# An Algebraic Approach for High-level Text Analytics

Xiuwen Zheng
xiz675@eng.ucsd.edu
University of California San Diego
La Jolla, Californa, USA

Amarnath Gupta
a1gupta@ucsd.edu
University of California San Diego
La Jolla, Californa, USA

## ABSTRACT

Text analytical tasks like word embedding, phrase mining and topic modeling, are placing increasing demands as well as challenges to existing database management systems. In this paper, we provide a novel algebraic approach based on associative arrays. Our data model and algebra can bring together relational operators and text operators, which enables interesting optimization opportunities for hybrid data sources that have both relational and textual data. We demonstrate its expressive power in text analytics using several real-world tasks.

## CCS CONCEPTS

• **Information systems → Information retrieval query processing**.

## KEYWORDS

associative array, text analytics, natural language processing

**ACM Reference Format:**
Xiuwen Zheng and Amarnath Gupta. 2020. An Algebraic Approach for High-level Text Analytics. In *32nd International Conference on Scientific and Statistical Database Management (SSDBM 2020), July 7–9, 2020, Vienna, Austria.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3400903.3400926

## 1 INTRODUCTION

A significant part of today's analytical tasks involve text operations. A data scientist who has to manipulate and analyze text data today typically uses a set of text analysis software libraries (e.g., NLTK, Stanford CoreNLP, GenSim) for tasks like word embedding, phrase extraction, named entity recognition and topic modeling. In addition, most DBMS systems today have built-in support for full-text search. PostgreSQL, for instance, admits a text vector (called tsvector) that extracts and creates term and positional indices to enable efficient queries (called tsquery). Yet, some common and seemingly simple text analysis tasks cannot be performed simply within the boundaries of a single information system.

**Example 1.** Consider a relational table $R$(newsID, date, newspaper, title, content) where $title$ and $content$ are text-valued attributes, and two sets $L_o, L_p$ that represent a collection of organization names and person names respectively. Now, consider the following analysis:

- $N1$ = Select a subset of news articles between dates $d_1$ and $d_2$
- $N2$ = Identify all news articles in $N1$ that have at least $c_1$ organization names from $L_o$ and $c_2$ persons from $L_p$
- $T1$ = Create a document-term matrix on $N2.text$
- $T2$ = Remove rows and columns of the matrix if either of their row or column marginal sums is below $\theta_1$ and $\theta_2$ respectively
- $M$ = Compute a topic model using $T2$

The intention of the analysis is to find the topic distribution of those news items that cover, for example, any two members of the senate (list $L_p$) and any one government organizations (list $L_o$). The analysis itself is straightforward and can be performed with a combination of SQL queries and Python scripts.

Our goal in this short paper is to present the idea that a novel *relation-flanked associative array data model* has the potential of serving as the underlying framework for the management and analysis of text-centric data. We develop the theoretical elements of the model and illustrate its utility through examples.

## 2 THE DATA MODEL

### 2.1 Text Associative Arrays

A number of current data systems, typically in the domain of polystore data systems, use associative arrays [3, 4] or its variants like associative tables [1] and tensor data model [5]. Many of these data models are used to support analytical (e.g., machine learning) tasks. In our setting, we specialize the essential associative model for text analytics. For our level of abstraction, our model reuses relational operations for all metadata of the associative arrays. While it has been shown [1] that associative arrays can express relational operations, we believe that using relational abstraction along with our text-centric algebraic operations makes the system easier to program and interpret. At a more basic level, since most text processing operations include sorting (e.g., by TF-IDF scores), our model is based on partially ordered semirings.

*Definition 2.1 (Semiring).* A semiring is a set $R$ with two binary operations addition $\oplus$ and multiplication $\odot$, such that, 1) $\oplus$ is associative and commutative and has an identity element $0 \in R$; 2) $\odot$ is associative with an identity element $1 \in R$; 3) $\odot$ distributes over $\oplus$; and 4) $\odot$ by 0 yields 0.

*Definition 2.2 (Partially-Ordered Semiring).* [2] A semiring $R$ is partially ordered if and only if there exists a partial order relation $\leq$ on $R$ satisfying the following conditions for all $a, b \in R$:

- If $a \leq b$, then $a \oplus c \leq b \oplus c$;
- If $a \leq b$ and $0 \leq c$, then $a \odot c \leq b \odot c$ and $c \odot a \leq c \odot b$.

*Definition 2.3 (Text Associative Array).* The Text Associative Array (TAA) $\mathbf{A}$ is defined as a mapping:

$$\mathbf{A} : K_1 \times K_2 \rightarrow R$$


*SSDBM 2020, July 7–9, 2020, Vienna, Austria*

ACM ISBN 978-1-4503-8814-6/20/07.
https://doi.org/10.1145/3400903.3400926

where $K_1$ and $K_2$ are two key sets (named row key set and column key set respectively), and $R$ is a partially-ordered semiring (Definition 2.2). We call $K_1 \times K_2$ "the dimension of $\mathbf{A}$", and denote $\mathbf{A}.K_1$, $\mathbf{A}.K_2$ and $\mathbf{A}.K$ as the row key set, column key set, and set of key pairs of $\mathbf{A}$, respectively.

Next, we define the basic operations on text associative arrays, to be used by our primary text operations (Sec. 2.2).

*Definition 2.4 (Addition).* Given two TAAs $\mathbf{A}, \mathbf{B} : K_1 \times K_2 \to R$, the addition operation $\mathbf{C} = (\mathbf{A} \oplus \mathbf{B}) : K_1 \times K_2 \to R$ is defined as,

$$\mathbf{C}(k_1, k_2) = (\mathbf{A} \oplus \mathbf{B})(k_1, k_2) = \mathbf{A}(k_1, k_2) \oplus \mathbf{B}(k_1, k_2).$$

Define $\mathbb{0}_{K_1,K_2}$ as a TAA where $\mathbb{0}_{K_1,K_2}(k_1, k_2) = 0$ for $\forall k_1 \in K_1, \forall k_2 \in K_2$. $\mathbb{0}_{K_1,K_2}$ serves as an identity for addition operation on key set $K_1 \times K_2$.

*Definition 2.5 (Hadamard Product).* Given two TAAs $\mathbf{A}, \mathbf{B} : K_1 \times K_2 \to R$, the Hadamard product operation $\mathbf{C} = (\mathbf{A} \odot \mathbf{B}) : K_1 \times K_2 \to R$ is defined as,

$$\mathbf{C}(k_1, k_2) = (\mathbf{A} \odot \mathbf{B})(k_1, k_2) = \mathbf{A}(k_1, k_2) \odot \mathbf{B}(k_1, k_2).$$

Define $\mathbb{1}_{K_1,K_2}$ as a TAA where $\mathbb{1}_{K_1,K_2}(k_1, k_2) = 1$ for $\forall k_1 \in K_1, \forall k_2 \in K_2$. $\mathbb{1}_{K_1,K_2}$ serves as an identity for Hadamard product on key set $K_1 \times K_2$.

*Definition 2.6 (Array Multiplication).* Define $\bigoplus$ as addition of a sequence of elements $e \in R$. Given two TAAs $\mathbf{A} : K_1 \times K_2 \to R$ and $\mathbf{B} : K_2 \times K_3 \to R$, the array multiplication operation $\mathbf{C} = (\mathbf{A} \otimes \mathbf{B}) : K_1 \times K_3 \to R$ is defined as,

$$\mathbf{C}(k_1, k_3) = (\mathbf{A} \otimes \mathbf{B})(k_1, k_3) = \bigoplus_{k_2 \in K_2} \mathbf{A}(k_1, k_2) \odot \mathbf{B}(k_2, k_3).$$

*Definition 2.7 (Array Identity).* Given two key sets $K_1$ and $K_2$, and a partial function $f : K_1 \hookrightarrow K_2$, the array identity $\mathbb{E}_{K_1,K_2,f} : K_1 \times K_2 \to R$ is defined as a TAA such that

$$\mathbb{E}_{K_1,K_2,f}(k_1, k_2) = \begin{cases} 1, & \text{if } k_1 \in \text{dom } f \text{ and } k_2 = f(k_1); \\ 0, & \text{otherwise.} \end{cases}$$

Specifically, if $\text{dom } f = K_1 \cap K_2$ and $f(k_1) = k_1$ for $\forall k_1 \in K_1$, $\mathbb{E}_{K_1,K_2,f}$ is abbreviated to $\mathbb{E}_{K_1,K_2}$.

In general, $\mathbb{E}_{K_1,K_2,f}(k_1, k_2)$ is not an identity for general array multiplication. However, $\mathbb{E}_{K,K}$ is an identity element for array multiplication on associative arrays $K \times K \to R$.

*Definition 2.8 (Kronecker Product).* Given two TAAs $\mathbf{A} : K_1 \times K_2 \to R$ and $\mathbf{B} : K_3 \times K_4 \to R$, their Kronecker product $\mathbf{C} = \mathbf{A} \otimes \mathbf{B} : (K_1 \times K_3) \times (K_2 \times K_4)$ is defined by

$$\mathbf{C}((k_1, k_3), (k_2, k_4)) = \mathbf{A}(k_1, k_2) \odot \mathbf{B}(k_3, k_4).$$

*Definition 2.9 (Transpose).* Given a TAA $\mathbf{A} : K_1 \times K_2 \to R$, its transpose, denoted by $\mathbf{A}^\mathsf{T}$, is defined by $\mathbf{A}^\mathsf{T} : K_2 \times K_1 \to R$ where $\mathbf{A}^\mathsf{T}(k_2, k_1) = \mathbf{A}(k_1, k_2)$ for $k_1 \in K_1$ and $k_2 \in K_2$.

## 2.2 Text Operations

We can express a number of fundamental text operations using the proposed TAA algebra. We first define three basic TAAs specifically for text analytics, then a series of text operations will be defined on general TAA or these basic structures.

*Definition 2.10 (Document-Term Matrix).* Given a text corpus, a document term matrix is defined as a TAA $\mathbf{M} : D \times T \to R$ where $D$ and $T$ are the document set and term set of a text corpus.

The term set in the document-term matrix can be the vocabulary or the bigram of the corpus, or an application-specific user-defined set of interesting terms. The matrix value $\mathbf{M}(d, t)$ can also take different semantics, in one application it can be the occurrence of $t$ in document $d$, while in another application, it can be the term frequency-inverse document frequency (tf-idf). Typically, elements of $D$ and $T$ will have additional relational metadata. A document may have a date and a term may have an annotation like a part-of-speech (POS) tag.

*Definition 2.11 (Term-Index Matrix).* Given a document $d$, the term index matrix is defined as a TAA, $\mathbf{N} : T_d \times I \to \{0, 1\}$ where $T_d = \{d\} \times T$ is the set of terms in document $d$ and $I = \{1, \cdots, I_d\}$ is the index set ($I_d$ is the size of $d$). Specifically, for $(d, t) \in T_d$ and $i \in I$,

$$\mathbf{N}((d, t), i) = \begin{cases} 1, & \text{if } i\text{-th word of document } d \text{ is } t; \\ 0, & \text{otherwise.} \end{cases}$$

**Example 2.** For a document $d$ = "Today is a sunny day", let its term index matrix be $\mathbf{N} : (\{d\} \times T) \times I \to \{0, 1\}$, then we have $T$ = {"today", is", "a", "sunny", "day"}, $I = \{1, 2, 3, 4, 5\}$. $\mathbf{N}$("today", 1) = 1, $\mathbf{N}$("is", 2) = 1, $\mathbf{N}$("a", 3) = 1, $\mathbf{N}$("sunny", 4) = 1, $\mathbf{N}$("day", 5) = 1, and for other $(t, i)$ pairs where $(t, i) \in T \times I$, we have $\mathbf{N}(t, i) = 0$.

*Definition 2.12 (Term Vector).* There are two types of term vectors. 1) Given a set of terms $T$ of a document $d$, the term vector is defined as a TAA $\mathbf{V} : \{d\} \times T \to R$. 2) Given a set of terms $T$ for a collection of documents $D$, $\mathbf{V} : \{1\} \times T \to R$ is a term vector for the corpus $D$.

The term vector represents some attribute of terms in the scope of one document or a corpus. For example, for a document $d$, the value of the term vector $\mathbf{V} : \{d\} \times T$ can be the occurrence of each term in this document. For a corpus $D$, the value of its term vector $\mathbf{V} : \{1\} \times T$ can be idf value for each term in the whole corpus, and the value is not specific to a single document.

Based on these structures, we can define our unit text operators as follows. Some operators are defined for general TAAs, while some are defined for a specific type of TAAs.

*Definition 2.13 (Extraction).* Given a TAA $\mathbf{A} : K_1 \times K_2 \to R$ and two projection sets $K_1' \subseteq K_1$, $K_2' \subseteq K_2$, we define the extraction operation as

$$\Pi_{K_1',K_2'}(\mathbf{A}) = \mathbb{E}_{K_1',K_1} \otimes \mathbf{A} \otimes \mathbb{E}_{K_2,K_2'}.$$

Let $\mathbf{B} = \Pi_{K_1',K_2'}(\mathbf{A})$, we have $B(k_1, k_2) = A(k_1, k_2)$, for $\forall (k_1, k_2) \in K_1' \times K_2'$.

When only extracting row keys, the operation can be expressed as $\Pi_{K_1',:}$ and when extracting column keys, it is expressed as $\Pi_{:,K_2'}$.

*Definition 2.14 (Rename).* Given a TAA $\mathbf{A} : K_1 \times K_2 \to R$, suppose $K_2'$ is another column key set and there exists a bijection $f : K_2 \to K_2'$. The column rename operation is defined as

$$\rho_{K_1,K_2 \to K_2',f}(\mathbf{A}) = \mathbf{A} \otimes \mathbb{E}_{K_2,K_2',f}.$$

Similarly, given another row key set $K_1'$ and a bijection $f : K_1 \to K_1'$, the row rename operation is defined as

$$\rho_{K_1 \to K_1', K_2, f}(\mathbf{A}) = \mathbb{E}_{K_1', K_1, f^{-1}} \otimes \mathbf{A}.$$

The subscript $f$ can be omitted if the bijection is clear, e.g., $|\text{dom } f| = 1$. In addition, the row rename operation and column rename operation can be combined together as $\rho_{K_1 \to K_1', K_2 \to K_2'}(\mathbf{A})$. Our rename operator is more general than the rename operation of relational algebra since it supports both row key set and column key set renaming.

*Definition 2.15 (Apply).* Given a TAA $\mathbf{A} : K_1 \times K_2 \to R$ and a function $f : R \to R$, define the apply operator by $\mathbf{Apply}_f(\mathbf{A}) : K_1 \times K_2 \to R$ where,

$$\mathbf{Apply}_f(\mathbf{A})(k_1, k_2) = f(\mathbf{A}(k_1, k_2)), \forall (k_1, k_2) \in K_1 \times K_2.$$

*Definition 2.16 (Filter).* Given a TAA $\mathbf{A} : K_1 \times K_2 \to R$ and an indicator function $f : R \to \{0, 1\}$, define the filter operation on $\mathbf{A}$ as

$$\mathbf{B} = \mathbf{Filter}_f(\mathbf{A}) = \sigma_f(\mathbf{A}) : K_{1f}, K_{2f} \to R,$$

where $K_{1f} \times K_{2f} = \{(k_1, k_2) | (k_1, k_2) \in K_1 \times K_2 \text{ and } f(\mathbf{A}(k_1, k_2)) = 1\}$, and $\mathbf{B}(k_1, k_2) = \mathbf{A}(k_1, k_2)$.

*Definition 2.17 (Rank).* Given a TAA $\mathbf{A} : K_1 \times K_2 \to R$, for any $k \in K_1$, we extract a TAA $\mathbf{V} = \Pi_{\{k\},:}(\mathbf{A}) : \{k\} \times K_2$. Since $R$ is a partially-ordered semiring (Definition 2.2), the value set $\{\mathbf{V}(k, x) | \forall x \in K_2\} \subseteq R$ inherits the partial order from $R$, which implies an order $\mathbf{V}(k, x_1) \le \mathbf{V}(k, x_2) \le \cdots \le \mathbf{V}(k, x_{|K_2|})$. Define $\mathbf{Idx}(k, x_i) = i$, then the rank by column operation is defined as

$$\mathbf{Rank}_2(\mathbf{A}) : K_1 \times K_2 \to \{1, \cdots, |K_2|\},$$

where $\mathbf{Rank}_2(\mathbf{A})(k, x) = \mathbf{Idx}(k, x)$. Similarly, we have rank by row operation defined as

$$\mathbf{Rank}_1(\mathbf{A}) : K_1 \times K_2 \to \{1, \cdots, |K_1|\}.$$

When the column key dimension or row key dimension is 1 (e.g., for a term vector), $\mathbf{Rank}_1$ or $\mathbf{Rank}_2$ is abbreviated to $\mathbf{Rank}$.

*Definition 2.18 (Merge).* Given two TAAs $\mathbf{A} : K_{A1} \times K_{A2}$ and $\mathbf{B} : K_{B1} \times K_{B2}$, if $(K_{A1} \times K_{A2}) \cap (K_{B1} \times K_{B2}) = \emptyset$, then merge operation can be applied on them, and it is defined as,

$$\mathbf{C} = \mathbf{Merge}(\mathbf{A}, \mathbf{B}) : K_1 \times K_2 \to R$$

where $K_1 = K_{A1} \cup K_{B1}$ and $K_2 = K_{A2} \cup K_{B2}$, and

$$\mathbf{C}(k_1, k_2) = \begin{cases} \mathbf{A}(k_1, k_2), & \text{if } (k_1, k_2) \in K_{A1} \times K_{A2}; \\ \mathbf{B}(k_1, k_2), & \text{if } (k_1, k_2) \in K_{B1} \times K_{B2}; \\ 0, & \text{otherwise.} \end{cases}$$

*Definition 2.19 (Expand).* Given an element-wise binary operator $\mathbf{OP}$ on associative arrays, e.g., $\oplus$ and $\odot$, a term vector $\mathbf{V} : \{1\} \times T \to R$ and a document-term matrix $\mathbf{M} : D \times T \to R$, the expand operator $\mathbf{Expand}_{\mathbf{OP}}(\mathbf{V}, \mathbf{M})$ implicitly expands the term vector $\mathbf{V}$ to generate another associative array $\mathbf{M'} : D \times T \to R$ where $\mathbf{M'}(d, t) = \mathbf{V}(1, t), \forall d \in D$ and $\forall t \in T$, and then applies $\mathbf{OP}$ on $\mathbf{M'}$ and $\mathbf{M}$.

Suppose that for a corpus $D$, there is a term vector $\mathbf{V} : \{1\} \times T \to R$ where $\mathbf{V}(1, t)$ is the mean occurrence of term $t$ in $D$ (i.e., $\frac{Count_t}{|D|}$ where $Count_t$ is the total occurrence of $t$ in $D$), and there is a document-term matrix $\mathbf{M} : D \times T$, then

$$\mathbf{Expand}_{\oplus}(\mathbf{Apply}_{f(x)=-x}(\mathbf{V}), \mathbf{M})$$

will generate the difference of terms occurrences for each document from their average occurrences.

*Definition 2.20 (Flatten).* Given an associative array $\mathbf{A} : K_1 \times K_2 \to R$, the flatten operation is defined by $\mathbf{Flatten}(\mathbf{A}) : \{1\} \times (K_1 \times K_2) \to R$ where

$$\mathbf{Flatten}(\mathbf{A})(1, (k_1, k_2)) = \mathbf{A}(k_1, k_2) \text{ for } \forall (k_1, k_2) \in K_1 \times K_2.$$

*Definition 2.21 (Left Shift).* Given a term-index matrix $\mathbf{N} : (\{d\} \times T) \times I \to \{0, 1\}$, and an integer $n \ge 1$, define the left shift operator by $\mathbf{LShift}_n(\mathbf{N}) : (\{d\} \times T) \times I \to \{0, 1\}$ where

$$\mathbf{LShift}_n(\mathbf{N})((d, t), i) = \begin{cases} \mathbf{N}((d, t), i + n), & \text{if } i + n \le |I|; \\ 0, & \text{if } i + n > |I|; . \end{cases}$$

For a term-index matrix $\mathbf{N}$ of document $d$, $\mathbf{LShift}_1(\mathbf{N})$ generates another term-index matrix $\mathbf{N'}$ where $\mathbf{N'}((d, t), i) = 1$ when $t$ is the $(i + 1)$-th word in $d$.

*Definition 2.22 (Intersection).* Suppose there are two term-index matrices with the same index set $I$, $\mathbf{N}_1 : (\{d\} \times T) \times I \to \{0, 1\}$ and $\mathbf{N}_2 : (\{d\} \times T) \times I \to \{0, 1\}$, let the intersection be $\mathbf{N} = \mathbf{Intersection}(\mathbf{N}_1, \mathbf{N}_2) : (\{d\} \times (T \times T)) \times I \to \{0, 1\}$ where

$$\mathbf{N}((d, (t_1, t_2)), i) = \begin{cases} 1, & \text{if } \mathbf{N}_1((d, t_1), i) = 1 \text{ and } \mathbf{N}_2((d, t_2), i) = 1; \\ 0, & \text{otherwise.} \end{cases}$$

The left shift and intersection operations can be composed to compute all bigrams of a document. Given a term-index matrix $\mathbf{N}$ of document $d$, let $\mathbf{N'} = \mathbf{Intersection}(\mathbf{N}, \mathbf{LShift}_1(\mathbf{N}))$, then $\mathbf{N'}((d, (t_1, t_2)), i) = 1$ when $(t_1, t_2)$ is the $i$-th bigram in document $d$.

*Definition 2.23 (Sum).* Given a TAA $\mathbf{A} : K_1 \times K_2 \to R$, the sum by row operation $\mathbf{Sum}_1$, sum by column operation $\mathbf{Sum}_2$ are defined as the following,

$$\mathbf{B} : K_1 \times \{1\} = \mathbf{Sum}_1(\mathbf{A}) \text{ where } B(k_1, 1) = \bigoplus_{k_2 \in K_2} \mathbf{A}(k_1, k_2);$$

$$\mathbf{B} : \{1\} \times K_2 = \mathbf{Sum}_2(\mathbf{A}) \text{ where } B(1, k_2) = \bigoplus_{k_1 \in K_1} \mathbf{A}(k_1, k_2).$$

# 3 TEXT ANALYTIC TASKS

## 3.1 Constructing a Document Term Matrix

As we state in Section 2.2, a document term matrix is a common representation model for a collection of documents where the terms can be a list of import terms or the whole vocabulary or bigrams. The entry of the matrix can be either the occurrence of each term or the tf-idf value.

**Example 3.** For a document set $D$, build a document term matrix where terms are all unigrams and bigrams in $D$, and the values should be the occurrence of each term in the whole corpus.

Suppose there is a tokenization function called **Tokenize** that takes a document $d$ as input and generates a term index matrix $\mathbf{N} : (\{d\} \times T) \times I$. The construction can be decomposed to two parts,

the first part shown in Fig. 1 is to construct a Term Vector for one single document $d$ containing all unigrams and bigrams together with their corresponding occurrences.

$$\mathbf{N} = \mathbf{Tokenize}(d) \quad : (\{d\} \times T) \times I \tag{1}$$

$$\mathbf{V_1} = \rho_{\{1\}\to\{d\},\{d\}\times T\to T}(\mathbf{Sum_1(N)})^{\mathsf{T}} \quad : \{d\} \times T \tag{2}$$

$$\mathbf{T} = \mathbf{N} \otimes \mathbf{LShift_1(N)}^{\mathsf{T}} \quad : (\{d\} \times T) \times (\{d\} \times T) \tag{3}$$

$$\mathbf{V_2} = \mathbf{Flatten}(T) \quad : \{1\} \times (\{d\} \times T) \times (\{d\} \times T)) \tag{4}$$

$$\mathbf{V_2} = \rho_{\{1\}\to\{d\},(\{d\}\times T)\times(\{d\}\times T)\to(T\times T)}(\mathbf{V_2}) \quad : \{d\} \times (T \times T) \tag{5}$$

$$\mathbf{V_2} = \sigma_{f:x\to\mathbb{1}(x>0)}(\mathbf{V_2}) \quad : \{d\} \times (T \times T) \tag{6}$$

$$\mathbf{V}_d = \mathbf{Merge(V_1, V_2)} \quad : \{d\} \times (T \cup (T \times T)) \tag{7}$$

**Figure 1: Algebraic representation for task in Example 3.**

Step 1 generates the term index matrix where each term is the unigram. The $\mathbf{Sum_1}$ operation in Step 2 generates the term vector where $\mathbf{V_1}(d, t)$ is the occurrence of unigram $t$ in document $d$. Steps 3–6 get the term vector $\mathbf{V_2}$ for all bigrams. Step 7 concatenates two term vectors to get the representation for $d$.

For each document $d_i$ in document set $D = \{d_1, \cdots, d_n\}$, we get its term vector $\mathbf{V_{di}} : \{d_i\} \times (T_i \cup (T_i \times T_i)) \to R$ using the above steps, then apply the **Merge** operation to get the document-term matrix $\mathbf{M} : D \times T \to R$ where $T = (T_1 \cup \cdots \cup T_n) \cup ((T_1 \times T_1) \cup \cdots \cup (T_n \times T_n))$ is the union of all unigrams and bigrams in the whole corpus, $\mathbf{Merge(V_{d1}, Merge(V_{d2}, \cdots, Merge(V_{d(n-1)}, V_{d(n)})))}$.

Besides word-occurrence as the values of term document matrix, one can also use a term's tf-idf value. If all terms are considered, term document matrix $\mathbf{M}$ would be of high dimension and sparse, which would be costly to manipulate. A simple and commonly adopted method to reduce dimension is to select out informative words. The following presents the queries to get document-term matrix $\mathbf{M}$ with the tf-idf values for only informative terms where the informativeness is measured by idf value.

**Example 4.** Given a collection of documents $D$, we have to generate a document-term matrix $\mathbf{M}$ for the top 1000 "informative words" where $\mathbf{M}(d, t)$ is the tf-idf value for term $t$ in document $d$. Suppose there is a term-document matrix $\mathbf{M_1}$ which stores the occurrence for all unigrams in each document (the construction is similar to that of example 2 and thus is skipped), $\mathbf{M}$ can be generated by the following steps. The function $idf$ in Step 3 is to calculate idf value, which is defined as $idf(x) = -\log \frac{x}{|D|}$ where $x$ is the number of documents that contains a specific term.

$$\mathbf{M_2} = \mathbf{Apply}_{f:x\to\mathbb{1}(x>0)}(\mathbf{M_1}) \quad : D \times T \tag{1}$$

$$\mathbf{V} = \mathbf{Sum_2}(\mathbf{M_2}) \quad : \{1\} \times T \tag{2}$$

$$\mathbf{I} = \sigma_{f:x\to\mathbb{1}(x\le 1000)}(\mathbf{Rank(V)}) \quad : \{1\} \times T' \tag{3}$$

$$\mathbf{V_1} = \mathbf{Apply}_{idf}(\Pi_{:,\mathbf{I}.K_2}(\mathbf{V})) \quad : \{1\} \times T' \tag{4}$$

$$\mathbf{M_3} = \Pi_{:,\mathbf{I}.K_2}(\mathbf{M_1}) \quad : D \times T' \tag{5}$$

$$\mathbf{M} = \mathbf{Expand}_{\odot}(\mathbf{V_1}, \mathbf{M_3}) \quad : D \times T' \tag{6}$$

**Figure 2: Algebraic representation for task in Example 4.**

## 3.2 Using TAAs

For Example 1 introduced in Section 1, we express this analysis using relational algebra and text associative array operations. Suppose that the maximum number of words for a term in $L_o \cup L_p$ is 3, now this analysis can be expressed as the following. Step 1 is expressed in relational algebra. **TopicModel** in the last step is a function which takes a document-term matrix and produce document topic matrix and topic term matrix, which are the standard outputs of topic modeling, represented by another two TAAs **DTM** and **TTM**. Let $\mathbf{T} = \rho_{f:x\to\mathbb{1}(x\ge|D|-k)}(\mathbf{Rank_2(DTM)})$, then $\mathbf{T}.K$ will return all $(d, t)$ pairs where $t$ is one of the top-$k$ topics for $d$.

$$D = \pi_{content}(\sigma_{d_1\le data\le d_2}(R)) \tag{1}$$

$$\mathbf{M} : \{\} \times \{\} \to R, \quad \mathbf{FV} : \{\} \times \{\} \to R \tag{2}$$

$$\text{for } d \in D : \tag{3}$$

$$\mathbf{N_1} = \mathbf{Tokenize}(d) \tag{3.1}$$

$$\mathbf{V} = \rho_{\{1\}\to\{d\},\{d\}\times T\to T}(\mathbf{Sum_1(N_1)})^{\mathsf{T}} \tag{3.2}$$

$$\mathbf{N_2} = \mathbf{Intersection(N_1, LShift_1(N_1))} \tag{3.3}$$

$$\mathbf{N_3} = \mathbf{Intersection(N_2, LShift_2(N_1))} \tag{3.4}$$

$$\mathbf{N} = \mathbf{Merge(N_1, Merge(N_2, N_3))} \tag{3.5}$$

$$\mathbf{V_f} = \rho_{\{1\}\to\{d\},\{d\}\times T'\to T'}(\mathbf{Sum_1(N)}^{\mathsf{T}}) \tag{3.6}$$

$$\mathbf{FV} = \mathbf{Merge(FV, V_f)} \tag{3.7}$$

$$\mathbf{M} = \mathbf{Merge(M, V)} \tag{3.8}$$

$$\mathbf{FV_o} = \mathbf{\Pi}_{:,L_o}(\mathbf{FV}) \tag{4}$$

$$\mathbf{FV_p} = \mathbf{\Pi}_{:,L_p}(\mathbf{FV}) \tag{5}$$

$$\mathbf{I_o} = \sigma_{f:x\to\mathbb{1}(x>c_1)}(\mathbf{Sum_1}(FV_o)) \tag{6}$$

$$\mathbf{I_p} = \sigma_{f:x\to\mathbb{1}(x>c_2)}(\mathbf{Sum_1}(FV_p)) \tag{7}$$

$$\mathbf{M} = \Pi_{I_o.K_1\cap I_p.K_1,:}(\mathbf{M}) \tag{8}$$

$$\mathbf{I_t} = \sigma_{f:x\to\mathbb{1}(x<\theta_2)}(\mathbf{Sum_2(M)}) \tag{9}$$

$$\mathbf{I_d} = \sigma_{f:x\to\mathbb{1}(x<\theta_1)}(\mathbf{Sum_1(M)}) \tag{10}$$

$$\mathbf{M} = \Pi_{I_d.K_1,I_t.K_2}(\mathbf{M}) \tag{11}$$

$$\mathbf{DTM, TTM} = \mathbf{TopicModel(M)} \tag{12}$$

**Figure 3: Algebraic representation for the task in Example 1.**

# 4 FUTURE WORK

This work has exploited the algebra for text analytics which would serve a solid theoretical foundation for our future work. In the near future, we will provide a formal language for both relational and textual data analysis and provide operators rewritting rules for optimization. These text operators introduced here will be implemented and experiments will be conducted to test the validity of this approach.

**Acknowledgment.** This work was partially funded by the National Science Foundation grant 1909875.

# REFERENCES

[1] Pablo Barceló, Nelson Higuera, Jorge Pérez, and Bernardo Subercaseaux. 2019. On the Expressiveness of LARA: A Unified Language for Linear and Relational Algebra. *arXiv preprint arXiv:1909.11693* (2019).

[2] Jonathan S Golan. 2013. *Semirings and affine equations over them: theory and applications*. Vol. 556. Springer Science & Business Media.

[3] Hayden Jananthan, Ziqi Zhou, Vijay Gadepally, Dylan Hutchison, Suna Kim, and Jeremy Kepner. 2017. Polystore mathematics of relational algebra. In *Int. Conf. on Big Data*. IEEE, 3180–3189.

[4] Jeremy Kepner, Vijay Gadepally, Hayden Jananthan, Lauren Milechin, and Siddharth Samsi. 2020. AI Data Wrangling with Associative Arrays. *arXiv preprint arXiv:2001.06731* (2020).

[5] Éric Leclercq, Annabelle Gillet, Thierry Grison, and Marinette Savonnet. 2019. Polystore and Tensor Data Model for Logical Data Independence and Impedance Mismatch in Big Data Analytics. In *Trans. on Large-Scale Data-and Knowledge-Centered Systems XLII*. Springer, 51–90.