A Query Algebra for Temporal Text Corpora

Jens Willkomm
Karlsruhe Institute of Technology
Department of Informatics
jens.willkomm@kit.edu

Christoph Schmidt-Petri
Karlsruhe Institute of Technology
Department of Humanities
and Social Sciences
christoph.schmidt-petri@kit.edu

Martin Schäler
Karlsruhe Institute of Technology
Department of Informatics
martin.schaeler@kit.edu

Michael Schefczyk
Karlsruhe Institute of Technology
Department of Humanities
and Social Sciences
michael.schefczyk@kit.edu

Klemens Böhm
Karlsruhe Institute of Technology
Department of Informatics
klemens.boehm@kit.edu

ABSTRACT

Researching the evolution of the concepts represented by words, like “peace” or “freedom”, named conceptual history, is an important discipline in the humanities, but still a laborious task. It normally consists of reading and interpreting a large number of carefully selected texts, without however always having a comprehensive knowledge of all the potentially relevant material. Thus, our objective is to design a query algebra to access temporal text corpora. It shall comprehensively allow domain experts to formalize hypotheses on how concepts manifest in large-scale digital text corpora targeting at the complete works of Reinhart Koselleck, a highly prominent researcher in conceptual history. In cooperation with domain experts, we first determine the primary information types used in conceptual history, such as word usage frequency or sentiment. Based on this, we define database operators formalizing these types, which can be combined to formulate arbitrarily complex queries representing hypotheses. The result is a novel query algebra that enables researchers in conceptual history to access large text corpora and extensively analyze word behaviors over time in a comprehensive way. In a proof of concept, we demonstrate how to use our algebra resulting in the first novel insights. This proves the suitability of our algebra.

KEYWORDS

Query algebra; temporal text corpora; database operators; conceptual history

ACM Reference Format:
Jens Willkomm, Christoph Schmidt-Petri, Martin Schäler, Michael Schefczyk, and Klemens Böhm. 2018. A Query Algebra for Temporal Text Corpora. In *JCDL ’18: The 18th ACM/IEEE Joint Conference on Digital Libraries, June 3–7, 2018, Fort Worth, TX, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3197026.3197044

1 INTRODUCTION

The digitization of large time-labeled bibliographies has resulted in corpora such as the Google Ngram data set. Such corpora capture how words are used over time in a comprehensive way. As a result, they are expected to reveal novel insights into the history of how language and thus also how society evolved. This however requires that adequate analysis systems are available. Providing a query algebra that allows domain experts to formalize complex hypotheses is therefore a key factor to successfully unlock this potential. The case of conceptual history serves as our example. In conceptual history, researchers examine the evolution of concepts represented by words such as “peace” or “freedom”. In exploring the history of a concept, scholars commonly make use of, but are not restricted to, word-usage frequencies, word contexts, sentiment analysis, how words refer and relate to and contrast with each other, or they look for word pairs or word families whose usage is correlated [2, 15]. As an example, think of conceptual history researchers who examine whether the words *Osten* and *Westen* (German for East and West) change from cardinal directions to a political meaning and their use in political contexts after 1945.

Existing query algebras, like the one for the Structured Query Language (SQL), have proven their worth, but do not feature specific support for analyses in this spirit. Other approaches from the literature, e. g., the Contextual Query Language [12], the Corpus Query Language [6], or the ANNIS Query Language [20], have similar issues.

In this paper, we propose a query algebra for empirical analysis for temporal text corpora. A *temporal text corpus* in our sense is a set of words and word chains, i. e., ngrams, together with their usage frequency at various points of time. In the humanities, approaching large amounts of texts using information systems is known as *Distant Reading* [9] and is contrasted with the conventional human close reading of individual texts. There now is the need for an approach that (1) is useful for domain experts, i. e., is descriptive and complete, and (2) bears optimization potential to allow fast query processing on large data sets. The ideal of *completeness* represents the ambition to provide an algebra that can match all actual and potential hypotheses of conceptual history. We address the problem of designing such a query algebra and examine its suitability. We focus on an algebra inspired by conceptual history, as exemplified by the work of Reinhart Koselleck [13]. Koselleck is the founding father of German conceptual history and one of the most frequently cited researchers in this field. This paper is part of an interdisciplinary collaboration between philosophers working on conceptual history and computer scientists.

1.1 Challenges

We have to solve two challenges: data characteristics and showing the completeness of our query algebra. We describe these challenges in more detail in the following.

Data Characteristics. Koselleck has given intuitive definitions of his concept types that are more concrete for some concept types and more abstract for others. In contrast, when working with real data, we need observable data characteristics that specify the behavior of individual words.

Completeness. To guarantee that domain experts can examine the whole conceptual history using our query algebra, we must show its completeness. We call our query algebra complete when one can formalize all potential hypotheses regarding Koselleck's conceptual history.

1.2 Contributions

In this work, we present a set of algebra operators that forms our CHQL algebra. It allows analyzing conceptual history related to arbitrary words and addresses the aforementioned challenges as follows.

We need to overcome the gap between the philosophical descriptions of information types and their implementation. Therefore, we define a set of data characteristics. Such a data characteristic is a concrete and quantifiable piece of information, e. g., how often has a word been used in a specific year, or which words are used around this word. The philosophical part is to propose data characteristics that allow to realize all of Koselleck's information types. The computer scientists part is to offer data characteristics and suggest alternative realizations. Our algebra is mostly based on these characteristics. We realize every data characteristic with one algebra operator. For example, our operator *surroundingwords* creates a set of words used around a target word. Another example is our *sentiment* operator. It maps every word to an integer that represents the sentiment value of this word.

Our completeness criterion is to cover all of Koselleck's hypotheses. We argue that using his hypotheses is a suitable and meaningful criterion, given his academic standing in the field. To show the completeness of our algebra in turn, we show that it completely covers the information types to analyze conceptual history.

Finally, we arrive at first novel insights from experimenting with a proof-of-concept implementation.

2 RELATED WORK

Developing methods (and systems) allowing to analyze large text corpora, e. g., from a linguistic or philosophic perspective, is a current trend that has created the *digital humanities*. We now review solutions from this field and declarative query languages in general.

Work in digital humanities mainly consists of data processing and the analysis of text corpora [3, 18]. For example, *distant reading* is a known idea for text analysis [9]. Existing solutions focus on linguistic and reflective properties as well as their evolution, i. e., changes over time, such as [4, 5, 14]. Respective systems cannot output the required information to conduct research on conceptual history in a comprehensive way. In addition, such systems do not provide a sufficiently abstract interface, a reason why experts are reluctant in using them [3].

A very common query algebra is the relational algebra [1, 8]. However, it does not contain sufficiently specific operators, e. g., temporal or linguistic operators. There are extensions to add temporal operators [16, 17], but not linguistic operators. To query relation between words, there are special-purpose query languages. For example, SQWRL is a language to query an ontology [11]. Querying word relations, e. g., from an ontology, does not include all linguistic relationships needed. Further, ontologies do not provide temporal information. SQWRL does not contain any temporal operator. — All of these algebras have in common that they do not cover both linguistic and temporal operators required for research on conceptual history.

3 CONCEPTUAL HISTORY FUNDAMENTALS

In this section, we explain fundamental terms of conceptual history. In the second part, we describe Koselleck's information types.

3.1 Terms of Conceptual History

Here we describe the meaning of important terms of conceptual history.

Concept A concept is a word with a wide range of social and political meanings. This wide range makes it ambiguous. This ambiguousness creates space for interpretation [21].

Concept Type A concept type is a group of concepts having a similar behavior. The objective of conceptual history is to determine which concept type a particular concept has fallen under during a particular period. For example, a *parallel concept* is a concept type containing concepts with a similar role in a particular discourse, such as "love" and "peace".

Information Type Koselleck distinguishes a number of concept types by various criteria. His criteria are for example changes in the sentence structure or changes in a word's linguistic context. We denote Koselleck's criteria as information types. We describe a full list of Koselleck's information types in Section 3.2.

3.2 Koselleck's Information Types

Koselleck does not explicitly give a set of information types he uses for his research. However, experts in this field have extensive knowledge about the information Koselleck uses, and how he argues. We, and the philosophers in particular, now assemble all information types Koselleck considers in his work, as follows.

Geography What is the geographical dispersion of words? In which country are specific words used, and which are not?

Conceptual Design Conceptual design means that words can be used as proxies for a more complex topic than their literal meaning. For instance, when talking about the topic war, one often does not mean the single word war but also words that belong to war, like soldiers and death.

Context The context refers to the linguistic context. A linguistic context is the set of words that are used around a target word. The context is of special interest for concepts and ambiguous words. Considering the context of a word allows to detect its meanings. This is not restricted to ambiguous words. For distinct words, the context might describe a different view on the same circumstance. Take the word *ecology*: It is used at least in political as well as in economic contexts. It has the same meaning, but triggers different targets, namely to take care of people's health — and a restriction to maximize the profit.

Sentence Structure The arrangement of words and phrases define the structure of a sentence. Keeping track of changes in the arrangement of phrases is an indicator of a social change. For example, the phrase *the history of the farmers* changed to *the history of the trade*. This might mean that trade has now taken the role agriculture used to have.

Neologisms A neologism is a word or phrase that either is completely new to a language or is used with a new meaning. It is not necessarily part of the mainstream language. A revolution usually goes along with new words to describe and categorize the new conditions.

World Affairs Conceptual history and the ability to interpret a concept and its meaning require knowledge about world affairs, e. g., changes of economic, political or social circumstances. Such historical events often trigger a social change which is reflected in the language.

Coverage in the Query Algebra. We decide to not deal with geographical information in our algebra. Using a certain language data set, e. g. the British English or the German one, implicitly contains important geographical information already. We also do not explicitly model world affairs, for two reasons. First, the user needs extensive knowledge about world affairs to correctly interpret the results. Second, a user may be looking for special world affairs and its impact on language. When we explicitly model some events, we would only facilitate the investigation of known events.

4 FROM CONCEPT TYPES TO OPERATORS

In this section, we describe how one can search for concept types with the operators which we formally define in Section 6. We show that one can search for concept types that follow the definitions of Koselleck [2]. We explain completeness in three steps, which Figure 1 illustrates. In Section 3.2, we show that Koselleck has come up with a relationship between concept types and a set of information types. In this section, since Koselleck's specifications of information types are rather abstract, we describe an interpretation of his information types. This interpretation is an original contribution of this article, based on the expert knowledge of the philosophers in the team of authors. As part of this step, we also describe a mapping of those types to so-called data characteristics. A data characteristic is a quantitative feature either explicitly present in the data, e. g., the usage frequency of word “peace” in 1969, or a derived piece of information, e. g., the difference between the usage frequency of words “peace” and “war” in 1941. In principle, we can create numerous data characteristics from a temporal text corpus. Hence, in the second step we describe which data characteristics are needed to simulate Koselleck's information needs. In the third step we explain our realization of all data characteristics and their implementation as operators.

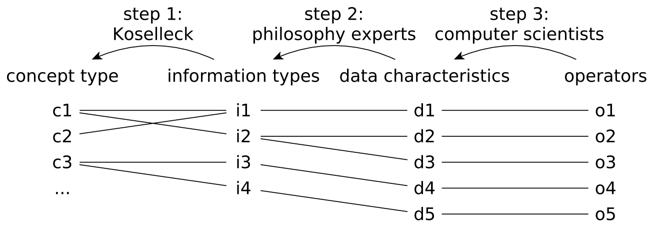


Figure 1: The relationship between concept types, information types, data characteristics and operators

4.1 Step 1: Relationship between Concept Types and Information Types

One of Koselleck's assumptions is that any concept type has its specific characteristic, i. e., any concept type can be described as a combination of information types. For example, words that form a parallel concept might have a significant number of equal surrounding words. However, for most concept types he does not specify this relationship explicitly. This means that we cannot directly look for concept types. We need to define operators to find the information types that are indicators for concept types.

Example 4.1. *We illustrate a relationship between concept types and information types, using counter concepts as example. Counter concepts shape an asymmetric relation between “us” and “them” [21]. – Examples that are part of the following subsections will continue this example and say what concept types and information types are in this specific case. We use counter concepts as a running example in this section wherewith we illustrate each of the three steps.*

4.2 Step 2: Relationship between Information Types and Data Characteristics

Every concept type has its own characteristics, e. g., roles and functions in text. Koselleck's information types are a set of such characteristics. If Koselleck's theory holds, one can describe every concept type c as a combination of information types which characterizes c. This combination always is a subset of all of Koselleck's information types. We therefore strive for a system that finds these information types in large corpora. To this end, we need a formal definition of any information type which is observable and quantifiable. We call such a definition *data characteristic*. Our philosophy experts define a mapping from information types to data characteristics. Before defining all our data characteristics, here is an example for Step 2.

Example 4.2. *This example continues Example 4.1. Words that are often used either near the word “us” or “them” potentially are counter concepts. Well-known counter concepts [21] are:*

- *the bourgeois opposing to the proletarian*
- *the socialist opposing to the liberal*
- *the West opposing to the East*
- *the Protestant opposing to the Catholic*

Hence, one must consider the context of the information type to find indications for counter concepts.

Our Data Characteristics. We now describe our data characteristics which identify each of Koselleck's information types. In Section 6, we define one operator for each of these characteristics.

Conceptual Design Words can have single or multiple meanings, i. e., be ambiguous. We realize finding conceptual design by checking whether a word describes a topic additionally to its own meaning. A topic is a group of words that are used to write about the same issue, e. g. politics or economy. Our experts define the data characteristic of conceptual design as the sum of the usage numbers of all words that belong to a topic. Hence, we propose an operator *topicgrouping* that groups words by its topic and sums their usage numbers. If a word has multiple topics, we sum it into all of these topics.

Context To determine a word's context in Koselleck's spirit requires two data characteristics: (1) a set of surrounding words for a target word, i. e., the linguistic context, and (2) the sentiment for this context, by summing up the sentiment values of the words of the context. Our *surroundingwords* operator and *sentiment* operator implement this.

Sentence Structure One needs to consider two data characteristics: (1) the function of a word, i. e., differentiate between the parts of speech, and (2) completing phrases, i. e., search for missing words in a phrase. The first data characteristic is implemented by our operator *pfilter*. We implement the second one as a pattern-matching operator which we call *textsearch*.

Neologisms The data characteristic of a neologism is an increasing word-usage frequency over time. To find this characteristic, we propose an operator *time series-based selection* that compares the time-series values with a constant. To allow for a temporal restriction, we also provide a *subsequence* operator that limits the selection to an arbitrary time interval. The combination of both operators facilitates the search for neologisms.

4.3 Step 3: Finding Data Characteristics with Operators

In the third step, we implement operators to find the specified data characteristic in a large text corpus. This allows to search for any of Koselleck's information types.

Example 4.3. *This example continues Example 4.2. To find potential counter concepts, we apply the* surroundingwords *operator to our data. It returns a set of surrounding words for every word in the corpus. We now use our* textfilter *operator to search these sets of surrounding words for words like "us" or "them". The more often one of these words is found, the stronger is the indication that this word is part of a counter concept.*

If our operators can identify all data characteristics, one can search for any of Koselleck's information types. This means that one can formalize and test all hypothetical relationships between information and concept types. Hence, we claim that our set of operators is complete with respect to the set of possible hypotheses.

w^1	p^1	ts^3		
		1980	1981	1982
Begriffsgeschichte	$\emptyset$	70	54	58
peace	NOUN	312,031	330,389	295,867
war	NOUN	875,479	878,696	873,246
soldier	NOUN	70,196	72,941	72,587

(a) Example elements of set G_1

w^2	p^2	ts^3		
		1980	1981	1982
Reinhart Koselleck	$\emptyset$ $\emptyset$	65	24	19
conceptual history	$\emptyset$ $\emptyset$	37	31	27
modern history	$\emptyset$ $\emptyset$	3,074	3,165	3,459
history modern	$\emptyset$ $\emptyset$	1	6	4
history books	$\emptyset$ $\emptyset$	2,248	2,205	2,333

(b) Example elements of set G_2

Table 1: Example sets for $gram_n$

5 DATA MODEL

In this section, we define the data model behind our query algebra. We then describe additional data sources which would be necessary to realize all information types described in Section 4. In the last part of this section, we describe notation shorthands and our data sources.

5.1 Ngram

There are three elementary types: words, parts of speech and time series. We use w to represent an individual word and W for a set of words, e. g., $W = \{\emptyset, peace, war, soldier, \ldots\}$. $\vec{w} \in W^n$ is a vector of n words. For example, $W^2 = \{(conceptual, history), \ldots\}$. A part of speech p is an element of $P = \{\emptyset, NOUN, VERB, ADJ, ADV, PRON, DET, ADP, NUM, CONJ, PRT, X\}$. $\vec{p} \in P^n$ is a vector of parts of speech of length n. For example $P^2 = \{(ADV, VERB), \ldots\}$. A time series $\vec{ts} \in \mathbb{Z}^T$ is a vector of integer values of length T. Using these basic types, we now define $gram_n$.

Definition 5.1. A $gram_n$ is a tuple $(\vec{w}, \vec{p}, \vec{ts}) \in W^n \times P^n \times \mathbb{Z}^T$ consisting of a vector of n words, a vector of n parts of speech and a time series of length T. We abbreviate a set of $gram_n$ as G_n.

Parts of speech are classes of words with similar grammatical properties, e. g., nouns or verbs. The part of speech elements are intuitive, e. g., NOUN denotes a noun, VERB a verb. A full description of the part of speech elements is in [7]. Table 1 shows example elements of sets G_1 and G_2.

Projecting on Single Elements. We define functions that project a $gram_n$ tuple to its components. To access the word vector we have function *projwords* of type $G_n \rightarrow W^n$, as follows:

$$\text{projwords}(gram_n) := \vec{w} \tag{1}$$

To access the parts of speech of a $gram_n$ tuple, we have function *projpos* of type $G_n \rightarrow P^n$:

$$\text{projpos}(gram_n) := \vec{p} \tag{2}$$

w_1	sentiment
peace	+1
war	-1

Table 2: An example of sentiment information

w_1	category
war	military
soldier	military
military	military

Table 3: An example of category information

To access the time series, we have function *projts* of type $G_n \longrightarrow \mathbb{Z}^T$:

$$\text{projts}(gram_n) := \vec{ts} \tag{3}$$

5.2 Notation Shorthands

We write $gram_n.w$ as shorthand for projwords($gram_n$), to access the word vector $\vec{w}$. We use $gram_n.p$ and $gram_n.ts$ to project on the parts of speech element and the time series, respectively. We use the notation w_i to access the i^{th} element of vector $\vec{w}$. The same holds for p_i and vector $\vec{p}$. We additionally define two shorthands to access time-series values. The first one ts_y projects a time series to its value of a specific year y. The second notation maps a time series $\vec{ts}$ to a subsequence $\mathbb{Z}^T \longrightarrow \mathbb{Z}^t$ where $0 \leq t \leq T$. The access to a subsequence from year y_s to year y_e inclusively is as follows.

$$\vec{ts}[y_s, y_e] := \begin{cases} \text{subsequence from } y_s \text{ to } y_e & \text{if } y_s < y_e \\ ts_{y_s} & \text{if } y_s = y_e \\ \text{empty subsequence} & \text{else} \end{cases} \tag{4}$$

5.3 Sentiment and Category

To realize all the information types, we need to combine information from different sources. We need to consider two additional kinds of information: the sentiment value of a word and its category. A word sentiment can be positive, negative or neutral. To implement such a contrast-word rating, we use a binary word-classification mechanism. A category is a group of words that are used in the same topic, such as religion or economy. To realize this, we need the information which category a word belongs to. We now define both kinds of additional information, the sentiment and the category.

Sentiment. In conceptual history, it is important to investigate the relations between concepts, e. g., differences in positive and negative sentiments, or their orientation towards the past or the future. Signal words typically make these differences explicit. To illustrate, in the phrase *the hope for peace* the word "hope" is a word that signals a positive phrasing of the sentence. The additional information is the knowledge about signal words and its positive or negative classification. To define the sentiment value for a word, we define a function *wordsentiment*: $W^n \longrightarrow \mathbb{Z}$. The function maps single words as well as word vectors to a sentiment value. For example, the single word "power" is a positive word and may have a sentiment value of +1. The 2-gram "electrical power" in turn does neither have a positive nor a negative sentiment value. The sentiment value for a neutral rating is 0. *wordsentiment* returns integer values to allow modeling different grades of positive and negative sentiment. Table 2 lists example words and their sentiments.

Category. A category is a word that is used as proxy for a more complex topic than its literal meaning, e. g., the category military includes the word "war" as well as "soldiers" and others. To implement information type "conceptual design", we need to model the relationship between words and its categories. We define *category* as a function of type $W^n \longrightarrow C$ that maps words to the category they belong to, where $C \subset W^1$ is a set of categories. Every category in C is described as a single word of the set of all words W^1. For example, the word "military" is a word ($military \in W^1$) and a category ($military \in C$). The word "soldier" in turn is a word $soldier \in W^1$, but does not describe a category $soldier \notin C$. Table 3 shows an example of such a category grouping.

Information Sources. We store the sentiment information as well as the category information independent of one specific source. Therefore, we are able to use every source that contains this information and also to replace a source if a better one is available. Due to their informedness regarding text and word interpretation, the philosophers in our team have decided for the following sources. We extract the sentiment information from the LIWC list [19]. This list is a common reference in computerized text analysis. We use the OpenThesaurus[1] database [10] for the mapping between words and categories.

6 QUERY OPERATORS

Section 3 lists a complete set of information types to analyze conceptual history. We now propose a corresponding set of query operators. Some operators realize an entire information type, while one needs a combination of operators to realize other, more complex types. This section introduces the operators, gives their definitions and shows how they implement the information types.

6.1 Topic Grouping Operator

We realize information type *conceptual design* by defining a topic grouping operator. To realize this abstraction, this operator groups all words from the same topic. The groups have names consisting of one word. Thus, the result is of type $gram_1$. This allows us to define word categories e. g., *military* or *religious*, and analyze or compare their developments over time. We first define the set of occurrence topics for a given set of words and afterwards we define our *topicgrouping* operator.

Definition 6.1. C_G is the set of categories that appear in set G.

$$C_G := \bigcup_{g \in G} \{\text{category}(g.\vec{w})\} \tag{5}$$

[1]https://www.openthesaurus.de

w^1	p^1	ts^3		
		1980	1981	1982
military	∅	945,675	951,637	945,833

Table 4: Example result set of the topic grouping operator

Definition 6.2. *topicgrouping* is an operator of type $\mathcal{P}(G_n) \rightarrow \mathcal{P}(G_1)$. It has the following semantics:

$$\text{topicgrouping}(G) := \bigcup_{c \in C_G} (c, \emptyset, \sum_{\{g \in G | \text{category}(g.\vec{w})=c\}} g.\vec{ts}) \tag{6}$$

The topicgrouping operator groups all ngrams by their topic and sums together their time series.

Example. This example, as well as the ones that follow, use the data in Tables 1, 3 and 2. *topicgrouping* sums up the time series of all words belonging to category military. The result is a time series with the number of times authors have written about a military topic. For our example data, the operator yields the result in Table 4.

6.2 Surrounding Words Operator

When computing the context of words, a *target word* is the word we determine the context for, whereas *context words* are the words used around the target word. Before defining the operator, we define two auxiliary functions. The first one maps a word vector to the set of words of the vector. The second function maps a word vector to a word vector of a higher dimensionality that includes the same elements as the original word vector. We first define the help function *split* and secondly use this function to define our *expandwords* operator.

Definition 6.3. *split* is a function of type $W^n \rightarrow \mathcal{P}(W^1)$.

$$\text{split}(w \in W^n) := \{w_1, w_2, \ldots, w_n\} \tag{7}$$

split maps a vector of n words to a set of n words.

Definition 6.4. *expandwords* is a function of type $\mathbb{N} \times W^n \rightarrow \mathcal{P}(W^m)$ with $n < m; n, m \in \mathbb{N}$.

$$\text{expandwords}(m, \vec{q} \in W^n) := \left\{ (w_1, w_2, \ldots, w_m) \in W^m \middle| \exists o : \bigwedge_{i=1}^{n} w_{i+o} = q_i \right\} \text{ with } 0 \leq o \leq m - n \tag{8}$$

expandwords maps a vector of words to a set of vectors that are of higher dimensionality and contain the original word vector. This includes word vectors with new words in the front or in the back or both. Parameter m defines the dimensionality of the target vectors. Function *expandwords* adds $m - n$ words to the input vector. For example, we expand the word vector "history" of length $n = 1$ to a word vectors of length $m = 2$. A partial result are the word vectors "the history", "conceptual history", or "history of".

w^1	p^1	ts^3		
		1980	1981	1982
conceptual	∅	75,586	78,319	84,518
modern	∅	523,599	510,492	532,338
books	∅	447,885	436,655	462,202

Table 5: Example result set of *surroundingwords*

Definition 6.5. *surroundingwords* is an operator of type $\mathbb{N} \times G_n \rightarrow \mathcal{P}(G_1)$. It has the following semantics:

$$\text{surroundingwords}(m, g \in G_n) := \bigcup_{i \in \text{expandwords}(m, g.\vec{w})} \bigcup_{j \in \text{split}(i)} \{\text{projwords}^{-1}(j)\} \tag{9}$$

The *surroundingwords* operator returns a set of context words occurring together with at least one of the target words in a window of size m. We split the description into three steps. First, we filter all multigrams of size m for those that contain at least one target word vector. Second, we split these multigrams into single words and third, we add the corresponding usage-frequency time series to the context words.

Here are the three steps in more detail.

(1) The help function *expandwords* selects all word vectors of size m which contain the target word vector $\vec{q}$, no matter at which position of the window it occurs. *expandwords* returns a set of word vectors.
(2) Having the set of word vectors that contain the target word vector, we split these word vectors into single words. To perform this splitting we need function *split*. This results in a set of single words.
(3) To return elements of type $gram_1$ instead of single words, we use function projwords^{-1} to get the $gram_1$ elements that contain the corresponding word. We do this for every single word that function *split* returns. The result is the union over all $gram_1$ elements of all surrounding words.

Example. Suppose that we want all surrounding words for the word *history* within a window of size $m = 2$. Then we get the result shown in Table 5.

6.3 Sentiment Operator

Another important information for conceptual history is a word's sentiment, i. e., the positive or negative emotions associated with a word. This operator can be used to determine the sentiment for a single word or for a set of context words. This operator is the second part to completely cover the *context* information types from Section 3.

Definition 6.6. *sentiment* is an operator of type $G_n \rightarrow G_n$. It has the following semantics:

$$\text{sentiment}(g \in G_n) := (g.\vec{w}, g.\vec{p}, g.\vec{ts} \cdot \text{wordsentiment}(g.\vec{w})) \tag{10}$$

This operator multiplies the usage frequency of a word by the word's sentiment value. This multiplication takes both into account, the word's sentiment value and the word's usage frequency in the

w^1	p^1	ts^3		
		1980	1981	1982
peace	NOUN	+312,031	+330,389	+295,867
war	NOUN	-875,479	-878,696	-873,246

Table 6: Example result set of the sentiment operator

w^2	p^2	ts^3		
		1980	1981	1982
history modern	∅ ∅	1	6	4
history books	∅ ∅	2,248	2,205	2,333

Table 7: Example result set of the textsearch operator

text. The sentiment operator changes the meaning of the time-series values. The time-series values then no longer stand for the usage frequency of the word, but its sentiment over time. More precisely, the time-series values represent the sentiment value of a ngram over time.

The sentiment operator is defined for all lengths of ngrams. However, it only matches the same ngram length that is specified in the sentiment mapping, but not shorter or longer ones. For instance, the sentiment for *war* only matches the 1-gram *war*, but not the 2-gram *civil war*. This allows a fine-grained definition of sentiment values, as it becomes possible to have different sentiment values for *War*, *Civil War*, *First World War*, *Second World War* and *Cold War*.

Example. Table 6 shows an example result of *sentiment*.

6.4 Text Search Operator

Having a large set of words, a fundamental need is to search for the words conceptual historians might be interested in which matches some pattern pt. Pt is short for a set of patterns. $Z = \{\exists, \forall\}$ is the set of standard quantifiers.

Definition 6.7. textsearch is an operator of type $Pt \times Z \times \mathcal{P}(G_n) \rightarrow \mathcal{P}(G_n)$. It maps its input as follows:

$$\text{textsearch}(pt, \zeta, G_n) := \{g \in G_n \mid \zeta i : g.w_i \text{ matches } pt\} \tag{11}$$

The operator selects all tuples that satisfy the given condition with $\zeta \in \{\exists, \forall\}$ as quantifier and pt as a search pattern. The quantifier controls whether all words need to match pt or just one.

Example. When searching for the Pattern "histo.*" the *textsearch* operator returns the set in Table 7.

6.5 Part of Speech Filtering

Analyses in conceptual history often only refer to specific parts of speech such as nouns. So we propose a filter to keep only ngrams having a specific parts of speech, e. g., it allows to select all nouns.

Definition 6.8. pfilter is an operator of type $P^n \times Z \times \mathcal{P}(G_n) \rightarrow \mathcal{P}(G_n)$. It maps its input as follows:

$$\text{pfilter}(p, \zeta, G_n) := \{g \in G_n \mid \zeta i : g.p_i = p\} \tag{12}$$

w^1	p^1	ts^3		
		1980	1981	1982
peace	NOUN	312,031	330,389	295,867
war	NOUN	875,479	878,696	873,246

Table 8: Example result set of the pfilter operator

w^1	p^1	ts^3		
		1980	1981	1982
peace	NOUN	312,031	330,389	295,867
war	NOUN	875,479	878,696	873,246

Table 9: Example result set of the time series-based selection

Like in the definition of the textsearch operator, we use the quantifier $\zeta \in \{\exists, \forall\}$ to match the part of speech for at least one element or for all elements. In contrast to the textsearch operator, we can only search for part of speech tags.

Example. Selecting all nouns returns a set like the one in Table 8.

6.6 Time Series-based Selection

We may want to exclude words with extreme time-series values from our analysis, e. g., very rarely used words, because they might falsify our results. So we need a filter for time-series characteristics. $\Theta = \{<, \leq, =, \neq, \geq, >\}$ is the set of standard comparison operators.

Definition 6.9. cond is a tuple $(\zeta, \theta, v) \in Z \times \Theta \times \mathbb{Z}$ of conditions *Cond*.

Definition 6.10. $F_{cond} \subseteq \mathbb{Z}^T$ is a set of time series that fulfill condition *cond*.

$$F_{(\zeta,\theta,v)} := \{\vec{ts} \in \mathbb{Z}^T \mid \zeta i : ts_i \, \theta \, v\} \tag{13}$$

Definition 6.11. tsselection is an operator of type $Cond \times \mathcal{P}(G_n) \rightarrow \mathcal{P}(G_n)$. It maps its input as follows:

$$\text{tsselection}(cond, G_n) := \{g \in G_n | g.\vec{ts} \in F_{cond}\} \tag{14}$$

The time series-based selection operator selects $gram_n$ elements based their time-series properties.

Example. Condition $cond = (\forall, >, 1000)$ is the condition to select all ngrams whose usage frequency is larger than 1,000 for every available year. The resulting set is shown in Table 9.

6.7 kNN Operator

Another information that is of interest for analyzing conceptual history is to find words with a similar evolution of usage frequency, sentiment or surrounding words. This leads us to the definition of a kNN operator, which returns the k ngrams having the most similar time series to a given ngram. This information type is part of analyzing the *grammatical structure*.

w^2	p^2	ts^3		
		1980	1981	1982
conceptual history	NULL NULL	37	31	27

Table 10: Example result set of the kNN operator

w^2	p^2	ts^2	
		1980	1981
Reinhart Koselleck	$\emptyset$ $\emptyset$	65	24
conceptual history	$\emptyset$ $\emptyset$	37	31

Table 11: Example result set of the subsequence operator

Definition 6.12. *kNN* is an operator of type $\mathbb{N} \times G_n \times \mathcal{P}(G_n) \to \mathcal{P}(G_n)$. It maps its input as follows:

$$\text{kNN}(k, q, G_n) := \underset{\{S \in \mathcal{P}(G_n): |S|=k\}}{\arg\min} \; \max_{s \in S} \left\| s.\vec{ts} - q.\vec{ts} \right\|_2 \tag{15}$$

Our kNN operator implements a kNN search based on the time-series values of the $gram_n$ elements. Input k defines the number of resulting elements, i. e., the number of nearest neighbors to search for. q defines the target $gram_n$ to search the neighbors for. Input set G_n defines the search space, i. e., the set of possible results. The kNN operator can be used with different distance measures. For example, our implementation currently supports the Euclidean distance and dynamic time warping (DTW).

Example. Searching for the $k = 2$ nearest neighbors of the target word "Reinhart Koselleck" yields the result in Table 10.

6.8 Subsequence Operator

Most time-specific information have a specific start or end date or are in a specific temporal range of historical events, e. g., in the period from 1945 to 1990. However, the operators above work over the whole time range. Hence, we need an operator that reduces time series to a certain time interval.

Definition 6.13. *subsequence* is an operator of type $G_n \times \mathbb{N} \times \mathbb{N} \to G_n$. It maps its input as follows:

$$\text{subsequence}(g, y_s, y_e) := (g.\vec{w}, g.\vec{p}, g.\vec{ts}[y_s, y_e]) \tag{16}$$

The subsequence operator has three parameters: the $gram_n$ element g, the start year y_s and the end year y_e. It takes the input ngram element and reduces its time series to the time window $[y_s, y_e]$.

Example. When applying function *subsequence* to consider only the years from 1980 to 1981, the result looks like the set in Table 11.

6.9 Absolute Value Operator

Using the sentiment operator allows generating positive and negative time-series values representing positive or negative word emotions. For some investigations of conceptual history one needs to quantify word's emotions, no matter whether they are positive or negative. To archive this, we first apply the sentiment operator and secondly the absolute value operator that maps all sentiment values to positive values, i. e., the absolute value of the sentiment value.

w^2	p^2	ts^3		
		1980	1981	1982
peace	NOUN	+312,031	+330,389	+295,867
war	NOUN	+875,479	+878,696	+873,246

Table 12: Example result set of the absolute operator

integer value
3

(a) Example result of the count operator

ts^3		
1980	1981	1982
$-563,448$	$-548,307$	$-577,379$

(b) Example result of operator sumup

Table 13: Example results of the operators count and sumup

Definition 6.14. *tsabs* is a function of type $\mathbb{Z}^T \to \mathbb{N}_0^T$, as follows:

$$\text{tsabs}(\vec{ts}) := (|ts_1|, |ts_2|, \ldots, |ts_T|) \tag{17}$$

tsabs maps every value of a given time series to its absolute value.

Definition 6.15. *absolute* is an operator of type $G^n \to G^n$. It maps its input as follows:

$$\text{absolute}(g) := (g.\vec{w}, g.\vec{p}, tsabs(g.\vec{ts})) \tag{18}$$

The absolute value operator turns every time-series value of a $gram_n$ element into its absolute value. Function *tsabs* maps the values for every single year to its absolute value and creates a new time series with absolute values.

Example. Applying this operator to the previous set yields the result in Table 12.

6.10 Count Operator

When working with huge data sets, which is preferred for statistical analysis, for many queries a human user simply cannot look at all elements or count them manually. The count operator counts the number of elements of a given set.

Definition 6.16. *count* is an operator of type $\mathcal{P}(G_n) \to \mathbb{N}$. It maps its input as follows:

$$\text{count}(G \in \mathcal{P}(G_n)) := |G| \tag{19}$$

Given an input set, *count* counts its elements and returns the number of containing elements.

Example. When we want to count the number of nouns in our data set in Table 1, we first filter for nouns, using the *pfilter* operator, followed by the *count* operator. Table 13a shows the result.

6.11 Sumup Operator

There is often the need to sum up the time series of ngrams, e. g., to compare the frequency of all nouns or adjectives. The sumup operator gets a set of ngrams, sums up their time series and returns the resulting time series. In comparison to the topicgrouping operator, the sumup operator does not group the elements, but sums up all received time series.

Definition 6.17. sumup is an operator of type $\mathcal{P}(G_n) \to \mathbb{Z}^T$. It maps its input as follows:

$$\text{sumup}(G \in \mathcal{P}(G_n)) := \sum_{g \in G} g.\vec{ts} \qquad (20)$$

The sumup operator iterates over the given set of ngrams and sums together their time series. In the case of summing up sentiment time series, the operator may return a time series containing negative values.

Example. The result set when applying this operator to the example set from the sentiment operator looks like the one in Table 13b.

6.12 Set Operators

Since we are working on sets, we are able to use the set operators intersection (∩), union (∪) and minus (\). The key on which these operators work is a combination of the word vector w and the corresponding part of speech, i. e., vector p. The time-series attribute of the result set is taken from the left input set.

6.13 Algebraic Expressions

So far, we have introduced individual operators. In order to formulate complex hypotheses revealing novel insights for conceptual history, one generally needs to combine operators. For instance, in case one wants to find the k nearest nouns to some given ngram in a specific time period, we can combine the existing operators to formulate an expression to compute the desired result. Operators are compatible if the output of the first operator is of the same type as the input of the second operator. Most of our operators result in a set of ngram tuples, but not all of them. There are two operators that do not result in such a set: *count* and *sumup*. Both operators yield a single value. These operators usually are the final operation in an algebraic expression, e. g., summing up the input tuples. How to come from a philosophical hypothesis to concrete queries, and how example results may look like is the topic of the next section.

7 PROOF OF CONCEPT

Our query algebra allows (1) hypothesis testing and (2) hypotheses engineering. We explain both in this section.

7.1 Hypothesis Testing

Hypothesis testing enables scholars to empirically test existing hypotheses regarding properties of different concept types. For example, scholars want to test a hypothesis that characterizes parallel concepts. This consists of the following steps: First, they formalize their hypothesis, i. e., translate it to an algebraic expression. Second, they let a machine evaluate it on a huge corpus. Third, they interpret the results. To illustrate, think of the two hypotheses:

Hypothesis H1 The words "Osten" and "Westen" (German for East and West) have acquired a political meaning, i. e,. a political context, during the Cold War, while the words "Norden" and "Süden" (German for North and South) have not changed their merely geographical meaning.

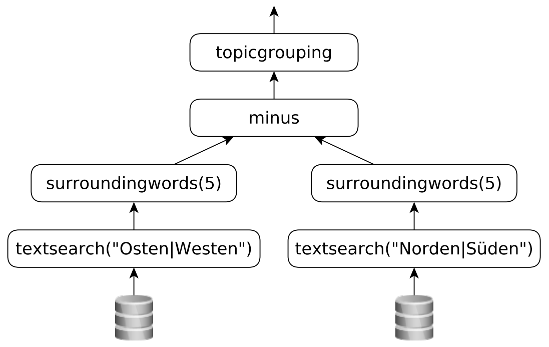
Hypothesis H2 The two words "Osten" and "Westen" have developed into contrasting classes, i. e., have changed from parallel concepts to counter concepts.

Formalizing Hypothesis H1. To formalize Hypothesis H1, we use several operators, see Figure 2a for the operator tree. First, we need the text-search operator for occurrences of the four cardinal directions. We then generate the common context of "Osten" and "Westen" as well as the one of "Norden" and "Süden". To remove the context words related to cardinal directions, we subtract the common context of "Norden" and "Süden" from the one of "Osten" and "Westen". This leads to the context of "Osten" and "Westen" that is not related to cardinal directions. In a final step, we categorize the remaining context words. Figure 2b visualizes the result of computing this operator tree.

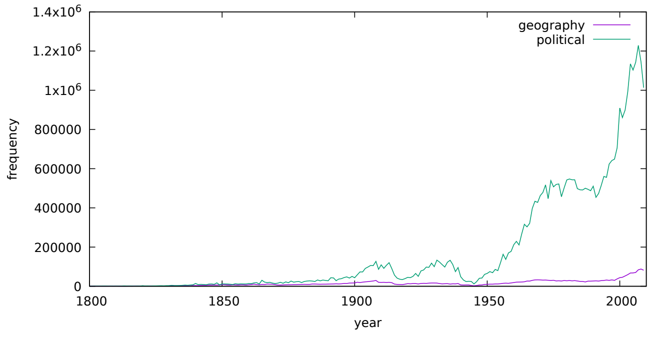
Formalizing Hypothesis H2. Given the validation of Hypothesis H1 that the words "Osten" and "Westen" become a political context, in Hypothesis H2 we check whether the semantics of the context for these two words develop into contrasting directions, i. e., if one word gets a positive context while the other one gets a negative one. We again start using our text-search operator to select the appropriate occurrences. We then separately generate the context for "Osten" and for "Westen" and perform a sentiment analysis. The operator tree to test Hypothesis H2 is shown in Figure 3a. Figure 3b visualizes the result. The result shows that the word "Westen" is used in a context mainly consisting of positive words while the words around "Osten" sum up to a negative sentiment. There can be two reasons for this: (1) The word "Westen" has more positive surrounding words that overcome the negative ones, or (2) the word "Osten" misses some surrounding words that the word "Westen" has and therefore sums to a negative sentiment. One can test these two possible reasons with a corresponding expression in our algebra.

7.2 Hypotheses Engineering

Hypotheses engineering means developing criteria and hypotheses to distinguish between concept types, i. e., speculating about their differences, formalizing and executing algebraic expressions, and interpreting the result. We, especially the philosophers team, are interested in checking hypothesis to get a better understanding of the differences between parallel and counter concepts. From close reading, they already have some hypotheses regarding the characteristics of parallel concepts. These hypotheses relate to the usage frequency, the surrounding words and the sentiment values of words. We have formalized and tested all these hypotheses, with the outcome that we could not confirm any of them. Our conceptual history experts then have had a closer look at the results of the expression. They have observed that some queries produce better results than others, i. e., contain more parallel concepts. This confirms that our approach gives way to important information for conceptual history experts, to develop new hypotheses. At the end

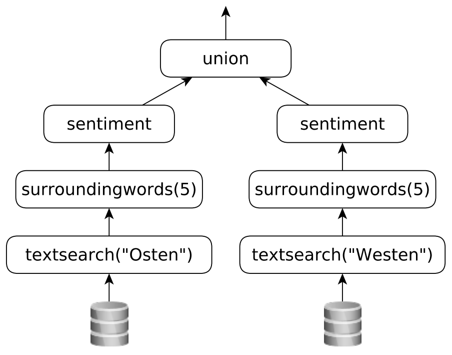


(a) The operator tree to realize the expression to test Hypothesis H1

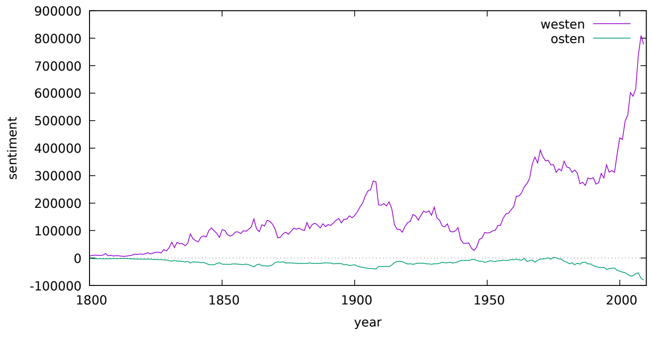


(b) Result when executing operator tree to test Hypothesis H1

Figure 2: Testing Hypotheses H1



(a) The operator tree to realize the expression to test Hypothesis H2



(b) Result when executing operator tree to test Hypothesis H2

Figure 3: Testing Hypotheses H2

of this process, our experts have been able to identify important information types of a parallel concept, as follows: A parallel concept features two concepts whose surrounding words have a similar sentiment, and they share numerous surrounding words. As part of our future work, we want to confirm this more rigidly, with a lot of close reading.

8 CONCLUSIONS

In this paper we propose the CHQL query algebra that supports exploring and understanding conceptual history. We have shown that our algebra completely covers all information types used by the pioneer of conceptual history, Reinhart Koselleck. We have demonstrated the usefulness of the algebra by formalizing two important hypotheses in conceptual history and testing it on Google's ngram data.

REFERENCES

[1] Serge Abiteboul, Richard Hull, and Victor Vianu. 1995. *Foundations of Databases: The Logical Level.* Addison-Wesley.
[2] Otto Brunner, Werner Conze, and Reinhart Koselleck. 2004. *Geschichtliche Grundbegriffe.* Vol. 1–8. Klett-Cotta.
[3] Shalin Hai-Jew. 2017. *Data Analytics in Digital Humanities.* Springer.
[4] William L. Hamilton, Jure Leskovec, and Dan Jurafsky. 2016. Cultural Shift or Linguistic Drift? Comparing Two Computational Measures of Semantic Change. In *Empirical Methods in Natural Language Processing.*
[5] William L. Hamilton, Jure Leskovec, and Dan Jurafsky. 2016. Diachronic Word Embeddings Reveal Statistical Laws of Semantic Change. (2016). arXiv:1605.09096v4
[6] Miloš Jakubíček, Adam Kilgarriff, Diana McCarthy, and Pavel Rychlỳ. 2010. Fast Syntactic Searching in Very Large Corpora for Many Languages. In *Pacific Asia Conference on Language, Information and Computation (PACLIC).*
[7] Yuri Lin, Jean-Baptiste Michel, Erez Lieberman Aiden, Jon Orwant, Will Brockman, and Slav Petrov. 2012. Syntactic Annotations for the Google Books Ngram Corpus. In *Annual Meeting of the Association for Computational Linguistics (ACL).* 169–174.
[8] David Maier. 1983. *Theory of Relational Databases.* Computer Science Press.
[9] Franco Moretti. 2013. *Distant Reading.* Verso Books.
[10] Daniel Naber. 2005. OpenThesaurus: Ein offenes deutsches Wortnetz. In *Sprachtechnologie, mobile Kommunikation und linguistische Ressourcen: Beiträge zur GLDV-Tagung 2005 in Bonn.* 422–433.
[11] Martin O'Connor and Amar Das. 2009. SQWRL: a Query Language for OWL. In *OWL: Experiences and Directions (OWLED).*
[12] The Library of Congress. 2013. The Contextual Query Language. (2013). https://www.loc.gov/standards/sru/cql/
[13] Niklas Olsen. 2012. *History in the Plural: An Introduction to the Work of Reinhart Koselleck.* Berghahn Books.
[14] Vinodkumar Prabhakaran, William L. Hamilton, Dan McFarland, and Dan Jurafsky. 2016. Predicting the Rise and Fall of Scientific Topics from Trends in their Rhetorical Framing. In *Annual Meeting of the Association for Computational Linguistics (ACL).* 1170–1180. https://doi.org/10.18653/v1/p16-1111
[15] Joachim Ritter, Karlfried Gründer, and Gottfried Gabriel. 2007. *Historisches Wörterbuch der Philosophie.* Vol. 1–13. Schwabe.
[16] Richard Snodgrass. 1987. The temporal query language TQuel. *ACM Transactions on Database Systems* 12, 2 (1987), 247–298. https://doi.org/10.1145/22952.22956
[17] Richard T. Snodgrass (Ed.). 1995. *The TSQL2 Temporal Query Language.* Springer.
[18] Claire Warwick. 2012. *Digital Humanities in Practice.* Facet Publishing.
[19] Markus Wolf, Andrea B. Horn, Matthias R. Mehl, Severin Haug, James W. Pennebaker, and Hans Kordy. 2008. Computergestützte quantitative Textanalyse. *Diagnostica* 54, 2 (2008), 85–98. https://doi.org/10.1026/0012-1924.54.2.85
[20] Amir Zeldes, Anke Lüdeling, Julia Ritz, and Christian Chiarcos. 2009. ANNIS: a search tool for multi-layer annotated corpora. In *Proceedings of Corpus Linguistics.* https://doi.org/10.18452/13437
[21] Niels Åkerstrøm Andersen. 2003. *Discursive Analytical Strategies: Understanding Foucault, Koselleck, Laclau, Luhmann.* Policy Press.