From Natural Language Processing to Neural Databases

James Thorne
University of Cambridge
Facebook AI
jt719@cam.ac.uk

Majid Yazdani
Facebook AI
myazdani@fb.com

Marzieh Saeidi
Facebook AI
marzieh@fb.com

Fabrizio Silvestri
Facebook AI
fsilvestri@fb.com

Sebastian Riedel
Facebook AI
University College London
sriedel@fb.com

Alon Halevy
Facebook AI
ayh@fb.com

ABSTRACT

In recent years, neural networks have shown impressive performance gains on long-standing AI problems, such as answering queries from text and machine translation. These advances raise the question of whether neural nets can be used at the core of query processing to derive answers from facts, even when the facts are expressed in natural language. If so, it is conceivable that we could relax the fundamental assumption of database management, namely, that our data is represented as fields of a pre-defined schema. Furthermore, such technology would enable combining information from text, images, and structured data seamlessly.

This paper introduces *neural databases*, a class of systems that use NLP transformers as localized answer derivation engines. We ground the vision in NeuralDB, a system for querying facts represented as short natural language sentences. We demonstrate that recent natural language processing models, specifically transformers, can answer select-project-join queries if they are given a set of relevant facts. However, they cannot scale to non-trivial databases nor answer set-based and aggregation queries. Based on these insights, we identify specific research challenges that are needed to build neural databases. Some of the challenges require drawing upon the rich literature in data management, and others pose new research opportunities to the NLP community. Finally, we show that with preliminary solutions, NeuralDB can already answer queries over thousands of sentences with very high accuracy.

PVLDB Reference Format:
James Thorne, Majid Yazdani, Marzieh Saeidi, Fabrizio Silvestri, Sebastian Riedel, and Alon Halevy. From Natural Language Processing to Neural Databases. PVLDB, 14(6): 1033-1039, 2021.
doi:10.14778/3447689.3447706

Proceedings of the VLDB Endowment, Vol. 14, No. 6 ISSN 2150-8097.
doi:10.14778/3447689.3447706

1 INTRODUCTION

Researchers have long considered the application of neural nets to data management problems, including learning indices [16], query optimization, data cleaning and entity matching [20, 23, 32]. In applying neural networks to data management, research has so far assumed that the data was modeled by a database schema.

The success of neural networks in processing unstructured data such as natural language and images raises the question of whether their use can be extended to a point where we can relax the fundamental assumption of database management, which is that the data we process is represented as fields of a pre-defined schema. What if, instead, data and queries can be represented as short natural language sentences, and queries can be answered from these sentences? Furthermore, what if relevant data from images can be seamlessly combined with text and structured data?

This paper describes a vision for neural databases and preliminary empirical evidence of its potential. Neural databases offer several benefits that database systems have struggled to support for decades. The first, and most important benefit, is that a neural database has no pre-defined schema. Therefore, the scope of the database does not need to be defined in advance, and any data that becomes relevant as the application is used can be stored and queried. The second benefit is that updates and queries can be posed in a variety of natural language forms, as is convenient to any user. In contrast, a traditional database query needs to be based on the database schema. Even when the data is modeled with a more flexible formalism such as RDF, there is still a single name for any given relation, and that name needs to be used in updates and queries. Third, with recent advances in machine translation, the language of queries and answers can be different from the language of the data in the neural database. A final benefit comes from the fact that the neural database is based on a pre-trained language model that already contains a lot of knowledge which can be exploited to generate better answers to more diverse queries.

To ground our vision, we built NeuralDB, a database system in which updates and queries are given as short natural language sentences. Figure 1 shows example facts and queries that NeuralDB can answer. Our preliminary experiments show that NeuralDB can answer select-project-join-aggregate queries over thousands of natural language sentences with very high accuracy.

By nature, neural databases are not meant to provide the same correctness guarantees of a traditional database system, i.e., that the answers to queries satisfy the precise binary semantics of the query

language. Hence, to be clear about the scope of the vision, neural databases should not be considered as an alternative to traditional databases in applications where such guarantees are required.

Given their benefits, neural databases are well suited for emerging applications where the schema of the data cannot be determined in advance and data can be stated in a wide range of linguistic patterns. A family of such applications arise in the area of storing knowledge for personal assistants that are currently available for home use and in the future will accompany Augmented Reality glasses. In these applications, users store data about their habits and experiences, their friends and their preferences, and designing a schema for such an application is impractical. Another class of applications is the modeling and querying of political claims [34] (with the goal of verifying their correctness). Here too, claims can be about a wide variety of topics and expressed in many ways.

The key technical insight underlying neural databases is that state of the art transformer models [36] can be used for query processing. More specifically, transformers can combine multiple facts, each expressed as a short natural language sentence, to answer simple queries. In effect, transformers can execute joins, selections, and projections. However, transformers are limited in the size of the inputs they can realistically process, and they fail when higher-level mathematical reasoning is involved, such as performing aggregation. Hence, to answer database-like queries, we need to run multiple transformer instances and combine their answers appropriately. These limitations raise the first two technical challenges for our vision: (1) finding suitable sets of facts from the database to feed to each transformer instance, and (2) further processing the answers of each transformer instance (e.g., union and aggregation) to produce the answer to the query.

In Section 2 we define the functionality of a neural database that answers queries over facts described as natural language sentences and describe the underlying NLP technology and its limitations. In Section 3 we describe NeuralDB, our first prototype neural database that provides initial solutions to the challenges described above. We also describe a set of experiments that validate the promise of transformers and of NeuralDB. Section 4 describes additional research challenges for realizing the vision of neural databases.

2 A NEURAL DATABASE FOR TEXT

We ground our vision for neural databases by building a specific instance, NeuralDB, a system for querying facts described in text. NeuralDB will demonstrate the challenges involved in building neural databases. Section 2.1 defines the functionality of NeuralDB. In Section 2.2, we explain how NLP technology, namely, transformers over pre-trained language models, can provide a key processing unit for realizing NeuralDB, and point out the key challenges in adapting transformers to data management.

2.1 Problem definition

In NeuralDB, data and queries are represented as sentences in natural language, providing two of the key benefits of neural databases. First, the database has no pre-defined schema – users can mention any relationship of interest. Second, the database is usable by a broader set of users because updates and queries can be specified in whatever linguistic form is most convenient to the user.

Facts: (4 of 50 shown)
Nicholas lives in Washington D.C. with Sheryl.
Sheryl is Nicholas's spouse.
Teuvo was born in 1912 in Ruskala.
In 1978, Sheryl's mother gave birth to her in Huntsville.
Queries:
Does Nicholas's spouse live in Washington D.C.?
`(Boolean Join)` ⟶ `TRUE`

Who is Sheryl's husband?
`(Lookup)` ⟶ Nicholas

Who is the oldest person in the database?
`(Max)` ⟶ Teuvo

Who is Sheryl's mother?
`(Lookup)` ⟶ `NULL`

Figure 1: In NeuralDB, facts and queries are posed with short natural language sentences. The above queries are answered by our prototype.

Data: Formally, the data in NeuralDB is a set of sentences. Intuitively, a sentence corresponds to a single fact, such as Sue is Mary's mom, or Gustavo likes espresso. But in many situations, especially when updates to the database are spoken by users, it is more convenient for sentences to express multiple facts, such as Kiara is married to Kyrone and they have 3 kids. We refer to the latter sentences as composite and the former as atomic. We mostly consider atomic sentences in this paper. In its current form, NeuralDB is aimed to support knowledge expressed in short sentences, not entire paragraphs.

Queries: Formally, a query Q over a database, D, produces a set of answers: $Q(D) = \{a_1, \ldots, a_l\}$. While queries are formulated in natural language, we only consider queries that, if translated to SQL, would involve a select-project-join (SPJ) component followed by an aggregation (predicates involving number comparisons are left for future work). In our discussion, *join queries* are ones that require combining two or more sentences to generate the answers (e.g., Who works in a company in France?), and *aggregation queries* are ones that end with an aggregation operator (e.g., how many kids does Pat have?). A *lookup query* is a query where each answer comes from a single fact in the database (e.g., Who is Susan's husband?), whether there is a single answer or several. If the query returns True/False, we refer to it as a Boolean query. Note that in our context, lookup queries are non-trivial because facts in the database are expressed in a variety of linguistic forms. We use the term *support set* for an answer to a query (or a sub-query) to refer to a minimal set of database facts that are needed in order to derive that answer.

2.2 Answering queries using NLP

The NLP problem of question answering from external knowledge such as Wikipedia, (a.k.a. *open-book QA*, or QA) is the most natural point to explore the application of NLP models to NeuralDB.

Common QA tasks, such as SQuAD, [28] and DROP [9], require models to answer reading comprehension queries such as When were the Normans in Normandy? and Which kicker had the most field goals? by extracting a contiguous span of text or performing reasoning over a given passage of text.

It is important to highlight two key differences between QA and NeuralDB, which will have a significant impact on the ultimate solutions. First, in QA, the fact needed to answer a given question is typically located in a paragraph or document with multiple sentences about the same subject where this additional context may help information recall. NeuralDBs do not enjoy this locality property because a query may be dependent on multiple facts that can be anywhere in the database. In fact, models in retrieval-based tasks that use Wikipedia as a knowledge source may also leverage the (unique) document title to aid retrieval (for example, predicting the document key with autoregressive retrieval [7]).

The second difference is that QA methods are designed to answer queries where a value is given as an input, and the expected result is small (e.g., *foreign minister of Malaysia*). In contrast, database queries can return sets (e.g., *cities of Malaysia*) and aggregations applied to these sets (such as Count and Max).

Keeping these differences in mind, we now describe how NLP techniques can be used to answer limited forms of database queries.

2.2.1 Pre-trained language models and transformers. Natural Language Processing (NLP) has made impressive progress in recent years by building on transformer architectures and pre-trained language models. In addition to QA, such models have led to major improvements on tasks such as text summarization and machine translation. Pre-trained language models such as BERT [8], GPT-3 [6], RoBERTa [21], T5 [27] are neural models trained on large corpora of text. The models are trained by randomly removing certain tokens from the corpus and training the neural net to predict the missing token, given its context (i.e., the words preceding and following the missing token). At an intuitive level, these models obtain two kinds of knowledge (1) the ability to make predictions independent of the exact linguistic form in which knowledge is stated [24, 33], and (2) some world knowledge that is mentioned frequently in text (e.g., London is the capital of the UK) [6, 26].[1] Pre-trained language models are usually fine-tuned to a given task. For example, for question answering from text, the system would be further trained with a set of (question, answer) pairs in order to produce the final model. Importantly, since the pre-trained language models capture world knowledge [26], fewer examples are needed for the fine-tuning compared to training a model from scratch.

The Transformer model [36] is the most common neural architecture to operate on pre-trained language models. Transformers [36] take as input a sequence of symbols $\mathbf{x} = (x_1, \ldots, x_n)$. They are typically trained in one of two configurations: encoder only or encoder-decoder. In the former, each token is encoded to a vector representation that is used to predict a label. In the latter, used in sequence-to-sequence applications (e.g., question answering or machine translation), the decoder produces the output sequence.

[1]Note that the second type of knowledge can also lead to implicit biases. We address this point in Section 4

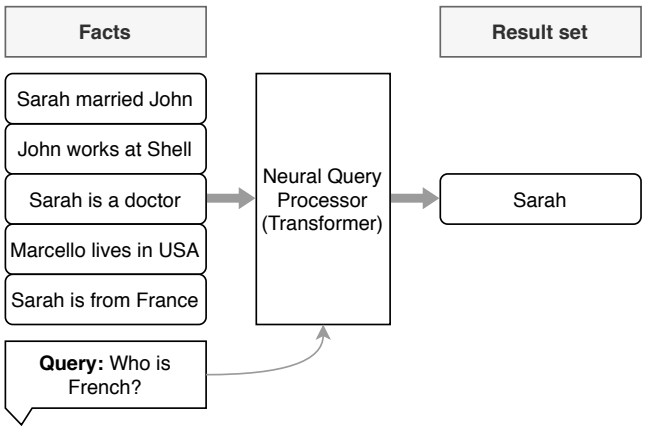

Figure 2: Prototype neural query processor using a T5 Transformer. The facts and the query are concatenated and given as input to the transformer. Our results show that transformers can answer lookup and join queries when given the support set of facts needed to generate an answer, but that the architecture does not scale to many facts.

In both configurations, the transformer works in two phases. In the first phase, the transformer encodes the input into an intermediate representation $\mathbf{Z} = (z_1, \ldots, z_n)$ where the dimension of the vector is fixed, typically where $z_i \in \mathbb{R}^{768}$. In the second phase, the transformer decodes $\mathbf{z}$ to produce the output. For example, in sequence-to-sequence generation, the output would be a sequence of tokens $\mathbf{y} = (y_1, \ldots, y_l)$, ending with a special token.

2.2.2 Using transformers to answer queries. Transformers have the capability to answer simple queries over a small amount of data expressed in natural language. To adapt them to NeuralDB, we would first train the transformer with triples of the form (D, Q, A), where D is a set of facts, Q is a query, and A is the answer to Q over D. As shown in Figure 2, at query time we feed the facts and the query to the transformer, and the output is the result set. In Section 3 we show that transformers produce high quality results even for queries that require joining multiple sentences.

Given the potential of transformers, we need to address two challenges in order to adapt transformers to large-scale query answering: organizing the data so it can be fed into transformers in smaller chunks, and planning to answer a query from multiple smaller query processing components. We consider each in turn.

Challenge 1: Support set generation. State-of-the-art transformer models cannot accept large inputs because their memory requirements are quadratic in repsect to the size of their inputs. In practice, it is common to use a maximum input size of 512 or 1024 tokens. Hence, we cannot encode the entire database using these models.

The implication for neural databases is that we need to run multiple transformer instances. Each instance needs a subset of facts from the database as input – a support set – from which the output can be derived. For example, for the query people who live in London, each support set would include enough facts to derive whether a person lives in London or not.

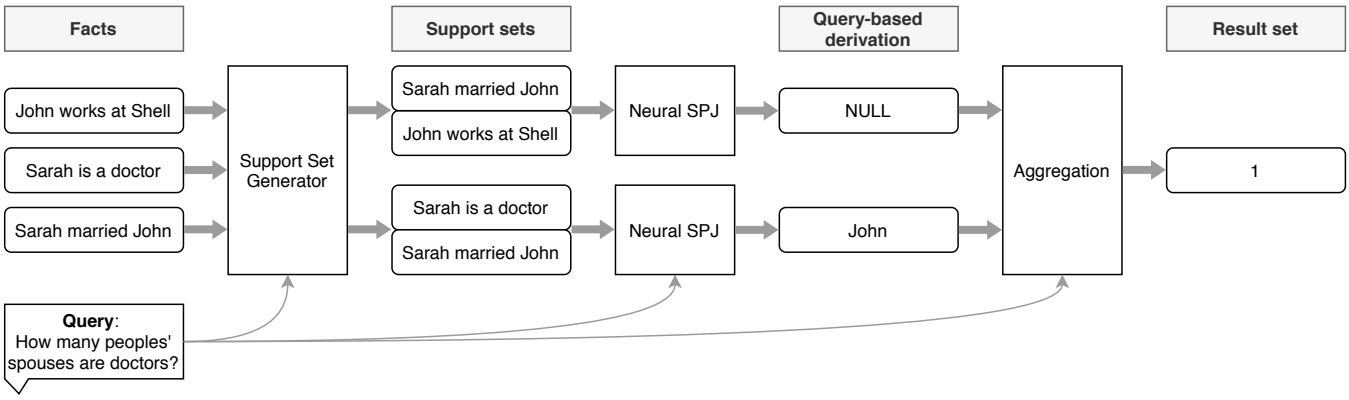

Figure 3: Query answering in NeuralDB. The support set generator creates small sets of facts that are each fed into a separate Neural SPJ operator that runs a single transformer. The results of the individual Neural SPJ operators are either unioned to produce the result or passed on to a traditional aggregation operator.

In the context of QA, this challenge is addressed by employing an information retrieval component that extracts a small subset of the relevant facts from the text corpus and feeds them to the transformer. The information retrieval component can either be a simple one (e.g., BM25), or trained jointly with the model to learn to extract relevant parts of the corpus [12, 15, 18, 25, 34, 39]. However, in our context, we need to create multiple support sets, each of which is fed into a transformer.

The retrieval problem is complicated by the fact that answering a query may require conditional retrieval from the database. For example, to answer the query Whose spouse is a doctor? we'd first need to fetch spouses and then their professions. In the NLP community, this is known as multi-hop query answering [5], which has recently become an active area of research but restricted to the case of retrieving or generating a single answer. While research in NLP focuses on expanding a single set of information to generate a single answer for a question, in NeuralDB we may need to perform multi-hop retrieval for sets of facts to support aggregation. For example, in the query How many people's spouses are doctors? multi-hop retrieval is required for every spouse fact.

Challenge 2: Answer composition. Since the transformer is performing local query answering, we need a component that can assemble the results of the individual transformers to produce an answer for the full query. Furthermore, as we show in Section 3, transformers are not adept at computing aggregates. Hence, if the query contains aggregation, the query plan needs to compute an intermediate result set that then gets aggregated using a traditional operator. In the next section, we describe an architecture that combines multiple transformers and is able to answer queries that can be expressed as a select-project-join followed by an aggregate. For more complicated queries, we may need a component that builds an explicit query plan as in a traditional database system.

3 A PROOF OF CONCEPT

We now describe NeuralDB's architecture that includes initial solutions to the two challenges outlined above. We experimentally validate that transformers can locally answer certain classes of database queries and then show that NeuralDB can build on that to answer queries on large databases. The architecture of NeuralDB is shown in Figure 3 and is based on the following ideas.

(1) Does Nicholas's spouse live in Washington D.C.?
{Nicholas lives in Washington D.C. with Sheryl.,
Sheryl is Nicholas's spouse.} ⟶ `TRUE`

(2) Who is the oldest person in the database?
{Teuvo was born in 1912.} ⟶ `(Teuvo, 1912)`

(3) Does Nicholas's spouse live in Washington D.C.?
{Teuvo was born in 1912.} ⟶ `NULL`

Figure 4: Examples of the intermediate results that are produced by the Neural SPJ operator.

Running multiple transformers in parallel: As noted earlier, in practice, transformers can only take a relatively small input. Hence, to scale to larger data sets, NeuralDB runs multiple copies of a *neural SPJ* operator in parallel, each outputting structured results. When queries don't involve aggregation, the union of the outputs of the neural SPJ operators is the answer to the query. When the query does involve aggregation, these machine-readable outputs are fed into the aggregation operator. Figure 4 shows examples of the output of the neural SPJ operator (and therefore the form of the training data it requires).

The facts given to each transformer are given by a support set generator (SSG). Intuitively, the support set of an answer to an SPJ query includes the leaves of its derivation tree. The SSG creates sets of facts that contain support for an answer, with minimal irrelevant facts.

Aggregation with a conventional operator: Since the neural SPJ was designed to output structured results, our architecture can use a separate conventional aggregation operator. Using a separate aggregation overcomes the limitation on transformers concerning aggregation queries. The aggregation operator is selected through a classifier that maps a query to an aggregation function.

3.1 Implementation

We implemented a version of NeuralDB following the above architecture. We developed a support set generator (described in detail in [35]) that incrementally builds support sets by starting with single facts and adding more. The decision about which facts to add to a support set is formulated as a classification problem that can be trained by the neural SPJ itself. Note that training this support set generator did not require additional labeled data as a distant supervision signal comes from measuring how changing the input facts change the answer generated by the model.

3.1.1 Experimental setup. Training any neural system requires a supervision signal. We generate training data in a controlled manner by converting knowledge base tuples from Wikidata [37] (e.g., (Zuckerberg, ceoOf, Facebook)) to natural language sentences (e.g, The CEO of Facebook is Zuckerberg). Because of the scale of Wikidata, it is possible to generate large numbers of training instances about a wide range of relationships and entities. For our first experiment (testing the transformers themselves, Section 3.2.1), it is sufficient to train with triples of the form (D, Q, A), where D is a set of facts, Q is a query, and A is the answer to Q over D. However, when testing NeuralDB (Section 3.2.2) we need slightly more refined labels in order to train the neural SPJ to generate intermediate results (see Figure 4).

3.1.2 Training data generation. For this proof of concept, we consider simple template-based data generation over 27 relations from Wikidata and their inverse. For every Wikidata relationship we consider, we create linguistic variations in both the facts and the queries by constructing multiple natural language templates that vary pronouns, temporal expressions, and phrasing of the fact and query. Each template has a placeholder for the subject and the object, e.g., $O is the place of birth of $S. We then generate different data instances by substituting entities from Wikidata for the placeholders in creating training, validation, and held-out test sets, which are disjoint by subject. Our data was generated using 632 templates: 115 for facts and 517 for the different query types over the 27 relations, for each fact and query, randomly choosing an appropriate template for the relation or its inverse. The held-out test database contains 8400 facts and 14000 queries over this database with reference answers that are used for scoring. This is orders of magnitude larger than can be feasibly encoded with the NLP-only model indicated in Figure 2. For training, we generate approximately 80,000 training instances, ensuring an equal balance for each of the aggregation operations. For aggregation, we consider Count, Min, and Max. The supervision for the classifier which selects the aggregation function is part of the templates.

3.2 Results

3.2.1 Testing transformers. We demonstrate the limitations of a standalone transformer model designed for conditional language generation when applied to neural query processing (Figure 2). Without adaptation, this model can only feasibly encode a maximum of 50 facts. We use a subset of our training data to fine-tune a T5 transformer model [27], constructing databases of 50 facts to avoid memory limitations when encoding the entire database. We jointly encode all facts by concatenating them with the query.

Table 1: Exact match scores for NeuralDB. Note that IR(k=5) with a single T5 transformer cannot accurately answer aggregation and join queries. We use 200 parallel SPJ operators (4xV100 GPUs, batch size = 50) over a database with 8400 facts, averaged over 14000 queries.

Method	Exact Match (%)				
	Count	Min/Max	Sets	Atomic	Joins
NeuralDB	**79.45**	**100.00**	**91.91**	**97.90**	**79.29**
TF·IDF+T5	31.06	0.00	44.25	98.05	68.02
DPR+T5	38.07	21.19	54.55	97.38	58.64

Our results show that for lookup and join queries the model attained near perfect scores (above 99% exact match) on the template-generated data. The fact that the model had high scores for queries that require the combination of multiple facts indicates that the transformer is able to combine information from multiple sources when generating an answer to the user query. Furthermore, generating correct answers when encoding the entire database without filtering out the irrelevant facts suggests that the model is resilient to exposure to irrelevant facts. However, the model performs poorly for queries that require an aggregation (for queries requiring a count operation, exact match was 57%) or when the query result is a large set. Importantly, the results indicate that the model can be robust to simple linguistic variations when processing queries.

3.2.2 Large-scale NeuralDBs. Table 1 compares NeuralDB with a baseline (bottom two rows) that first applies IR techniques to retrieve the relevant facts and feeds those to a transformer. In the NeuralDB version, the support sets generated by the SSG are input to SPJ operators in parallel and then aggregated. As the table shows, NeuralDB achieves the highest EM accuracy on all types of queries. While these numbers cannot be directly compared (as NeuralDB requires different supervision to generate intermediate results), NeuralDB makes substantial improvements over several query types from the same source data.

3.2.3 Summary. The initial experiments confirm the basic tenets of our approach to neural databases that are embodied in the architecture of NeuralDB: (1) if there were a way to feed the transformer the relevant facts from the database, it can produce results with reasonable accuracy, (2) aggregation queries need to be performed outside of the neural machinery, and (3) in order to handle queries that result in sets of answers and in order to prepare sets for subsequent aggregation operators, we need to develop a neural operator that can process individual (or small sets of) facts in isolation and whose results outputted as the answer or fed into a conventional (i.e. non-neural) aggregation operator.

4 A RESEARCH AGENDA

Our NeuralDB implementation provided initial solutions to Challenges 1 & 2. This demonstrated that it is possible to build a neural database that accurately answers queries over large numbers of facts stated with short sentences. Further addressing Challenges 1 & 2 draws upon the rich literature developed in the data management community regarding indexing, view materialization, and query

processing. In particular, support set generation can greatly benefit from indexing and view materialization techniques that enable the system to efficiently refine the support sets at query time. With additional advances, we should be able to scale neural databases to larger data, support more complex queries, and increase their accuracy. This section outlines additional research challenges.

Challenge 3: *Deeper understanding of semantics.* To support a wider range of applications, a neural database needs to correctly handle the semantics of numbers (comparisons and operations) and data types (e.g., addresses, time points). An additional challenge concerns understanding dependencies and hence identifying which updates should replace previous facts and which should not. For example, if the fact, Mariah is unemployed, was in the database and later the fact, Mariah works for Apple, was added, then the first fact should be removed (or at least, apply only to queries about the past). However, the same does not hold for the facts Kasper likes tea followed by the fact Kasper likes coffee. All of these issues pose challenges for the development of transformers in NLP, but they are guided by the needs of neural databases.

Challenge 4: *Multi-modal neural databases.* Combining multiple modalities such as images, text, structured data and audio in a neural database is an exciting area of investigation, building on the fact that transformers have also been applied to other data types. For example, consider a query that combines an image database with knowledge from text, asking for *photos that include an object used to brew coffee*. Such extensions would benefit from recent progress on visual query answering systems [2, 4], that can already answer certain classes of queries on the images themselves.

Challenge 5: *Obtaining training data and transfer learning.* Finding training data for a neural database can be challenging in some contexts. We expect that over time, the community will create public sets of training data for common relationships. A promising approach for creating training data sets is to find sentences in Wikipedia that express known triples from Wikidata or to use Wikidata triples to generate synthetic sentences as in T-REx [10] and KELM [1] datasets, respectively. Leveraging similar parallel corpora of text and structured data can yield additional training data. Advances in NLP on learning without parallel data [17], few-shot learning and large language models should considerably reduce the number of training instances needed.

A key factor in assessing the cost of training data is whether it needs to cover all the relationships that occur in the queries. For example, if the training data has no mention of people's hobbies, can we still answer a query such as, what are Ruth's hobbies? Developments in transfer learning [14] could be applied to this context. It is also reasonable to expect that as we obtain training data for many relations, transfer will occur more naturally because the linguistic patterns in queries and facts are limited. An initial experiment described in [35] substantiates this intuition.

Challenge 6: *Mitigating biases.* A possible downside of using neural techniques in a database is the potential for bias from the underlying language model. For example, suppose our database included facts saying that John and Jane work at a hospital, but when we asked for their profession, the system answers `doctor` for John and `nurse` for Jane. Currently, there is no good way of distinguishing biased from unbiased knowledge in a language model. A possible approach to addressing this issue is to design a separate module that *attacks* the database with queries that attempt to discover biases. Then, we could devise safeguards within the database that ensure that we don't use such biased knowledge in answering queries.

Challenge 7: *Applying neural components to existing data management architectures.* In NeuralDB, the entire database was text. However, there are settings in existing DBMSs where structured data is mixed with text and images. The challenge is to extend existing DMBSs and their query processors to incorporate neural operators, thereby allowing a wider range of queries over these data. For example, consider querying a product user report database for reports about a malfunctioning battery that also have an image of a battery accompanying them.

5 RELATED WORK

Space does not permit a comprehensive description of related work. The differences between neural databases and question answering over text was covered in Section 2.2. Bridging the gap between unstructured natural language data and database-style querying has been a long-standing theme in database research [3, 13, 19, 30, 39, 40]. Unlike the above works, neural databases do not try to map data or queries into a pre-defined schema but operate directly on facts described as short natural language sentences.

In the spirit to Neural Turing Machines [11] and Memory Networks [31] architectures, an alternative way of building neural databases is to encode all the facts in the database to a neural memory and build machinery to read, write, and reason on top of this neural memory. However, such an approach would not have control and transparency: It is challenging to remove facts from the database or check whether a particular fact exists. Also, it would not be possible to explain query results. Furthermore, these architectures perform well on bAbI [38] tasks where the number of facts is limited, and mainly lookup or simple reasoning is needed. There also have been considerable efforts in mixing traditional symbolic reasoning or data management algorithms with neural network architectures, such as differentiable backward chaining [29] and differentiable prolog [22]. Instead of "neuralizing" existing symbolic reasoners, in our work, we start with a scalable neural architecture and support it with symbolic computation only where necessary. This enables us to directly leverage the rapid progress made in retrieval augmented QA models and ensures scalability.

6 CONCLUSIONS

We introduced neural databases, a new class of database systems that use neural reasoning, and are therefore able to answer queries from data expressed as natural language sentences that do not conform to a pre-defined schema. The design of the NeuralDB architecture was based on a careful examination of the strengths and weaknesses of current models used for natural language processing, namely transformers. Neural databases open up many avenues of research because they provide a path for leveraging the latest advances in NLP and computer vision to enable queries over more flexible and varied representations of data.

REFERENCES

[1] Oshin Agarwal, Heming Ge, Siamak Shakeri, and Rami Al-Rfou. Large scale knowledge graph based synthetic corpus generation for knowledge-enhanced language model pre-training, 2020.

[2] Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Neural module networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 39–48. IEEE Computer Society, 2016.

[3] I Androutsopoulos, G D Ritchie, and P Thanisch. Natural Language Interfaces to Databases - an Introduction. *Natural Language Engineering*, 1(1):29–81, 1995.

[4] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. VQA: visual question answering. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pages 2425–2433. IEEE Computer Society, 2015.

[5] Akari Asai, Kazuma Hashimoto, Hannaneh Hajishirzi, Richard Socher, and Caiming Xiong. Learning to retrieve reasoning paths over wikipedia graph for question answering. In *International Conference on Learning Representations*, 2020.

[6] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language Models are Few-Shot Learners. 2020.

[7] Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. Autoregressive entity retrieval. In *International Conference on Learning Representations*, 2021.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 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, Volume 1 (Long and Short Papers)*, Minneapolis, Minnesota, 2019.

[9] Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. DROP: A Reading Comprehension Benchmark Requiring Discrete Reasoning Over Paragraphs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2368–2378, Minneapolis, Minnesota, jun 2019. Association for Computational Linguistics.

[10] Hady Elsahar, Pavlos Vougiouklis, Arslen Remaci, Christophe Gravier, Jonathon Hare, Frederique Laforest, and Elena Simperl. 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 2018. European Language Resources Association (ELRA).

[11] Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *CoRR*, abs/1410.5401, 2014.

[12] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-wei Chang. REALM : Retrieval-Augmented Language Model Pre-Training, 2020.

[13] Alon Y. Halevy, Oren Etzioni, AnHai Doan, Zachary G. Ives, Jayant Madhavan, Luke K. McDowell, and Igor Tatarinov. Crossing the structure chasm. In *CIDR 2003, First Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 5-8, 2003, Online Proceedings*. www.cidrdb.org, 2003.

[14] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

[15] Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense Passage Retrieval for Open-Domain Question Answering. 2020.

[16] Tim Kraska, Alex Beutel, Ed H. Chi, Jeffrey Dean, and Neoklis Polyzotis. The case for learned index structures. In Gautam Das, Christopher M. Jermaine, and Philip A. Bernstein, editors, *Proceedings of the 2018 International Conference on Management of Data, SIGMOD Conference 2018, Houston, TX, USA, June 10-15, 2018*, pages 489–504. ACM, 2018.

[17] Guillaume Lample, Alexis Conneau, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. Word translation without parallel data. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[18] Patrick Lewis, Ethan Perez, Aleksandara Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In *NeurIPS*, 2020.

[19] Fei Li and H V Jagadish. Constructing an Interactive Natural Language Interface for Relational Databases. *Proceedings of the VLDB Endowment2*, 8(1):73–84, 2014.

[20] Yuliang Li, Jinfeng Li, Yoshihiko Suhara, AnHai Doan, and Wang-Chiew Tan. Deep entity matching with pre-trained language models. *Proc. VLDB Endow.*, 14(1):50–60, September 2020.

[21] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A Robustly Optimized BERT Pretraining Approach. 2019.

[22] Pasquale Minervini, Matko Bosnjak, Tim Rocktäschel, Sebastian Riedel, and Edward Grefenstette. Differentiable reasoning on large knowledge bases and natural language. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 5182–5190. AAAI Press, 2020.

[23] Sidharth Mudgal, Han Li, Theodoros Rekatsinas, AnHai Doan, Youngchoon Park, Ganesh Krishnan, Rohit Deep, Esteban Arcaute, and Vijay Raghavendra. Deep learning for entity matching: A design space exploration. In Gautam Das, Christopher M. Jermaine, and Philip A. Bernstein, editors, *Proceedings of the 2018 International Conference on Management of Data, SIGMOD Conference 2018, Houston, TX, USA, June 10-15, 2018*, pages 19–34. ACM, 2018.

[24] Matthew Peters, Mark Neumann, Luke Zettlemoyer, and Wen-tau Yih. Dissecting Contextual Word Embeddings: Architecture and Representation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1499–1509, Brussels, Belgium, 2018. Association for Computational Linguistics.

[25] Fabio Petroni, Aleksandra Piktus, Angela Fan, Patrick Lewis, Majid Yazdani, Nicola De Cao, James Thorne, Yacine Jernite, Vassilis Plachouras, Tim Rocktäschel, et al. Kilt: a benchmark for knowledge intensive language tasks. *arXiv preprint arXiv:2009.02252*, 2020.

[26] Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H. Miller, and Sebastian Riedel. In *Proceedings of EMNLP-IJCNLP*, Hong Kong, China.

[27] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research*, 21:1–67, 2020.

[28] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for SQuAD. *ACL 2018 - 56th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers)*, 2:784–789, 2018.

[29] Tim Rocktäschel and Sebastian Riedel. End-to-end Differentiable Proving. In I Guyon, U V Luxburg, S Bengio, H Wallach, R Fergus, S Vishwanathan, and R Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 3788–3800. Curran Associates, Inc., 2017.

[30] Marzieh Saeidi, Max Bartolo, Patrick Lewis, Sameer Singh, Tim Rocktäschel, Mike Sheldon, Guillaume Bouchard, and Sebastian Riedel. Interpretation of natural language rules in conversational machine reading. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2018.

[31] Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. End-to-end memory networks. In *Advances in neural information processing systems*.

[32] Nan Tang, Ju Fan, Fangyi Li, Jianhong Tu, Xiaoyong Du, Guoliang Li, Sam Madden, and Mourad Ouzzani. Relational pretrained transformers towards democratizing data preparation [vision]. *CoRR*, abs/2012.02469, 2020.

[33] Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R. Thomas McCoy, Najoung Kim, Benjamin Van Durme, Samuel R Bowman, Dipanjan Das, and Ellie Pavlick. What do you learn from context? Probing for sentence structure in contextualized word representations. *ICLR*, pages 1–17, 2019.

[34] James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. FEVER: a large-scale dataset for Fact Extraction and VERification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 809–819, New Orleans, Louisiana, 2018. Association for Computational Linguistics.

[35] James Thorne, Majid Yazdani, Marzieh Saeidi, Fabrizio Silvestri, Sebastian Riedel, and Alon Y. Halevy. Neural databases. *CoRR*, abs/2010.06973, 2020.

[36] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Lilon Jones, Aidan Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *31st Conference on Neural Information Processing Systems (NIPS 2017)*, Long Beach, CA, USA, 2017.

[37] Denny Vrandečić and Markus Krötzsch. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85, 2014.

[38] Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. Towards ai-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*, 2015.

[39] Tomer Wolfson, Mor Geva, Ankit Gupta, Matt Gardner, Yoav Goldberg, Daniel Deutch, and Jonathan Berant. Break It Down: A Question Understanding Benchmark. *Transactions of the Association for Computational Linguistics*, 8:183–198, 2020.

[40] Jichuan Zeng, Xi Victoria Lin, Caiming Xiong, Richard Socher, Michael R. Lyu, Irwin King, and Steven C. H. Hoi. Photon: A robust cross-domain text-to-sql system.