A review of query systems for temporal n-gram corpora

Fabian Richter[1,*], Benjamin Schäfer[1] and Klemens Böhm[1]

[1] *Karlsruhe Institute of Technology, Kaiserstraße 12, 76131 Karlsruhe, Germany*

Abstract

Natural languages evolve over time and with increasing digitalization these evolutions are quantitively studied in humanities and social sciences. One important observable is the frequency of individual words, as well as word tuples (n-grams) over time. Different tools exist to analyze these changing frequencies in large text corpora, with different levels of complexity and efficiency. However, a systematic overview and evaluation of the expressiveness and practical usability of these different tools is missing. In this article, we present a structured approach to such an evaluation by defining a query algebra and a set of information needs expressed therein, followed by a comparison of 12 different query systems. Overall, we identify several systems as similar to the Google Books Ngram Viewer (GBNV) or as systems specific to a subcorpus, and find that the theoretically most potent and flexible systems lack a practical implementation, pointing out further research needs.

Keywords

Google Books Ngram Corpus, Query System, Query Algebra

1. Introduction

The development of word frequencies over time can give researchers insight into linguistic and cultural shifts. Many such analyses have been published in recent years [1, 2, 3, 4, 5, 6], an interesting future application could be the analysis of changes Large Language Models (LLMs) bring into written language. Different tools exist to support researchers in their work with large text corpora. For pure frequency analyses, however, a different and simpler kind of corpora can be used: temporal n-gram corpora (TNC) like the *Google Books Ngram Corpus* (GBNC) [7, 8, 9]. Instead of full texts, they only contain n-grams and their respective frequencies over time, and—while losing information—have two main advantages: (1) reduced memory usage, making the analysis of larger document collections possible, and (2) fewer copyright issues, allowing for the inclusion of more recent texts which cannot be freely published in full.

The most well-known tool for analyzing TNCs is the *Google Books Ngram Viewer*[1] (GBNV) [7], but it is not the only one. Over the years, other query systems have been developed and published, mostly independent of each other. Our goal with this work is to collect information about such systems, to organize it and to give researchers interested in TNCs a concise overview of the available tools they can choose from. To achieve this, several steps and contributions are necessary:

- a thorough *literature survey* to identify existing TNC query systems,
- a formal data model and *query algebra* to precisely express query types across different systems,
- a *systematic review* of the existing query systems regarding their expressiveness and practical usability; finally leading to
- a *concise overview* of existing TNC query systems and their capabilities.

Related work Our review focuses on query systems specifically built for TNCs. More generic corpus analysis tools like the *Sketch Engine* [10] or *Korp* [11] are not included, as their scope is much broader

SCOLIA '25: First International Workshop on Scholarly Information Access (SCOLIA), April 10, 2025, Lucca, Italy

*Corresponding author.

fabian.richter@kit.edu (F. Richter); benjamin.schaefer@kit.edu (B. Schäfer); klemens.boehm@kit.edu (K. Böhm)

https://fr2501.github.io/ (F. Richter); https://www.benjaminschaefer.org/ (B. Schäfer)

0000-0002-5694-9480 (F. Richter); 0000-0003-1607-9748 (B. Schäfer); 0000-0002-1706-1913 (K. Böhm)

[1] https://books.google.com/ngrams/

and they focus more on the analysis of full texts. Metadata search engines and query systems like *SchenQL* [12, 13] and *LexiDB* [14, 15] are also not included into our review. Even though some of the TNC query operators defined in the following rely on metadata, it is not the core concept of TNCs.

Paper outline In Section 2, we define a formal model of TNCs and a query algebra for them. Section 3 presents the results of our query system review, with a focus on expressive power in Section 3.3 and practical usability concerns in Section 3.4. Section 4 discusses limitations of our review and gives an outlook on future work.

2. Fundamentals

In this section, we first describe a formal data model for TNCs. We then define a query algebra operating on that data model, which will later be used to express common information needs across different query systems.

2.1. Data model

In this section, we construct a formal model of temporal n-gram corpora, starting from diachronic document collections and leading up to an extension of relational algebra [16]. This model allows us to precisely express abstract information needs and query types across different concrete systems.

Definition 1 (Temporal text corpus)**.** A temporal text corpus C consists of documents $d_i = (t_i, m_i, \tau_i)$. Document d_i comprises the text t_i, its metadata as a (partial) key-value-mapping m_i, and a specific metadata entry τ_i, the timestamp. The set of all timestamps in C is called T and must admit a natural ordering. Furthermore, we use $C_\tau := \{d_i \in C \mid \tau_i = \tau\}$ to denote the set of all documents with the specific timestamp $\tau \in T$.

Typical keys for the metadata map m_i are, e.g., *author* or *title*, and the timestamp usually represents the *publication date*. The metadata of a document is modeled as a *partial* key-value-mapping because not every metadata field needs to be populated for every document. We do not specify a granularity of the timestamps in T. For the GBNC, the timestamps correspond to years, while other corpora and systems might use finer-grained timestamps or coarser aggregates.

Definition 2 (n-gram)**.** Given a set Σ, $g \in \Sigma^n$ is a tuple of length n over Σ, an n-gram. To denote the set of all n-grams over Σ, without fixing n, we write Σ^*. Given some document d, we use $f_d(g) : \Sigma^* \to \mathbb{N}_0$ to obtain the number of occurrences of an n-gram g in the document's text.

In the context of natural language processing, the set Σ in Definition 2 is usually the *vocabulary* of a natural language, possibly annotated with linguistic metadata, e.g., *part-of-speech tags* [7, 8]. An n-gram is then an ordered sequence of n words from that vocabulary.

Definition 3 (Temporal n-gram corpus (TNC))**.** Consider a temporal text corpus C and the set T of its timestamps. Let Σ be the set of all words that occur in the texts of C. We then construct

$$f_C : \Sigma^* \times T \to \mathbb{N}_0, (g, \tau) \mapsto \sum_{d \in C_\tau} f_d(g)$$

to capture the number of occurrences of an n-gram g in all documents d with timestamp τ. Fixing C and an n-gram g, $f_g := f_C(g, \cdot)$ becomes a function over T, a time series. We say that $g \in C$ if and only if there is some $\tau \in T$ such that $f_g(\tau) \neq 0$. Letting $G := \{g \in \Sigma^* \mid g \in C\}$ and $F := \{f_g \mid g \in G\}$, $C' := (T, G, F)$ is the temporal n-gram corpus over C.

Constructing a TNC C' out of a temporal text corpus C may seem redundant, but it greatly simplifies notation and analysis in the following sections. Furthermore, in practice, C may not always be available, even if C' is. This is the case for the GBNC.

Finally, we define a relation $R_{C'}$ in the sense of [16] and using notation from [17] that represents a TNC $C' = (T, G, F)$, as follows: $R_{C'}(G, F) = \{(g_i, f_{g_i})\}_i \subseteq G \times F$. Neither of the two attributes of $R_{C'}$ are atomic, as the first one represents an n-gram, so a tuple of words, and the second one a frequency map from T to $\mathbb{N}_0$. In both cases, we use square brackets to denote member access, either via the 1-based tuple index or the map key, i.e., for some n-gram g, $g[2]$ shall represent the second word in g, and $f_g[2025]$ its frequency at timestamp 2025.

2.2. Query types

After defining our data model for TNCs, we now present different abstract query types. These are derived from the capabilities of existing query system implementations. First, we focus on filtering TNCs, using different types of constraints. Then, we focus on analytic queries. Finally, we describe how to incorporate external knowledge into queries and how to construct TNCs for specific purposes. Even though we introduce the different kinds of queries and operators independently, the underlying algebra allows for flexible combinations.

In the following, a TNC C' over a text corpus C is always given through its corresponding relation $R_{C'}(G, F)$. We use (g, f_g) to refer to specific elements of $R_{C'}(G, F)$.

2.2.1. Filtering

A user is not necessarily interested in the full TNC C', but maybe rather a subset of it or even a single time series. We express this subcorpus filtering as a selection $\sigma_p(R_{C'})$, where p is an arbitrary boolean predicate on n-grams and their frequencies. The general semantics of this operation is as follows:

$$\sigma_p(R_{C'}) = \{(g, f_g) \mid p(g, f_g) = \mathit{true}\}.$$

The structure of p can be used to further categorize filtering operations. For example, some p may only consider g and be independent of f_g, or vice versa. This is an important distinction, as many of the existing query systems only allow for the first, but not the second kind of predicates In our abstract algebra, this strict distinction does not exist and predicates can be freely mixed.

n-gram filtering First, we focus on filter predicates p that are independent of f_g and only depend on g itself. The simplest kind is strict equality with a given n-gram g_{ref}:

$$\sigma_{g=g_{ref}}(R_{C'}) = \{(g_{ref}, f_{g_{ref}})\},$$

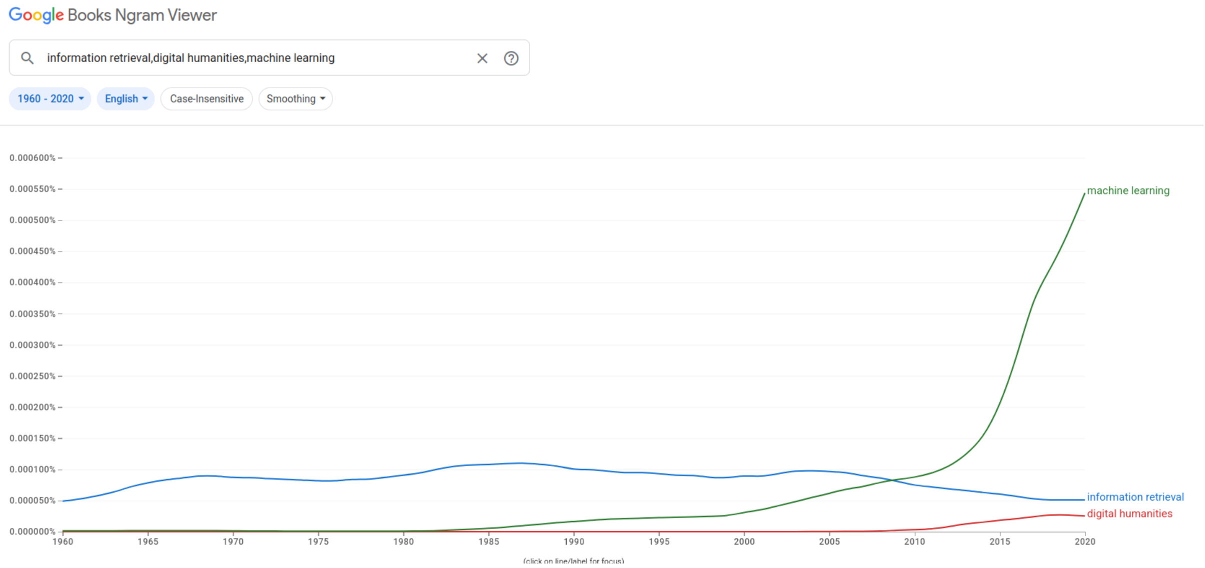
which yields the singleton set of g_{ref} and its frequency series. In a similar way, one can define p as strict set membership for some set G_{ref} to filter for several n-grams at once. An example for this is shown in Figure 1, showing the frequencies of the 3 2-grams *information retrieval*, *digital humanities*, and *machine learning*.

A more complex type of filtering employs *external* or *internal* wildcards. When employing the external wildcard operator $*$, two n-grams do not need to be exactly equal, but only in all non-wildcard positions. For an n-gram $g_{ref} = (w_1, \dots, *, \dots, w_n)$, the semantics of equality up to wildcards (or *matching*) is as follows:

$$g \simeq g_{ref} \equiv \forall 0 < i \leq n \,:\, g_{ref}[i] = * \vee g_{ref}[i] = g[i],$$

with $\sigma_{g \simeq g_{ref}}(R_{C'})$ as above. Another type of wildcards, which we call internal wildcard and denote as $\hat{*}$, does not operate on n-grams themselves, but on the words therein: For example, '$\hat{*}$*ing*' matches all words that end with '*ing*'; and consequentially, $(*, \hat{*}ing)$ matches all 2-grams starting with an arbitrary word and ending with a word that ends in '*ing*'.

While strict equality and wildcards are the most straightforward methods, n-gram filtering is not limited to these. Aleksandrov and Strapparava [18] include linguistic information from *WordNet* [19, 20], allowing for queries about, for example, synonym or antonym relationships between words. Willkomm et al. [6] apply sentiment analysis to identify and select n-grams carrying positive or negative sentiments.



Google Books Ngram Viewer

information retrieval,digital humanities,machine learning

1960 - 2020 ▾ English ▾ Case-Insensitive Smoothing ▾

Figure 1: Frequencies of *information retrieval*, *digital humanities*, and *machine learning* as plotted by the GBNV

Frequency filtering Second, we introduce predicates operating on the n-grams' frequencies. A simple example for a frequency-based predicate is a pointwise threshold: Setting $p = (f_g[2000] \geq 100)$, for example, selects only those n-grams that occurred at least 100 times at timestamp 2000. Another option is a global threshold, i.e., selecting n-grams g where $\sum_{t \in T} f_g(t) > c$ for some fixed value c.

A more complex predicate uses information about all of C', to select the c most frequent n-grams in the corpus:

$$top(c, g) \equiv \sum_{\tau \in T} f_g(\tau) < \sum_{\tau \in T} f_{g'}(\tau) \text{ for fewer than } c \text{ n-grams } g'.$$

Frequency filtering can become almost arbitrarily complex: Willkomm et al. [6] introduce a *kNN* operator, that selects, for a given reference n-gram and some constant k, the k most similar n-grams according to the Euclidean distance of their frequencies. Frequency filtering can also be combined with different operators, for example those presented in the next section.

2.2.2. Data analysis and computation

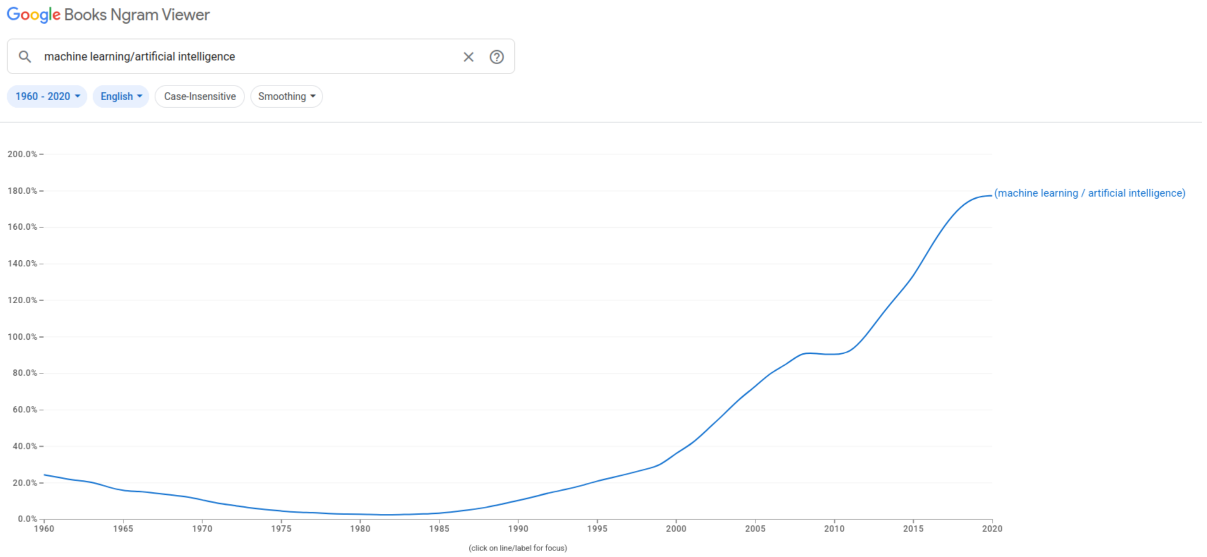
While the frequency of a single n-gram can be interesting in some research applications, being able to combine different n-grams' frequencies into aggregate time series or even single statistical values opens up many new possibilities. For example, it becomes possible to not only analyze specific words, but rather concepts, by summing the frequencies of synonyms. Another possibility is to identify n-grams with similar frequency patterns by computing pairwise correlations.

Arithmetic operations The easiest way to combine the frequencies of two n-grams is applying pointwise arithmetic operations, as follows:

$$(g_1, f_{g_1}) \circ (g_2, f_{g_2}) := (g_1 \circ g_2, f_{g_1 \circ g_2}[\tau] = g_1[\tau] \circ g_2[\tau], \tau \in T), \circ \in \{+, -, \cdot, /\}.$$

Here, $g_1 \circ g_2$ denotes a symbolic identifier for the combination of g_1 and g_2 using $\circ$, which is not an n-gram in itself, but ensures compatibility with other frequency-based operators of our algebra. An example of this can be seen in Figure 2, showing the ratio of the frequencies of *machine learning* and *artificial intelligence*, which was close to 0 in 1980, but has rapidly increased since then—showing a significant increase in popularity for the term *machine learning* over *artificial intelligence*. Replacing (g_2, f_{g_2}) with a constant c and using multiplication as operation defines a constant scaling operator.

Since the result of an arithmetic operation in this sense is structurally similar to its operands—albeit with a symbolic identifier instead of an n-gram—arithmetic operations can be nested and combined



Google Books Ngram Viewer

machine learning/artificial intelligence

1960 - 2020 · English · Case-Insensitive · Smoothing

Figure 2: Ratio of the frequencies of *machine learning* and *artificial intelligence* as plotted by the GBNV

following the usual rules of arithmetics. The definition of other operations to use as $\circ$ is also possible, as long as their signature fits the same pattern.

Statistical analysis The general goal of statistical analysis is to extract knowledge from data. This can take many different forms, we want to focus on *numerical* knowledge: Given one or several n-gram frequency series, compute a number representing an 'interesting' property v; or, more formally $f_v : F^k \to \mathbb{R}$ is a k-ary function mapping k n-grams to one real number quantifying the property v. Examples for f include unary ($k = 1$) functions like *min*, *max* and *mean*, as well as binary ($k = 2$) functions like ρ (Pearson's Correlation Coefficient) or *Mutual Information* [21].

2.2.3. External knowledge integration

TNCs are based on textual data, usually written in natural languages. There exists a wealth of linguistic knowledge that one might want to include into queries on TNCs, for example by grouping synonyms together or matching different spellings of the same word.

External knowledge can take many forms, so it is difficult to give a general formalization. The relational data model [16] has, however, proven to be flexible enough to represent almost any kind of data. It is therefore reasonable to assume that the external knowledge is available as some relation $K(K_1, \ldots, K_m)$. Then, combining $R_{C'}$ and K can be expressed as

$$R_{C'} \bowtie_p K := \sigma_p(R_{C'} \times K),$$

an *(inner) join* over some predicate p—or, equivalently, a cartesian product followed by a selection.

2.2.4. Metadata and corpus construction

Until now, C and C' have been fixed. In some cases, users might want to restrict a large text collection C to some smaller text collection $\hat{C}$ and then consider the TNC $\hat{C}'$ separately. This is fundamentally different from the previously described operators, as the filtering here does not operate on n-grams, but on entire documents and texts. On a purely conceptional level, however, the algebraic operators are similar: Instead of operating on $R_{C'}$, we now consider $R_C(T, M)$, where T contains the texts and M the metadata map (including the timestamp τ_i) of the documents d_i. A selection on R_C then yields a smaller relation $R_{\hat{C}}$, which in turn defines the smaller TNC $\hat{C}'$.

3. Query system review

Having defined our data model and a formal framework for query formulation, we can now proceed to review 12 existing query systems for TNCs. We begin by describing our literature research process and justifying our selection of the 12 query systems. We then consider different properties of these systems: the corpora they are meant to analyze, the expressive power of their query languages, and finally their practical usability.

3.1. Literature selection

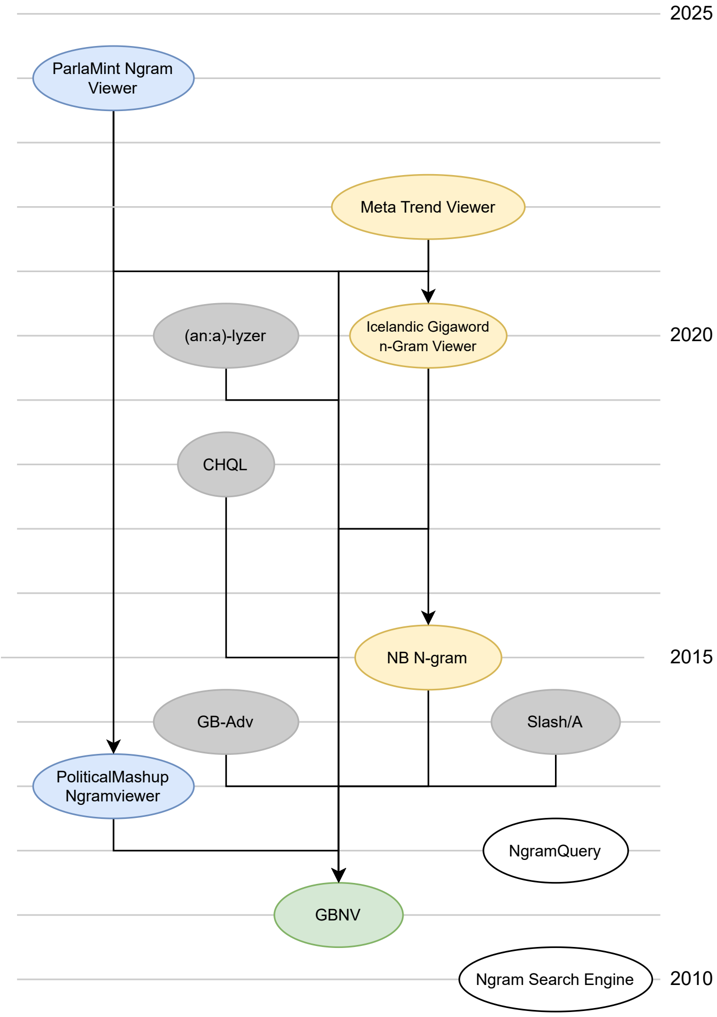


Figure 3: Citation graph of publications related to different query systems

For selecting the literature relevant for our review, we attempted to perform targeted keyword searches in *Google Scholar*. This, however, proved problematic: The GBNV is significantly more popular than any competitors, and thus heavily dominates the result sets. Excluding terms like ‘Google’ was not useful to restrict the result, as competing query systems almost necessarily mention the GBNV. The most useful query we found was `["ngram viewer" -intitle:google]`, yielding *NB N-Gram* by Birkenes et al. [26] and *Slash/A* by Todorova et al. [25]. Following bi-directional citation links from

Table 1
Existing TNC query systems

Year	Query System	**Website** (last accessed March 18, 2025)
2010	Ngram Search Engine [22]	http://nlp.cs.nyu.edu/nsearch (unavailable)
2011	GBNV [9, 7, 8]	https://books.google.com/ngrams/
2012	NgramQuery [18]	—
2013	PoliticalMashup Ngramviewer [23]	http://ngram.politicalmashup.nl (unavailable)
2014	GB-Adv [24]	https://www.english-corpora.org/googlebooks/
2014	Slash/A [25]	http://linguistics.chrisculy.net/lx/vistola/tools/slasha.html
2015	NB N-gram [26]	https://www.nb.no/ngram/
2018	CHQL [27, 6, 28]	—
2020	(an:a)-lyzer [1]	https://osf.io/ht8se/
2020	Icelandic Gigaword n-Gram Viewer [29]	https://n.arnastofnun.is/
2022	Meta Trend Viewer [30]	https://meta-trend-viewer.elte-dh.hu/
2024	ParlaMint Ngram Viewer [31]	https://debateabase.wooverheid.nl/ (unavailable)

there, we further identified the *Icelandic Gigaword n-Gram Viewer* (Steingrímsson et al. [29]) and the *Meta Trend Viewer* (Indig et al. [30]).

This motivated the following selection procedure: Starting from 8 *seed articles* (including the two mentioned above, [26] and [25]) we were aware of from previous research, we followed bi-directional citation links to other articles describing query systems. We did not include the GBNV itself into the seed set, as its corresponding papers have been cited several thousand times, but most of these citations do not describe query systems. We intended to continue this procedure transitively—but the only further query system we found was the *PoliticalMashup Ngramviewer* [23]. This may be a consequence of the generally sparse citation graph between the considered articles: Figure 3 illustrates that many of the query system articles do not cite others, except for the GBNV. This fragmentation makes it difficult to systematically identify and evaluate relevant literature, finally leading us to settle on our selection of 12 systems.

3.2. Underlying text corpora

Table 2
Different query systems and their underlying text corpora and TNCs

Query system	Text corpus	TNC
Ngram Search Engine	English Wikipedia	
GBNV	library collections, unknown	GBNC
NgramQuery	crawled websites, unknown	Google Web 1T
PoliticalMashup Ngramviewer	Dutch parliament proceedings	
GB-Adv	same as GBNV	GBNC, 10-year aggregates
Slash/A	customizable	
NB N-Gram	Norwegian National Library	
CHQL	same as GBNV	GBNC (2012)
(an:a)-lyzer	same as GBNV	GBNC, partial
Icelandic Gigaword n-Gram Viewer	Icelandic Gigaword Corpus [32, 33]	
Meta Trend Viewer	customizable	
ParlaMint Ngram Viewer	ParlaMint.ana 3.0 [34]	

One of the first decisions to make for any n-gram frequency analysis is which corpus is suitable for the concrete research questions. This already restricts the choice of available query systems, as most of the existing ones operate on one, predefined corpus and cannot easily be applied to other corpora, as summarized in Table 2. If the table lists no TNC, it is the one constructed from the corresponding text corpus as described in Section 2.1. While, in theory, all systems could be applied to any TNCs, most implementations do not allow changing the corpus and might not scale well.

The GBNC is the most well-known and by far the biggest TNC. It is based on numerous library collections, digitized as part of the *Google Books* project. It spans more than 500 years and contains at least around 6% of all books ever published [8]. Due to copyright reasons, neither the full texts nor any metadata of the included documents are available, making it impossible to reconstruct the GBNC's exact composition. Thus, the GBNC is a case where the TNC C'_{GBNC} is available, even though the text corpus C_{GBNC} is not. Nonetheless, it has attracted widespread attention from researchers.

A similar, but older and smaller corpus, is the *Google Web 1T 5* corpus [35, 36]. Released in 2006, it contains 1 trillion tokens, extracted from crawled websites. Lacking a temporal dimension, it is not strictly a TNC in our sense, but rather a snapshot of a single point in time. Again, the underlying documents are unavailable.

Most of the other query systems operate on smaller, domain-specific corpora: a corpus of international European parliament proceedings for the ParlaMint Ngram Viewer; specifically Dutch parliament proceedings for the PoliticalMashup Ngram Viewer; Norwegian and Icelandic texts for NB-N-Gram and the Icelandic Gigaword n-Gram Viewer, respectively; and the English Wikipedia for the Ngram Search Engine.

One special case are Slash/A and the Meta Trend Viewer. While their demos contain predefined corpora, letters between Elizabeth Barrett and Robert Browning for Slash/A, and Hungarian news articles for the Meta Trend Viewer, both are explicitly designed for different corpora as well. Slash/A uses the *TCF* format[2] for its corpora, the Meta Trend Viewer stores them in a relational database and can import a format also used by the *Sketch Engine* [10]. Four systems have source code available, so local instances of these systems can potentially also handle custom corpora (more detail is found in Section 3.4, under *Extensibility*). Another special case are NgramQuery and CHQL, for which no implementations are available at all—in this case, Table 2 lists the corpora the systems were designed for.

3.3. Expressive power of query languages

The expressive power of a query language defines what kinds of queries and information needs can be formulated within it. In this section, we investigate the languages of the different query systems and give insights into their expressive powers.

First, we define a set of information needs we want to evaluate. This set is shown in Table 3. It covers the types of information needs we defined in Section 2 and sufficiently differentiates the query systems.

Table 3
Common information needs and their formalizations

		Information need	Algebraic expression
n-gram operations	(i)	keyword search	$\sigma_{g=g_{ref}}(R_{C'})$
	(ii)	single wildcard	$\sigma_{g\simeq g_{ref}}(R_{C'})$, 1 $*$ in g_{ref}
	(iii)	multiple wildcards	$\sigma_{g\simeq g_{ref}}(R_{C'})$, multiple $*$ in g_{ref}
	(iv)	internal wildcard	$\sigma_{g\simeq g_{ref}}(R_{C'})$, where some w_i in g_{ref} contains $\hat{*}$
	(v)	edit distance	$d_{edit}(g_1, g_2)$
frequency operations	(i)	frequency threshold	$\sigma_{\sum_{t\in T} f_g(t)>c}(R_{C'})$, for some constant c
	(ii)	most frequent	$\sigma_{top(c,g)}(R_{C'})$, for some constant c
	(iii)	similarity search	$d_{sim}(f_{g_1}, f_{g_2})$
	(iv)	correlation	$\rho(f_{g_1}, f_{g_2})$
	(v)	arithmetic	$(g_1, f_{g_1}) \circ (g_2, f_{g_2})$
knowledge, metadata	(i)	synonym grouping	$R_{C'}(G, F) \bowtie_{R.G=S.G} S(G, Syn)$, where Syn is a list of synonyms of G.
	(ii)	dictionaries	$\sigma_{g\in D}(R_{C'})$, for some dictionary D
	(iii)	sentiment	$R_{C'}(G, F) \bowtie_{R.G=S.G} S(G, V)$, where S is a sentiment dictionary
	(iv)	metadata filtering	$\sigma_{m(k)=x}(R_C)$, for some metadata key k and value x

[2] https://weblicht.sfs.uni-tuebingen.de/englisch/tutorials/html/index.html

Table 4
Different query systems and supported query types as reported (GBNV-like systems in blue, subcorpus-oriented systems in red)

Query system	n-gram operations					frequency operations					knowledge, metadata			
	i	ii	iii	iv	v	i	ii	iii	iv	v	i	ii	iii	iv
Ngram Search Engine	✓	✓	✓			✓								
GBNV	✓	✓								✓				
NgramQuery	✓	✓	✓	✓							✓	✓		
PoliticalMashup Ngramviewer	✓													✓
GB-Adv	✓	✓	✓	✓			✓							
Slash/A	✓	✓	✓											
NB N-Gram	✓	✓	✓	✓						✓				
CHQL	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	
(an:a)-lyzer														✓
Icelandic Gigaword n-Gram Viewer	✓	✓	✓							✓				✓
Meta Trend Viewer	✓													✓
ParlaMint Ngram Viewer	✓													✓

The results of our review regarding expressive power are summarized in Table 4. First, we want to highlight common patterns across different systems, and then go into further detail on some of the systems that do not fit those patterns.

The first group of systems we identified are the *GBNV-like* systems. This group comprises the GBNV itself, the Ngram Search Engine, GB-Adv, Slash/A, NB N-Gram and the Icelandic Gigaword n-Gram Viewer. While there are some differences within the group, they share common characteristics: Search and filtering operations are (mostly) limited to n-grams and cannot take frequencies into account. Most of the systems support simple arithmetic operations to aggregate a small number of frequency series into a combined one. Their output is usually a visualization of frequencies over time, most often as a line plot. Within this group, the main difference between the systems is their flexibility regarding wildcards.

The second group are the *subcorpus-oriented* systems. This group includes the PoliticalMashup Ngramviewer, the Meta Trend Viewer and the ParlaMint Ngram Viewer. Their unifying characteristic is their focus on subcorpora: They offer the possibility to construct subcorpora according to documents' metadata, and operators to visualize and compare frequencies across these subcorpora. Their user interfaces are similar, offering exact keyword searches and separate fields for metadata-based restrictions.

The three remaining systems are NgramQuery, the (an:a)-lyzer and CHQL. The (an:a)-lyzer is a very specific case. Its focus is the investigation of one very specific linguistic phenomenon, the pronunciation of the letter *h* at the beginning of English words. It is not a general-purpose query system, so it stands out compared to all the others, not even offering keyword searches on its corpus. NgramQuery focuses mainly on combining TNCs with WordNet. It offers a rich selection of operators mapping relations in WordNet, e.g., synonym and antonym relationships between words. Finally, CHQL is a very flexible query language and offers a lot of unique operators. An actual system implementing this language is, however, not available, only the formal definitions of these operators. A query system based on CHQL would be vastly more expressive than any of the other systems we investigated.

3.4. Practical usability

We consider four dimensions of practical usability: *availability extensibility*, *scalability*, and *query language complexity*. As already mentioned in Section 3.2, implementations of NgramQuery and CHQL are not available, so in the following we assume these systems would behave in the way they are described in their corresponding articles.

Availability A query system can only be used if some implementation of it is available to researchers. This usually comes in one of two forms: as a hosted web application or as a repository of source code.[3] Both options have their own advantages: A hosted web application requires no user setup and entry barriers are very low, but it is usually less flexible. A source code repository is more difficult to set up and requires the user to have sufficiently powerful hardware, but can be more easily adapted for specific use cases. The ideal case is therefore a combination of both: a web demonstrator of the query system, together with source code allowing for modifications.

NgramQuery and CHQL are not available in any form. At the time of writing, hosted web applications are accessible for the GBNV, GB-Adv, NB N-gram, the Icelandic Gigaword n-Gram Viewer and the Meta Trend Viewer. The web demonstrators of the Ngram Search Engine, PoliticalMashup Ngramviewer, (an:a)-lyzer and the ParlaMint Ngram Viewer exist or existed in the past, but could not be accessed. Source code is available for Slash/A, the Icelandic Gigaword n-Gram Viewer, the Meta Trend Viewer and the ParlaMint Ngram Viewer. Web demonstrators or source code archives, respectively, are linked in Table 1.

Extensibility No query system developer is able to anticipate all potential information needs of their users. Therefore, extensibility of such systems is useful in adapting them for a wider range of research questions. We consider three kinds of extensibility: (1) the possibility to analyze custom corpora, (2) extensions of the query language itself, and (3) the export of data to enable processing in different tools.

(1) None of the web demonstrators support the upload of custom corpora, so we focus on the four systems for which source code is available. Slash/A and the Meta Trend Viewer address user-defined corpora explicitly and the required file format is documented in the respective papers [25, 30]. The Icelandic Gigaword n-Gram Viewer is based on *Korp* [11] and the *Stuttgart Corpus WorkBench* (CWB) [37]. It might be possible to adapt the viewer to custom corpora stored within these tools, but this is not intended and the required effort is unclear. The ParlaMint Ngram Viewer is based on XML files and *ElasticSearch* [38]—a potentially flexible technology, but the exact process to use the viewer for custom corpora is not described.

(2) None of the systems we reviewed offers any defined mechanisms for query language extensions. For pure web applications, this makes any extensions impossible, as it would require the execution of custom code on foreign servers, which is usually not allowed. The systems available as source code might offer the possibility to be extended, but not in a documented and intended way, so a deeper technical understanding of their implementations is required—a significant effort.

(3) Being able to export query results from a query system to a common format like *CSV* enables further analyses which are not possible within the system itself. Of the systems we investigated, 2 offer this possibility: NB N-Gram and the Icelandic Gigaword n-Gram Viewer. While not the raw data, at least plots can be exported from Slash/A, the (an:a)-lyzer and the Meta Trend Viewer. For other systems, workarounds to extract numerical data sometimes exist,[4] but these are not an official part of the query system and potentially outdated.

Scalability Experimentally evaluating the scalability of different query systems would require access to comparable implementations of the systems in question. Since that is not available, we take a more theoretical approach towards evaluating scalability, based on the size of the corpora the systems were designed for. These sizes are shown in Table 5, in terms of number of distinct n-grams or number of documents, depending on what was reported by the respective authors. We want to highlight two main points about the table: (a) the corpus sizes differ by several orders of magnitude, so it is to be expected that not all query systems can efficiently handle the bigger corpora; and (b) there is no clear trend towards larger corpora over time—recall from Table 1 that the rows are ordered by year of first publication.

[3] A third option would be the distribution of compiled binary files and non-disclosure of source code, but none of the systems we investigated follow that model.

[4] https://github.com/econpy/google-ngrams, https://github.com/prasa-dd-vp/google_ngram_api

Table 5
Corpus sizes for different query systems, as reported

Query System	Number of n-grams	Number of documents
Ngram Search Engine	4.6 billion [22]	-
GBNV	≥ 860 billion [28]	≥ 8.1 million [8]
NgramQuery	3.8 billion [22]	-
PoliticalMashup Ngramviewer	2.2 billion [23]	-
GB-Adv	730 million [24]	-
Slash/A	-	557 [25]
NB N-Gram	-	770,000 [26]
CHQL	860 billion [28]	8.1 million [8]
(an:a)-lyzer	233 million [1]	-
Icelandic Gigaword n-Gram Viewer	120 million [32]	-
Meta Trend Viewer	-	2.8 million [30]
ParlaMint Ngram Viewer	5.8 billion [31]	7.5 million [31]

Query language complexity Measuring the complexity of a query language is not trivial. We will not define a formal model for it, but rather give a plausible intuition regarding the different query systems. By 'complexity', we do not mean the computational complexity, but rather complexity in the process of query formulation.

On one end of the spectrum are simple keyword queries within a graphical interface: Users only have to supply an n-gram they are interested in to retrieve a plot of its frequency. Most of the GBNV-like systems follow this pattern. The syntax of wildcard queries is equally simple within these systems, replacing words or parts of words with the respective wildcard characters.

Two systems with significantly more complex query language are NgramQuery and CHQL. NgramQuery contains many operators relating to WordNet and semantic similarity, for example the query '`~#food#n`' "retrieves the hyponyms of the noun food for all of its senses" [18] and '`food#L#15#0.6`' "retrieves the similar terms of food with cosine value at least 0.6 among the first 15 most similar terms" [18]. A summary of all the operators defined in the language is given in [18]. CHQL, as already mentioned in Section 3.3, defines a query algebra instead of a query language implementation. A query language representing that algebra would be of similar complexity to a relational query language like *SQL*, so significantly more complex than the other systems.

3.5. Summary of review findings

We have reviewed 12 different query systems for TNCs with regards to their expressive power, scalability and practical usability. We identified two groups of systems, namely the *GBNV-like* and the *subcorpus-oriented* systems. We found that most systems support keyword and wildcard queries, with different levels of flexibility. Of the 12 systems we investigated, 5 were accessible as web demonstrators at the time of writing, 4 more offered web demonstrators in the past that could not be accessed any more. Source code is available for 4 of the systems. None of the systems offer defined mechanisms for extension, but some let users analyze custom corpora and export data for use in different systems. The size of the underlying corpora varies significantly between the different systems, with no clear trend towards larger corpora over time. Most of the systems offer simple query languages through graphical user interfaces, making them attractive to users without a strong technical background.

When conducting a study on TNCs, the choice of a suitable tool is mainly limited by three factors: (1) the corpus of interest, as most systems are specifically tailored to one corpus; (2) the complexity of the research question itself, as not every tool's query language is expressive enough for any information need; and (3) the complexity of expressing the specific information needs within a system, including the willingness and ability of researchers to do so. Furthermore, an implementation of the chosen system needs to be available.

4. Conclusion

We conclude by giving a brief summary of our work, discussing its limitations and give an outlook towards future research.

Summary We have defined a data model and query algebra for TNCs. We used this query algebra to express abstract information needs that can be satisfied using specific query systems. A literature review of these systems allowed us to categorize them and evaluate their capabilities, emphasizing different aspects: the corpora the systems can analyze, their expressiveness, and their practical usability.

While the GBNV is by far the most popular tool to analyze TNCs, similar tools exist for smaller corpora. Furthermore, there are query systems which are significantly more flexible in what information needs they can express. Our review is the first to systematically evaluate these systems and bring them together in a concise way, summarizing the state of the art for TNC query systems.

Limitations While we are confident to have given a proper overview of the field of TNC query systems, we are aware of limitations of our approach:

- The literature survey we conducted was not systematic. For reasons we detailed in Section 3.1, a systematic study was not feasible. Nonetheless, we cannot guarantee that we did not miss potentially interesting query systems. Our selection of query systems is rooted in an unsystematic, but thorough literature research. The systems are diverse and implement very different query languages, so we are confident that our review covers the most important classes, even if we may have missed certain systems.
- We only investigated existing query systems, so some interesting ideas for potential query systems could not be included. For example, to the best of our knowledge, there has been no investigation into whether pure SQL can be used to query TNCs efficiently and flexibly. While relational databases are the backend for several systems [24, 30], none of these systems allow free query formulation. Another example is due to the recent development of Large Language Models and their impressive performance on query synthesis tasks [39, 40]: No existing query system leverages these new capabilities, but being able to transfer natural language queries into actual database queries might lower the user-facing complexity of such tasks considerably.
- We evaluated practical usability in a rather abstract manner instead of conducting user studies. We consider our criteria to be well-defined and objective enough to still yield valuable insights, especially because a user study to compare 12 different systems would be a very substantial effort.

Future work We plan to make use of our results by implementing a query system based on the algebra we defined in Section 2. Such a system would combine the strengths of the existing query systems into one unified interface, and—if implemented efficiently—enable more complex TNC analyses than previously possible.

Acknowledgments

This work was supported by the pilot program Core-Informatics of the Helmholtz Association (HGF) and the Initiative and Networking Fund through Helmholtz AI.

Declaration on Generative AI

The author(s) have not employed any Generative AI tools.

References

[1] J. Schlüter, F. Vetter, An interactive visualization of Google Books Ngrams with R and Shiny: Exploring a(n) historical increase in onset strength in a(n) huge database, Journal of Data Mining & Digital Humanities (2020).

[2] F. Breit, The Distribution of English Isograms in Google Ngrams and the British National Corpus, Opticon1826 (2017).

[3] N. Younes, U.-D. Reips, The changing psychology of culture in German-speaking countries: A Google Ngram study, International Journal of Psychology 53 (2018) 53–62.

[4] O. O'Sullivan, R. Duffy, B. Kelly, Culturomics and the history of psychiatry: testing the Google Ngram method, Irish Journal of Psychological Medicine 36 (2019) 23–27.

[5] G. W. Teepe, E. M. Glase, U.-D. Reips, Increasing digitalization is associated with anxiety and depression: A Google Ngram analysis, PLOS One 18 (2023) e0284091.

[6] J. Willkomm, C. Schmidt-Petri, M. Schäler, M. Schefczyk, K. Böhm, A query algebra for temporal text corpora, in: Proceedings of the 18th ACM/IEEE Joint Conference on Digital Libraries, 2018, pp. 183–192.

[7] J.-B. Michel, Y. K. Shen, A. P. Aiden, A. Veres, M. K. Gray, G. B. Team, J. P. Pickett, D. Hoiberg, D. Clancy, P. Norvig, et al., Quantitative analysis of culture using millions of digitized books, Science 331 (2011) 176–182.

[8] Y. Lin, J.-B. Michel, E. A. Lieberman, J. Orwant, W. Brockman, S. Petrov, Syntactic annotations for the Google Books Ngram Corpus, in: Proceedings of 50th Annual Meeting of the Association for Computational Linguistics: System Demonstrations, 2012, pp. 169–174.

[9] J. Mann, D. Zhang, L. Yang, D. Das, S. Petrov, Enhanced search with wildcards and morphological inflections in the Google Books Ngram Viewer, in: Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, 2014, pp. 115–120.

[10] A. Kilgarriff, V. Baisa, J. Bušta, M. Jakubíček, V. Kovář, J. Michelfeit, P. Rychlỳ, V. Suchomel, The Sketch Engine: ten years on, Lexicography 1 (2014) 7–36.

[11] L. Borin, M. Forsberg, J. Roxendal, Korp-the corpus infrastructure of Språkbanken, in: LREC, volume 2012, 2012, pp. 474–478.

[12] C. K. Kreutz, M. Wolz, R. Schenkel, SchenQL: A concept of a domain-specific query language on bibliographic metadata, in: 21st International Conference on Asia-Pacific Digital Libraries, ICADL 2019, Kuala Lumpur, Malaysia, November 4–7, 2019, Proceedings 21, Springer, 2019, pp. 239–246.

[13] C. K. Kreutz, M. Wolz, J. Knack, B. Weyers, R. Schenkel, SchenQL: in-depth analysis of a query language for bibliographic metadata, International Journal on Digital Libraries 23 (2022) 113–132.

[14] M. Coole, P. Rayson, J. Mariani, LexiDB: A scalable corpus database management system, in: 2016 IEEE International Conference on Big Data (Big Data), IEEE, 2016, pp. 3880–3884.

[15] M. Coole, P. Rayson, J. Mariani, LexiDB: Patterns & methods for corpus linguistic database management, in: Proceedings of the Twelfth Language Resources and Evaluation Conference, 2020, pp. 3128–3135.

[16] E. F. Codd, A relational model of data for large shared data banks, Communications of the ACM 13 (1970) 377–387.

[17] J. D. Ullman, Principles of database and knowledge-base systems, Volume I, 1988.

[18] M. Aleksandrov, C. Strapparava, NgramQuery-Smart Information Extraction from Google N-gram using External Resources., in: LREC, 2012, pp. 563–568.

[19] G. A. Miller, Wordnet: a lexical database for english, Communications of the ACM 38 (1995) 39–41.

[20] C. Fellbaum, Wordnet: An electronic lexical database, MIT Press google schola 2 (1998) 678–686.

[21] T. E. Duncan, On the calculation of mutual information, SIAM Journal on Applied Mathematics 19 (1970) 215–220.

[22] S. Sekine, K. Dalwani, Ngram search engine with patterns combining token, POS, chunk and NE information., in: LREC, 2010.

[23] B. de Goede, J. van Wees, M. Marx, R. Reinanda, PoliticalMashup Ngramviewer: Tracking who said what and when in parliament, in: Research and Advanced Technology for Digital Libraries:

International Conference on Theory and Practice of Digital Libraries, TPDL 2013, Valletta, Malta, September 22-26, 2013. Proceedings 3, Springer, 2013, pp. 446–449.

[24] M. Davies, Making Google Books n-grams useful for a wide range of research on language change, International Journal of Corpus Linguistics 19 (2014) 401–416. doi:10.1075/ijcl.19.3.04dav.

[25] V. Todorova, M. Chinkina, R. de Haan, Slash/A n-gram tendency viewer-visual exploration of n-gram frequencies in correspondence corpora, in: Proc. of the ESSLLI, 2014, pp. 229–239.

[26] M. B. Birkenes, L. G. Johnsen, A. M. Lindstad, J. Ostad, From digital library to n-grams: NB N-gram, in: Proceedings of the 20th Nordic Conference of Computational Linguistics (NODALIDA 2015), 2015, pp. 293–295.

[27] C. Schmidt-Petri, M. Schäler, M. Schefczyk, K. Böhm, J. Willkomm, The CHQL Query Language for Conceptual History Using Google Books Ngrams, in: Data for History, 2021.

[28] J. Willkomm, Querying and Efficiently Searching Large, Temporal Text Corpora, Ph.D. thesis, Dissertation, Karlsruhe, Karlsruher Institut für Technologie (KIT), 2021, 2021.

[29] S. Steingrímsson, S. Barkarson, G. T. Örnólfsson, Facilitating corpus usage: Making icelandic corpora more accessible for researchers and language users, in: Proceedings of the Twelfth Language Resources and Evaluation Conference, 2020, pp. 3399–3405.

[30] B. Indig, Z. Sárközi-Lindner, M. Nagy, Use the metadata, Luke!–an experimental joint metadata search and n-gram trend viewer for personal web archives, in: Proceedings of the 2nd International Workshop on Natural Language Processing for Digital Humanities, 2022, pp. 47–52.

[31] A. de Jong, T. Kuzman, M. Larooij, M. Marx, ParlaMint Ngram Viewer: Multilingual comparative diachronic search across 26 parliaments, in: Proceedings of the IV Workshop on Creating, Analysing, and Increasing Accessibility of Parliamentary Corpora (ParlaCLARIN)@ LREC-COLING 2024, 2024, pp. 110–115.

[32] S. Steingrímsson, S. Helgadóttir, E. Rögnvaldsson, S. Barkarson, J. Guðnason, Risamálheild: A very large Icelandic text corpus, in: Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018), 2018.

[33] S. Barkarson, S. Steingrímsson, H. Hafsteinsdóttir, Evolving large text corpora: Four versions of the Icelandic Gigaword Corpus, in: Proceedings of the Thirteenth Language Resources and Evaluation Conference, 2022, pp. 2371–2381.

[34] T. Erjavec, M. Kopp, M. Ogrodniczuk, P. Osenova, D. Fišer, H. Pirker, T. Wissik, D. Schopper, M. Kirnbauer, N. Ljubešić, et al., Multilingual comparable corpora of parliamentary debates ParlaMint 3.0 (2023).

[35] T. Brants, Web 1T 5-Gram Version 1, Philadelphia Linguistic Data Consortium (2006).

[36] T. Brants, A. Franz, The Google Web 1T 5-Gram corpus version 1.1, LDC2006T13 17 (2006).

[37] S. Evert, A. Hardie, Twenty-first century corpus workbench: Updating a query architecture for the new millennium, in: Proceedings of the Corpus Linguistics 2011 conference, Citeseer, 2011, pp. 1–21.

[38] C. Gormley, Z. Tong, Elasticsearch: the definitive guide: a distributed real-time search and analytics engine, " O'Reilly Media, Inc.", 2015.

[39] Z. Hong, Z. Yuan, Q. Zhang, H. Chen, J. Dong, F. Huang, X. Huang, Next-generation database interfaces: A survey of LLM-based text-to-SQL, arXiv preprint arXiv:2406.08426 (2024).

[40] L. Shi, Z. Tang, N. Zhang, X. Zhang, Z. Yang, A survey on employing large language models for text-to-SQL tasks, arXiv preprint arXiv:2407.15186 (2024).