A Foundation for Spatio-Textual-Temporal Cube Analytics

Mohsin Iqbal
Aalborg University
mohsin@cs.aau.dk

Matteo Lissandrini
Aalborg University
matteo@cs.aau.dk

Torben Bach Pedersen
Aalborg University
tbp@cs.aau.dk

ABSTRACT

Large amounts of *spatial, textual, and temporal (STT) data* are being produced daily. This is data containing an *unstructured* component (text), a *spatial* component (geographic position), and a *time* component (timestamp). Therefore, there is a need for a powerful and *general* way of analyzing *STT data together*. In this paper, we define and formalize the *Spatio-Textual-Temporal Cube (STTCube)* structure to enable *combined* effective and efficient analytical queries over *STT data*. Our novel data model over *STT objects* enables novel *joint and integrated* STT insights that are hard to obtain using existing methods. Moreover, we introduce the new concept of *STT measures* with associated novel STT-OLAP operators. To allow for efficient large-scale analytics, we present a pre-aggregation framework for exact and approximate computation of *STT measures*. Our comprehensive experimental evaluation on a real-world Twitter dataset confirms that our proposed methods reduce query response time by 1-5 orders of magnitude compared to the *No Materialization* baseline and decrease storage cost between 97% and 99.9% compared to the *Full Materialization* baseline while adding only a negligible overhead in the STTCube construction time. Moreover, *approximate computation* achieves an accuracy between 90% and 100% while reducing query response time by 3-5 orders of magnitude compared to *No Materialization*.

1 INTRODUCTION

Due to the increased usage of mobile devices and advancements in accurate geo-tagging, more and more geo-tagged data is being produced [8]. In particular, social media platforms like Twitter and Facebook are some of the main sources of geo-tagged data, usually in the form of posts, comments, and reviews (e.g., Figure 1). This type of data contains spatial, textual, and temporal (STT) information. As a result, *STT data* analysis is becoming increasingly important [9] since it allows to extract new insights regarding customer satisfaction, user-generated content shared online, and brand reputation [27].

STT data contains information regarding topics discussed w.r.t. time and location, hence presenting an invaluable link between user opinions and the real world. For example, STT data can help us analyze an advertisement campaign to identify the best locations for ad placements. Traditionally, this information is accessed through spatial keyword-queries [4], e.g., to retrieve topics within a certain location, or identify in which locations some topic is discussed. However, keyword or topic search are *point-wise* search tasks. Instead, there a significant need to *provide more extensive analytics analogous to traditional OLAP-style analytics*. An example STT query is *"find the top-k trending hashtags aggregated by topic within a user-defined region (i.e., polygon) around Paris this month"*.

The traditional data cube model is one of the most widely used tools to analyze structured data. Since their introduction, data cubes have been extended to analyze different types of data,

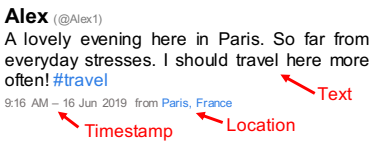


Figure 1: Geo-tagged tweet: An example of a STT object

like sales [14], locations [15], time-series [6], and text [22], but *separately*. In particular, some works propose OLAP operators to analyze either textual data [3, 37] or spatial data [14, 15]. However, no previous work proposed a unified model and set of operators enabling *integrated* and *joint* analysis of *STT data*. Moreover, as we propose to *jointly* analyze STT dimensions together with other dimensions, we are also able to define novel families of measures that have not been studied before, namely *STT measures*. These measures, as we show later, allow to produce more advanced analytics instead of, e.g., simple keyword frequency.

Contributions. In this paper, we introduce the Spatio-Textual-Temporal Cube (STTCube) to analyze *STT data*. Adding spatial, textual, and temporal support to a traditional data cube is not straight-forward due to the presence of n–n relationships in textual hierarchies and because existing families of measures cannot support *joint* and *integrated* analysis involving spatial, textual, and temporal dimensions, e.g., finding the trending keywords grouped by regions, defined by geometry shapes, over a time interval (Section 3.3). Hence, we introduce new families of measures and OLAP operators that extract *combined insights* from STT dimensions and measures. STTCube provides specialized *spatio-textual* and *spatio-textual-temporal measures* such as *Top-k Dense Keywords within an area* and *Top-k Volatile Keywords within an area* that deliver the integrated aggregates over *STT data*. Moreover, a set of analytical operators, namely STT slice, dice, roll-up, and drill-down are proposed. This results in a data model able to support *spatio-textual-temporal OLAP* (STTOLAP) operators. Furthermore, we propose *Partial Exact Materialization (PEM)* and *Partial Approximate Materialization (PAM)* methods for efficient exact and approximate computations of *STT measures*, respectively. Among other things, we also provide a systematic set of solutions to handle n–n relationships in textual hierarchies.

In this work, we present the following contributions: I) We extend the standard cube model to add support for *spatial, textual, and temporal* dimensions and hierarchies and *spatio-textual* and *spatio-textual-temporal* measures (Sections 3.1 to 3.3). II) We propose a set of analytical operators (STTOLAP) over *spatio-textual-temporal data* (Section 3.4). III) We introduce *keyword density* and *keywords volatility* as prototypical *spatio-textual* and *spatio-textual-temporal measures* (Section 3.3). IV) We propose a pre-aggregation framework (STTCube materialization) for efficient, exact (*PEM*) and approximate (*PAM*), computation of the

Table 1: Spatio-Textual-Temporal Sample Dataset

	Time	Location	Terms
1	11:12:13 20-10-2019	57.016254, 09.991203	Apple, fruit, #love
2	11:18:23 24-10-2019	56.187421, 10.171410	Potato, #NewYear
3	11:35:56 20-10-2019	56.151078, 10.204762	Banana, Season
4	16:12:14 24-10-2019	57.016254, 09.991203	Potato, Salad, #Fresh
...	...	...	...

Table 2: Presence (✔) or absence (✘) of support for spatial and textual data, dimensions, hierarchies, and measures in existing methods

	Textual Support				Spatial Support				ST	STT
Method	Data	Dimension	Hierarchies	Measures	Data	Dimension	Hierarchies	Measures	Measures	Measures
EXODuS [7]	✔(JSON)	✘	✘	✘	✘	✘	✘	✘	✘	✘
TextCube [22]	✔	✔	✔	✔	✘	✘	✘	✘	✘	✘
Text OLAP [36]	✔	✔	✔	✘	✘	✘	✘	✘	✘	✘
TextCubeTopKCells [10]	✔	✔	✔	✔	✘	✘	✘	✘	✘	✘
Geo Miner [15]	✘	✘	✘	✘	✔	✔	✔	✔	✘	✘
SpatialCube [14]	✘	✘	✘	✘	✔	✔	✔	✔	✘	✘
StreamCube [12]	✔	✘	✘	✔	✔	✔	✔	✘	✘	✘
TwitterSand [32]	✔	✔	✔	✘	✔	✘	✘	✔	✘	✘
TextStreams [34]	✔	✘	✘	✔	✔	✘	✘	✘	✘	✘
TopicExploration [39]	✔	✘	✘	✔	✔	✔	✔	✘	✘	✘
SocialCube [24]	✔	✘	✘	✔	✔	✔	✔	✘	✘	✘
TopicCube [38]	✔	✔	✔	✔	✔	✔	✔	✘	✘	✘
ContextualizedWarehouse [29]	✔	✘	✘	✔	✔	✔	✔	✘	✘	✘
STTCube	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔

proposed *STT measures* (Section 4). V) We propose techniques for processing *spatio-textual-temporal objects* and the construction of the *STTCube* (Section 5). VI) We evaluate the pre-aggregation framework's (*PEM* and *PAM*) query response time, storage cost, and accuracy by comparing it with the *No STT Cube*, *Full Materialization*, and *No Materialization* baselines. Our pre-aggregation framework provides 1-5 orders of magnitude improvement in query response time and a 97% to 99.9% reduction in storage cost with an accuracy between 90% and 100% (Section 6).

2 RELATED WORK

OLAP and the *Data Cube* [18] are used heavily in business intelligence to obtain insights over the historical, current, and future state of business. With the emergence of web and social media, an immense amount of unstructured data is being produced, which must be included in the analytical process. Table 2 summarizes the state of the art on spatial, textual, and temporal analytics by listing the properties and gaps in the current methods.

The *Text-Cube* [22, 30] allows OLAP-like queries on text data by providing dimensions and hierarchies for terms. Moreover, it supports the computation of two information retrieval (IR) measures: *inverted index* and *term frequency*. *EXODuS* [7] processes semi-structured document stores (i.e., JSON) using a schema-on-read approach to allow exploratory OLAP on text. Text OLAP [36] extends traditional OLAP to support textual dimensions and keyword-based top-k search [10]. *Yet, all these approaches lack support for spatial and temporal data and the advanced measures and operators required for spatio-textual-temporal analytics.*

For spatial data, GeoMiner [15] proposes a cube structure for mining characteristics, comparisons, and association rules from geo-spatial data. The coupling of GIS and OLAP is known as Spatial OLAP (SOLAP) [31] and Spatial cube [14] allows to perform SOLAP on the semantic web. *Yet, these solutions focus on spatial data only and lack support for textual and temporal data.*

There are solutions that combine more than one component of data, e.g., spatio-temporal [35], into the same model but do not provide combined *STT* analytics. Among those, the contextualized warehouse [29] combines traditional OLAP with a textual warehouse. This allows the user to provide some keywords, select a market (country or region), retrieve documents matching the keywords as context, and then analyze the facts related to those keywords and documents. Similarly, *Topic Cube* [38] extends the functionality of a traditional cube and combines probabilistic topic modeling with OLAP by introducing the *topic hierarchy*. TwitterSand [32] and StreamCube [12] exploit textual and spatial information to gain insights by clustering twitter hashtags and tweets in a region, respectively. *STT data* is also analyzed to extract events and topics information in TextStreams [34] and TopicExploration [39]. Finally, SocialCube [24] tries to capture human, social, and cultural behavior by performing linguistic analysis (sentiment analysis) over tweets. All these approaches focus on the unstructured nature of text along with spatial and temporal data but they *do not provide Integrated STT analytics*, for example, they do not provide the ability to *compute aggregate spatial, textual, temporal, and spatio-textual-temporal measures over spatial, textual, and temporal dimensions and hierarchies.*

Spatial top-k keyword-queries [5, 9, 25] *answer only point-wise* queries and do not support aggregation functions or hierarchies. Thus, they do not support more complex OLAP-style analytical tasks, which we do. There are methods that solve a very specific task for a specific type of data [2, 21, 28]. These methods are fundamentally different from STTCube because *STTCube provides a generic framework for a wide range of STT analytics over different kinds of STT data sources*, including, but not limited to, geo-tagged tweets. Also, STTCube can take advantage of the improvements suggested over other cubes, e.g., Nanocubes [23] and DICE [17], making it a powerful tool for OLAP-style *STT analytics*.

Our summary of related work in Table 2 shows that no existing method provides integrated support for *STT data*, unlike STTCube. To the best of our knowledge, a proper formalization of a data cube model for *STT data* able to support complex analytics for *STT objects* at scale is missing. In particular, no previous method studies dimensions, hierarchies, and measures that allow processing STT data *jointly*. Furthermore, the main novel challenge for *STT-OLAP* is handling n–n relationships inside the *STT* dimensions effectively since n–n relationships do not allow traditional pre-aggregation techniques to be used. Moreover, arbitrary temporal ranges with multiple levels of granularity adds complexity to *STT* measures computations. As a remedy, we propose *STTCube* which enables the *joint* and *integrated* analysis of *STT objects* by introducing new sets of STT measures to gain in-depth insights using STTOLAP operators.

3 SPATIO-TEXTUAL-TEMPORAL CUBES

Here, we define the *STTCube*, an extension of the traditional data cube to allow storage and analysis of *STT objects*. Data cubes are used to model and analyze multi-dimensional data. The basic building block of a data cube is the cell that contains *fact*. Each fact is the observational object for analysis, with one or more numerical *measures* associated with it.

Definition 1 (Data Cube). *An n-dimensional data cube $\mathcal{CS}_{dc}$ is a tuple $\mathcal{CS}_{dc}=(D, M, F)$, with a set of dimensions $D=\{d_1, d_2, \cdots, d_n\}$, a set of measures $M=\{m_1, m_2, \cdots, m_k\}$, and a set of facts F. A dimension $d_i \in D$ has a set of hierarchies H_{d_i}. Each hierarchy $h \in H_{d_i}$ has a set of hierarchy steps (discussed in Section 3.1) and is organized into a set of levels L_h. Each level $l \in L_h$ contains a set of members and has a set of attributes A_l. Each attribute $a \in A_l$ is defined over a domain. Each measure $m \in M$ is a function defined over a domain which can return either a single value or a complex object. The domain of a dimension d_i is denoted by $\delta(d_i)$*

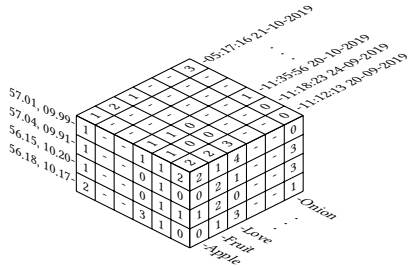


Figure 2: STTCube Example

Spatio-Textual-Temporal (STT) Objects An *STT object* records place (geo-coordinates or location where it was created), text (a review, or a user comment), and time (when it was created). Social networks with geo-tagged micro-blog posts are typical *STT data* sources (e.g., the geo-tagged tweet in Figure 1).

Definition 2 (STT object). *A spatio-textual-temporal object is a tuple* $obj_{st}=\langle \lambda, \varphi, \tau \rangle$ *where* λ, φ, *and* τ *represent the location, text, and time components, respectively.*

The *Location* is represented as the latitude and longitude pair $\lambda \in (\mathbb{R} \times \mathbb{R})$. The *Text* is an ordered list $\varphi=\langle w_1, w_2, \ldots, w_n \rangle$ where $w_i \in \mathcal{W}$ is a string and is called a *Term*. Among all Terms, keywords are a user-defined subset of important Terms $W_k \subseteq \mathcal{W}$. For instance, the user can decide that hashtags (terms starting with #) have special meaning and are a special type of keyword. *Time* specifies a precise instant (a timestamp) to some resolution (e.g., seconds). Table 1 contains examples of *STT objects* with their location, a set of keywords extracted from the text, and timestamp.

3.1 The STTCube Schema

For analytical processing of *STT objects* we propose to model them as an *STTCube*. An STTCube $\mathcal{CS}_{stt}=(D, M, F)$ is a data cube (Definition 1) with three special dimensions, namely *Location*, *Text*, and *Time*, along with zero or more traditional dimensions that is $D = \{d_{Location}, d_{Text}, d_{Time}, d_4, \ldots, d_n\}$.

Dimensions. An *STTCube* stores *STT objects* as facts modeling their spatial, textual, and temporal features in the corresponding dimensions. Figure 2 shows a *3*-dimensional *STTCube* built on the sample dataset in Table 1 where each row represents one fact (i.e., the members of F) with dimensions $D=\{d_{Location}, d_{Text}, d_{Time}\}$. Domains for the respective dimensions are

$$\delta(d_{Location}) = \{(57.016,\ 09.991), (56.187,\ 10.171), \ldots\}$$
$$\delta(d_{Text}) = \{apple, Fruit, \#love, \ldots\}$$
$$\delta(d_{Time}) = \{11{:}12{:}13\ 20\text{-}10\text{-}2019, 11{:}18{:}23\ 24\text{-}10\text{-}2019, \ldots\}$$

Hence w.r.t. Definition 2, the dimensions capturing λ, φ, and τ are the spatial, textual, and temporal dimensions, respectively. STTCube supports one spatial, textual, and temporal dimension with the possibility of having multiple hierarchies for each.

Dimension Hierarchy. A hierarchy is spatial, textual, or temporal if it contains spatial, textual, or temporal levels, respectively. In Figure 2, the *Location* dimension is a spatial dimension with a spatial hierarchy going from λ to *City, Region*, and *Country* and the *Text* dimension is a textual dimension aggregating φ into the *Term, Theme, Topic*, and *Concept* levels. Similarly, *Time* is a temporal dimension. Hierarchy steps $HS_h=\{hs_1, hs_2, hs_3, \ldots, hs_n\}$ define the mechanism of moving from a lower (child) level to an upper (parent) level and vice versa. A hierarchy step $hs_i=(l_{\downarrow}, l_{\uparrow}, cardinality) \in HS_h$ entails that members of a child level $l_{\downarrow}$ can be aggregated together if they correspond to the same member at the parent level $l_{\uparrow}$ and that this correspondence between children to parent members has the given $cardinality \in \{1{-}1, 1{-}n, n{-}1, n{-}n\}$. For instance, the step from *Date* to *Month* has an $n{-}1$ cardinality, while *Term* to *Topic* has an $n{-}n$ cardinality (e.g., the Carrot *Term* correspond both to the Gardening and Food *Topics*, while the Food *Topic* has as child members not only Carrot but also Apple).

Level Attributes. As mentioned earlier, a level l is associated with a set of attributes $A_l=\{a_1, a_2, \ldots, a_n\}$ and has a set of members $l=\{l_1, l_2, \ldots, l_3\}$. Attribute values describe the different characteristics of each member from that level. Spatial, textual, and temporal levels are then usually characterized by spatial, textual, and temporal attributes. For instance, at the *City* level, all members have the *Boundary* attribute whose value is the polygon defining the boundary of respective city. An example of a textual attribute is *Sentiment* which captures the polarity of the associated textual member. Similarly, an *integer value* representing the number of days in a specific month is a temporal attribute.

3.2 Managing STT Hierarchies

We now describe the STTCube's dimensions and hierarchies.

Spatial Dimensions. Spatial information can be analyzed at different levels and granularities. It is important to note that facts in an STTCube are composed only by geographical *points* (i.e., each tweet or user post is associated with a coordinate, not with shapes or polygons). Points can be aggregated either within a predefined spatial grid or based on semantic information.

Grid-Based Hierarchy. Here, the geographic area being analyzed is divided into small equal size cells with a predefined resolution, e.g., $1{\times}1\ km^2$. At the lowest level, each latitude and longitude point is assigned to the cell they fall in. To analyze data at a coarser granularity, neighboring cells are combined into a larger cell at the parent level (e.g., $3{\times}3\ km^2$). This hierarchy can be built automatically, without the need for any meta-data.

Semantic-Based Hierarchy. Here, data is analyzed in a predefined taxonomy, e.g., an administrative division. Therefore, we move within the taxonomy, e.g., from the *Location* to the *City* level, from the *City* level to the *Region* level, and so on up to the *All* level. This hierarchy requires each object coordinate to be associated with a member in the lowest level in the hierarchy (usually in a pre-processing step) and requires the taxonomy information to build the entire hierarchy.

Textual Dimensions. Hierarchies in the textual dimension move from specific concepts to general ones. This follows a generic taxonomic structure connecting more specific terms to more general ones (i.e., hypernyms) [20]. Textual hierarchies are implemented using WordNet [11] which is discussed in Section 6. In particular, *Terms* are the base level which are grouped into *Themes*, *Themes* into larger categories called *Topics*, and *Topics* in turn grouped into *Concepts. Differently from most hierarchies, the members in the levels of a textual hierarchy are typically in an n−n relationship.* Hence, when moving between textual levels we need to decide how measure values get aggregated. Below we propose a set of aggregation techniques to address this issue.

Replication-Based Hierarchy. This is a common approach where each member of a child level is aggregated into all the parent members. Hence, its value is effectively replicated. This approach leads to a *counting problem* when parent levels are further aggregated. For example, the first data instance in Table 1 will be part of two *Themes*: 1) Fruits because it contains *Term* {apple and fruit} and 2) Emotion because of *Term* {#love}.

Majority-Based Hierarchy. If a fact can be mapped to more than one parent member, then that fact will be part of the parent

member which has the most representation (e.g., in terms of frequency). This scheme avoids double counting of facts in parent members. In case of ties, some tie-breaking heuristic or a user-defined criterion can be employed instead, e.g., the first fact in Table 1 will be part of only the Fruits *Theme* because it has the two representative *Term* {apple, fruit}, as compared to Emotion having only one *Term* {#love}.

Custom Hierarchy. In general, other user-specified criteria and rules can be defined to establish how child-parent level steps will be aggregated in case of ambiguities. For instance, a domain-specific *importance score* can be assigned to the hierarchy members during the *STTCube* construction. In this way, facts will be part of only the parent member with the highest importance.

Temporal Dimensions. Similarly, temporal dimension allows to analyze *STT objects* at different levels of granularity w.r.t. time and has the following two temporal hierarchies: $\tau \rightarrow Day \rightarrow Month \rightarrow Quarter \rightarrow Year \rightarrow all$ and $\tau \rightarrow Second \rightarrow Minute \rightarrow Hour \rightarrow all$. Here, the first contains a hierarchy of *Date* aggregated by the temporal levels *Day, Month, Quarter*, and *Year* (total 5 levels including *All*), whereas the second is a hierarchy for *TimeOfDay* having 4 levels in total.

3.3 Spatial, Textual, and Temporal Measures

As defined earlier, an *n-dimensional STTCube* has a set of measures $M=\{m_1, m_2, m_3, \ldots, m_k\}$, which permit to analyze *STT objects* by computing values at different levels of granularity. For instance, the *STTCube* in Figure 2 models *Location*, *Text*, and *Time* with *Fact Count* as a measure (i.e., *Fact Count*$\in M$). In practice, it maintains the count of *STT objects* at given spatial, textual, and temporal aggregation levels. Measure values at different levels in the hierarchies are obtained by applying an aggregation function over the *STT objects*. Examples of aggregation functions are *SUM*, *COUNT*, *MIN*, *MAX*, and *AVG*. The *STTCube* in Figure 2 uses *COUNT* as an aggregation function. For example, it reports that on *September, 20th* at *AAU Bus Terminal* the *Term apple* was mentioned in 2 facts.

A measure is spatial if it is defined over a spatial domain. A spatial measure is then computed over a collection of spatial values (e.g., geographical points, or geometry shapes like polygons). A spatial measure can be a simple value, e.g., the (numeric) area of the convex hull of multiple shapes, or a complex spatial object, e.g., the polygon representing the convex hull itself. A measure is textual if it is defined over a textual domain, and can be either a simple numeric value or a complex textual object. Analogously, a measure is temporal if it is defined over a temporal domain, A measure is spatio-textual if it is defined over a spatial and textual domain and is a combination of spatial and textual measures. Finally, a measure is spatio-textual-temporal if it is defined over a spatial, textual, and temporal domain and is a combination of spatial, textual, and temporal measures. Below, we propose a list of *spatio-textual and spatio-textual-temporal measures* to be used as part of *STTCube* to analyze *STT objects* effectively.

Top-k Keywords within an Area is a spatio-textual measure which returns a list of tuples $\langle \xi, \overrightarrow{kw} \rangle$ consisting of a geometry shape ξ representing a geographical area and the list of top-k most frequent keywords $\overrightarrow{kw}=\langle w_1, w_2, \ldots, w_k \rangle$ in that area. Analogous to previous measures, it can also be computed at different levels of aggregation, so that it can return the top-k keywords for each *City* or each *Region*.

Keyword Density is a spatio-textual measure which returns a list of tuples $\langle \xi_i, w_j, \rho_{ij} \rangle$ consisting of a geometry shape ξ_i representing a geographical area, a keyword w_j, and its density ρ_{ij} in the area ξ_i. The density ρ_{ij} of a keyword w_j over an area ξ_i is computed as $\rho_{ij} = \frac{freq(\xi_i, w_j)}{SurfaceArea(\xi_i)}$, in which $freq(\xi_i, w_j)$ is the frequency of the keyword w_j in the area ξ_i (i.e., the number of objects located within ξ_i in which w_j appears) and *SurfaceArea* is the surface area of ξ_i. For example, if we have two *Regions* r_1, r_2 with $SurfaceArea(r_1)=10m^2$, $SurfaceArea(r_2)=100m^2$, and the term *Apple* with frequency 5 and 30 in r_1 and r_2, respectively (see Figure 3), then, keyword densities are $\rho_1=0.5$, $\rho_2=0.3$ for r_1 and r_2, respectively.

Top-k Dense Keywords within an Area is a spatio-textual measure which returns a list of tuples $\langle \xi_i, \overrightarrow{kw} \rangle$ computing the keyword density as described in the measure above, but in this case, it returns the top-k keywords $\overrightarrow{kw}=\langle w_1, w_2, \ldots, w_k \rangle$ with the highest density.

Keyword Volatility is a spatio-textual-temporal measure (becomes textual-temporal If no region is specified) which returns a list of tuples $\langle \xi_i, w_j, T_k, \Delta\rho_{ijk} \rangle$ consisting of a geometry shape ξ_i representing a geographical area, a keyword w_j, a time interval T_k, and its change in density $\Delta\rho_{ijk}$ in the area ξ_i over the time interval T_k (divided into k equal intervals). The change in density $\Delta\rho_{ijk}$ of a keyword w_j in an area ξ_i over a time interval T_k is computed as $\Delta\rho_{ijk}=\frac{\sum_{z=1}^{k} |\rho_{ij_z} - \rho_{ij_{z-1}}|}{k}$, where ρ_{ij_z} represents the density of the keyword w_j in the area ξ_i at a specific time instance T_{k_z}. Furthermore, the change in density computation formula can be updated depending on the analysis requirements, e.g., it can be changed to weighted density (assign different weights to each interval in T_k) or to rate of change computation using linear regression [19].

Top-k Volatile Keywords within an Area is a spatio-textual-temporal measure which returns a list of tuples $\langle \xi_i, \overrightarrow{kw} \rangle$ computing the keyword volatility as described above, but in this case, it returns the top-k volatile keywords $\overrightarrow{kw}=\langle w_1, w_2, \ldots, w_k \rangle$ with the highest change in density.

Distributive, Algebraic, and Holistic Measures. There are three types (also known as additivity) of measures: distributive, algebraic, and holistic, depending on whether it is possible to compute the value of a measure at a parent level directly from the values at the child level [13]. For distributive and algebraic measures, this is possible. For instance, the *Fact Count* at the *State* level can be computed by summing up the *Fact Counts* at the *City*. *Keyword Density* is instead an algebraic measure. We can compute the higher-level aggregate values of this measure if we store for each child level both the frequency of each keyword and the *SurfaceArea*. The *Top-k Keywords*, the *Top-k Dense Keywords*, and *Top-k Volatile keywords within an area* measures, instead, are holistic, since the value at a parent level cannot be computed directly from the values at the child level, but it is necessary to recompute them directly from the base facts every time.

Consider the computation of *Top-3 Dense Keywords within an Area* in Figure 3 given the two *Regions* r_1 and r_2 with *SurfaceArea* $10m^2$ and $100m^2$, respectively, and the computation at the parent level $r_3=r_1 \cup r_2$ (grayed-out rows are not part of the computed measure value). The values in the top-3 for the members r_1 and r_2 at the child level are not sufficient to compute the correct densities for region r_3. Both, some of the computed density (in column ρ_{Top-3}, while the correct values are reported in ρ_{all}) and consequently the final ranking, would be wrong. For instance, the keyword *Strawberry* would not have been returned (if computed algebraically) because it is neither in the top-3 for r_1 nor r_2. To

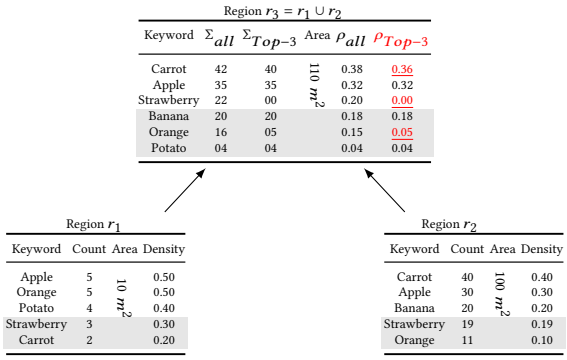


Figure 3: Example: Merging of Holistic Measure

compute the correct response, either we have to store all the aggregate values for each possible cell or we have to reprocess all the facts covered by the query. When dealing with large datasets these approaches are not feasible. Hence, in Section 4 we provide a framework for the computation of an exact and approximate solution with accuracy guarantees.

3.4 STTOLAP Operators

A data cube allows different **O**n**L**ine **A**nalytical **P**rocessing (OLAP) operators to group, filter, and analyze cells and subsets of cells at different levels of granularity and under different perspectives. Those operators are known as *Slice*, *Dice*, *Roll-Up*, and *Drill-Down* [18]. We extend the basic OLAP operators to *STTOLAP operators*, i.e., for spatial, textual, and temporal dimensions, hierarchies, and measures (Handing of n–n relationships is explained in Section 4 and 5). In general, an OLAP (and STTOLAP) operator *OP* accepts as input a cube C', some parameters *params* and outputs a new cube C'', i.e., $OP(C', params)=C''$. In this way, a new OLAP operator can be applied to C''. Among all cubes, we distinguish the initial or *base cube* C as the cube containing all the original information at the base level.

4 CUBE MATERIALIZATION

Cube materialization is the process of pre-aggregating measure values at different levels of granularity in the cube to compute query responses from pre-aggregated results instead of the raw data, and hence improve query response time for *STTOLAP operators* [16]. In a data cube, a *cuboid* is a collection of *level members* and associated *measure values* for a unique combination of dimension hierarchy levels. Each unique combination is represented by a separate cuboid. For instance, if we request the *Fact Count* for the *State* of Denmark and have stored *Fact Count* at the *Region* level, we can avoid accessing the raw data and compute the aggregation from much fewer rows. This is an example of *partial materialization*, i.e., the actual cuboid at the *State* level, containing the answer to the query was not materialized, but the system was still able to exploit the cuboid for *Region*.

What to materialize and how much to materialize depends on the trade-off between query response time and storage cost. *Full Materialization (FM)* is obtained by pre-computing measure values for *all* combinations of levels in *all* hierarchies. This approach requires huge storage but achieves the best query response time since every operation can just look up already pre-computed results. At the other extreme, *No Materialization (NM)* only materializes the base cuboid and does not require any extra space, but will require aggregated measure values to be recomputed from the base cuboid every time, hence incurring much slower response times. A middle-ground solution is to partially materialize the cube, i.e., to materialize only some of the possible cuboids. In this strategy, some queries will be able to exploit pre-aggregated values at the current level, while other queries can exploit pre-aggregated values at lower levels for distributive or algebraic measures.

4.1 Cost Model

The core of the proposed *partial materialization* approach depends on the trade-off between the storage cost of materializing any particular cuboid and the actual *benefit* that the materialization of the cuboid provides. To evaluate this benefit, we have to estimate the (run time) cost of a query. To devise a cost model for this estimation, we performed a micro-benchmark which confirmed that the running time is directly proportional to the data size (the number of rows). Hence we can use the following linear cost model for benefit calculation

$$Benefit(c) = \sum_{c' \in descendants(c) \cup \{c\}} cost(c') - size(c)$$

4.2 Partial Exact Materialization

We propose an exact partial materialization technique for precomputing the *spatio-textual* and *spatio-textual-temporal measure* values. To answer an *STT query* for these measures we materialize two other distributive measures, namely *Keyword Frequency* f and *SurfaceArea* a. Then, since *Keyword Density* ρ and *Keyword Volatility* $\Delta\rho$ are algebraic measures, they can be computed from the values of *Keyword Frequency* f and *SurfaceArea* a. Finally, *Top-k Dense Keywords* and *Top-k Volatile keywords* are holistic but for an exact solution we materialize *Top-ALL* and hence, compute it from the materialized measure values (Figure 3).

We adopt the chosen linear cost model (Section 4.1) and extend the greedy algorithm approach [16] to our task (Algorithm 1). *Additionally, and different from [16], Algorithm 1 accepts an input parameter K and materializes only the top-K measures values in each cuboid.* For instance, for $K = 10$, it will materialize the top-10 keywords in each cuboid. Then, any top-k query, with $k \leq K$, for a materialized cuboid will return the pre-computed answer.

Algorithm 1, given a size budget B (measured in rows, cuboids, or GB), proceeds until the size of the current cube is as large as possible within the budget (Line 6). At each step, it selects among all the non-materialized cuboids (Line 3) the one with the highest benefit (Line 4) and materializes it (Line 5). The difference between the exact (PEM) and approximate (PAM) materialization using Algorithm 1 is the value of K. When $K{=}\infty$ the full sorted list of measure values will be stored so that all top-k queries can be answered for that cuboid. We set $K{=}\infty$ and $K{=}n$ (to materialize only top n measure values) for *PEM* and *PAM*, respectively.

Query rewriting. Finally, as in [16], after *STTCube* materialization, queries are still formulated in terms of the base cuboid but rewritten by the system to be evaluated over the smallest cuboid.

Algorithm 1: Greedy Materialization

```
1 GreedyMaterialization (B, 🧊, K)
      Input: Budget B, STTCube 🧊, desired top-k K
      Output: Partially Materialized STTCube 🧊
2     do
3         Candidates ← {V ∈ 🧊 | ¬V.isMaterialized};
4         V̄ ← max_{V ∈ Candidates} Benefit(V);
5         🧊.materialize(V̄, K);
6     while size(🧊) ≤ B;
7     return 🧊;
```

Algorithm 2: Top-K Volatile Keywords in an Area

1 **TopKVolatile** (Φ=$\{\langle \xi_1, \overrightarrow{kw_1}, T_1\rangle, \ldots, \langle \xi_n, \overrightarrow{kw_n}, T_n\rangle\}$,$T_x$,$k$)
  Input: Set of Top-k+1 Volatile Keywords lists Φ, Set of x Timestamps T_x, Integer k
  Output: $\langle \xi, \overrightarrow{kw}, T_n\rangle$ top-k keywords $\overrightarrow{kw}$ in the merged area ξ over time interval T_x, δ number of guaranteed top positions
2 $\xi \leftarrow \bigcup_{i\in[1,n]} \xi_i$, $A \leftarrow SurfaceArea(\xi)$; // Merge areas
3 $\overline{kw} \leftarrow \{\}, \Delta f \leftarrow \{\}, prev_f \leftarrow \{\}$; // Empty dictionaries
4 **foreach** $t \in T_x$ **do**
5 **foreach** $\langle \xi_i, \overrightarrow{kw_i}, T_i\rangle \in \Phi$ **do**
6 **foreach** $j \in [1, \ldots, k{+}1]$ **do**
7 **if** $t \in T_i$ **then**
8 $w \leftarrow \overrightarrow{kw_i}.get(j)$; // keyword at j
9 $f \leftarrow \overrightarrow{kw_i}.freq(j)$; // frequency at j
10 $\overline{kw}[w] \leftarrow \overline{kw}[w] + f$;
11 $\Delta f[w] \leftarrow \Delta f[w] + |prev_f[w] - f|$;
12 $prev_f[w] \leftarrow f$;
13 $\epsilon \mathrel{+}= \overrightarrow{kw_i}.freq(k+1)$;
14 $\overrightarrow{kw} \leftarrow topK(\overline{kw}, A, \Delta f)$; // top-k volatile keywords
15 $\delta \leftarrow \max_{j\in[1,\ldots,k]} \overrightarrow{kw}.freq(j) \geq \epsilon$;
16 **return** $\langle \xi, \overrightarrow{kw}, T_n\rangle, \delta$

4.3 Partial Approximate Materialization

As a result of the materialization performed by Algorithm 1, when querying a non-materialized cuboid, we can directly exploit values in the cuboid's materialized ancestors when computing all distributive and algebraic measures. On the other hand, for holistic measures, we have to perform some additional computation. For instance, as mentioned earlier, to compute the value for the *Top-k Dense Keywords in an area* we can exploit the pre-computed *Keyword Density* values, but then we need to perform the top-k selection. That is, if the top-k for the current view is not materialized, we cannot exploit the materialized top-k of the ancestor views without incurring the risk of returning the wrong result.

Yet, it is possible to exploit the top-k computation in some materialized cuboid to retrieve an *approximate* top-k and estimate the result's accuracy [33]. In practice, for the *Top-k Dense Keywords within an area*, given a target k for the top-k computation, when materializing a cuboid, we materialize the top-k+1 most dense keywords for that cuboid (i.e., set K=k+1 in Algorithm 1). Then, to compute the top-k dense keywords for a descendant cuboid by exploiting a materialized ancestor cuboid, we determine which members of the list are guaranteed to be correct.

Algorithm 2 implements this computation for *Top-k Volatile Keywords within an area*. It receives as input the set Φ=$\{\langle \xi_1, \overrightarrow{kw_1}, T_1\rangle, \langle \xi_2, \overrightarrow{kw_2}, T_2\rangle, \ldots, \langle \xi_n, \overrightarrow{kw_n}, T_n\rangle\}$ of lists of top-K (i.e., k+1) dense keywords in a specific area with respective time stamps, time interval T_x divided into x equal-sized interval (e.g., day or month), and the value for k. The output is the ranked list of top-k volatile keywords in the area ξ that is composed by the merging of the areas $\xi_1, \xi_2, \ldots, \xi_n$. It computes the *SurfaceArea* of the merged area ξ (lines 2). Then it merges all the aggregated keyword frequencies (line 10) and change in keyword frequencies (line 11) for each time instance in T_x(line 7) in respective dictionaries $\overline{kw}$ and Δf (lines 4-13) by getting each keyword in each list (line 8) and the corresponding frequencies (line 9). If a keyword is not found in the $\overline{kw}$, Δf, or $prev_f$ dictionary then its value is considered to be *zero*. Moreover, it keeps track of the upper-bound ϵ frequency for keywords outside the current materialized ranking for possible error reporting (line 13). Once all frequencies and changes in frequencies are merged, we compute the top-k volatile keywords using the aggregated values (line 14). Finally, by comparing the value of ϵ with the frequencies of keywords in the aggregated top-k, we report how many positions in the current ranking are guaranteed to be exact (line 15). In the best case, the frequency of the keyword at position k will be at least ϵ and thus the computed top-k is guaranteed to be correct.

Algorithm 3: STTCubeConstruction

1 **ConstructSTTCube** ($\mathcal{X}, \mathcal{T}, \mathcal{G}, B, K$)
  Input: Collection of Spatio-Textual-Temporal Objects $\mathcal{X}$, Knowledge Source $\mathcal{T}$, Geographical Information $\mathcal{G}$, Materialization Budget B, desired top-k K
  Output: Spatio-Textual-Temporal Cube $\square$
2 $\square \leftarrow$ load empty **or** existing cube;
3 $\square.d_{Time} \leftarrow$ initialize **or** load temporal dimension;
4 $\square.d_{Location} \leftarrow$ initialize **or** load spatial dimension;
5 $\square.d_{Text} \leftarrow$ initialize **or** load textual dimension;
6 $\square.F \leftarrow$ initialize empty **or** load existing *Fact Table*;
7 **foreach** $x \in \mathcal{X}$ **do**
8 UpdateTemporalHierarchies($x.\tau, \square.d_{Time}$);
9 $\lambda' \leftarrow$ProcessLocation($x.\lambda$);
10 UpdateSpatialHierarchies($\lambda', \mathcal{G}, \square.d_{Location}$);
11 $\varphi' \leftarrow$ProcessText($x.\varphi$);
12 UpdateTextualHierarchies($\varphi', \mathcal{T}, \square.d_{Text}$);
13 InsertFact($x.\tau, \lambda', \varphi', \square.F$);
14 GreedyMaterialization($B, \square, K$);
15 **return** $\square$;

5 STTCUBE CONSTRUCTION

Here, we describe the proposed approach for constructing an *STTCube*. Algorithm 3 takes a collection $\mathcal{X}$ of *STT objects* to be analyzed, a textual taxonomy $\mathcal{T}$ with semantic information about the terms, themes, topics, and concepts, and a geographical taxonomy $\mathcal{G}$ for cities, regions, and countries. Standard *date functions* are used for the temporal dimension processing. Moreover, it also receives as input the parameters B and K as the budget and number of top-K keywords for the partial materialization.

Algorithm 3 constructs the STTCube in an incremental way, it initializes an empty cube (line 2), and then the corresponding spatial, textual, and temporal dimensions (lines 3-5) as well as the *Fact Table* (line 6). If the cube is already constructed, i.e., the cube is being updated instead of constructed for the first time, then Algorithm 3 loads the existing *STTCube* (lines 2-6) and updates it with new information. In particular, the spatial dimension has the grid-based hierarchy and the hierarchy with the base level at each object's *Location* (i.e., the geographical point), and then the levels *City*, *Region*, *Country*, and *All* (5 levels in total). The textual dimension, instead, has the hierarchy build from the base level *Term*, and then *Theme*, *Topic*, *Concept*, and *All* (5 levels in total). Finally, the temporal dimension contains the *Date* and *TimeOfDay* hierarchies mentioned in Section 3.

Once the basic structure is prepared, Algorithm 3 loops through each STT object in $\mathcal{X}$ (lines 8-13). In this loop, it extracts and initializes from each *STT object* the base-level members for each dimension. Then, once the base level data has been extracted, it proceeds with building the various dimension hierarchies starting from the existing base-level members and exploiting the provided spatial and textual taxonomies (lines 8-12). Once the dimension hierarchies are built, the *STT object* itself is then inserted in the fact table of the *STTCube* (line 13) so that each fact is linked to the lowest (base) level members in the respective dimensions. In this step (line 13), the fact measure values are also computed (e.g., the keyword count). As the last step (line 14) Algorithm 3 executes the (partial) materialization procedure.

Spatial Hierarchies Construction. In our proposed *STTCube* the base level for the spatial hierarchies is the *Location* present in the raw data, i.e., the longitude and latitude points. Hence, we use Military Grid Reference System (MGRS) for grid-based hierarchy and when building the semantic-based hierarchy, individual points are linked to the respective cities using the information in

the available geographical taxonomy $\mathcal{G}$, or to a special member for points that link to unknown locations. Therefore, this corresponds to the step function from *Location* to *City*. The spatial taxonomy $\mathcal{G}$ is also used to generate the spatial hierarchy step functions for the higher levels.

Textual Hierarchies Construction. The unstructured nature of the text makes it a challenging task to convert it into a dimension of a cube. In Algorithm 3, the `ProcessText` function (line 11) implements the following steps: (1) splits the text into individual words, (2) removes *stop words*, and (3) converts the remaining words to their base form (e.g., "works" and "working" have the same base form "work"). The final processed text is used to populate the *Term* base-level in the textual dimension. This implements the base step function in the textual hierarchies, and links every fact to one or more *Terms*, hence it has an $n-n$ cardinality. Moreover, while constructing the higher levels, using the semantic taxonomy $\mathcal{T}$ (e.g., WordNet), each *STT object* is linked to one or more *Themes*, and similarly for *Topics*, and *Concepts*.

6 EXPERIMENTAL EVALUATION

Now, we report on the performance of *STTCube* analysis. In particular, we compare the different materialization strategies for *STTCube* and *No STTCube (NC)* implementations, in terms of query response time (QRT) and storage cost. *NC* answers the queries by computing the query response from base data without constructing the *STTCube*. Also, we compare QRT and hierarchy construction time for different combinations of hierarchy schemes. Moreover, we also report on the accuracy of *PAM* and demonstrate the advantage in performance when compared to *PEM*. Lastly, we compare QRTs for different spatial and textual hierarchy schemes, showing that combinations of *Grid-based spatial and Majority-based textual (GM)* hierarchy scheme achieves the fastest QRTs among all hierarchy combinations.

Experimental Setup. We evaluate the *STTCube* on a real-world Twitter dataset containing 125 million tweets collected over *six* weeks. Each tweet contains the tweet location, text, and time. We implemented the *STTCube* in a leading commercial RDBMS, called *RDBMS-X* as we cannot disclose the name. The proposed design is realized using a *snowflake schema* to avoid redundancy in the dimension data.

We implemented the *Pre-Processing (PP)* component, where the whole raw dataset is parsed and the relational tables are populated, in Java (v11). All tests are run on a Windows Server machine with 2 Intel Xeon 2.50GHz CPUs and 16GB RAM.

We extracted the taxonomy for the spatial dimension from GeoNames [1]. For the *City* level, we considered all the cities having $population > 1000$ and for the *Region* level, we use administrative divisions information available in the GeoNames dataset. We use the reverse geocoding process to find the city name for the *Location* coordinates.

For the textual dimension, as a taxonomy for *Terms*, *Themes*, *Topics*, and *Concepts*, we use the widely used WordNet [11]. We use the direct *HYPERNYM* link of WordNet to decide the parent member for a *Term*, *Theme*, and *Topic*. If a term is present in WordNet and has a super-class (*HYPERNYM*) then the super-class becomes the parent of the term. Otherwise, it becomes its own parent (this avoid unbalanced hierarchies and UNKNOWN values in the hierarchy). For text pre-processing –tokenization, lemmatization, and stop word removal– we use the Stanford Core NLP library [26]. We implemented the temporal dimension using the standard *Date* and *Time* functions supported in *RDBMS-X*.

We implemented the *semantic-based* and *grid-based* hierarchy schemes for the spatial dimension, *replication-based* and *majority-based* hierarchy schemes for the textual dimension (Section 3.2), and *Date* hierarchy for the temporal dimension.

Spatial, Textual, and Temporal Levels Members. The base levels contain 40.1 million unique *Location Points* and 9.8 million unique *Terms*. The GeoNames taxonomy contains 132K cities, divided into 4K administrative divisions (regions) for 247 countries. Among those, we have tweets for 104K cities, 3.8K regions, and 246 distinct countries. In the textual hierarchy, terms are grouped into 23.8K *Themes*, 19.4K *Topics*, and 17.6K *Concepts*. Furthermore, the temporal dimension spans over 37 days. Finally, for *PAM* we materialize K=31 densest keywords.

We compare *PEM* and *PAM* strategies with the following three baselines. **No STTCube (NC):** is the traditional RDBMS setup with all textual, spatial, and temporal functions implemented as built-in or user-defined functions. Specifically, *NC* uses user-defined functions for text (for retrieving individual terms) and location processing (e.g., identification of the city a particular longitude, latitude point belongs to) and built-in functions for timestamp. Further, *NC* filters on location and timestamp for the queried *area* and time and performs a series of joins, e.g., 4 joins for *Concept* level, to retrieve information for the requested textual level. Finally, it groups results on the textual and temporal columns, computes the *STT measure values*, and performs the top-k selection. *NC* is the *traditional* solution one would go for without the *STTCube*. **No Materialization (NM):** constructs the *STTCube* and minimizes the storage cost by only materializing the base cuboid and computing all query responses from it. **Full Materialization (FM):** minimizes the QRT by materializing every cuboid in the *STTCube*. With this approach queries are answered through a lookup in the pre-computed cuboid. *These three baselines are at the extreme ends of the space-time trade-off and are usually infeasible for large datasets.*

Queries. We perform experiments on five different sizes of datasets using nine different *STT queries*. Each *STT query*, described in Table 3, targets a different level of spatial, textual, and temporal granularity. Each query requests either dense or volatile keywords with a range of time which is used for volatile but not used for dense keywords queries. We execute each query ten times with randomly generated parameters for each method and report mean and standard deviation.

Query Response Time. For *Top-k Dense and Top-k Volatile Keywords within an area* measures, we compare the QRT of *PEM* and *PAM* with the *NC*, *NM*, and *FM* baselines. For *Keyword Density* and *Keyword Volatility*, no approximate solution is possible so we only compare *PEM* with *NC*, *NM*, and *FM*. As the *Majority-based* textual hierarchy scheme does not process *Terms* (Section 3.2), we only evaluate *five* out of *nine* queries requesting *Theme*, *Topic*, and *Concept* for it (Figures 4c, 4d, 4g, and 4h).Furthermore, we cannot evaluate *PAM* for Q9 as no approximate solution is possible for it.

Table 3: Spatio-Textual-Temporal Queries

Query	Description
Q1	Top-k Dense/Volatile *Terms* in a *City [time span]*
Q2	Top-k Dense/Volatile *Topics* in a *City [time span]*
Q3	Top-k Dense/Volatile *Concepts* in a *Country [time span]*
Q4	Top-k Dense/Volatile *Terms* in a *Region [time span]*
Q5	Top-k Dense/Volatile *Concepts* in a *Region [time span]*
Q6	Top-k Dense/Volatile *Themes* in a *Region [time span]*
Q7	Top-k Dense/Volatile *Terms* in a *Country [time span]*
Q8	Top-k Dense/Volatile *Terms* in a *Country* Group by *Region [time span]*
Q9	Top-*ALL* Dense/Volatile *Topics* in a *Country* Group by *Region [time span]*

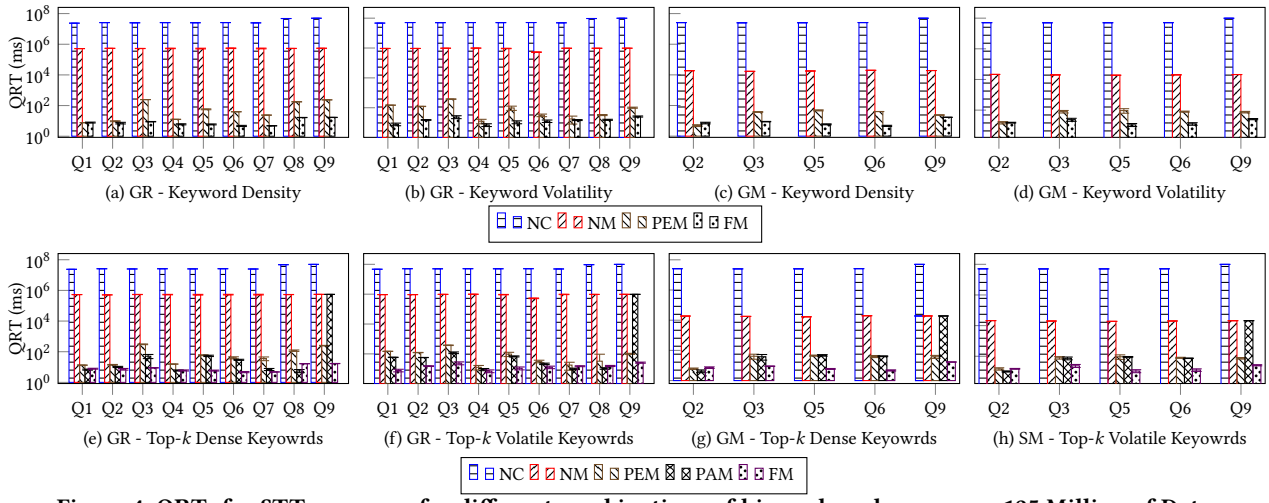


Figure 4: QRTs for STT measures for different combinations of hierarchy schemes over 125 Million of Data

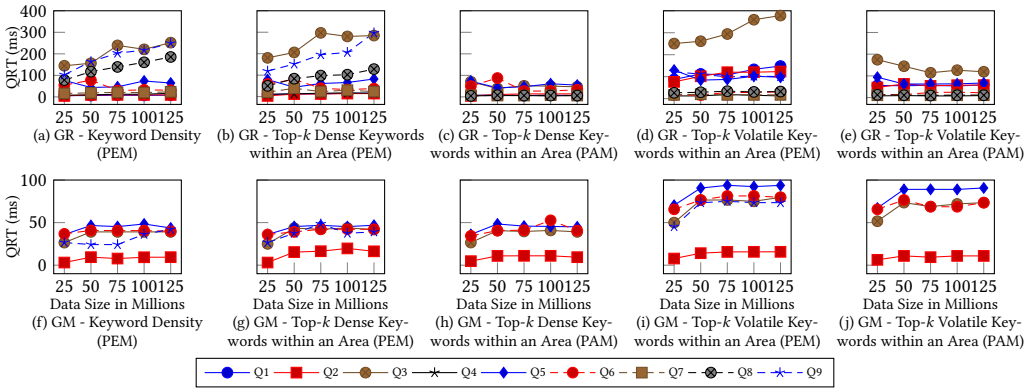
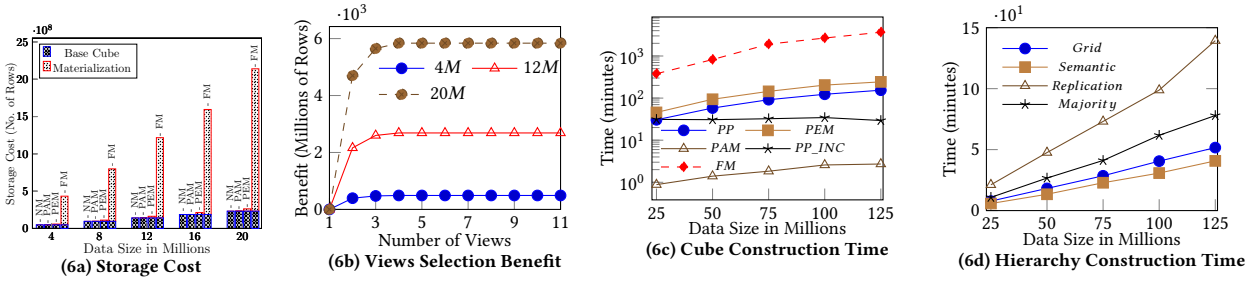


Figure 5: QRT Vs Data Size

We plot results in Figures 4a—4h for 100% (125M) of data, as the results are similar for smaller data sizes. Each row in Figure 4 shows the QRTs for one particular combination of spatio-textual hierarchy schemes. Specifically, Figures 4a, 4b, 4e, and 4f show the QRTs for the Grid-based spatial and Replication-based textual (**GR**) hierarchy combination for all *measures*. Similarly, Figures 4c, 4d, 4g, and 4h show QRTs for Grid-based spatial and Majority-based textual (**GM**) combinations. Figure 4 has queries on the x-axis and QRTs in msec on the y-axis (note: log scale). Figure 4 confirms that *NC* is 1−5 orders of magnitude slower than *NM*. Specifically, regardless of the spatial hierarchy scheme, it is 1−2 and 3−5 orders of magnitude slower than *NM* for *Replication-based* and *Majority-based* textual hierarchy, respectively. The *Majority-based* textual hierarchy scheme achieves faster QRTs because it does not process individual *Terms* but directly links *Theme* to the *fact*, hence, drastically reducing the number of rows to process (from millions to thousands). Furthermore, *NM* is 1−4 and 3−5 orders of magnitude slower than *PEM* and *PAM*, respectively, for all measures and combinations of hierarchy schemes. *PEM* is on average six times slower than *FM* which achieves its fast QRTs at the expense of a highly increased storage cost (Figure 6a). *PAM* achieves *near-optimal* QRTs because it materializes only *the K densest keywords* in the cuboid, hence it has much fewer rows to process. QRTs for *Q9* for *Top*-k *Volatile Keyword within an area* and *Top*-k *Dense Keywords within an area* measures for all combinations of hierarchy schemes are the worst for *PAM* (same as *NM*) because it requests *ALL* keywords' densities instead of *top-k* which cannot be computed from the approximate pre-aggregated information. To generate a response for *Q9*, we have to process all detail data directly from the base facts. In comparison, *PEM* and *PAM* materialize a subset of views (also a subset of rows for *PAM*) and use the pre-aggregated measure values in those views to efficiently generate a response for a query instead of processing base facts, thus improving the overall QRT. *NC* is the slowest of all (1 − 5 orders of magnitude slower than the slowest *STTCube NM*) because it has to process the complete dataset for computing each query response, and cannot take advantage of the *STTCube* optimizations for *STT measures*. Among all the hierarchy scheme combinations, *GM* has the fastest QRTs mainly because of *Majority-based* which drastically reduces the row count by linking the *Theme* directly to each *Fact* instead of individual *Terms*, whereas, *GR* has the slowest QRTs due to *Replication-based* having far more rows to process than *Majority-based* textual hierarchy. Furthermore, *Grid* and *Semantic-based* spatial hierarchies have similar QRTs.

Figure 5 shows the scalability of *PEM* and *PAM* over growing data sizes for different combinations of hierarchy schemes and confirms that the QRTs are almost constant as the data grows. This is because the sizes of materialized views do not increase a lot as the data grows. Only new dimension members, e.g., new cities or topics, increase the size of materialized views, but only by a small fraction. Figures 5f—5j confirm that the *GM* hierarchy combination results in the fastest QRTs, i.e., all QRTs < 100 msec. On the contrary, Figures 5a—5e show that *GR* yields the slowest



(6a) Storage Cost (6b) Views Selection Benefit (6c) Cube Construction Time (6d) Hierarchy Construction Time

Figure 6

QRTs, with QRT as high as 400 msec. Figure 5 confirms that *PAM* consistently achieves the fastest QRTs (mostly < 100 msec with few a bit over 100 msec) regardless of hierarchy schemes. Figure 5 shows that *PEM* and *PAM* scale linearly w.r.t. data size.

Storage Cost. We now compare the storage cost for *FM*, *PEM*, *PAM*, and *NM*. We do not compare *NC*'s storage cost because it does not construct STTCube, and hence does not materialize anything. We only show the storage cost for up to 20 million because *FM* takes an unfeasible amount of time (shown in Figure 6c) while for the other methods and over the larger datasets we observe the same trend. We use the number of rows in a view as its storage cost.The base cube's storage cost is always needed. Besides that, every additional materialized view adds to the storage cost, as displayed in Figure 6a, that shows the storage cost of *NM*, *PAM*, *PEM*, and *FM* over growing data sizes. The materialization of the STTCube using *PEM* and *PAM* only adds 13% and 0.1% to the storage cost of the base cube, respectively. Whereas, using *FM* increases the storage cost by more than an order of magnitude. *PEM* reduces the storage cost by only materializing a subset of views (four views) and still achieves 2-5 orders of magnitude improvement in QRT (Figures 4). *PAM* further reduces the storage cost by only materializing a subset of rows in a view (top-k) and gains an additional order of magnitude improvement in QRT. On the other hand, *FM* materializes all views in a cube, i.e., 500 ($5 \times 5 \times 5 \times 4$) views in our case, which makes the view materialization storage cost much higher (one order of magnitude) than the base cube itself, as shown in Figure 6a. Figure 6a confirms that our proposed methods *PEM* and *PAM* reduce the storage cost between 97% and 99.9% compared to *FM*.

Views Selection for Materialization. Our proposed methods *PEM* and *PAM* are partial materialization methods that materialize only a subset of the cuboids. Hence, an important trade-off to be understood is between the number of cuboids to materialize, the corresponding storage cost, and the gain in query response time achieved. We empirically evaluate the benefit gained (improvements in QRT for all dependent cells which can be answered using this view) against the cost of materializing the view (Algorithm 1). We consider the base cube as a necessary view to be materialized and consider its benefit as zero. Figure 6b shows that materializing three cuboids (*(Day, City, Term)*, *(Day, Location, Theme)*, and *(Day, Region, Term)*) on top of the base cube gain the most benefit after which we do not get a significant advantage of materializing further cuboids. The reason is that the materialized cuboids are already small enough, so the benefit of materializing any descendant cuboid is small. Hence, materializing 4 cuboids represents the best trade-off between QRT and storage cost.

Pre-Processing and Cube Construction. Here, we report the time for the construction of *STTCube*. Construction of an STTCube is divided into two steps: 1) *Pre-Processing (PP)* of base facts (*STT objects*) and population of the relational tables and 2) materialization of views. Further, the materialization of views can be done either using *FM*, *PEM*, or *PAM*. In Figure 6c, we have data sizes on the x-axis and time in minutes on the y-axis (note: log scale). *FM* is the most time consuming among all and adds significant overhead on top of *PP* time and does not scale. On the contrary, *PEM* and *PAM* time is negligible compared to the *FM* time. Hence, with *PEM* and *PAM* *STTCube* construction time scales linearly. To evaluate *STTCube's* ability to handle updates (maintenance wall-clock time), we performed several updates of 25M tweets each (*PP_INC* in Figure 6c). Experiments confirm that STTCube's update time grows linearly with the amount of new *STT objects* because it only processes the new STT objects and updates respective fact and dimensions tables.

Furthermore, we compare the different hierarchy schemes w.r.t. their construction time. Figure 6d shows the hierarchies' construction time for different hierarchy schemes. It is evident from Figure 6d that, among all, the *Replication-based* textual hierarchy scheme takes the longest to construct because for each single *spatio-textual-temporal* object it has to process each individual *Term* and construct hierarchy for it. Whereas, for all other schemes, for each spatio-textual-temporal object only one hierarchy instance is processed. Figure 6d confirms that all of the hierarchy schemes are constructed in linear time w.r.t. data size, allowing STTCube to support multiple hierarchy schemes.

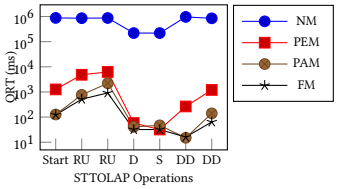
Accuracy. Given that *PAM* efficiently computes the approximate measure values, it becomes necessary to evaluate its accuracy. To evaluate the accuracy of *PAM*, we use *NM*'s results as ground truth. Our evaluation result in Table 7a confirms that it achieves high accuracy. Specifically, it is 100% for 6 out of 8 queries, and 90-97% for 2. Queries with 90-97% accuracy request as many keywords as are materialized and the risk of having wrong results near the border (bottom of the top-k list) is higher.

QRT of STTOLAP Operators. Our proposed materialization strategies (*PEM and PAM*) improves the QRTs for *STTOLAP operators*. To demonstrate this, we perform a series of STTOLAP operations and measure their QRT for different materialization strategies. Figure 7b shows the QRTs for multiple *STTOLAP operations* for different materialization strategies. We have STTOLAP operators on the x-axis (RU, D, S, and DD represents STT Roll Up, Dice, Slice, and Drill Down operators, respectively) on QRT in msec on the y-axis. It is evident that *NM* is on average 3-5 orders of magnitude slower than *PEM* which is one order of magnitude slower than *PAM*. Furthermore, *PAM* achieves near-optimal QRTs, just a fraction higher than *FM*. These experiments confirm that *STTCube's* materialization methods (*PEM and PAM*) improves *STTOLAP operators'* QRTs by materializing only a subset of cuboids.

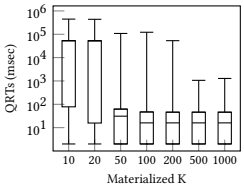
Top-K Value Estimation. Here, we study the relationship between QRT and the value of materialized K. We create *seven*

Query	Data Size in Millions				
	25	50	75	100	125
Q1	100.0	100.0	100.0	100.0	100.0
Q2	100.0	100.0	100.0	100.0	100.0
Q3	100.0	100.0	90.0	95.0	90.0
Q4	92.3	100.0	100.0	100.0	100.0
Q5	100.0	100.0	100.0	100.0	100.0
Q6	100.0	100.0	100.0	100.0	100.0
Q7	93.3	96.7	90.0	93.3	93.3
Q8	100.0	100.0	100.0	100.0	100.0

(7a) PAM's Accuracy



(7b) STTOLAP Operations' QRTs



(7c) Materialized-K Vs QRT

Figure 7

different *STTCube* materialization versions using 10, 20, 50, 100, 200, 500, and 1000 as the value of K. Next, we use the Gamma distribution to generate 100 random numbers, to be used as top-k values, in the range of 1 and 1000. We chose the Gamma distribution because it resembles a common long-tail distribution for top-k values. We execute each query for all the 100 generated top-k values over all *seven* materialization versions. Figure 7c shows the QRT for all queries over different materialization versions. For K=10 and 20 the median value is the same as the box top, hence not visible in the plot. It is evident from Figure 7c that a larger value of materialized K achieves faster QRTs (lower median value) because almost all the queries are answered using the pre-computed measure values. But, in the case of smaller K, all the queries requesting k>K need to be answered using the non-pre-computed measure values from the base cuboid. Hence, resulting in slower QRTs (higher median value). A larger value of K such as 1000 is not recommended because 1) there will be very few queries requesting a larger top-k and 2) it will require more storage cost (Figure 6a). Specifically, between K=50 and 100 and K=100 and 200 QRT decrease by 35% and 0% but storage increase 250% and 200%, respectively. Hence, these experiments confirm that choosing a value between 20—50 for K in our current experiments settings is a near-optimal choice.

7 CONCLUSION AND FUTURE WORK

In this paper, we defined and formalized the *STTCube* structure to effectively perform *STTCube analytics. We introduced STT hierarchies, STT measures, and STTOLAP operators to analyze STT data together.* For efficient, exact and approximate, computation of *STT measures*, we proposed a pre-aggregation framework able to provide faster response times by requiring a controlled amount of extra storage to store pre-computed measure values. We observed how the partial materialization provides 1 to 5 orders of magnitude reduction in query response time, with between 97% and 99.9% reduced storage cost compared to full materialization techniques. Moreover, the approximate materialization provides accuracy between 90% and 100%, while requiring considerably less space compared to no materialization techniques. In future work, we plan to enhance STTCube with additional *STT measures* and distributed implementation.

REFERENCES

[1] 2020. GeoNames. http://download.geonames.org/. Accessed: 2020-09-09.
[2] A. Almaslukh, A. Magdy, A. M. Aly, M. F. Mokbel, S. Elnikety, Y. He, S. Nath, and W. G. Aref. 2019. Local trend discovery on real-time microblogs with uncertain locations in tight memory environments. *GeoInformatica* (2019).
[3] M. Azabou, K. Khrouf, J. Feki, C. Soulé-Dupuy, and N. Vallès. 2016. Analyzing textual documents with new OLAP operators. *AICCSA* (2016).
[4] X. Cao, L. Chen, G. Cong, C. S. Jensen, Q. Qu, A. Skovsgaard, D. Wu, and M. L. Yiu. 2012. Spatial Keyword Querying. *ER* (2012).
[5] L. Chen, G. Cong, C. S. Jensen, and D. Wu. 2013. Spatial Keyword Query Processing: An Experimental Evaluation. *PVLDB* (2013).
[6] Y. Chen, G. Dong, J. Han, B. W. Wah, and J. Wang. 2002. Multi-dimensional Regression Analysis of Time-series Data Streams. *VLDB* (2002).
[7] M. L. Chouder, S. Rizzi, and R. Chalal. 2019. Exploratory OLAP over Doc Stores. *IS* (2019).
[8] G. Cong, K. Feng, and K. Zhao. 2016. Querying and mining geo-textual data for exploration: Challenges and opportunities. *ICDEW* (2016).
[9] G. Cong and C. S. Jensen. 2016. Spatial Keyword Queries and Beyond. *SIGMOD* (2016).
[10] B. Ding, B. Zhao, C. X. Lin, J. Han, C. Zhai, A. Srivastava, and N. C. Oza. 2011. Efficient Keyword-Based Search for Top-K Cells in Text Cube. *TKDE* (2011).
[11] C. Fellbaum. 1998. WordNet: An Electronic Lexical Database. *MIT Press* (1998).
[12] W. Feng, C. Zhang, W. Zhang, J. Han, J. Wang, C. Aggarwal, and J. Huang. 2015. STREAMCUBE: Hierarchical spatio-temporal hashtag clustering for event exploration over the Twitter stream. *ICDE* (2015).
[13] J. Gray, A. Bosworth, A. Lyaman, and H. Pirahesh. 1996. Data cube: a relational aggregation operator generalizing GROUP-BY, CROSS-TAB, and SUB-TOTALS. *ICDE* (1996).
[14] N. Gür, T. B. Pedersen, E. Zimanyi, and K. Hose. 2017. A foundation for spatial data warehouses on the Semantic Web. *Semantic Web* (2017).
[15] J. Han, K. Koperski, and N. Stefanovic. 1997. GeoMiner: A System Prototype for Spatial Data Mining. *SIGMOD* (1997).
[16] V. Harinarayan, A. Rajaraman, and J. D. Ullman. 1996. Implementing data cubes efficiently. *SIGMOD* (1996).
[17] P. Jayachandran, K. Tunga, N. Kamat, and A. Nandi. 2014. Combining User Interaction, Speculative Query Execution and Sampling in the DICE System. *ICDE* (2014).
[18] C. S. Jensen, T. B. Pedersen, and C. Thomsen. 2010. *Multidimensional Databases and Data Warehousing*. Morgan & Claypool Publishers.
[19] J. F. Kenney and E. S. Keeping. 1962. Mathematics of Statistics, Part 1, chapter 15. *van Nostrand* (1962).
[20] J. D. Knijff, F. Frasincar, and F. Hogenboom. 2013. Domain taxonomy learning from text: The subsumption method versus hierarchical clustering. *DKE* (2013).
[21] M. D. Lieberman, H. Samet, J. Sankaranarayanan, and J. Sperling. 2007. STEWARD: Architecture of a Spatio-textual Search Engine. *GIS* (2007).
[22] C. X. Lin, B. Ding, J. Han, F. Zhu, and B. Zhao. 2008. Text Cube: Computing IR Measures for Multidimensional Text Database Analysis. *ICDM* (2008).
[23] L. Lins, J. T. Klosowski, and C. E. Scheidegger. 2013. Nanocubes for real-time exploration of spatiotemporal datasets. *TVCG* (2013).
[24] X. Liu, K. Tang, J. Hancock, J. Han, M. Song, R. Xu, and B. Pokorny. 2013. A Text Cube Approach to Human Social and Cultural Behavior in the Twitter Stream. *SBP* (2013).
[25] A. Magdy, L. Abdelhafeez, Y. Kang, E. Ong, and M.F. Mokbel. 2020. Microblogs data management: a survey. *The VLDB Journal* (2020).
[26] C. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. Bethard, and D. McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. *ACL* (2014).
[27] R. Othman, R. Belkaroui, and R. Faiz. 2016. Customer Opinion Summarization Based on Twitter Conversations. *WIMS* (2016).
[28] B. Pat and Y. Kanza. 2017. Where's Waldo?: Geosocial Search over Myriad Geotagged Posts. *SIGSPATIAL* (2017).
[29] J. M. Pérez-Martínez, R. Berlanga-Llavori, M. J. Aramburu-Cabo, and T. B. Pedersen. 2008. Contextualizing Data Warehouses with Documents. *DSS* (2008).
[30] F. Ravat, O. Teste, R. Tournier, and G. Zurfluh. 2008. Top_keyword: An aggregation function for textual document OLAP. *DaWaK* (2008).
[31] S. Rivest, Y. Bédard, and P. Marchand. 2001. Toward better support for spatial decision making: defining the characteristics of spatial on-line analytical processing (SOLAP). *Geomatica* (2001).
[32] J. Sankaranarayanan, H. Samet, B. E. Teitler, M. D. Lieberman, and J. Sperling. 2009. TwitterStand: News in Tweets. *SIGSPATIAL* (2009).
[33] A. Skovsgaard, D. Sidlauskas, and C. S. Jensen. 2014. Scalable top-k spatio-temporal term querying. *ICDE* (2014).
[34] M. Walther and M. Kaisser. 2013. Geo-spatial Event Detection in Twitter Stream. *ECIR* (2013).
[35] S. Wang, J. Cao, and P. Yu. 2020. Deep learning for spatio-temporal data mining: A survey. *TKDE* (2020).
[36] D. Yu, D. Xu, D. Wang, and Z. Ni. 2019. Hierarchical Topic Modeling of Twitter Data for Online Analytical Processing. *IEEE Access* (2019).
[37] C. Zhang and J. Han. 2019. Multidimensional Mining of Massive Text Data. *DMKD* (2019).
[38] D. Zhang, C. X. Zhai, J. Han, A. Srivastava, and N. Oza. 2009. Topic Modeling for OLAP on Multidimensional Text Databases. *Stat. Anal. Data Min.* (2009).
[39] K. Zhao, L. Chen, and G. Cong. 2016. Topic Exploration in Spatio-Temporal Document Collections. *SIGMOD* (2016).