WannaDB: Ad-hoc SQL Queries over Text Collections

Just tell it what you want, what you really, really want

Benjamin Hättasch,[1,2,3] Jan-Micha Bodensohn,[1,2] Liane Vogel,[1,2] Matthias Urban[2] and Carsten Binnig[2,3]

Abstract: In this paper, we propose a new system called *WannaDB* that allows users to interactively perform structured explorations of text collections in an ad-hoc manner. Extracting structured data from text is a classical problem where a plenitude of approaches and even industry-scale systems already exist. However, these approaches lack in the ability to support the ad-hoc exploration of texts using structured queries. The main idea of *WannaDB* is to include user interaction to support ad-hoc SQL queries over text collections using a new two-phased approach. First, a superset of information nuggets from the texts is extracted using existing extractors such as named entity recognizers. Then, the extractions are interactively matched to a structured table definition as requested by the user based on embeddings. In our evaluation, we show that *WannaDB* is thus able to extract structured data from a broad range of (real-world) text collections in high quality without the need to design extraction pipelines upfront.

Keywords: interactive text exploration; text to table; matching embeddings

1 Introduction

A question like "What were the days with a COVID-19 incidence rate higher than 750 in Germany?" can be answered with a simple SQL query if the relevant information is present in a database. Yet, in case there are only written (i.e., textual) reports available such as those published by governmental organizations like the RKI in Germany,[4] the situation is much more complex: answering such queries over collections of textual documents that each contain only a part of the information needed requires that first the relevant attributes are extracted from each document, before they are stored in a structured form (i.e., a spreadsheet or a database table) in order to make them available for structured queries.

One could now argue that extracting structured data from text is a classical problem for which there is a plethora of approaches and where even several industry-scale systems already exist: for example, DeepDive [Sa16] that was acquired by Apple or System-T

[1] Primary Authors
[2] Technical University of Darmstadt, Systems Group, 64287 Darmstadt, Germany, firstname.lastname@cs.tu-darmstadt.de
[3] DFKI
[4] https://www.rki.de/DE/Content/InfAZ/N/Neuartiges_Coronavirus/Situationsberichte/Gesamt.html

doi:10.18420/BTW2023-08

```
SELECT report_date WHERE incidence_rate > 500;
SELECT region, AVG(incidence_rate) GROUP BY region HAVING AVG(incidence_rate) > 500;
SELECT AVG(vaccinated_twice) WHERE report_date > 21-01-01 AND report_date < 21-02-01;
```

Fig. 1: Exemplary ad-hoc information needs phrased as SQL-like queries in *WannaDB*. Two classes of ad-hoc queries are supported: Queries that extract facts from individual documents (e.g., first query) as well as queries that involve aggregation and grouping (e.g., the latter two queries).

[Le20a] from IBM are such systems that have developed rather versatile tool suites to extract structured facts from textual sources. However, these systems require a team of highly-skilled engineers that compile extraction pipelines, which often includes training particular machine learning models, and then populate a structured database from the given text collection. And even more importantly, the resulting extraction pipelines are typically static and can only be used to extract a pre-defined (i.e., fixed) set of attributes and tables for a certain text collection. This prevents exploratory scenarios where users can ask ad-hoc queries regardless of whether a pipeline has been set up to extract the attribute or not.

Hence, being able to ad-hoc execute SQL-like queries over a text collection without the need to manually compose extraction pipelines would be a major step forward compared to existing approaches for structured data extraction from text. Use cases with needs for such ad-hoc structured querying of unstructured text can be found in various domains beyond the example mentioned before, e.g., data scientists together with medical doctors looking for new insights through medical reports or data journalists examining hundreds of documents as part of their investigations. Structured queries provide a higher expressiveness (e.g., aggregation and filtering operations), and more rigorousness in the calculation of the results compared to the usage of natural language queries in classical question answering systems.

Contributions. In this paper, we hence propose *WannaDB*, a system that can execute SQL-like queries on text collections in an ad-hoc manner. Examples for queries that *WannaDB* supports can be found in Figure 1. Overall, *WannaDB* supports two classes of queries: *(1) Ad-hoc Fact Queries:* queries that extract facts from text documents to construct table rows. This also involves applying filter predicates and projection operations, as shown by the first query in Figure 1. *(2) Ad-hoc Aggregate Queries:* queries that in addition involve aggregations and grouping over multiple documents as shown by the two other queries in Figure 1, which come with additional challenges like named entity disambiguation/cross-document co-reference resolution that we discuss later in this paper. *WannaDB* can therefore directly produce tables stating information that is not explicitly mentioned in the documents and hence not discoverable by pure extraction or search approaches. To enable such ad-hoc SQL queries over a given text collection, *WannaDB* implements a novel extraction and querying pipeline that builds on two key ideas:

The first key idea of *WannaDB* is that, different from existing approaches which aim to extract information for a specific (i.e., fixed) information need from a given text collection, *WannaDB* instead implements a *holistic extraction* approach that aims to extract a wide spectrum of information from a given text collection (called information nuggets in the

sequel). For this holistic extraction, *WannaDB* implements a framework approach and relies on a set of different general-purpose extraction methods, such as approaches for named-entity recognition. Moreover, during extraction, *WannaDB* computes embeddings for all the information nuggets, taking several signals such as the textual mentions itself, and the position in the text into account.

As a second key idea, to answer ad-hoc queries on top of the extracted information nuggets, *WannaDB* implements a novel *interactive matching* approach that aims to map the information nuggets to the information needs specified by the user in form of an SQL query: embeddings of the extracted information nuggets together with the embeddings of the query attributes are used to decide which information nuggets qualify for answering the query. For this matching, *WannaDB* requests feedback from the user whether certain information nuggets are the correct values for the required query attributes. The system carefully selects these requests to minimize the amount of required feedback. The query attributes can be of a much finer granularity than the labels of the extraction approaches used in the first stage (e.g., `airline` instead of `ORG`) and *WannaDB* can even distinguish between similar attributes with just a small semantic difference (e.g., the amounts of people vaccinated once and twice).

While other approaches that can extract tables from text such as learned sequence-to-sequence models [WZL22] often suffer from a phenomenon called hallucination (i.e., they generate values that are not in the actual source document), our approach can guarantee that the contents of the produced result tables always originate from the queried documents. Moreover, compared to learned question answering approaches, *WannaDB* can perform numerical reasoning on the data without the need to rely on the limited mathematical abilities [He21] of a language model.

In order to evaluate the abilities of *WannaDB*, we conduct a wide range of experiments on text collections from different domains ranging from aviation reports over daily COVID-19 situation reports to multiple text collections created from Wikipedia that cover different categories (Nobel laureates, countries, and skyscrapers). We show that *WannaDB* not only outperforms other baselines that can be used for ad-hoc query answering on text collections, but is also competitive with approaches that are trained or refined on domain-specific data. Moreover, our evaluation shows that typically only a few interactions per query attribute are sufficient to answer a query over hundreds or thousands of source documents. Overall, answering an SQL query over text documents with *WannaDB* (by providing minimal interactive feedback) only takes a few minutes, compared to hours and hours of manually extracting information or refining an extraction pipeline without *WannaDB*. Finally, to make the results reproducible, we will make our source code and the data sets used for evaluation available at `https://link.tuda.systems/wannadb`.

Outline. Next, we describe the functions of *WannaDB* in an exemplary usage scenario, before we explain the different components in Section 3. In Section 4, the algorithms behind the interactive components are discussed in further depth, followed by a short overview of

the current limitations in Section 5. We provide an evaluation of *WannaDB* in Section 6 and an overview of existing and related work in Section 7, before we conclude in Section 8.

2 Exemplary Usage

In this exemplary usage scenario, we aim to show how *WannaDB* can be used to satisfy an information need based on a text collection. Imagine, e.g., a data journalist who just obtained a large collection of airline incident reports and is now looking for noticeable events, like a high rate of incidents for a certain carrier or airport. They use *WannaDB* for that purpose. The data journalist starts by loading the collection of text files into *WannaDB* for processing and triggers the pre-processing of the files, a process that needs to be done only a single time for each text collection.

Next, the data journalist enters an SQL-like query as a starting point for their exploration (e.g., `SELECT airline, airport, COUNT(*) GROUP BY airline, airport`). As there is no pre-existing table yet, the FROM-part of a typical SQL query can be omitted, simplifying the query syntax. After entering the query, *WannaDB* presents a list of possible matches for each required attribute (e.g., airline) found in texts of the collection, as shown in Figure 2. Not all the found matches will be correct right away, therefore *WannaDB* relies on some user input to adjust the results. The data journalist confirms a few of the correctly found matches, corrects wrong matches by choosing the relevant extraction or marks if the required attribute does not occur in a given text (see Figure 2). Meanwhile, *WannaDB* continuously updates the list of all guessed matches during this interactive phase, leveraging the feedback. The user interface allows to quickly identify entries that stand out and get an impression of the quality already achieved. Once the data journalist is satisfied with the quality of the matches, they continue with the next attribute of their query.

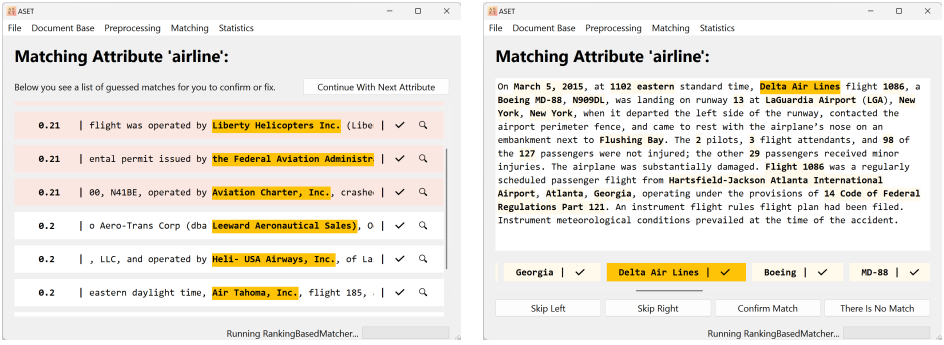


Fig. 2: Graphical user interface of *WannaDB*, more details can be found in our SIGMOD'22 demo [HBB22]. Left: potential matches over and under the threshold are shown, the user is asked to either confirm or fix them. Right: Inspect a document and fix by selecting the correct match.

After all attributes are processed, *WannaDB* will execute the query on the resulting table. If the query contains grouping operations, the data journalist might be asked again for some interactive feedback (e.g., to confirm that *Lufthansa* and *LH* refer to the same airline, but *LHS* does not). *WannaDB* will again try to transfer this feedback to other rows. In the end, the data journalist will receive an answer to their query and can export the resulting table to a spreadsheet, an SQLite table, or a Pandas Dataframe for further investigation. If they have further queries to submit to *WannaDB*, the interactive matching process only needs to be repeated for new attributes, as *WannaDB* leverages existing results from previous queries.

3 System Overview & Architecture

In this section, we describe the architecture of *WannaDB*. It consists of two stages: an offline stage to extract information nuggets (i.e. short information-bearing text snippets), followed by the interactive stage to answer the query by table extraction and if required interactive filtering or grouping. The overall workflow is visualized in Figure 3. Here, we give an overview of both stages and the relevant components of *WannaDB*. More details of the table extraction as well as grouping and filtering, which are the main contributions of *WannaDB*, are described in Section 4.

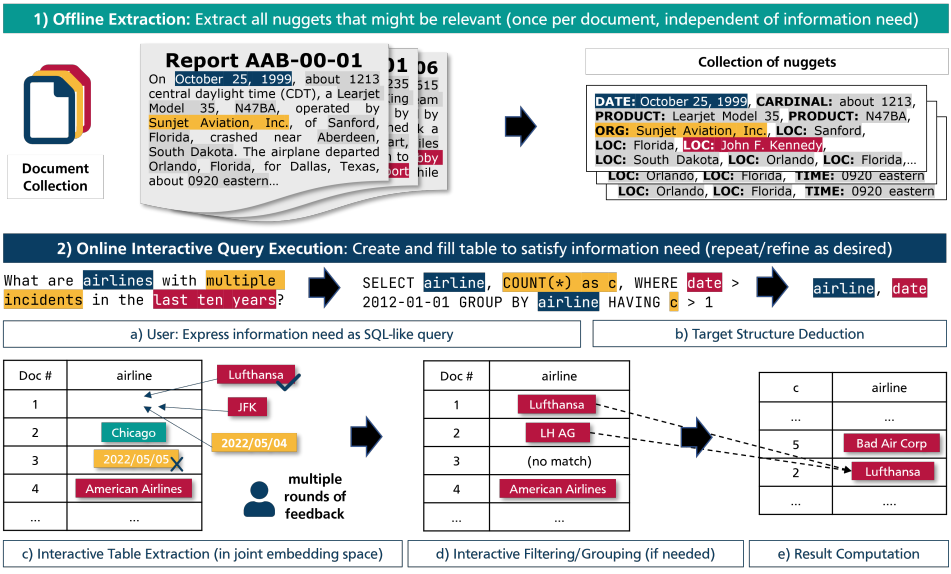


Fig. 3: Architecture & exemplary usage: The offline extraction phase obtains information nuggets from the documents. The online phase then infers the required structure from a query, matches between the extracted information nuggets and the user's schema, performs the grouping and executes the query.

3.1 Stage 1: Offline Extraction

In the first stage we employ off-the-shelf information extractors to extract a superset of potentially relevant information nuggets (e.g., named entities) from the given text collection. This step is independent of user queries and can thus be executed offline to prepare the text collection for ad-hoc exploration by the user. The extractors process the collection document-by-document to generate the corresponding extractions. Clearly, a limiting factor of *WannaDB* is which kinds of information nuggets can be extracted in the extraction stage, since only this information can be used for the subsequent matching stage. As a default, we use named entity recognizers from *Stanza* [Qi20] and *spaCy* [Ho20]. In general, *WannaDB* can be used with any extractor that produces label-mention pairs; i.e. a textual mention of an information nugget in the text (e.g., *American Airlines*) together with a natural language descriptor representing its semantic type called label (e.g., *Company*). Moreover, additional information about the extraction (e.g., its position in the document and the surrounding sentence) is also stored and used for computing the embeddings, as we describe below.

After extraction, the information nuggets are pre-processed to derive their actual data values (i.e., a canonical representation, e.g., for timestamps) from their mentions. For this we also rely on state-of-the-art systems for normalization [Ma14]. The nuggets are then represented based on the following signals: (1) *label* – the entity type determined by the information extractor (e.g. *Company*),[5] (2) *mention* – the textual representation of the entity in the text (e.g., *Lufthansa*), (3) *context* – the sentence in which the mention appears, (4) *position* – the position of the mention in the document. Each information nugget representation comprises embeddings for the individual signals (1-4). We compute semantic representations for the natural language signals using FastText [Mi18] (1), Sentence-BERT [RG19] (2) and BERT [De19] (3) and normalize the position by dividing it by the document length.

3.2 Stage 2: Interactive Query Execution

At runtime, a user issues queries and interacts with the system. *WannaDB* infers the table structure required to answer a query, and employs a novel interactive matching stage to map the information nuggets extracted in the first stage to the required query attributes.

Interactive Table Extraction. The first step of the interactive query execution of *WannaDB* is the interactive table extraction from the text documents. In this step, a table with attributes is filled by *WannaDB* to answer a given user query. The required table structure is automatically inferred from the user's SQL query. *WannaDB* checks which attributes are mentioned explicitly as attributes to return, and as part of aggregation operations, or implicitly in filter predicates or group-by statements. Then, *WannaDB* starts to fill the table with the derived schema by executing the interactive table extraction algorithm.

[5] We map the named entity recognizers' labels like *ORG* to suitable natural language expressions according to the descriptions in their specification.

In the interactive table extraction, the user interacts with *WannaDB* in order to fill the required attributes of the result table with the information nuggets extracted before. To find matching nuggets, *WannaDB* first computes embeddings for the target attributes similar to the ones computed for the information nuggets in the offline phase.

A classical approach to determine a mapping between information nuggets and attributes of the user table would be to train a machine learning model in a supervised fashion to classify to which attribute the extracted information nugget should be mapped to. However, learning such a classification model would require a substantial set of labeled training data for each attribute and thus prevent ad-hoc queries. Instead, our approach leverages embeddings to quantify the intuitive semantic closeness between information nuggets and the attributes of the user table. For the attributes of the target table, only the attribute names are available to derive an embedding, while for the extracted nuggets we can make use of more information as we described above.

WannaDB therefore employs a novel interactive matching strategy that incorporates user feedback and operates in the joint embedding space of nuggets and target attributes. This strategy works in an attribute-by-attribute fashion and collects user feedback (e.g., confirming or correcting a possible match). *WannaDB* uses distances between possible and confirmed matches to populate the remaining cells. This process is steered by carefully selecting potential matches that are presented to the user for feedback to reach a high matching quality with as little feedback as possible.

Interactive Filtering & Grouping. After the interactive table extraction step, *WannaDB* executes the interactive filtering and grouping stage for answering a user query. Remember, *WannaDB* has the aim to work on text collections from domains without pre-existing resources like refined language models or custom knowledge bases. Grouping and filtering the extracted table thus is challenging, since it is filled with mentions from the text directly, hence applying these operations might lead to faulty query results if entities are not correctly resolved: e.g., the table might contain entries such as *Deutsche Lufthansa* and *German Lufthansa Airline* which both refer to the same entity. Applying `GROUP BY` or a `WHERE` directly on such an extracted table would return multiple lines (i.e., one for each different mention even though they refer to the same entity). *WannaDB* therefore again uses interaction to perform those operations on the level of embeddings instead of string representations, as will be described in detail in the next section.

4 Interactive Query Execution

WannaDB introduces novel embedding-based algorithms for interactive table extraction as well as filtering and grouping. In this section, we describe these algorithms in further detail (see Figure 4 for a pseudo-code representation).

4.1 Interactive Table Extraction

In the interactive table extraction stage, *WannaDB* populates the attributes of the table one by one. To fill the cells of a certain attribute, *WannaDB* aims to select one matching information nugget from each of the documents. To do so, *WannaDB* associates each information nugget with a *cached distance* that corresponds to the certainty with which it believes that the nugget matches the attribute. For each document, *WannaDB* considers the information nugget with the lowest cached distance as the document's currently *guessed match*. Furthermore, *WannaDB* uses a distance *threshold* for each attribute to decide when a cell should be left empty instead. The details of how this threshold is calculated and interactively adapted are explained in Section 4.2. The overall procedure of the table extraction is shown in Figure 4.

In the beginning, each nugget's cached distance is initialized as the cosine distance between the nugget's label embedding (e.g., *Organization*) and the embedding of the attribute name (e.g., *Airline*) (Figure 4, line 2-3). After initialization, the interactive feedback phase starts. *WannaDB* presents a ranked list of documents with their currently guessed matches to the user for feedback (see Figure 2) and will continuously update the list after every given feedback. This allows the user to quickly identify (incorrect) entries that stand out and to get an impression of the quality already achieved. The ranked list is centered around the threshold and thus hopefully shows both correct guesses with a low certainty, and incorrect guesses, where *WannaDB* would profit most from feedback.

The user can then provide feedback for any of these guesses (line 7): they may either confirm the guess, select another information nugget from the document, or state that the document does not contain a matching information nugget. In case their feedback results in a confirmed match, this matching information nugget is used to update the cached distances of all other remaining information nuggets (line 13-16). To compute the distance between two information nuggets, *WannaDB* calculates the mean of the cosine distances between their individual signal embeddings. The distance updates ensure that a nugget's cached distance is always the distance to the closest confirmed match. Considering distances between information nuggets allows *WannaDB* to capitalize on more signals like the textual mentions (e.g., *American Airlines*) of other matching information nuggets.

Next, *WannaDB* updates the documents' currently guessed matches by selecting the information nuggets with the lowest cached distances (line 21). Finally, *WannaDB* then adjusts the threshold accordingly (see Section 4.2 for more details). Moreover, the user can at any time decide to terminate the interactive feedback phase and continue with the next attribute. All remaining documents' cells without explicitly confirmed matches will then be populated with their currently guessed matches (line 24-28) if there is at least one with a distance that is low enough (i.e., below the threshold).

```
for attribute in query.attributes: # Process each attribute separately
  for nugget in all_nuggets:
    nugget.distance = compute_distance(attribute, nugget) # Compute initial distances

  while interactive_feedback_phase:  # Interactively get user feedback
    ranked_list = make_ranked_list(threshold, documents)
    feedback = get_user_feedback(ranked_list)
    match feedback:
      # Positive feedback (confirmation or manually correction):
      case ConfirmNugget(document, confirmed_nugget):
        # Mark this particular cell as manual confirmed...
        set_match(document, confirmed_nugget)
        # ... and update distances for all nuggets based on user feedback
        for nugget in all_nuggets:
          new_distance = compute_distance(nugget, confirmed_nugget)
          nugget.distance = min(new_distance, nugget.distance)
      # Negative feedback:
      case NoMatchInDocument(document):
        # Direct effect only on the given document...
        leave_empty(document)
    update_guessed_matches(documents)
    adjust_threshold(feedback)  # ... but both feedback types can have effects indirectly
    ↪  through threshold adjustment on other document's rows, too

  for document in documents:  # Only consider values up to a given maximum distance
    if current_guess(document).distance < threshold:
      set_match(document, current_guess(document)) # compute final result table
    else:
      leave_empty(document)

def adjust_threshold(feedback):  # Feedback can be further exploited in certain cases
  match feedback:
    case ConfirmNugget(document, confirmed_nugget):
      if confirmed_nugget.distance > threshold:
        increase_threshold(confirmed_nugget)
    case NoMatchInDocument(document):
      if current_guess(document).distance < threshold:
        decrease_threshold(document)

def decrease_threshold(document):  # Consider fewer matches as valid (especially those
↪  above last marking as incorrect that are currently accepted nevertheless)
  nuggets = ranked_list.between(threshold, document)
  min_dist = min(n.distance for n in nuggets)
  threshold = min(min_dist, threshold)

def increase_threshold(confirmed_nugget):  # Consider more matches as valid (especially
↪  those below last confirmation that are currently discarded because of the threshold)
  nuggets = ranked_list.between(confirmed_nugget, threshold)
  max_dist = max(n.distance for n in nuggets)
  threshold = max(max_dist, threshold)
```

Fig. 4: Pseudo-Code representation of our interactive algorithm for table extraction, including threshold adjustment.

4.2 Threshold Adjustment

WannaDB uses a threshold for two purposes: (a) to decide when it is better to leave a cell empty than to use a very unlikely guess (mostly because the desired value is not mentioned in the document) and (b) to select guesses to present to the user where feedback will have as much effect as possible. This threshold is automatically tuned during the runtime of *WannaDB* to fit the data at hand. Given the approximate query setting *WannaDB* is built for, we decided to use a common threshold for all regions forming in the embedding space instead of individually tuning it, to keep the number of interaction cycles low.

The adjustment of the threshold is shown in Figure 4 (line 30-47). The general idea is to incorporate the additional knowledge gained from the user confirming a nugget even though it was above the threshold or correcting an entry below the threshold. This feedback action will only affect a certain nugget directly, but other similarly well fitting nuggets from other documents might still be accepted or discarded wrongly because of the threshold, which is therefore carefully adapted after feedback actions: If the user confirms a nugget from the ranked list that is above the threshold, all nuggets between the threshold and this nugget should be considered as a good guess. In the case that any of the nuggets is still above the threshold after the calculation of the new distances, the threshold is adapted accordingly. In contrast, if the user states that for a nugget with a distance below the threshold there is no match in the document, the threshold is decreased to also exclude other matches that are in the list above the nugget if necessary. The threshold is only adapted in these two cases, where implicit hints about the quality assessment by the user can be incorporated.

4.3 Interactive Filtering & Grouping

In the following, we explain how interactive grouping is supported in *WannaDB* to tackle the problem of different surface forms for the same entries. Filtering works similarly, but we omit the details due to space limitations.

To resolve entities correctly, the interactive grouping algorithm is based on agglomerative clustering using the distances between the information nugget embeddings for an attribute. Entries with the same string representation are merged without interaction. For the remaining ones, the different signals from the extraction phase are utilized. *WannaDB* presents all distinct members of two clusters that should potentially be merged to the user and asks them to confirm whether these all describe the same entity. If that is the case, the clusters are merged and the distances are recalculated. To minimize the amount of necessary interactions with the user, *WannaDB* does not always ask for the pair of clusters with the lowest distance, but chooses a pair with a higher distance, using a step size that is adapted based on the last interactions. If the user confirms the equivalence of the candidates, not only that pair but also those with a substantially lower distances are merged. If the entries of the merging candidates are marked as different, *WannaDB* continues to search for a better threshold for the distance between clusters using a binary search pattern.

5 Current Limitations of *WannaDB*

In order to build a system that can quickly compute query results on various domains, we introduce two limitations: First, *WannaDB* currently can only answer single-table queries on top of document collections; i.e., we extract one table per document collection where each row of the table corresponds to one document. However, this is not a severe limitation, since the extracted table can be seen as the materialized result of a join. *WannaDB* will extract a wide table (e.g., containing information about an incident itself but also the airlines and airports involved)—but only with the attributes that are required for a given query.

Second, the results produced by *WannaDB* are always approximate. While *WannaDB* can achieve a high F1-score for all attributes (as we will show below), query results might be incomplete (i.e., values of attributes might be missing) or the extracted values might be dirty (e.g., a group-by statement might result in two instead of one group due to a not fully correct clustering). However, we believe that the query results of *WannaDB* are still of high value to users, providing them with a trend and allowing them to decide if something interesting is contained in the document collection in a short time.

6 Experimental Evaluation

In this evaluation, we aim to show the abilities of *WannaDB* on text collections from different domains. We will demonstrate the end-to-end performance, compare our table filling approach to non-interactive and learned models, and evaluate the effects of interaction, and the scalability of *WannaDB*. To the best of our knowledge, there is no system working like *WannaDB* yet. Therefore, we cannot compare our results end-to-end with existing systems. As the whole task of running SQL queries over text collections is quite complex, there is no simple baseline for comparison either. However, we evaluate the components of our approach individually, and show that *WannaDB* performs better compared to various baselines. We perform our evaluation on three data sets from very different domains. Each of them consists of a document collection as well as a ground-truth extraction of structured data that we can use to evaluate the results of executing ad-hoc queries with *WannaDB*.

Aviation. The first data set is based on aviation accident reports published by the United States National Transportation Safety Board (NTSB).[6] Each report documents a severe aviation accident and provides details like the prevailing circumstances, probable causes, conclusions, and recommendations. For the experiments, we use the executive summaries that the NTSB publishes with each report. As a ground-truth, we compiled a list of twelve attributes based on frequently occurring facts from the summaries. We then manually created annotations that capture where the summaries mention the attributes' values. The final data set comprises 100 annotated documents and a table which provides the ground-truth structured data for all attributes.

[6] https://www.ntsb.gov/investigations/AccidentReports/Pages/Reports.aspx?mode=Aviation

COVID19. The second data set is based on the German RKI's daily reports outlining the situation of the Covid-19 pandemic in Germany.[7] We again used the summaries of the full documents, which contain information like the number of new laboratory-confirmed Covid-19 cases or the number of Covid-19 patients in intensive care. We compiled a list of seven all-numeric attributes, which is in particular challenging compared to string-valued attributes, since these are harder to separate into different attributes in the embedding space. As a ground-truth for the experiments, we manually annotated the occurrences of all these seven attributes again in 100 reports.

T-REx: Countries, Nobel & Skyscrapers. In addition to the data sets before that we explicitly created for evaluating *WannaDB*, we adapted the T-REx data set [El18] that was also used in other papers. The original data set consists of 11 million Wikidata triples aligned with 3.09 million Wikipedia abstracts. We extracted three subsets based on article categories from different domains: *Countries* consists of 187 documents with three annotated attributes, *Nobel* challenges to extract four attributes (date of birth and death, field of work and country) for 209 Nobel Prize laureates, and *Skyscrapers* is by far the largest data set with 2683 documents containing annotations for three attributes. All these data sets are quite sparse, since most of the time only a subset of the attributes is contained in a document. Therefore, this data set is valuable to test how well *WannaDB* can work when information in documents is missing.

Metrics. As a main metric, we report the F1 score in most experiments (values between 0 and 1, higher is better) as an aggregated value that incorporates both the precision (i.e., the correctness of the table cell values) of our approach and its recall (i.e., the extent to which table cells are filled as expected). The F1 scores we report are calculated based on the ground truth and predictions in the filled tables. We thereby consider cells (i.e., an attribute value) as true positives when they are correctly filled with information from the text corresponding to that row, and as true negatives when they are correctly left empty, in case the required information is not present in the corresponding text. False positive predictions occur, when a cell is filled incorrectly. False negatives occur when a cell is left empty that should have been filled with data from the text, and also for incorrectly filled cells, as the correct nugget has not been found.

6.1 Exp. 1 – End-to-end Queries

To provide an indication of how *WannaDB* works end-to-end, we perform a qualitative analysis on queries involving aggregation and grouping over multiple documents before we later-on show quantitative results for *WannaDB*. For the experiments, we assume that a user always provides correct feedback for *WannaDB* to execute the matching of extractions to query attributes. However, we do not expect optimal feedback, i.e., the simulated feedback actions are not chosen in a way to maximize speed of convergence. We report the results after

[7] https://www.rki.de/DE/Content/InfAZ/N/Neuartiges_Coronavirus/Situationsberichte/Gesamt.html

using 20 simulated user interactions (i.e., 20 times confirming an extraction or choosing an alternative one as a match for a query attribute). We discuss the interaction effort that is needed for *WannaDB* to perform extractions in a separate experiment.

Figure 5 shows the first five rows of the query results for two aggregation queries executed on the *T-REx Nobel* and the *T-REx Countries* data sets. Additionally, precision and recall, as well as a numeric score of the correctness of the clusters, can be seen. While *WannaDB* delivered the correct values for the group-by operation, the aggregation (COUNT) deviates slightly from the ground-truth. The reason is that for some documents, *WannaDB* could not extract the requested information. As such, the results of *WannaDB* can be seen as an approximation of the true query result that can be used for quickly gaining (initial) insights into text collections. Moreover, it is important to note that existing extraction baselines—that in contrast to *WannaDB* do not support ad-hoc queries—also do not provide perfect extractions (as we show in the following experiments).

T-REx nobel laureate:

```
SELECT country,
COUNT( * ) AS count
GROUP BY country
SORT BY count
```

P: 1.0000
R: 0.5714
MJI: 0.4775

country	count *(correct)*
American	**88** *(84)*
Soviet	**9** *(10)*
Swiss	**5** *(4)*
Japan	**3**
Germany	**2** *(3)*
...	...

T-REx country:

```
SELECT continent,
COUNT( * ) AS count
GROUP BY language
SORT BY count
```

P: 0.7500
R: 1.0000
MJI: 0.5004

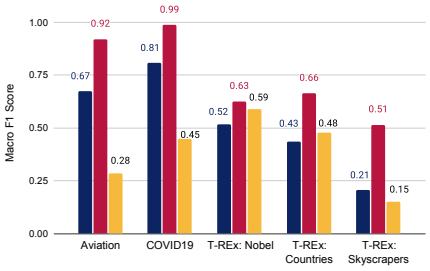
continent	count *(correct)*
Africa	**33** *(34)*
Europe	**19** *(20)*
Asia	**12** *(14)*
South America	**11** *(13)*
North America	**4** *(5)*
...	...

Fig. 5: End-to-end results for two queries executed on T-REx data sets. The tables show the first five rows of the resulting table (one attribute column filled by *WannaDB* plus aggregation results). The bracketed values indicate the ground truth values. Additionally, precision (P) and recall (R) computed at cluster level, and mean Jaccard Index (MJI) averaged over all clusters are reported.

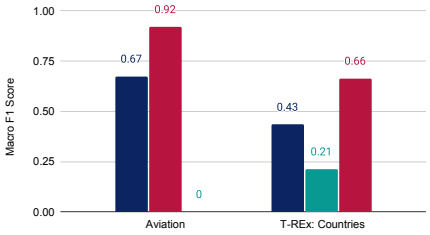
6.2 Exp. 2 – Interactive Table Extraction

In the second experiment, we quantitatively evaluate how well *WannaDB* can fill a table specified by a user's query with information from the texts. For this, we focus on the quality of the interactive table extraction, which is the most important step for *WannaDB* to provide high-quality query results; i.e., if the table extraction is not able to provide high accuracy, grouping and filtering will also not be able to provide high accuracy. For showing the quality of *WannaDB*, we run the experiments in this section on all three data sets (Aviation, COVID19 and T-REx).

Baselines. To put the results of *WannaDB* into perspective, we compare it to two baselines based on BART [Le20b]. BART is a state-of-the-art pre-trained transformer model, with a high capacity to learn text-based tasks with minimal overhead of fine-tuning. Its robust architecture outperformed older transformers, especially on tasks like question answering. We use the openly available *bart-large* model from the Huggingface [Wo19] library and formulate information extraction for individual query attributes as a sequence-to-sequence



(a) Table filling results. It can be seen that ■ *WannaDB* performs comparable to the ■ BART model trained explicitly on the data (fine-tuned per topic individually) and outperforms the generic ■ BART model (fine-tuned on SQuAD) most of the time.



(b) Upper Baseline BART models show low generalization abilities on unseen data sets—the performance of a ■ model trained on one data set drops drastically when applied to ■ the other data set. ■ *WannaDB* for comparison.

Fig. 6: Text-to-Table Results

task (i.e., the input is a text document and the output is the structured data extracted from the text). For fine-tuning BART for the information extraction task on a particular data set (i.e., transforming a text into a table) we use the following procedure: We split each data set into 75% that we use as train set for fine-tuning, 15% as validation set and 10% as a holdout test set. We then fine-tune one BART model on each data set for 50 epochs with a learning rate of $1e-5$ and batch size of 2, which yielded the best performance in our experiments. Moreover, we select the best checkpoint from the 50 epochs based on the validation set for evaluation. Important to note here is that the resulting fine-tuned BART models are an upper baseline for *WannaDB*, as they are trained supervised on the annotated data and all possible query attributes; i.e., with this baseline we do not test the ad-hoc scenario that we envision for *WannaDB*, but instead assume that all query attributes are known in advance.

For comparing *WannaDB* to a baseline that supports ad-hoc queries on a new (unseen) text collection, we use a second variant that is also based on BART but not pre-trained on the particular data set and query attributes. For this baseline, we instead use a BART model[8] that is already fine-tuned for extracting structured information from the SQuAD 2.0 data set [RJL18].[9] For the experiment, we use this fine-tuned model on an unseen data set and extract attributes that the model has not seen during fine-tuning.

WannaDB vs. Baselines. The results of *WannaDB* in comparison with the two BART models are shown in Figure 6a. For *WannaDB*, we report the median over 20 randomized runs, and again use 20 simulated user interactions per attribute. As baselines, we use the two variants of BART discussed before.[10] BART models fine-tuned per data set (red bars)

[8] Used Checkpoint: *phiyodr/bart-large-finetuned-squad2* from Huggingface [Wo19]

[9] In particular the fine-tuning task is QA on text collections which can be used to extract query attributes.

[10] The results of *WannaDB* and the second BART model that is used out-of-the-box are calculated on the whole data sets, whereas the results of the first BART model that is fine-tuned for the given data set are computed only on the 10% holdout test sets.

are able to achieve high F1 scores on the data and query attributes they were trained on, outperforming *WannaDB* on all data sets. Nevertheless, this approach is relying on the availability of annotated training data, which prevents ad-hoc queries. In comparison to the BART model that is used without fine-tuning on a given data set and set of query attributes (yellow bar), *WannaDB* achieves substantially better results. Especially for the Aviation and COVID19 data sets, *WannaDB* clearly outperforms this BART baseline. On the T-REx data sets, *WannaDB* provides competitive or better performance depending on the subset of data. We assume that BART's performance on the T-REx data is influenced by the fact that both the SQuAD data set it was fine-tuned on and the T-REx data set are based on Wikipedia.

Generalization of BART. As we have seen, while fine-tuning a BART model per data set yields the best performance, the BART model that is not fine-tuned for a data set provides inferior performance up to a point that it cannot extract any attributes correctly. To understand the generalization capabilities of BART in more depth and see if this is a systematic problem of BART, we now systematically use BART on data sets it has not been fine-tuned for. To be more precise, Figure 6b shows the results of two fine-tuned BART models: one fine-tuned on the *Aviation* data set and then used on the *T-REx Countries* data set and another model that we used vice versa; i.e., we applied both of them to the respective other data set, for which they have not been fine-tuned. The model fine-tuned on the *Aviation data* (reaching an F1 score of 91.95% tested in-domain on the *Aviation* data) only achieves 21.23% when tested on the *T-REx Countries* data set. At the same time, the model fine-tuned on the *T-REx Countries* data set (reaching an F1 score of 0.6633 on the in-domain test set) fails completely for extracting information correctly from the unseen aviation data domain with an F1-score of 0.0. This shows that a fine-tuned BART model is a valid approach to information extraction when annotated data is available and a fixed set of attributes is queried, but the resulting models are not able to generalize ad-hoc to other domains. In contrast, the results of *WannaDB* show that it can generalize well across data sets even without any particular training per data set and that the interactive approach provides an advantage over using generic embeddings or transformers directly.

Detailed Analysis of *WannaDB*. As a last point, we now zoom into the performance of *WannaDB* and analyze the results for all data sets on a per-attribute level to show that *WannaDB* can provide stable high performance and not just high performance for some query attributes. We used a combination of two different named entity recognizers,[11] Stanza [Qi20] and SpaCy[12] [Ho20] followed by our interactive matching approach.
Figure 7 shows that *WannaDB* can provide high accuracy and recall (measured by the combining F1 score, blue bars, right axis) for a wide spectrum of attributes from the three different data sets used in our evaluation. However, for some attributes the table is filled with a much lower quality than for others or not at all (e.g., for weather conditions). One

[11] *WannaDB* allows using multiple extractors at the same time, even if they produce overlapping nuggets. As default configuration for *WannaDB* and our experiments, we employ a combination of two robust general purpose extractors that are designed to work for a broad variety of domains. However, any other (combination of) extractors could be used in *WannaDB* as well.

[12] Using the *en_core_web_lg* model

reason can be that the currently employed information extractors are not able to extract the necessary information nuggets from the text (yellow bars). In particular, *aircraft_damage* and *weather_condition* are examples, where not only a large heterogeneity of mentions can be found but also very domain-specific terminology is used. Another reason for low table filling quality can be that the attributes occur in only a small fraction of the documents, as in the case of the attribute *owned_by* (which only occurs in 6% of the documents).

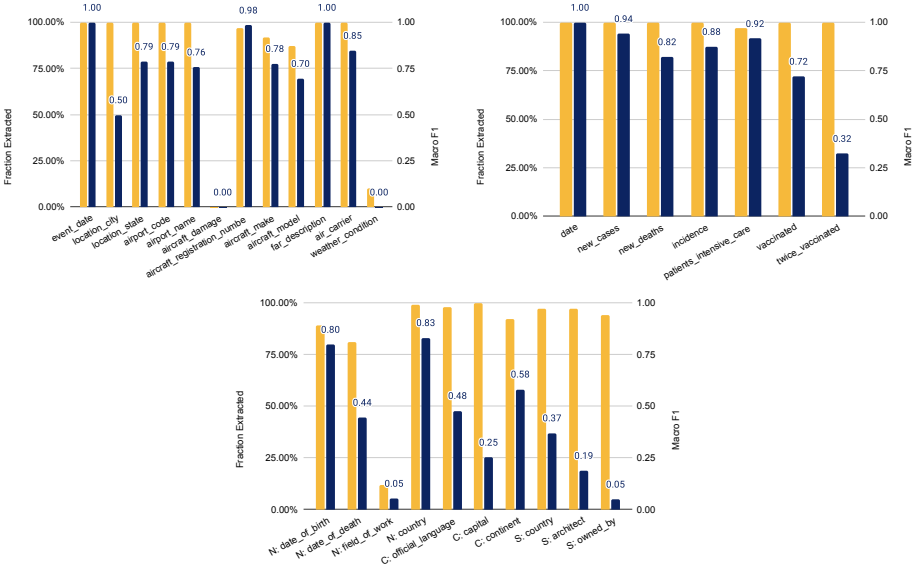


Fig. 7: ■ Fraction of values that could be extracted successfully and ■ table filling results per attribute of the Aviation, *COVID19* and T-REx data sets (in this order). *WannaDB* produces high scores for the majority of attributes, more than half are 0.7 or above.

In conclusion, *WannaDB* has the advantage over fine-tuned BART models, that it neither requires annotated training data, nor several hours of training time in order to work on unseen text collections. Furthermore, it does not suffer from the problem of *hallucination* [Ma20] that transformer-like models regularly experience, since they aim to also generate values for attributes even if no information nugget is present in the text. *WannaDB* instead generates an empty value in that case.

6.3 Exp. 3 – Effects of Interaction

In the previous experiments, we assumed a fixed amount of user interaction. In the third part of our evaluation, we instead investigate how the amount of interactive feedback given affects the table filling performance of *WannaDB*. We therefore simulate the interactive matching process with different interaction limits (i.e., the number of interactions per extracted query attribute). The resulting F1 scores can be seen in Figure 8.

As we can see, for some attributes, *WannaDB* achieves very high F1 scores with only one interaction with the user (e.g., for *event date* or *aircraft registration number* in the *Aviation* data set). These are attributes where the entity type of the extracted information nugget is very similar to the attribute name or the pattern of the extracted information nugget is rather unique. For example, the extraction has the named entity tag *DATE* which is similar to *event date*. For other attributes though, the performance of *WannaDB* strongly depends on the amount of interactive feedback. However, important is that *WannaDB* can typically provide high quality with only a few interactions. For most attributes, the first $5 - 10$ interactions massively improve the F1-score to achieve gains of up to 0.5. This overall confirms the interactive matching procedures we presented in Section 4 and the algorithm to select the right threshold. Yet, as we can additionally see, for a few attributes (e.g., weather condition), even many interactions cannot further improve the F1 scores. As we showed in the last experiment, the reason is that none of the extractors used in *WannaDB* can provide the information nugget for this attribute. Thus, as a future direction we want to combine *WannaDB* with a much broader set of existing extraction approaches beyond the named entity recognizers which we currently use, such as approaches for open information extraction.

6.4 Exp. 4 – Scalability

In our final experiment, we aim to assess the scalability of *WannaDB* to large text collections. Since *WannaDB* is an interactive system, the response times experienced by users are the most important performance metric. Across all used data sets, we measure that *WannaDB* takes on average 0.43 seconds to process a single user interaction.[13] This latency includes all computations between two user interactions; i.e., updating the cached distances and guessed matches as well as presenting the next set of candidate matches to the user for feedback. In general, we find that the interaction latency scales linearly with the number of nuggets. To measure the offline extraction phase, which has to be executed only once per text collection, we report the runtime on our largest data set *T-REx Skyscrapers*, which comprises $2,683$ documents. Running our default extraction phase takes about 48 minutes and produces $102,467$ nuggets. Comparing runtimes across data sets, we again find that the extraction runtime scales linearly with the number of generated nuggets.

In summary, it can be seen that *WannaDB* can scale to extensive text collections with thousands of documents and more than $100,000$ information nuggets by finishing the offline phase in a reasonable time and providing response times that allow for an interactive usage of the system [LH14].

[13] We executed this and all other of our experiments on a consumer desktop machine (CPU: AMD Ryzen 9 3900X; RAM: 32GB @3000MHz; GPU: NVIDIA GeForce RTX 2070 SUPER with 8GB VRAM).

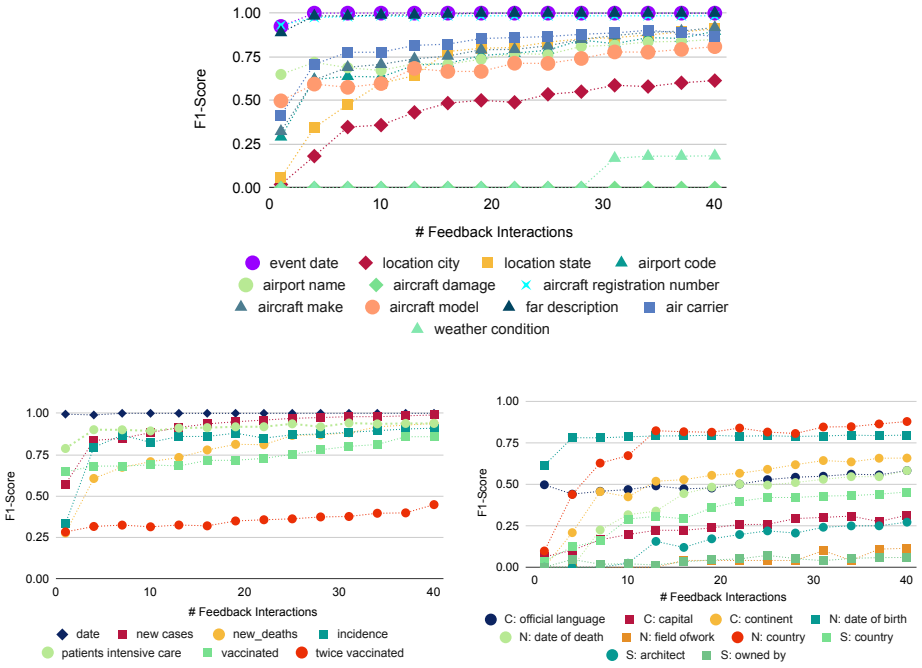


Fig. 8: F1 scores of *WannaDB* for the different attributes of the *Aviation*, *COVID19* and *T-REx* data sets for different amounts of feedback iterations per attribute (1-40). For most attributes, already a small amount of interactions drastically improves the quality, and more interactions lead to continuous improvements.

7 Related Work

Running SQL queries on text collections is a new task, and to the best of our knowledge, there is no other system yet working in the same way as *WannaDB*. However, some parts of the task resemble existing tasks and for some components of our approach there is previous work. Therefore, in this section, we give an overview of the related work of different areas, including knowledge base population and schema matching based on embeddings.

Information Extraction Systems. Existing approaches to answer queries over text collections heavily rely on manual labor, requiring users either to read through vast amounts of texts and extract relevant information manually, or to build specific extraction pipelines. One category of information extraction systems focuses on the task of knowledge base population, where a graph-structured knowledge base is constructed or expanded based on knowledge from natural language texts. Extractive approaches like DeepDive [Sa16], SystemT [Ch10], DefIE [BTN15], and QKBFly [Ng17] build upon (open) information extractors like ClausIE [CG13] and also perform the adaption, cleaning, and combination

stages of the knowledge base building process. Most of these approaches require high manual efforts to design extraction pipelines for each knowledge base and domain specifically. Google Squared could be used to create fact-tables similar to the ones we propose from web contents, but was unfortunately discontinued without publications about the underlying techniques. Closest to our work are recent approaches for query-driven on-the-fly knowledge base construction, such as QKBFly. Yet, QKBFly extracts general subject-predicate-object triples and does not populate a user-defined table as *WannaDB* does. The vision of INODE [Am21] is to provide an end-to-end data exploration system that is also able to include information from natural language texts. For this task, the knowledge base population approach LILLIE [Sm22] extracts triples from text domain-independently. However, the system has not been thoroughly evaluated for generalization to unseen domains. Recent approaches use transformer models to tackle information extraction tasks like relation extraction [EU21, CN21, Ng20] in an end-to-end fashion to avoid the errors accumulating in pipeline-based approaches. However, transformer-based methods are costly to train and suffer from issues like hallucination [Ma20]. A more explainable approach to information extraction is introduced by [Ko22, Re21] with a framework for learning text classifiers with a human-in-the-loop. Recently, [Sa22] introduced an interactive system that allows users to specify templates that are then used to perform zero-shot information extraction.

Text-To-Table. The idea of automatically transforming a text into a table was also approached by [WZL22] as text-to-table task, which inversely tackles the well studied table-to-text problem. Yet, their work is not directly comparable, since they assume that each text fills one or more entire tables, while we assume that a text collection fills one table in which each text corresponds to a row.

Template Filling & Named Entity Recognition. The goal of slot or template filling is similar to our objective [GS96], yet in contrast to our approach, most template filling approaches are specifically crafted for a fixed set of slots. A common approach to extract a fixed set of attributes from a text is to learn a named entity recognizer specifically for the desired entity types (e.g., [SJ19]). Named entity recognizers extract a set of entity types like organizations, locations, or products from natural language texts. However, the training requires a substantial amount of annotated data, and the learned system will not generalize to entity types not present in the training data. Some approaches (e.g., [Ch15, We19, Kh17]) attempt to avoid this problem by using active learning, which allows the learning algorithm to query the user, for example by selecting training instances that the user then labels by hand. Another strategy is distantly-supervised or weakly-supervised named entity recognition (e.g., [Fr17, Li20]). In contrast to our system, these approaches train named entity recognizers specifically for the desired set of entity types, whereas we use the output of conventional named entity recognizers to populate the user-provided attributes. Together with the interactive matching, this allows *WannaDB* to generalize to unseen domains without the costly training of domain-specific named entity recognizers.

Other Matching Tasks. Approaches for schema matching (e.g., [Hä20, He20]), are related to *WannaDB*, too, since we frame the mapping between the information extractors' output

and the user-provided list of attributes as a matching problem, but try to find correspondences between attributes and possible values, and not between columns or even full tables. Another recent approach focuses on matching texts to structured data, in particular also matching texts to table rows [ASP21]. Yet, this task differs from the matching task in *WannaDB*, as it assumes the tables are given, whereas in *WannaDB* a table is filled through the matching.

Entity Disambiguation & Cross Document Co-Reference Resolution. The surface form of an entity in a text is often not sufficient to uniquely identify it. Yet, knowing whether two mentions of the same type describe the same entity is relevant for correct grouping in our case, but also existing tasks like entity linking/knowledge base alignment. For the latter there are three main challenges (see [Dr10]): name variations (e.g., different mention forms, abbreviations, alternate spellings, and aliases), entity ambiguity (same written form for different entities), and absence (i.e., the text mentions a previously unknown entity). The last one is not relevant for our use-case, since we do not rely on a given KB but build tables only based on the current text collection. We can concentrate on the problem of ambiguity, i.e., decide, whether two nuggets that were matched as different rows of the same attribute are in fact the same or represent different concepts. The field of computing equivalence classes of textual mentions for the same entity is called cross-document co-reference resolution (CCR). It was, e.g., tackled by [DW15, KCP18, Ca21], but these existing approaches often concentrate only on entities from certain domains or of certain types (like events).

Prior Results of *WannaDB*. A first version of the matching component of *WannaDB* including an initial evaluation on two real-world data sets was published at [HBB21]. In this paper, we pick up the vision of the whole application cycle presented at [Hä21]. As such, we present the integration of the table extraction procedure of *WannaDB* into a full system. Moreover, compared to the original submission, we also developed a new interactive matching procedure where we leverage the human ability to quickly find patterns by presenting multiple guessed matches at once, which allows users to quickly correct wrong matches. Multiple ways to give feedback (confirm, fix, or mark that there is no match in the document) further enhance quality and flexibility of matching. A demo of the interactive GUI for this matching process was presented at [HBB22].

8 Conclusions

In this paper, we presented *WannaDB*, a novel tool to explore the contents of unstructured data (text) using SQL-like queries in an ad-hoc fashion and without the need to manually design extraction pipelines upfront. It builds on embeddings and a novel interactive query execution strategy and consists of components to infer the required table structure from the query, extract and organize the required information from the text, group results on the embedding level and execute the query. Our evaluation shows that the individual components of *WannaDB* can achieve similar performance to models trained on large data sets for partial or related tasks, and gives an impression of the end-to-end quality that makes *WannaDB* suitable for many exploratory use cases.

Acknowledgments

This work has been supported by the German Federal Ministry of Education and Research (BMBF) as part of the Project Software Campus 2.0 under grant ZN 01IS17050, by the BMBF and the state of Hesse as part of the NHR Program, the Hochtief project AICO (AI in Construction), and the HMWK cluster project 3AI. Finally, we want to thank hessian.AI, and the Centre Responsible Digitality (ZEVEDI) at TU Darmstadt, as well as DFKI Darmstadt for their support.

Bibliography

[Am21] Amer-Yahia, Sihem; Koutrika, Georgia; Braschler, Martin; Calvanese, Diego; Lanti, Davide; Lücke-Tieke, Hendrik; Mosca, Alessandro; de Farias, Tarcisio Mendes; Papadopoulos, Dimitris; Patil, Yogendra; Rull, Guillem; Smith, Ellery; Skoutas, Dimitrios; Subramanian, Srividya; Stockinger, Kurt: INODE: Building an End-to-End Data Exploration System in Practice. SIGMOD Rec., 50(4):23–29, 2021.

[ASP21] Ahmadi, Naser; Sand, Hansjorg; Papotti, Paolo: Unsupervised Matching of Data and Text. CoRR, abs/2112.08776, 2021.

[BTN15] Bovi, Claudio Delli; Telesca, Luca; Navigli, Roberto: Large-Scale Information Extraction from Textual Definitions through Deep Syntactic and Semantic Analysis. Trans. Assoc. Comput. Linguistics, 3:529–543, 2015.

[Ca21] Cattan, Arie; Johnson, Sophie; Weld, Daniel S.; Dagan, Ido; Beltagy, Iz; Downey, Doug; Hope, Tom: SciCo: Hierarchical Cross-Document Coreference for Scientific Concepts. In: 3rd Conference on Automated Knowledge Base Construction, AKBC 2021, Virtual, October 4-8, 2021. 2021.

[CG13] Corro, Luciano Del; Gemulla, Rainer: ClausIE: clause-based open information extraction. In: 22nd International World Wide Web Conference, WWW '13, Rio de Janeiro, Brazil, May 13-17, 2013. International World Wide Web Conferences Steering Committee / ACM, pp. 355–366, 2013.

[Ch10] Chiticariu, Laura; Krishnamurthy, Rajasekar; Li, Yunyao; Raghavan, Sriram; Reiss, Frederick; Vaithyanathan, Shivakumar: SystemT: An Algebraic Approach to Declarative Information Extraction. In: ACL 2010, Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, July 11-16, 2010, Uppsala, Sweden. The Association for Computer Linguistics, pp. 128–137, 2010.

[Ch15] Chen, Yukun; Lasko, Thomas A.; Mei, Qiaozhu; Denny, Joshua C.; Xu, Hua: A study of active learning methods for named entity recognition in clinical text. J. Biomed. Informatics, 58:11–18, 2015.

[CN21] Cabot, Pere-Lluís Huguet; Navigli, Roberto: REBEL: Relation Extraction By End-to-end Language generation. In: Findings of the Association for Computational Linguistics: EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 16-20 November, 2021. Association for Computational Linguistics, pp. 2370–2381, 2021.

[De19] Devlin, Jacob; Chang, Ming-Wei; Lee, Kenton; Toutanova, Kristina: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers). Association for Computational Linguistics, pp. 4171–4186, 2019.

[Dr10] Dredze, Mark; McNamee, Paul; Rao, Delip; Gerber, Adam; Finin, Tim: Entity Disambiguation for Knowledge Base Population. In: COLING 2010, 23rd International Conference on Computational Linguistics, Proceedings of the Conference, 23-27 August 2010, Beijing, China. Tsinghua University Press, pp. 277–285, 2010.

[DW15] Dutta, Sourav; Weikum, Gerhard: Cross-Document Co-Reference Resolution using Sample-Based Clustering with Knowledge Enrichment. Trans. Assoc. Comput. Linguistics, 3:15–28, 2015.

[El18] ElSahar, Hady; Vougiouklis, Pavlos; Remaci, Arslen; Gravier, Christophe; Hare, Jonathon S.; Laforest, Frédérique; Simperl, Elena: T-REx: A Large Scale Alignment of Natural Language with Knowledge Base Triples. In: Proceedings of the Eleventh International Conference on Language Resources and Evaluation, LREC 2018, Miyazaki, Japan, May 7-12, 2018. European Language Resources Association (ELRA), 2018.

[EU21] Eberts, Markus; Ulges, Adrian: An End-to-end Model for Entity-level Relation Extraction using Multi-instance Learning. In: Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, EACL 2021, Online, April 19 - 23, 2021. Association for Computational Linguistics, pp. 3650–3660, 2021.

[Fr17] Fries, Jason A.; Wu, Sen; Ratner, Alexander; Ré, Christopher: SwellShark: A Generative Model for Biomedical Named Entity Recognition without Labeled Data. CoRR, abs/1704.06360, 2017.

[GS96] Grishman, Ralph; Sundheim, Beth: Message Understanding Conference - 6: A Brief History. In: 16th International Conference on Computational Linguistics, Proceedings of the Conference, COLING 1996, Center for Sprogteknologi, Copenhagen, Denmark, August 5-9, 1996. pp. 466–471, 1996.

[Hä20] Hättasch, Benjamin; Truong-Ngoc, Michael; Schmidt, Andreas; Binnig, Carsten: It's AI Match: A Two-Step Approach for Schema Matching Using Embeddings. In: 2nd International Workshop on Applied AI for Database Systems and Applications (AIDB20). In conjunction with the 46th International Conference on Very Large Data Bases, Virtual, August 31 - September 4, 2020. 2020.

[Hä21] Hättasch, Benjamin: WannaDB: Ad-hoc Structured Exploration of Text Collections Using Queries. In: Proceedings of the Second International Conference on Design of Experimental Search Information REtrieval Systems, Padova, Italy, September 15-18, 2021. volume 2950 of CEUR Workshop Proceedings. CEUR-WS.org, pp. 179–180, 2021.

[HBB21] Hättasch, Benjamin; Bodensohn, Jan-Micha; Binnig, Carsten: ASET: Ad-hoc Structured Exploration of Text Collections. In: 3rd International Workshop on Applied AI for Database Systems and Applications (AIDB21). In conjunction with the 47th International Conference on Very Large Data Bases, Copenhagen, Denmark, August 16 - 20, 2021. 2021.

[HBB22] Hättasch, Benjamin; Bodensohn, Jan-Micha; Binnig, Carsten: Demonstrating ASET: Ad-hoc Structured Exploration of Text Collections. In: SIGMOD '22: International Conference on Management of Data, Philadelphia, PA, USA, June 12 - 17, 2022. ACM, pp. 2393–2396, 2022.

[He20] Hernández, Daniel Ayala; Hernández, Inma; Ruiz, David; Rahm, Erhard: LEAPME: Learning-based Property Matching with Embeddings. CoRR, abs/2010.01951, 2020.

[He21] Hendrycks, Dan; Burns, Collin; Kadavath, Saurav; Arora, Akul; Basart, Steven; Tang, Eric; Song, Dawn; Steinhardt, Jacob: Measuring Mathematical Problem Solving With the MATH Dataset. In: Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual. 2021.

[Ho20] Honnibal, Matthew; Montani, Ines; Van Landeghem, Sofie; Boyd, Adriane: spaCy: Industrial-strength Natural Language Processing in Python. 2020.

[KCP18] Kenyon-Dean, Kian; Cheung, Jackie Chi Kit; Precup, Doina: Resolving Event Coreference with Supervised Representation Learning and Clustering-Oriented Regularization. In: Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics, *SEM@NAACL-HLT 2018, New Orleans, Louisiana, USA, June 5-6, 2018. Association for Computational Linguistics, pp. 1–10, 2018.

[Kh17] Kholghi, Mahnoosh; Vine, Lance De; Sitbon, Laurianne; Zuccon, Guido; Nguyen, Anthony N.: Clinical information extraction using small data: An active learning approach based on sequence representations and word embeddings. J. Assoc. Inf. Sci. Technol., 68(11):2543–2556, 2017.

[Ko22] Kovács, Ádám; Gémes, Kinga; Iklódi, Eszter; Recski, Gábor: POTATO: exPlainable infOrmation exTrAcTion framewOrk. CoRR, abs/2201.13230, 2022.

[Le20a] Lembo, Domenico; Li, Yunyao; Popa, Lucian; Scafoglieri, Federico Maria: Ontology mediated information extraction in financial domain with Mastro System-T. In: Proceedings of the Sixth International Workshop on Data Science for Macro-Modeling, DSMM 2020, In conjunction with the ACM SIGMOD/PODS Conference, Portland, OR, USA, June 14, 2020. ACM, pp. 3:1–3:6, 2020.

[Le20b] Lewis, Mike; Liu, Yinhan; Goyal, Naman; Ghazvininejad, Marjan; Mohamed, Abdelrahman; Levy, Omer; Stoyanov, Veselin; Zettlemoyer, Luke: BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020. Association for Computational Linguistics, pp. 7871–7880, 2020.

[LH14] Liu, Zhicheng; Heer, Jeffrey: The Effects of Interactive Latency on Exploratory Visual Analysis. IEEE Trans. Vis. Comput. Graph., 20(12):2122–2131, 2014.

[Li20] Liang, Chen; Yu, Yue; Jiang, Haoming; Er, Siawpeng; Wang, Ruijia; Zhao, Tuo; Zhang, Chao: BOND: BERT-Assisted Open-Domain Named Entity Recognition with Distant Supervision. In: KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020. ACM, pp. 1054–1064, 2020.

[Ma14] Manning, Christopher D.; Surdeanu, Mihai; Bauer, John; Finkel, Jenny Rose; Bethard, Steven; McClosky, David: The Stanford CoreNLP Natural Language Processing Toolkit. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, System Demonstrations. The Association for Computer Linguistics, pp. 55–60, 2014.

[Ma20] Maynez, Joshua; Narayan, Shashi; Bohnet, Bernd; McDonald, Ryan T.: On Faithfulness and Factuality in Abstractive Summarization. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020. Association for Computational Linguistics, pp. 1906–1919, 2020.

[Mi18] Mikolov, Tomás; Grave, Edouard; Bojanowski, Piotr; Puhrsch, Christian; Joulin, Armand: Advances in Pre-Training Distributed Word Representations. In: Proceedings of the Eleventh International Conference on Language Resources and Evaluation, LREC 2018, Miyazaki, Japan, May 7-12, 2018. European Language Resources Association (ELRA), 2018.

[Ng17] Nguyen, Dat Ba; Abujabal, Abdalghani; Tran, Khanh; Theobald, Martin; Weikum, Gerhard: Query-Driven On-The-Fly Knowledge Base Construction. Proc. VLDB Endow., 11(1):66–79, 2017.

[Ng20] Nguyen, Minh-Tien; Le, Dung Tien; Son, Nguyen Hong; Minh, Bui Cong; Duong, Do Hoang Thai; Linh, Le Thai: Understanding Transformers for Information Extraction with Limited Data. In: Proceedings of the 34th Pacific Asia Conference on Language, Information and Computation, PACLIC 2020, Hanoi, Vietnam, October 24-26, 2020. Association for Computational Linguistics, pp. 478–487, 2020.

[Qi20] Qi, Peng; Zhang, Yuhao; Zhang, Yuhui; Bolton, Jason; Manning, Christopher D.: Stanza: A Python Natural Language Processing Toolkit for Many Human Languages. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations, ACL 2020, Online, July 5-10, 2020. Association for Computational Linguistics, pp. 101–108, 2020.

[Re21] Recski, Gábor; Lellmann, Björn; Kovács, Ádám; Hanbury, Allan: Explainable Rule Extraction via Semantic Graphs. In: Joint Proceedings of the Workshops on Automated Semantic Analysis of Information in Legal Text (ASAIL 2021) and AI and Intelligent Assistance for Legal Professionals in the Digital Workplace (LegalAIIA 2021) held online in conjunction with 18th International Conference on Artificial Intelligence and Law (ICAIL 2021). volume 2888 of CEUR Workshop Proceedings, CEUR-WS.org, Sao Paolo, Brazil (held online), pp. 24–35, 2021.

[RG19] Reimers, Nils; Gurevych, Iryna: Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019. Association for Computational Linguistics, pp. 3980–3990, 2019.

[RJL18] Rajpurkar, Pranav; Jia, Robin; Liang, Percy: Know What You Don't Know: Unanswerable Questions for SQuAD. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 2: Short Papers. Association for Computational Linguistics, pp. 784–789, 2018.

[Sa16] Sa, Christopher De; Ratner, Alexander; Ré, Christopher; Shin, Jaeho; Wang, Feiran; Wu, Sen; Zhang, Ce: DeepDive: Declarative Knowledge Base Construction. SIGMOD Rec., 45(1):60–67, 2016.

[Sa22] Sainz, Oscar; Qiu, Haoling; Lopez de Lacalle, Oier; Agirre, Eneko; Min, Bonan: ZS4IE: A toolkit for Zero-Shot Information Extraction with simple Verbalizations. In: Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: System Demonstrations. Association for Computational Linguistics, Hybrid: Seattle, Washington + Online, pp. 27–38, July 2022.

[SJ19] Sharma, Shreyas; Jr., Ron Daniel: BioFLAIR: Pretrained Pooled Contextualized Embeddings for Biomedical Sequence Labeling Tasks. CoRR, abs/1908.05760, 2019.

[Sm22] Smith, Ellery; Papadopoulos, Dimitris; Braschler, Martin; Stockinger, Kurt: LILLIE: Information extraction and database integration using linguistics and learning-based algorithms. Inf. Syst., 105:101938, 2022.

[We19] Wei, Qiang; Chen, Yukun; Salimi, Mandana; Denny, Joshua C.; Mei, Qiaozhu; Lasko, Thomas A.; Chen, Qingxia; Wu, Stephen; Franklin, Amy; Cohen, Trevor; Xu, Hua: Cost-aware active learning for named entity recognition in clinical text. J. Am. Medical Informatics Assoc., 26(11):1314–1322, 2019.

[Wo19] Wolf, Thomas; Debut, Lysandre; Sanh, Victor; Chaumond, Julien; Delangue, Clement; Moi, Anthony; Cistac, Pierric; Rault, Tim; Louf, R'emi; Funtowicz, Morgan; Brew, Jamie: HuggingFace's Transformers: State-of-the-art Natural Language Processing. ArXiv, abs/1910.03771, 2019.

[WZL22] Wu, Xueqing; Zhang, Jiacheng; Li, Hang: Text-to-Table: A New Way of Information Extraction. In: Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022. Association for Computational Linguistics, pp. 2518–2533, 2022.