The visualization also highlights that many relevant ER

components are already effectively captured by the rule-based method, indicating that the underlying rules are well-defined and cover a broad range of relevant grammatical structures.

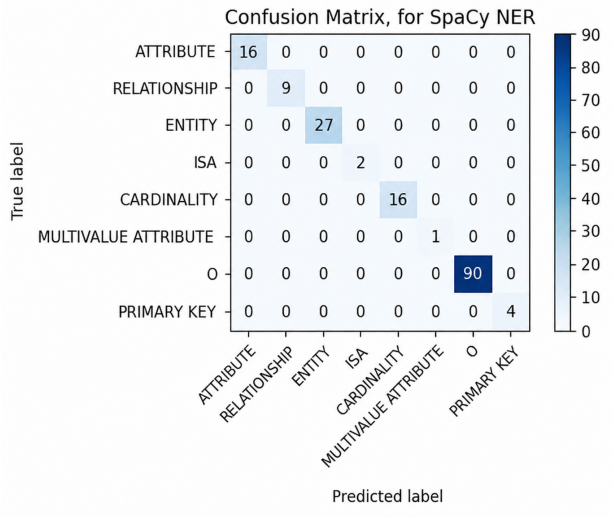

Fig. 15. Ideal reclassification result in the case of training and testing with the same data.

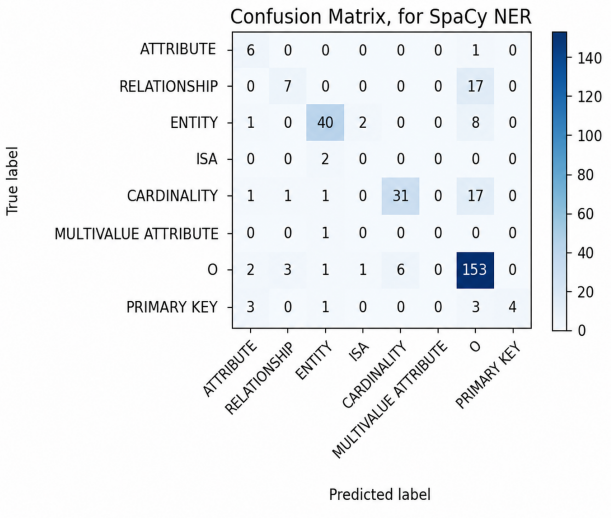

Fig. 16. Results of Crossvalidation with larger and different training and test data sets.

The Confusion Matrix in Figure 15 shows the reclassification case in which all previously labeled ER components are correctly classified during testing. However, the Confusion Matrix in Figure 16 provides more information about the generalization capability of the *SpaCy* model, because the cross-validation also examines text data that the model does not yet know and was therefore not part of the training process.

As shown in Figure 17, one of the main reasons why the metrics stagnate in this experiment is the small amount of training data. With only 77 training sentences, the model does not see enough examples to learn robust patterns, which limits its ability to improve beyond a certain point. Furthermore,

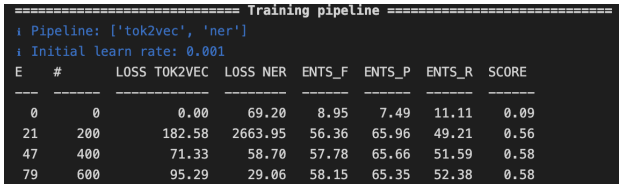

Fig. 17. Model performance with limited training data.

there is a class imbalance problem, where some entity types occur far less frequently than others. This makes it harder for the model to generalize and leads to uneven performance across different labels. As a result, both precision and recall remain stuck at moderate levels instead of improving over time.

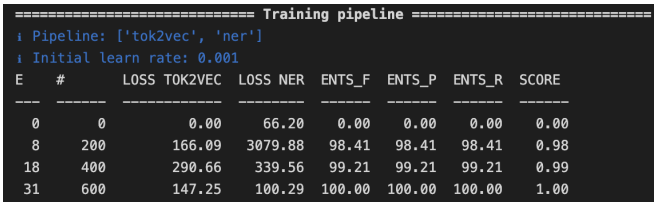

Fig. 18. Model performance with extended training data.

In Figure 18, it can be observed that the *SpaCy* model reached very high precision and recall values after only 8 epochs, both around 98%. This shows that the model was able to generalize effectively and recognize entities with high accuracy at an early stage of training. The model was trained with an extended dataset of 206 training sentences and evaluated on 77 test sentences. Such a strong performance after a relatively small number of epochs underscores the quality of the dataset, while the effect of a potential imbalance in the data set played only a minor role.

Similarly to the *SpaCy* model, the *LSTM* model is also evaluated using the learning curves shown in Figure 19. After 10 training epochs, a solid validation accuracy of 92.1% and a validation error of 0.53 are achieved.

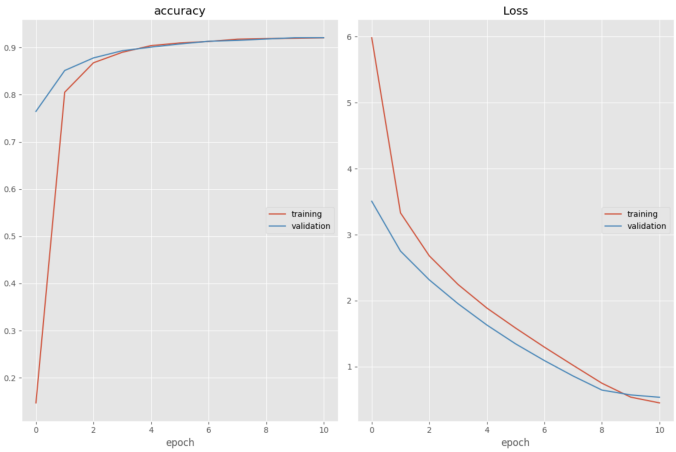

Fig. 19. Learning curves from the *LSTM* model training.

This evaluation already shows promising results, but can only serve as a first step. As part of our future work, we plan to compare our results to the results created by domain experts for a larger set of different models.

The results of the two approaches, both rule-based and model-based, complement each other very well to form a complete tool due to their complementarity. It can be stated that the challenge of the rule-based method lies in the assignment of the nouns found to the individual components. Furthermore, the connection of attributes and relationships to certain entities was not entirely trivial. On the other hand, the model-based comparison offers possibilities to correctly capture different cardinalities or ISA inheritances due to the increased flexibility in sentence structure and grammar structure.

VII. Conclusion

Extracting information from unstructured text is a complex task. Manual processing of large amounts of text is time-consuming, error-prone, and not scalable. Recently, numerous approaches for automating this task have been proposed. However, there exist many cases where very specific information has to be extracted from unstructured text. These imply challenges based on the specifics and rules applicable for the desired result. One of these cases is the extraction of ER models. There exist several approaches, but they still lack different features for a complete and usable automation.

We proposed a hybrid approach to extract meaningful ER data from unstructured text. Two subsystems were developed, both of which can extract ER components from unstructured texts. Special NLP methods were used for the rule-based extraction of entities, attributes, and relationships. Due to the higher reliability of the rule-based results, these were used for the final ER model. *SpaCy* and *LSTM* models can be used to validate the rule-based results. In the future, these results can be supplemented by an automated comparison with the results of the model-based approach. Additionally, this extended article provides a more detailed view of the LSTM model architecture, includes more illustrative examples, and presents a meaningful evaluation of the model-based results.

In the literature, there are still research gaps in the area of automatic extraction of ERM from texts using NLP techniques. This includes the integration of contextual information, the consideration of ambiguity, and the additional complexity posed by the German language, such as compound nouns and flexible sentence structure. Furthermore, current approaches mainly focus on the extraction of ERM, while object-oriented representations such as class diagrams are rarely considered. Modern software systems and web frameworks commonly rely on object-oriented paradigms and frequently use Object-Relational Mappers (ORMs) to derive ERM directly from source code. Investigating the automatic generation of class diagrams and other object-oriented representations from textual descriptions is planned as part of future work. Another fundamental improvement is the training of the models with even more text data, so that even rarely occurring ER components, such as multi-value attributes, can be better learned. The pretrained models can be trained for other domains according to the principle of transfer learning so that the models can be used for other purposes, e.g., to create knowledge graphs.

Our future work will include the mentioned gaps: We will expand our approach to include more specific ER concepts. In addition, we will focus on enabling a broad applicability for different domains. We will also investigate options for automated integration of the results from the two approaches. To enhance the robustness of our evaluation, we will also include more use cases with real-world texts and compare the results of our approach with ER models created by human experts. Another area of future work is the extension of the approach to also incorporate other models. As a first step, we plan to integrate UML class diagram creation.

Recent advances in large language models (LLMs) have shown that such models are capable of extracting entities and relationships from textual descriptions with promising results, often even without task-specific fine-tuning. Nevertheless, the focus of this work was intentionally placed on a hybrid approach combining rule-based and classical model-based techniques. An important reason is the improved explainability and traceability of the extraction process, as rule-based methods allow a clearer understanding of why specific entities or relationships were identified. In contrast, LLM-based approaches often operate as black-box systems, making validation and error analysis more difficult. Furthermore, our approach enables a more controlled integration of linguistic rules tailored to German language characteristics. Therefore, integration and evaluation of LLM-based extraction methods will be considered as part of future work.

References

[1] V. Vyramuthu and G. Grambow, "Towards extracting entity relationship diagrams from unstructured text using natural language processing," in *Proc 17th International Conference on Advances in Databases, Knowledge, and Data Applications (DBKDA), Lisbon, Portugal, March 09-13*, 2025, pp. 42–47.

[2] M. Kanakaraj and R. M. R. Guddeti, "Performance analysis of ensemble methods on twitter sentiment analysis using NLP techniques," in *Proceedings of the 2015 IEEE 9th International Conference on Semantic Computing (IEEE ICSC 2015)*. IEEE, 2015, pp. 169–170.

[3] A. Rajput, "Natural language processing, sentiment analysis, and clinical analytics," in *Innov in Health Informatics*. Elsevier, 2020, pp. 79–97.

[4] R. Suganya, R. Krupasree, S. Gokulraj, and B. Abinesh, "Product review analysis by web scraping using NLP," in *Smart Data Intelligence: Proceedings of ICSMDI 2022*. Springer, 2022, pp. 427–436.

[5] V. T. N. Chau and S. Chittayasothorn, "A bitemporal SQL database design method from the enhanced entity-relationship model," in *2021 7th International Conference on Engineering, Applied Sciences and Technology (ICEAST)*. IEEE, 2021, pp. 85–90.

[6] S. Ghosh, P. Mukherjee, B. Chakraborty, and R. Bashar, "Automated generation of ER diagram from a given text in natural language," in *2018 Int'l Conf on ML and Data Eng (iCMLDE)*. IEEE, 2018, pp. 91–96.

[7] V. Bryl, C. Giuliano, L. Serafini, and K. Tymoshenko, "Supporting natural language processing with background knowledge: Coreference resolution case," in *The Semantic Web–ISWC 2010: 9th International Semantic Web Conference, ISWC 2010, Shanghai, China, November 7-11, 2010, Revised Selected Papers, Part I 9*. Springer, 2010, pp. 80–95.

[8] P. Kashmira and S. Sumathipala, "Generating entity relationship diagram from requirement specification based on NLP," in *2018 3rd Int'l Conf on Information Technology Research (ICITR)*. IEEE, 2018, pp. 1–4.

[9] M. K. Habib, "On the automated entity-relationship and schema design by natural language processing," *Int'l J Eng Sci*, vol. 8, no. 11, pp. 42–48, 2019.

[10] N. Omar, J. Hanna, and P. McKevitt, "Heuristic-based entity-relationship modelling through natural language processing," in *Proc. of the 15th Artificial Intelligence and Cognitive Science Conference (AICS-04)*. Artificial Intelligence Association of Ireland, 2004, pp. 302–313.

[11] M. Omar and A. Abdulla, "The entities extraction for entity relationship models from natural language text via machine learning algorithms," in *Proc 4th Int'l Conf of Basic Sci and Their Appl, Elbeida, Libya*, 2020.

[12] E. S. Btoush and M. M. Hammad, "Generating ER diagrams from requirement specifications based on natural language processing," *Int'l J of Database Theory and Application*, vol. 8, no. 2, pp. 61–70, 2015.

[13] X. Li, K. Chen, Y. Long, and M. Zhang, "Llm with relation classifier for document-level relation extraction," *IEEE Transactions on Big Data*, 2026.

[14] G. Banjac, D. Brdjanin, and D. Banjac, "Towards automatic conceptual database design based on heterogeneous source artifacts," in *European Conference on Advances in Databases and Information Systems*. Springer, 2023, pp. 487–498.

[15] M. Honnibal, I. Montani, S. Vanlandeghem, and A. Boyd, "spacy: Industrial-strength natural language processing in python," doi: 10.5281/zenodo.1212303, 2020.

[16] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994.

[17] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[18] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017, pp. 5998–6008.

[19] C. Manning, P. Raghavan, and H. Schutze, *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[20] H. P. Luhn, "The automatic creation of literature abstracts," in *IBM Journal of Research and Development*. Ibm, 1958, vol. 2, no. 2, pp. 159–165.

[21] F. Chollet *et al.*, "Keras," https://github.com/fchollet/keras, 2015.

[22] E. Zvornicanin, "What are embedding layers in neural networks?" May 2023. [Online]. Available: https://www.baeldung.com/cs/neural-nets-embedding-layers

[23] S. Weidman, "Deep learning from scratch." [Online]. Available: https://www.oreilly.com/library/view/deep-learning-from/9781492041405/ch04.html