WannaDB: Using ad-hoc SQL Queries for Information Retrieval

DT-DB42-M: The Question to or the Better Answer on 42? (Summer term 2023)

Jakob Ernesti
University of Bamberg
jakob.ernesti@stud.uni-bamberg.de

ABSTRACT

WannaDB is a new way for ad hoc SQL-like Queries over (unstructured) text collections. The system is able to extract information without prior training out of a variety of collections. The results are then provided as tables. The system works with already existing extractors to collect the necessary information. The innovative part, enabling an ad-hoc exploration, is two interactive steps through the extraction process: A feedback loop ensures that the automatic extraction matched the right mentioning with a requested attribute, and also the user is asked about grouping matches. The structure of the result tables is defined by the SQL query itself. The evaluation demonstrates that WannaDB is able to handle different kinds of existing text collections with good results, but still struggles with some queried attributes or collections. Summarizing, WannaDB is a new tool for Information Retrieval over hardly-known text collections that returns structured information.

1 INTRODUCTION

To be honest, nobody likes reading a pile of papers. If it is enjoyable and a paper is well-written, it is nice. But often, you have to read it multiple times to extract all the information. And yes, the author is aware of the irony present in it, but you are also still reading, so we can go a little further and look at the problem in more detail.
The most frustrating part is that you might not only read one paper to get an overview of a whole subject; you have to read a bunch and figure out which one might hide the most needed information. But it's not just scientists who struggle. Journalists, for example, have to read tons of reports or articles. However, time is a valuable resource, and often it is not clear at all if the information you need is in that text collection.
WannaDB is designed to satisfy information needs on (unstructured) text document collections in an ad hoc manner and to present the results in tables. The system was presented in Hättasch et al.[2] at the BTW'23. This paper used the example of journalists looking up information in a large document collection, and we might stick with that. It works within two big steps, that will be shown in more detail in the following chapters. First step is the extraction of so called information nuggets out of each documents with a Named Entity Recognition. After that, the user can ask a question with a SQL-like query. For example, they want to know the covid incident in a certain time period in Upper-Franconia or they examines the airlines with multiple incidents in the last years. At just this moment, the tables are formed based on the query's attributes. For tuning the results, WannaDB uses the user's input within an interactive user feedback loop.
Of course, there are quite many Extraction Systems and some of them also sort information into tables. But the big advantage of WannaDB is its flexibility of use, as it is not a domain specific approach. Other systems are often trained on special domains and sets or, if they are able to extract tables too, use pre-designed templates. On contrast, this system is able to adapt to unknown data sets after the nugget extraction and is even capable to handle difficult disambiguation problems with the interactive user feedback. But it has also significant limitations that should be known before implementing it. The system will only answer single-table queries, because it builds just one table on a document collection with each row representing one document. The table's attributes must also be given within the SQL-query. But most of all, one must be aware that the result shown may not be complete or values might be dirty (maybe through incorrect clustering). And, of course, you have to be able to form a SQL-like statement to use it.

2 STAGE 1: OFFLINE EXTRACTION

The first step can be done as a pre-processing before any user interaction. Even if it might take its time, especially for big collections, it only has to be done once for each collection as long as it remains unchanged. Now so-called nuggets are created, which will be used for query processing later. Each document of a collection is processed with a named entity recognition tool; almost any tool can be used, as long as it can create label-mention-pairs. It is even possible to use multiple extractors together at the same time. This step already represents a conceptual bottleneck for WannaDB, as only the information contained in the nuggets can be processed further. The nuggets store following signals: Label expresses the

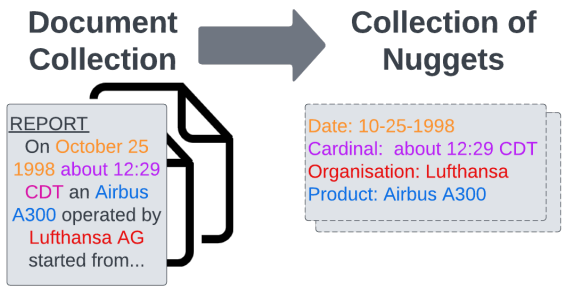

Figure 1: Offline extraction: Extract all nuggets that might be relevant (once per document, independent of information need).[2, 161]

entity type defined by the extractor (e.g. "region"), mention is the entity's representation in the text (e.g. "Upper-Franconia"), context is the specific sentence with the mention and the position describes

the position of the mentioning in the document. After the extraction, the information is preprocessed, to get normalized data from e.g. dates.

3 STAGE 2: INTERACTIVE QUERY EXECUTION AT RUNTIME

3.1 Creating a schema by the query's attributes

One main key of WannaDB are the interactive processes. A rough idea how this interactions looks like from a user's perspective is shown in a demonstration by the research group presenting a quite similar system, ASET. It is a quite related approach to structure text collections in tables using the same technologies as in WannaDB but the attributes are not given within a SQL-like query.[1]

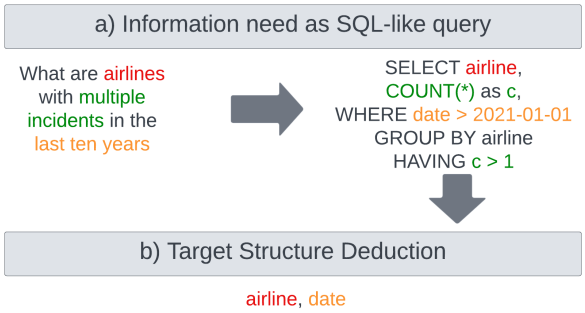

Figure 2: Online Interactive Query Execution part 1: Create table[2, 161]

WannaDB supports two classes of SQL-like queries: A simple extraction of facts (arguments) and grouping / aggregations. The tables for answering the user's query are only created, when the request is made. The table's structure is given by the query's attributes and aggregation operations or group-by-statements. In the default table used, every column represents one attribute and every row one document. However, it is important to note that users are required to possess knowledge about the text collection's structure. For instance, when inquiring about the authors of documents, the authors must also be listed within the respective documents. This structured approach, with one row per document, implies that the requested entities must be individually listed in separate documents. For example, if a user requests information on Covid incidences per state, there must be a dedicated document for each state in order to provide the desired data.

Then the table is filled with the correct content based on the information extraction available in the nuggets. However, the assignment by machine learning algorithms requires training with corresponding training data. WannaDB establishes the semantic similarity between the nuggets and the table attributes through user interaction in order to be able to encounter different subject areas in an ad hoc manner.

For every query attribute, WannaDB calculates the semantic closeness (*cached distance*) between the attribute (e.g. "Government District") and nugget (e.g. "region").[1] The lowest distance within a document is considered the document's guessed match at that moment.

3.2 Interactive user feedback loop

The interactive user feedback phase is the true innovative point about this approach. To ensure WannaDB selects the nuggets relevant to the query, it assumes that the user has domain knowledge, even if limited. In this way, the software overcomes the problem of limited contextual knowledge and no specialized training on certain collections by simply sticking to the text and relying on user's knowledge.

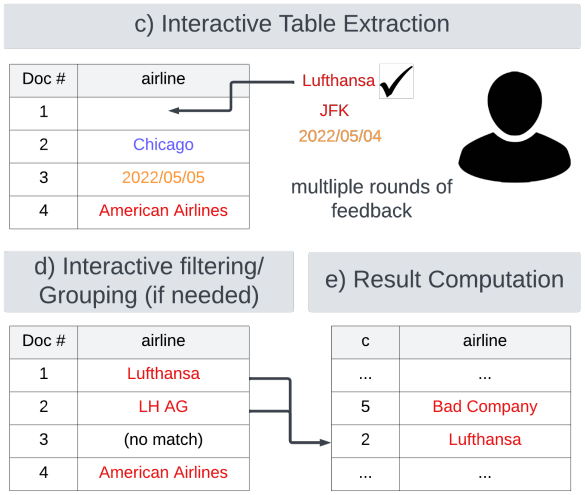

Figure 3: Online Interactive Query Execution part 2: Fill table to satisfy information need (repeat/refine as desired)[2, 161]

A ranked list of the documents with their current guessed match is presented to the user. They may confirm the guess, may select another nugget from the the specific document or can reject the it, if the document does not contain any searched content. The user also gets a first impression of the current state of the quality of the query results. Additionally, they can independently terminate the interactive phase at any time, ensuring no frustration from continuously 'work' for the system, the evaluation shown below demonstrates that the WannaDB benefits quite quickly from sometimes only a few user inputs.

The feedback is continuously used to update the current guessed matches and also updates the list presented to the user. In this way, incorrect matches can be corrected and results with a low probability can be confirmed or rejected. In case of a confirmed match, the feedback is used to update the cached distance of all remaining nuggets.[2]

3.3 Threshold Adjustment

A threshold is used for two purposes. First, it ensures a guess that is too uncertain will not be used and it might be better to leave a cell

[1]The cached distance is represented with the cosinus distance between the nugget's label embedding and the attributed name embedding.

[2]The distance between two nuggets is calculated by the mean of the cosine distances between their own signal embeddings.

empty. Second, it presents guesses with maximum benefit in the feedback loop to the user. The threshold is not fixed. It is adjusted permanently through the feedback phase. If the threshold value is higher than the distance of a nugget confirmed by the user, the threshold value is raised. On the other hand, if the user confirms that there is no match in the document, and the distance of the current guess is lower than the threshold value, the threshold will be decreased. The idea is, if the user confirms a nugget that is above the threshold, then all nuggets that fall between the confirmed and the threshold should be an adequate result.

3.4 Interactive Filtering & Grouping

The Filtering / Grouping Process is designed like the feedback loop above to use the user's knowledge through an interactive process. The paper does only explain grouping, but the filtering shall work similar. First, entries with the identical string representation will be merged without any user interaction. The other ones are clustered by the distance between the nuggets. The system will present all members of a pair of clusters that might be merged to the user for approval. For a better result and to keep the necessary interactions small, WannaDB choose to present a pair with a higher distance. If they confirm, all clusters shown and any clusters with a lower distance are merged and additionally all distances will be computed again. If the user rejects the suggestion, the system will look up for a better threshold for the distance by using a binary search pattern.

4 EVALUATION

The authors tested WannaDB's abilities to show of their system's performance. Therefore they used three different collections: Summaries of the aviation accidents reports by the United States Transportation Safety Board, summaries of daily Covid reports by the german RKI and three T-REx data sets about countries, Nobel Prize laureates and skyscrapers. All collections provide different challenges. For the Covid reports the search for numerical attributes in texts was supposed to be the difficulty, while the requested attributes in the various T-REx datasets were rarely all in one document. The system should therefore have to cope with various problem cases, which are, of course, also part of the requirements for such an 'Jack-of-all-trades' system.

The first evaluation involved conducting end-to-end queries to test its performance, using metrics such as Recall, Precision, and mean Jaccard Index, similar to classic Information Retrieval systems. The authors simulated 20 user interactions for each query. The paper presents the initial results of two queries over the T-REx nobel laureate set and the country set. The extraction differs slightly from the actual values, especially concerning the count(*) function. However, the authors note that other extraction systems do not provide 100% accuracy either and users gain in any case a faster overview of existing data in the collections.

One of the most important steps in WannaDB is the user feedback for improving the results. To evaluate the effectiveness of the interactive extraction, the authors compared it against structured data extracted from the test collections using BART, a very reliable pre-trained system. They fine tuned one BART model for each collection separately and further used another already-tuned one. In

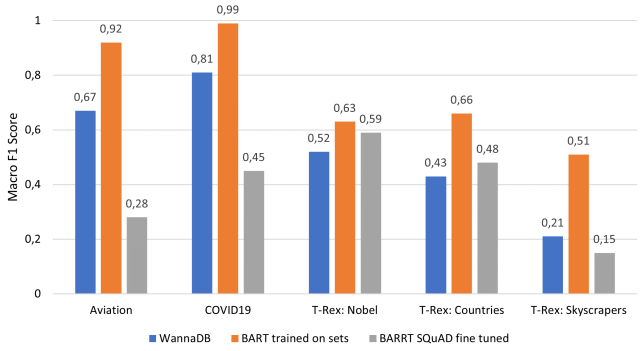

Figure 4: F1-Score for table filling results from WannaDB, BART trained on explicit sets and BART tuned with SQuAD with different sets[2, 170]

comparison to a run with 20 user interactions the fine-tuned system yielded the best results. The performance of the already-tuned model was very dependent on the underlying data sets. Although WannaDB did not always achieves the best results and with some data sets also underwent the system not being finely tuned, the authors highlighted that their system was able to perform well across many different datasets and without needing prior training. This demonstrates that WannaDB meets the requirement of being able to answer ad hoc queries effectively.

The last experiment tested the performance of WannaDB on a per-attribute level to show stable accuracy and recall for very different attributes. The results demonstrate, that WannaDB's tables hat lower quality for certain attributes compared to others. The reason might be the quality of current extractor models that may struggle with extracting some kind of information, especially in fields with very domain-specific languages or different wordings to describe the same concepts. Another factor mentioned is the occurrence of attributes in only a few documents, which made the table filling challenging.

The evaluation of the user interaction demonstrated, that for high performance the system often needs just 5 to 10 interactions to improve its score. In some cases, even a single interaction gained very good results. However, there were also attributes where additional interaction did not significantly improve the output. Overall, even if WannaDB's user interaction is a crucial part, only a few iterations are enough to gain good results.

In terms of scalability the group evaluates the response time within the user interaction and the compute time for the Offline Extraction. They assume that these might be the most crucial time costs for potential users, as the system should be able to extract nuggets from large document sets within an acceptable amount of time. The latency to compute the user input and present the next set of guessed matches was only a fraction of a second on average for a single user interaction, so that another one could have been done immediately. Even with the largest document set with over 100,000 nuggets, the extraction took approximately 48 minutes. These seemed tolerable time costs and exemplified that the system can handle even larger collections within a reasonable time frame for users. However, the calculations were done with a relatively

powerful desktop computer, which exceeds the processing power of many common PCs. A comparison with weaker machines was not made.[3]
The evaluation results indicate that WannaDB can effectively handle the majority of queries on the aviation and covid collections, producing satisfactory outcomes. These collections closely align with the use case of journalists seeking information. However, WannaDB may face difficulties with specific queried arguments and certain text collections. Despite these limitations, the authors of the paper express their overall satisfaction with the evaluation results. It is important to note that while a specialized system trained on specific data will always outperform WannaDB, it still serves its purpose by providing a general impression of the data and enabling ad hoc information retrieval.

5 CONCLUSION

Is this the end of reading? Probably not! WannaDB, as it was presented, allows a user to get an impression of the information in a document collection, mainly because there are queried terms and even whole collections that are more difficult to handle than others. And you always need to have at least some domain knowledge, at least to formulate a proper query. The system design is aware of this and is using the users knowledge of the world and adapt this through an interactive cycle to improve its results. But also to formulate a SQL-query that can proceed further very well users will need at least some information about the text collection (and of course, must have some basic SQL knowledge). Beside that, one bottleneck is the nugget extraction: If the extractors struggle with the documents, the output might be poor as well. Nevertheless the evaluation proofed that WannaDB can handle most queried attributes in an ad hoc manner. Besides the nugget collection at 'index time' the system needs no specific training. The authors demonstrated how a query language made for relational databases can be used for Information Retrieval approaches. The results presented as a table that the user are able to shape within the SQL-query is a good and very unique idea. To the best of the author's knowledge, there is no system that works in the way WannaDB works and this paper's author was not able to find any, either.[2, 167]
Briefly, WannaDB stands out as a valuable tool due to its ability to handle ad hoc queries and generate tables based on those queries. It offers unique features that can complement other related approaches.[4]

REFERENCES

[1] Benjamin Hättasch, Jan-Micha Bodensohn, and Carsten Binnig. 2022. Demonstrating ASET: Ad-Hoc Structured Exploration of Text Collections. In *Proceedings of the 2022 International Conference on Management of Data* (Philadelphia, PA, USA) *(SIGMOD '22)*. Association for Computing Machinery, New York, NY, USA, 2393–2396. https://doi.org/10.1145/3514221.3520174

[2] Benjamin Hättasch, Jan-Micha Bodensohn, Liane Vogel, Matthias Urban, and Carsten Binnig. 2023. WannaDB: Ad-hoc SQL Queries over Text Collections. In *BTW 2023*, Birgitta König-Ries, Stefanie Scherzinger, Wolfgang Lehner, and Gottfried Vossen (Eds.). Gesellschaft für Informatik e.V. https://doi.org/10.18420/BTW2023-08

[3] If a computer with following specifications is a 'normal' consumer desktop machine, might be questionable: AMD Ryzen 9 3900X; 32GB @3000MHz RAM; NVIDIA GeForce RTX 2070 SUPER 8GB VRAM[2, 173]
[4] The WannaDB code is available at https://github.com/DataManagementLab/wannadb/