Approximate Semantic Matching of Events for the Internet of Things

SOULEIMAN HASAN and EDWARD CURRY, National University of Ireland, Galway

2

Event processing follows a decoupled model of interaction in space, time, and synchronization. However, another dimension of semantic coupling also exists and poses a challenge to the scalability of event processing systems in highly semantically heterogeneous and dynamic environments such as the Internet of Things (IoT). Current state-of-the-art approaches of content-based and concept-based event systems require a significant agreement between event producers and consumers on event schema or an external conceptual model of event semantics. Thus, they do not address the semantic coupling issue. This article proposes an approach where participants only agree on a distributional statistical model of semantics represented in a corpus of text to derive semantic similarity and relatedness. It also proposes an approximate model for relaxing the semantic coupling dimension via an approximation-enabled rule language and an approximate event matcher. The model is formalized as an ensemble of semantic and top-k matchers along with a probability model for uncertainty management. The model has been empirically validated on large sets of events and subscriptions synthesized from real-world smart city and energy management systems. Experiments show that the proposed model achieves more than 95% F_1Score of effectiveness and thousands of events/sec of throughput for medium degrees of approximation while not requiring users to have complete prior knowledge of event semantics. In semantically loosely-coupled environments, one approximate subscription can compensate for hundreds of exact subscriptions to cover all possibilities in environments which require complete prior knowledge of event semantics. Results indicate that approximate semantic event processing could play a promising role in the IoT middleware layer.

Categories and Subject Descriptors: C.2.5 [**Computer-Communication Networks**]: Local and Wide-Area Networks—*Internet*; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Information filtering*; H.3.4 [**Information Storage and Retrieval**]: Systems and Software—*Distributed systems*

General Terms: Algorithms, Experimentation, Performance, Languages

Additional Key Words and Phrases: Approximate matching, distributional semantics, event processing, Internet of Things, semantic matching, uncertainty

ACM Reference Format:
Souleiman Hasan and Edward Curry, 2014. Approximate semantic matching of events for the internet of things. ACM Trans. Internet Technol. 14, 1, Article 2 (July 2014), 23 pages.
DOI: http://dx.doi.org/10.1145/2633684

1. INTRODUCTION

The Internet of Things (IoT) builds upon the success story of Internet technologies and the Web in order to connect physical things to the world wide Internet [Atzori et al. 2010]. It is estimated that 50 billion devices will be connected to the Internet by 2020 [OECD 2012]. In recent years, there has been an increasing number of deployments of devices connected to the Internet via IoT protocols such as the Constrained Application

This work has been supported in part by Science Foundation Ireland under Grant Number SFI/12/RC/2289 and conducted in the Insight Centre for Data Analytics at the National University of Ireland, Galway.
Authors' addresses: Souleiman Hasan (corresponding author), National University of Ireland, Galway; email: souleiman.hasan@insight-centre.org; Edward Curry, National University of Ireland, Galway; email: edcurry@acm.org.

DOI: http://dx.doi.org/10.1145/2633684

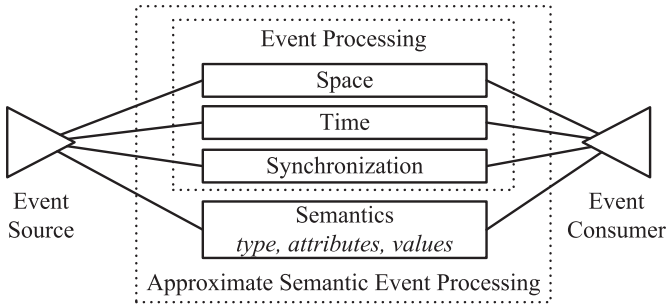

Fig. 1. Decoupling dimensions.

Protocol (CoAP). IoT projects such as the SmartSantander smart city project have deployed tens of thousands of Internet-connected sensor devices in large cities among Europe [Sanchez et al. 2011]. The sensing capabilities of these devices are wide ranging, including solar radiation, wind speed and direction, temperature, water flow, noise, traffic, public transport, rainfall, parking, and others.

Connecting physical objects, or things, to the Internet would enable a plethora of applications, such as assisted driving, monitoring environmental parameters, augmented reality, smart and comfortable homes, etc. [Atzori et al. 2010]. A basic requirement for realizing the IoT is an infrastructure of communication solutions and standards, such as IPv6 over Low power Wireless Personal Area Networks (6LoWPAN) and CoAP protocols by the Internet Engineering Task Force (IETF) [Atzori et al. 2010]. There is also a need for middleware that can abstract the application developers from the underlying technologies, which is crucial to the adoption and evolution of IoT applications [Atzori et al. 2010]. Event-based technology has played an important role in the middleware space, specifically in Enterprise Application Integration (EAI) [Hohpe and Woolf 2004], as an enabler for building distributed systems.

Event-based technologies are based on a loosely-coupled interaction model which supports scalability. Nevertheless, they assume a high level of semantic agreement between event producers and consumers. This might be challenging for largely heterogeneous environments, such as IoT smart buildings and cities, due to difficulties in establishing such agreements. This article tackles this aspect and proposes an approximate matching model for single events. It could be suitable in many scenarios, particularly when partial agreements exist, such as on a measurement unit of interest while leaving most of an event's payload to be approximated by the matcher. We think that there is a need to research these event-based technologies to contribute, among other technologies, in playing the middleware role and enabling applications of the IoT.

1.1. Background

In the event-based paradigm, event sources fire instantaneous and atomic information items called events. Event consumers use rules with condition and action parts to detect events and react to them. Events are the only means of interaction between sources and consumers. This results in decoupling the production and consumption of events and thus increasing scalability by "removing explicit dependencies between the interacting participants" [Eugster et al. 2003]. Event-based systems decouple participants on three dimensions, as shown in Figure 1.

—*Space decoupling* suggests that participants do not need to know each other.
—*Time decoupling* means that participants do not need to be active at the same time.

Table I. Approaches to Semantic Coupling

	Content-Based	Concept-Based	Approximate Semantic Matching
Matching	exact string matching	boolean semantic matching	approximate semantic matching
Semantic Coupling	term-level full agreement	concept-level shared agreement	loose agreement
Semantics	not explicit	top-down ontology-based	statistical model based on distributional semantics
Effectiveness (F_1Score)	100%	depends on the domains and number of concept models	depends on the corpus
Cost	defining a large number of rules	establishing shared agreement on ontologies	minimal agreement on a large textual corpus
Efficiency (throughput)	high	medium to high, refer to Section 1.2.2	medium to high, refer to Section 6.4.4

—*Synchronization decoupling* suggests that event producers and consumers are not blocked while producing or consuming events [Eugster et al. 2003].

Nevertheless, event-based systems can be at the same time tightly coupled by the semantics of exchanged events. Traditional deployments of event systems assume a mutual agreement on event types, attributes, and values, which forms an explicit dependency between participants. For example, if a smart city event source marks an event with the type *'parking space occupied'*, all event consumers of this event would have to use this exact event type in their rules. A new event consumer to the system cannot use a rule with the term *'garage spot occupied'* to handle such events. Within environments with a high level of semantic heterogeneity, the definition and maintenance of all possible rules becomes difficult for users and might not be feasible.

Events can be represented in various data models, such as ordered tuples, attribute-value records, hierarchical structured and semi-structured records, and graph data models [Mühl et al. 2006; Wang et al. 2004]. Attribute-value events are simple and widely used. They may also be used to convey other types via some adaptation, such as the use of dotted naming schemes for hierarchical models. Thus we scope the rest of this article to the attribute-value-based event model.

Event semantic coupling has three common dimensions resulting from mutual agreement on *event types*, *event attributes*, and *event values*. Thus, one core challenge that needs to be tackled in order to prepare event processing systems for the Internet of Things is the relaxation of semantic coupling in the paradigm. This challenge forms the main theme and high-level requirement discussed in this article.

1.2. Current Approaches and Proposed Model

The following categories of approaches can be recognized, as shown in Table I.

1.2.1. Content-Based Approach. Event sources and consumers use the same event types, attributes, and values without any extra description of meaning external to the rules and events. This is the case assumed in traditional content-based publish/subscribe and event processing systems, such as SIENA [Carzaniga et al. 2000], where the matcher performs string exact comparison between terms. The approach has high semantic coupling between parties and works quite well in environments with a low level of data heterogeneity. However, it is limited as for as scaling out to more heterogeneous environments due to the effort required to keep the agreement on shared schemata and to develop a large number of rules according to such an agreement.

1.2.2. Concept-Based Approach. In this category, participants can use different terms and values and still expect matchers to be able to match them properly thanks to an explicit knowledge representation that encodes semantic relationships between terms. Example knowledge representations are thesauri and ontologies which describe the meaning of each concept and its properties and relationships with other concepts. Building and agreeing upon such a knowledge representation suggests an explicit dependency between parties, not directly but via the conceptual model. Thus, relatively high semantic coupling and effort for management and maintenance are still required, as the agreement has to happen on the level of each individual concept.

Given a knowledge representation, a traditional event processing system can be leveraged by translating incoming events to all possible synonymous and semantically related events before matching, such as in S-TOPSS [Petrovic et al. 2003]. The translation process in real time and the large number of translated events can significantly decrease the throughput of the system. Rewriting and expansion of rules or subscriptions based on the conceptual model can be used instead, as in Zeng and Lei [2004]. However, a significantly large and possibly exponential number of possible rules may result in low throughput, as shown in Section 6. Nonetheless, these two approaches hold the merit of possibly sitting on top of currently existing systems. On the other hand, because underlying systems follow a Boolean model of matching, the resulting precision and recall might be affected due to the lack of inherent ranking of results.

Besides, matchers can be redesigned to leverage an index based on an ontology and follow a Boolean matching model, as in Wang et al. [2004], or index for continuous querie as, in Le-Phuoc et al. [2011]. These approaches can enhance throughput by delaying the decision of semantic matching until the actual time of matching, thus reducing the space of possible comparisons. Nonetheless, tying the matcher to a specific conceptual model limits its re-usability with different models of meaning.

1.2.3. Proposed Approximate Semantic Matching Approach. In the absence of an agreement on event schema or a conceptual model, participants may loosely agree on topics represented in large corpora of texts. Such corpora can be used to construct distributional models of meaning to derive semantic similarity and relatedness based on terms occurrence in similar contexts in the corpora [Harris 1954]. Such an approach provides a loose semantically-coupled paradigm, and distributional models can be built automatically, as opposed to manually-created knowledge representations. Distributional-based semantic similarity presents a promising compromise to loose semantic coupling and at the same time have acceptable levels of effectiveness and efficiency. In a previous work [Hasan et al. 2012], we proposed and set the main lines for the approximate semantic event processing approach. This article extends our previous work with a formal framework, empirical validation for time efficiency, and insights on several aspects of the proposed model, such as the effect of the degree of approximation on the model.

1.3. Contributions

The contributions of this article are manifold.

—We present an effective and efficient approximate single-event processing model for addressing semantic coupling in the Internet of Things.
—We show a formal framework for the proposed model based on an ensemble of semantic and top-k matchers in addition to a probability model for uncertainty management.
—We provide an efficient algorithm for finding top-k matchings based on evolving Pareto frontier in a vector space.
—We give an evaluation framework based on synthetic event loads and approximate subscriptions from real-world IoT deployments.

The rest of this article is organized as follows. Section 2 describes a motivational scenario. Section 3 states the requirements and questions tackled. The proposed model is discussed in Sections 4 and 5. Section 6 details the evaluation methodology and results. Related work is discussed in Section 7. Section 8 discusses the implications of the proposed model for future work, and Section 9 concludes.

2. MOTIVATIONAL SCENARIO

Alice is a sustainability officer in a large corporation in the electronics industry. The organization has many offices and facilities all over its city. Alice's job is to ensure that the company sticks to its Corporate Social Responsibility (CSR) programs, such as saving energy and lowering its overall CO_2 emissions. Most of the company's buildings are equipped with sensor nodes for energy consumption, temperature, and other environmental parameters.

Alice wants to set up a rule that notifies her when an excessive energy consumption scenario in public spaces of the buildings is detected. The intended alert should fire when "energy consumption from heating public halls of the buildings increases." Such a scenario may happen when employees tend to open windows if it is warm despite the fact that the heater is still turned on. This rule can be expressed using an Event Processing Language (EPL) such as Esper's language [EsperTech 2013] as follows.

pattern [**every** a=BuildingsEvents(a.type= 'heater energy consumption increased'
and a.location='public hall')].

While the sources of the required information are available from the buildings' IoT nodes, the semantics of the events differ from one building to another due to different manufacturers of sensors. For instance, events containing terms such as *'energy consumption'*, *'energy usage'*, and *'power consumption'* to refer to the same thing.

Alice is not interested in an exact number of such behavioral patterns but rather in a rough estimate which helps her take action. Alice asks the IT department to realize the intended detection scenario. The IT department reports that the scenario requires a large set of rules, such as the preceding one, to be deployed on an existing event processing engine with all possible variations of semantics in order to cover the semantic heterogeneity that exists. These rules take time to define and also to maintain when the environment or the requirements change, such as adding new sensors.

3. REQUIREMENTS AND RESEARCH QUESTIONS

The following user requirements for an IoT event processing engine are identified.

—*R1.* Usability of the event processing system by nontechnical users
—*R2.* Low cost and short time for integrating and accessing IoT devices
—*R3.* Near real-time and effective event processing of IoT events

Those user requirements can be elaborated into the following high-level research questions scoped to single-event processing.

—*Q1.* How can loose semantic coupling between event sources and consumers be achieved?
—*Q2.* How can effective and efficient single-event processing with a high detection rate and throughput within a loosely semantically coupled paradigm be achieved?

This article focuses the discourse on single-event matching, although impacts on related aspects, such as complex event processing, are also discussed.

4. APPROXIMATE EVENT PROCESSING MODEL

With respect to the requirements and questions in Section 3, we propose a loosely semantically-coupled model. The proposed model requires the participants to agree on a topic or set of topics which are represented as a large corpus of text. The corpus is then used to build a distributional semantic model for deriving semantic similarity and relatedness. The model also introduces the *tilde* $\sim$ semantic approximation operator to the event processing language. For example, a subscription to energy events, such as the one required in Section 2, can be expressed as {**type**= heater energy consumption increased $\sim$} to let the engine match events of the mentioned type or any other type semantically related to it. The proposed model is realized based on:

—the use of distributional semantics relatedness measures, such as the Wikipedia-based Explicit Semantic Analysis (esa), to parametrize the *tilde* $\sim$ operator;
—a matching model rooted in uncertain schema matching;
—a probability model for uncertainty management.

4.1. Distributional Semantics as a Loosely-Coupled Semantic Model

Distributional semantics is based on the hypothesis that similar and related words appear in similar contexts [Harris 1954]. Distributional models are quite useful for the task of assessing semantic similarity and relatedness between terms. A semantic measure is a function *sm* that quantifies the similarity/relatedness between two terms. Typically *sm* has its values in $\mathbb{R}$. Distributional models can be constructed automatically from statistical cooccurrence of words in a corpus of documents, for example, the measure based on the Explicit Semantic Analysis (esa) of the Wikipedia corpus. The formalism of such models as a vector space provides a computationally efficient framework for calculating similarity scores. Thus, they represent a good fit for the requirements of loose semantic coupling and real-time performance for event-based systems.

The IoT event cloud would include events from various domains, which suggests that domain-agnostic measures have a potential for IoT. We scope this article to the domain-agnostic distributional semantic measure *esa* [Gabrilovich and Markovitch 2007], constructed from the Wikipedia corpus as of 2013[1], because of its relative ease of construction, as it is based on statistical analysis of unstructured document corpus. However, the model is generic and suitable for other measures as well.

In a nutshell, Wikipedia-based *esa* builds an index of words based on the Wikipedia articles they appear in. A word becomes a vector of Wikipedia titles, and the more common titles that exist between two words, the more related the words are. For example, *esa('parking', 'garage') > esa('parking', 'energy')*, as the formers appear frequently in common articles. We assume that semantic measures are external services to the event engine and that they are constructed independently. This assumption simplifies the interface between the event engine and the service and makes the embedding of different services relatively easy.

4.2. Event Flow Model

The event processing engine plays a key role of filtering and making sense out of the event cloud. Cugola and Margara present an abstract functional model for information flow processing systems [2012]. The core components of an event engine in their model are the event *receiver*, *decider*, *producer*, and event *forwarder*. Event *sources*, *consumers*, and *users* interact with the engine through protocols and condition/action *Rules*.

[1] http://en.wikipedia.org/wiki/Wikipedia:Database_download

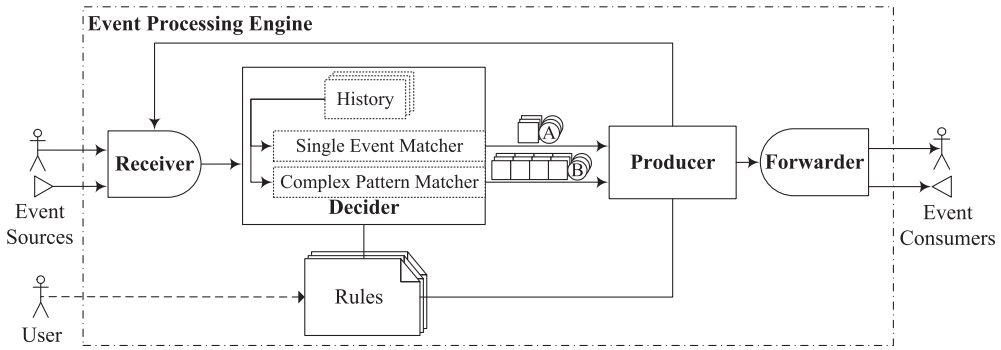

Fig. 2. Event flow model (adapted from [Cugola and Margara 2012]).

Figure 2 presents an elaboration of Cugola and Margara's model. Event sources which can be human, software, or hardware agents creating events. These events propagate to the engine and are received by the receiver, which sends the events to the decider. The decider is responsible for detection of conditions or patterns that hold in single events or in a set of events according to the condition parts of the rules registered in the engine. When a condition is detected in the decider, the participating events which caused the detection are propagated along with the condition to the producer, refer to A and B in Figure 2.

The producer generates an event as dictated by the action parts of the rules whose conditions or patterns are detected with possibly binding of placeholders with actual values from events. The generated events may feed back the receiver and/or propagate to the forwarder which sends them to the external event consumers which may be human agents, software applications such as user interfaces, or hardware agents. The decider keeps in its working memory a history of events which may be eligible to trigger a detection. The *single event matcher* detects only single events which match some filtering conditions, while the *complex pattern matcher* detects a pattern of events, such as the sequence of two or more events which have passed single-event matching.

Our proposed model extends the event processing engine as follows.

—Rules are equipped with the *tilde* $\sim$ semantic approximation operator. Rules which consist only of a detection part for single events are called *subscriptions* throughout the rest of this article.
—The single event matcher is equipped with matching and mapping algorithms to detect events semantically relevant to approximate subscriptions. The single event matcher works in two modes
  —top-1, which forwards the best mapping between an event and a subscription to the consumers;
  —top-k, which results in a list of top-k possible mappings between an event and a subscription. The top-k mappings of various events to various subscriptions go to the complex pattern matcher.
—The complex pattern matcher performs a probabilistic reasoning to deduce the probabilities of occurrences of the derived events in the action parts of the complex rules.

The focus of this article is on the single event matcher subcomponent of the decider.

4.3. Event Model

The event model used in this work is an attribute-value model, but the discussion is as relevant to other models, such as hierarchical or graph-based event models. Each

event is a set of tuples. Each tuple consists of an attribute-value pair. Example 4.1 represents an event complying to the model.

Example 4.1 (*Increased Energy Consumption Event*). {**type**: increased energy consumption event, **measurement unit**: kilowatt per hour, **device**: computer, **desk**: desk 112c, **office**: room 112, **floor**: ground floor, **zone**: building, **city**: Galway, **country**: Ireland, **continent**: Europe}.

The formal definition of the event model is as follows: Let E be the set of all events and let A and V be the sets of possible attributes and values, respectively. Let T be the set of possible attribute-value pairs, that is, tuples, such that $T = \{(a, v) : a \in A \wedge v \in V\}$. We define two functions $Attribute : T \rightarrow A$ and $Value : T \rightarrow V$, which give the attribute and value, respectively, when applied to a tuple such that $Attribute(a, v) = a$ and $Value(a, v) = v$. An event $e \in E$ is a set of tuples where no two distinct tuples can have the same attribute, as in Equation (1).

$$e = \{t : t \in T \wedge \forall t_1, t_2 \in e, t_1 \neq t_2 \Rightarrow Attribute(t_1) \neq Attribute(t_2)\}. \quad (1)$$

4.4. Language Model

Subscriptions follow a conjunctive query form of attribute-value predicates. Each predicate uses the equality operator to signify exact equality or approximate equality when indicated. Other Boolean and numeric operators such as $!=$, $>$, and $<$ are kept out of the language for the sake of simplicity and for focusing the discourse on semantic matching. Nonetheless, the model can be extended to encompass such operators, as discussed in Section 5.5.

Each predicate consists of an attribute, a value, and specifications of the semantic approximation for the attribute and the value. The most notable feature of the language is the *tilde* $\sim$ operator, which helps specify the approximation for an attribute/value when it follows it. The *tilde* $\sim$ operator also helps specify optionally the semantic measure to be used for approximation, as shown in Example 4.2.

Example 4.2 (*Approximate Subscription*). {**type** = increased energy consumption event, **device** = laptop$\sim$, **room**$\sim$esa = room 112}.

The author of the subscription in Example 4.2 is interested in an event of exactly the type *'increased energy consumption event'*. The subscription specifies that the device can be *'laptop'* or something related semantically to *'laptop'* with no specification of what semantic measure to use, meaning that the default should be used. The subscription also states that the event room must be *'room 112'*, however, it states that the attribute *'room'* itself can be semantically relaxed using the *esa* semantic measure.

The formal definition of the language model is as follows: Let S be the set of subscriptions and let A and V be the sets of possible attributes and values, respectively, which can be used in a subscription. Typically there are no restrictions on A or V, and the user is free to use any term or combination of terms. Let SM be the set of all possible semantic relatedness measures available for approximate subscriptions. Each predicate is a sextuple which consists of the attribute, the value, whether or not the attribute/value are approximate, and the semantic measure to relax the attribute/value if applicable. Let P be the set of possible predicates. Thus, P is a subset of a Cartesian product, as shown in Equation (2).

$$P = \{p : p = (a, v, app_a, app_v, sem_a, sem_v) \in A \times V \times \{0, 1\} \times \{0, 1\} \times SM \times SM\}. \quad (2)$$

A subscription $s \in S$ is a set of predicates such that $s = \{p : p \in P\}$. We define two functions $Attribute : P \rightarrow A$ and $Value : P \rightarrow V$, which give the attribute and value, respectively, when applied to a predicate. We define a Boolean function $App_A^{\sim}$:

$P \rightarrow \{0, 1\}$, which specifies if the attribute of a predicate $p \in P$ must be approximated if $App_A^{\sim}(p) = app_a = 1$. We define a function $Sem_A^{\sim} : P \rightarrow SM$, which specifies for a predicate $p \in P$ the semantic measure $Sem_A^{\sim}(p) = sem_a$ to be used to approximate its attribute if the predicate is approximated, that is, if $App_A^{\sim}(p) = 1$. We define functions $App_V^{\sim}$ and $Sem_V^{\sim}$ for value approximation in a similar way. An exact subscription is a special case of approximate subscriptions where all attributes and values are not approximated.

5. MATCHING

Given an approximate subscription $s \in S$ and an event $e \in E$, an approximate semantic single-event matcher $\mathcal{M}$ decides on the semantic relevance between s and e. The relevance results from semantic mapping between attribute-value predicates of s and attribute-value tuples of e. Example 5.1 shows a possible mapping between the event in Example 4.1 and the approximate subscription in Example 4.2.

Example 5.1. $\sigma = \{$(**type** = **increased energy consumption event** $\leftrightarrow$ type: increased energy consumption event), (**device** = **laptop**$\sim$ *esa* $\leftrightarrow$ device:computer), (**room**$\sim$ *esa* = **room 112** $\leftrightarrow$ office: room 112)$\}$.

$\mathcal{M}$ works in two modes: the top-1 mode which decides on the most probable mapping between s and e, and the top-k mode which decides on the top-k probable mappings to be used later for complex event processing. It has been shown [Gal 2006] that producing the top-k mappings increases the chance of hitting the correct mapping due to the statistical monotonicity principle which roughly states that mappings with higher similarities tend to have higher precisions but with a statistical distribution such that a mapping with a slightly smaller similarity can have a better precision than that of higher similarity [Gal 2006]. Uncertain mapping between predicates and tuples is inherent in both matching modes with probabilities being the final outcome.

The formal definition of the matching model is as follows: Let $C = s \times e$ be the set of all possible correspondences between the predicates of s and the tuples of e. $\forall c = (p, t) \in C \Rightarrow p \in s \wedge t \in e$. $\Sigma = 2^C$ is the power set of C and represents all the possible mappings between s and e. Let $\Gamma : \Sigma \rightarrow \{0, 1\}$ be a Boolean constraint function which defines the validity of a mapping $\sigma \in \Sigma$. We adopt in this work an $n : 1$ cardinality constraint function which allows every predicate to be mapped to one and only one event tuple. We denote the set of all valid mappings according to Γ as Σ_Γ. There are exactly n correspondences in any valid mapping σ, where n is the number of predicates in the subscription s.

For any valid mapping $\sigma \in \Sigma_\Gamma$, there exists a probability function which quantifies the probability of every predicate-tuple correspondence $(p, t) \in \sigma$, such as (**device** = **laptop**$\sim$ *esa* $\leftrightarrow$ device:computer). The probability of (p, t) is denoted as $p_{(p,t)}$, where $p_{(p,t)} \in [0, 1]$. The probabilities $p_{(p,t)}$ form a probability space $\mathcal{P}_\sigma$ over all $(p, t) \in \sigma$, as shown in Equation (3).

$$\sum_{(p,t)\in\sigma} p_{(p,t)} = 1. \tag{3}$$

For any valid mapping $\sigma \in \Sigma_\Gamma$, there exists a probability function which quantifies the probability of the overall mapping σ among other possible mappings. The probability of σ is denoted as p_σ, where $p_\sigma \in [0, 1]$. To realize a matcher such as the matcher $\mathcal{M}$ previously described, we propose an ensemble of matchers, illustrated in Figure 3.

5.1. First-Line Matchers and Similarity Matrices

First-line matchers operate on actual attributes and values of s and e and output similarity matrices according to the semantic measures $sm \in SM$ in s. A similarity matrix

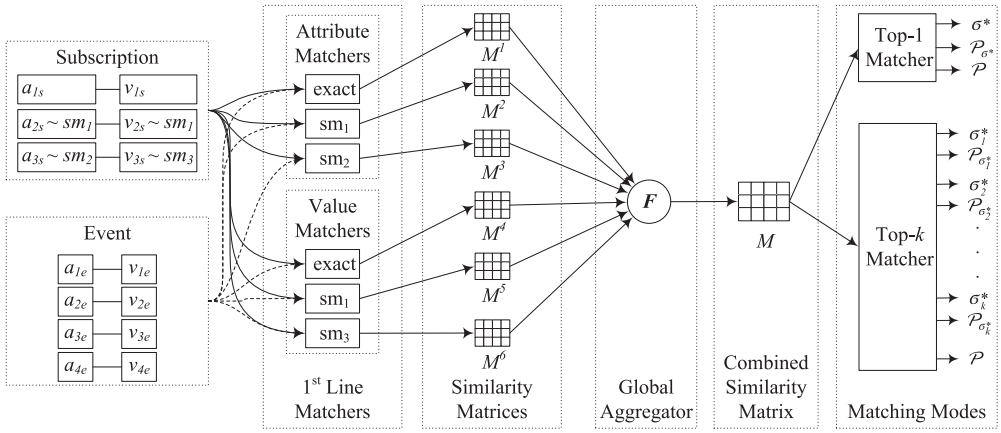

Fig. 3. Matcher model.

M^l is an $n \times m$ matrix, where n is the number of predicates in s and m is the number of tuples in e. Each element $M^l_{i,j}$ of M^l represents the degree of similarity between predicate $p_i \in s$ and tuple $t_j \in e$ according to the matcher l. Typically, $M^l_{i,j} \in \mathbb{R}$. For instance, the cell $M^{l_1}_{i,j}$ of the correspondence (**device** = **laptop**$\sim esa \leftrightarrow$ device:computer) would be assigned the value 1 by the matcher l_1 responsible for attribute exact matching. Another cell $M^{l_2}_{i,j}$ in another matrix M^{l_2} would be assigned a value <1 by the matcher l_2 responsible for value approximate matching.

There are two sets of first-line matchers: matchers which operate on the attributes of predicates/tuples and those which operate on values. There is an exact matcher for attributes and an exact matcher for values. These exact matchers handle the predicates' attributes/values which do not have any approximation specification and ignore the rest. An exact matcher operates on attributes or values and produces a Boolean similarity matrix, that is, $M^{exact}_{i,j} \in \{0, 1\}$. Let the matchers labeled exa and exv be the attributes and values exact matchers, respectively. Let $p_i \in s, t_j \in e$, Equation (4) shows how the attributes exact matcher assigns similarities. The same applies to the values' exact matcher.

$$M^{exa}_{i,j} = \begin{cases} 0, & \text{if } App^{\sim}_{A}(p_i) = 0 \wedge Attribute(p_i) \neq Attribute(t_j), \\ 1, & \text{otherwise.} \end{cases} \tag{4}$$

The remaining first-line matchers are approximate matchers, each of which uses one of the semantic measures used in the subscription. An approximate first-line matcher handles the predicates' attributes/values, which are relaxed by its corresponding semantic measure, and ignores the rest. It operates on attributes or values and produces a similarity matrix, as shown in Equation (5), which explains the behavior of an approximate attribute matcher l which corresponds to a semantic measure sm. The same applies to values' approximate matchers. Let $p_i \in s, \forall t_j \in e$:

$$M^l_{i,j} = \begin{cases} sm(Attribute(p_i), Attribute(t_j)), & \text{if } App^{\sim}_{A}(p_i) = 1 \wedge Sem^{\sim}_{A}(p_i) = sm, \\ 1, & \text{otherwise.} \end{cases} \tag{5}$$

The inner working and order of first-line matchers can be changed according to optimization strategies, as discussed in Section 5.5.2.

5.2. Global Aggregator and the Combined Similarity Matrix

The global aggregator F operates on the resulting similarity matrices from first-line matchers and produces a single combined similarity matrix M, as shown in Figure 3. For example, the correspondence (**device** = **laptop**$\sim$ *esa* $\leftrightarrow$ tool:computer) would be assigned a similarity value of 0 by the attribute-exact first-line matcher because *'device'* $\neq$ *'tool'*, and they are not approximated. It would be assigned a similarity value $x > 0$ by the value-approximate first-line matcher of *esa*, as *'laptop'* is related to *'computer'*, and they are approximated. The global aggregator shall combine the 0 similarity from the first matrix with the similarity x from the other matrix and conclude a judgment of 0 as an overall similarity according to matching semantics, as the correspondence violates it for attributes.

Matrix M represents an overall judgment on the similarity between the subscription's predicates and the event's tuples. The global aggregator chosen for the model is the element-wise matrix multiplication operator $\circ$, also called the Hadamard product, as defined in Equation (6).

$$M_{i,j} = (M^1 \circ M^2 \circ \ldots \circ M^L)_{i,j} = \prod_{l=1}^{l=L} M^l_{i,j}. \tag{6}$$

The Hadamard product is commutative and associative. It is also efficient to be implemented, as it can be computed in $O(n \times m \times L)$ time. The zero and identity elements of the Hadamard product easily extend from the familiar zero and identity elements of the multiplication operator $\times$, that is, 0 and 1. This makes it easy to pass information from the first-line matchers to the aggregator, that is, to neglect or skip a correspondence (p_i, t_j) by assigning 0 or 1 as its similarity.

5.3. Top-1 Matcher

In the top-1 matching mode, the top-1 matcher operates over the combined similarity matrix M. It produces the best mapping σ^* and the space $\mathcal{P}_{\sigma^*}$, which defines the probabilities of correspondences $c_i \in \sigma^*$. It also produces the space $\mathcal{P}$, which defines the probability that σ^* is the correct mapping between the subscription s and the event e, as illustrated in Figure 3. Given the combined similarity matrix M, the best mapping σ^* can be computed by choosing the tuple t_{j_i}, which has the maximal similarity for every predicate p_i, as shown in Equation (7).

$$\sigma^* = \left\{ (p_i, p_{j_i}) : 1 \leq i \leq n \wedge j_i = \arg\max_j (M_{i,j}) \right\}. \tag{7}$$

According to Equation (7), σ^* can be found in $O(n \times m)$ operations. σ^* contains exactly n predicate-tuple correspondences under the $n : 1$ matching semantics.

To create the probability space $\mathcal{P}_{\sigma^*}$, Equation (8) defines the n probabilities of the correspondences (subscription predicate $\leftrightarrow$ event tuple) of σ^*. That is done by dividing each (predicate $\leftrightarrow$ tuple) similarity in the mapping by the sum of all similarities so the sum of probabilities becomes 1. These probabilities can be computed in $O(n)$ time.

$$p_{i,j_i} = \frac{M_{i,j_i}}{\sum_{i=1}^{i=n} M_{i,j_i}}, 1 \leq i \leq n. \tag{8}$$

To create the probability space $\mathcal{P}$ which defines the probability that σ^* is the correct mapping between s and e among other possible mappings, there is a need to normalize the similarity matrix M among other possible matrices. The maximal possible similarity value max_{sm} of each measure $sm \in SM$ is used, as they are universal among all

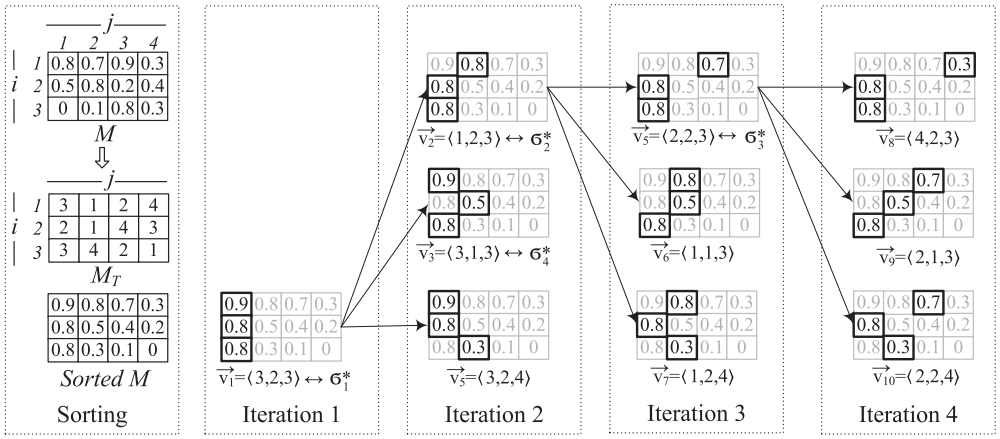

Fig. 4. Top-k by an evolving frontier algorithm.

mappings, so the probability that σ^* is correct ≤ 1. The maximum value of any element in M is $max_{SM} = \prod_{sm \in SM} max_{sm}$. Thus, the probability that σ^* is correct is defined in Equation (9) and can be computed in $O(n)$ time.

$$p_{\sigma^*} = \sum_{(p_i, t_{j_i}) \in \sigma^*} \frac{M_{i,j_i}}{n * max_{SM}}. \tag{9}$$

5.4. Top-k Matcher

In the top-k matching mode, the matcher $\mathcal{M}$ produces a ranked list of the best k mappings $\sigma_1^*, \sigma_2^*, \ldots, \sigma_k^* \in \Sigma_\Gamma$ along with the probability spaces of correspondences $\mathcal{P}_{\sigma_r^*}$ and the probability space of mappings $\mathcal{P}$, as illustrated in Figure 3. Given the combined $n \times m$ similarity matrix M between a subscription s and an event e, we propose an efficient algorithm for finding the top-k mappings σ_r^* based on an evolving Pareto frontier in a vector space, as shown in Figure 4.

Consider the set V of all n-dimensional vectors, where the components of each vector are tuples of e, that is, $V = \{\vec{v} : \vec{v} \in \{1, 2, \ldots, m\}^n\}$. Each vector $\vec{v} \in V$ encodes a valid mapping $\sigma \in \Sigma_\Gamma$, as shown in Equation (10). We denote this as $\vec{v} \leftrightarrow \sigma$.

$$\vec{v}^i = j \Leftrightarrow (p_i, t_j) \in \sigma : \sigma \in \Sigma_\Gamma, p_i \in s, t_j \in e. \tag{10}$$

For example, let an approximate subscription be {**type** = increased energy consumption event,**device** = laptop∼,**room**∼esa = room 112}. Let an event be {**type**: increased energy consumption event,**office**: room 112,**device**: computer }. Let a mapping σ = {(**type**=**increased energy consumption event** ↔ type:increased energy consumption event), (**device** = **laptop**∼ *esa* ↔ device:computer), (**room**∼ *esa* = **room 112** ↔ office: room 112)}. A vector $\vec{v} = \langle 1, 3, 2 \rangle$ corresponds to this mapping between the subscription's predicates 1, 2, and 3 to the event's tuples 1, 3, and 2, respectively.

To quantify a vector, and hence its corresponding mapping, we define an operator $\|\ldots\|_M : V \to \mathbb{R}$ given the similarity matrix M, as in Equation (11).

$$\|\vec{v}\|_M = \sum_{i=1}^{i=n} M_{i,j} : j = \vec{v}^i. \tag{11}$$

The more similarity a vector $\vec{v}$ encodes according to M, the more it becomes $\|\vec{v}\|_M$.

We say that a vector $\vec{v} \in V$ is dominated by a vector $\vec{u} \in V$ if and only if the similarity encoded by all components of $\vec{v}$ is greater than or equal to the similarity encoded by the corresponding components of $\vec{u}$ with at least one similarity to be greater than and not equal. That means that the mapping $\sigma_{\vec{v}} \leftrightarrow \vec{v}$ is better than the mapping $\sigma_{\vec{u}} \leftrightarrow \vec{u}$, and that $\|\vec{v}\|_M > \|\vec{u}\|_M$. We denote this as $\vec{v} \prec \vec{u}$. We say that a vector $\vec{u}$ directly dominates a vector $\vec{v}$ if there exists no vector $\vec{w}$ different from $\vec{u}$ and $\vec{v}$, where $\vec{v} \prec \vec{w} \prec \vec{u}$. We denote this as $\vec{v} \prec\prec \vec{u}$. For instance, $\vec{v_2} \prec\prec \vec{v_7}$ in Figure 4. In terms of similarity, this means that $\vec{v_2}$ encodes a mapping that has more similarity between predicates and tuples than the mapping encoded by $\vec{v_7}$.

The proposed algorithm which we call *Top-k by an Evolving Frontier* is depicted in Figure 4. It starts by sorting the rows in the similarity matrix M in descending order and keeping track of the new locations of tuples in a matrix called M_T. The best mapping σ_1^* is represented by the elements of the sorted matrix which have $j = 1$. This is equivalent to a vector $\vec{v_1}$ which is dominated by all other vectors of V. Vector $\vec{v_1}$ is a Pareto frontier. A Pareto frontier is a set of vectors which are dominated by all other non-searched vectors not in the frontier, and which do not dominate each other.

ALGORITHM 1: Top-k by an Evolving Frontier

Input: the similarity matrix M, the required number of mappings k
Result: Top-k mappings Σ_k

```
1  begin
2      Σ_k ⟵ Φ;
3      SortedM ⟵ SortRows(M);
4      M_T ⟵ TupleIndicesOf(SortedM);
5      Frontier ⟵ Φ ;            /* A priority queue of vectors whose key is ‖v⃗‖_M */
6      v⃗_1 ⟵ ⟨1, 1, ..., 1⟩;
7      Enqueue(v⃗_1, Frontier);
8      for 1 ≤ r ≤ k do
9          v⃗_r ⟵ Head(Frontier) ;  /* Get the best vector v⃗_r from head of the queue */
10         Σ_k ⟵ Σ_k ∪ σ*_r : v⃗_r ↔ σ_r;
11         Frontier ⟵ Frontier\v⃗_r ;          /* Remove v⃗_r from head of the queue */
12         D ⟵ {d⃗ : v⃗_r ≺≺ d⃗} ;        /* Set of n vectors directly dominating v⃗_r */
13         Enqueue(D, Frontier);
14     end
15     return Σ_k;
16 end
```

Because the dominance, as we define it, is equivalent to the quality of mapping, then the best mapping of non-searched mappings must lie on the frontier. The algorithm works in iterations, and the frontier keeps evolving. When a vector is found to correspond to the best mapping, it is removed from the frontier. All vectors which directly dominate the removed vector are candidates for search, and thus they are added to the frontier. Vectors which directly dominate a vector can be found by changing one of its n components at a time by moving one step rightwards in the rows of the sorted similarity matrix. As a result, the algorithm is able to find the top-k best mappings within k iterations, and the search space is kept to a minimum and updated with n vectors at each iteration. The correctness of the algorithm follows from the previous discussion. Algorithm 1 shows its main steps.

The frontier is presented as a priority queue of vectors $\vec{v} \in V$ on the key $\|\vec{v}\|_M$, so searching the frontier is quite efficient, as the best vector sits at the head of the queue. Sorting M in Step 3 has a time complexity of $O(n.m.log(m))$. Taking the head of the queue

in Step 9 k times has an overall complexity of $O(k)$. Generating set D in Step 12 k times has an overall complexity of $O(k.n^2)$. Enqueuing n vectors of D in Step 13 k times has an overall complexity of $O(n.log(n) + k.log(k) - k)$. That makes the overall time complexity of the proposed algorithm proportional to $O(n.m.log(m) + n.log(n) + k.n^2 + k.log(k))$. Creating the probability space $\mathcal{P}_{\sigma_r^*}$ of correspondences and the probability space $\mathcal{P}$ of mappings σ_r^* is achievable in the same way as in the top-1 mode by normalizing M, as shown in Equations (8) and (9), with the difference being that in the top-k mode, there are k probabilities $p_{\sigma_r^*}$ in $\mathcal{P}$ to be calculated.

5.5. Matcher Extensibility

This section tackles the extensibility of the matcher to include Boolean and numeric operators and to leverage common optimization strategies in event processing.

5.5.1. Boolean and Numeric Operators. The current language, as described in Section 4.4, is confined to the equality operator. However, Boolean and numeric operators such as $!=$, $<$, $\leq$, $>$, and $\geq$ can be added as exact first-line value matchers in Figure 3. Let (**temperature**> 25) be a predicate and let {**location**:first floor, **temperature**:26} be an event of two tuples. Then an exact matcher for the $>$ operator will produce a Boolean matrix which contains 0 for the cell which corresponds to the predicate and first tuple, while it contains 1 for the predicate and the second tuple. If the predicate contains an approximate attribute, that is, (**temperature** $\sim esa > 25$), then the approximate first-line matcher of attributes produces the similarities for the mappings ('temperature'$\leftrightarrow$ 'location') and ('temperature'$\leftrightarrow$ 'temperature'). This result will need to be combined then with the matrix produced by the $>$ operator first-line matcher.

5.5.2. Optimization. A distinguishing aspect of matching in event processing systems is that there are typically a large number of subscriptions S to be matched against every event e. There are two main types of optimization strategies which can be recognized in the literature: leveraging commonalities between subscriptions and changing the evaluation order of predicates [Fabret et al. 2000; Liu et al. 2008].

Leveraging commonalities is based on the observation that two subscriptions $x, y \in S$ may share one or more predicates. Thus, it is more efficient to evaluate unique atomic predicates first and then propagate the results to subscriptions. In our model in Figure 3, this can be achieved by decomposing registered subscriptions into their predicates before entering the matcher. The set of predicates then forms the entries of the similarity matrices. The top-1 and top-k matchers then aggregate the matching results according to each subscription. This affects the creation of probability spaces which shall consider only those predicates which are a part of the subscription in question. We call a matcher equipped with this strategy a *commonalities-based* matcher.

The idea of ordering the evaluation of predicates stems from interdependencies between predicates. In our proposed model, we have two distinct types of predicates: exact and approximate. If an exact predicate of a subscription evaluates to *False* then there is no need to evaluate the rest of the subscription's predicates if they do not belong to other subscriptions. Thus, we order the execution of first-line matchers by starting with the exact matchers first. Another observation is that when an approximate attribute/value of a predicate evaluates to 0, then the whole predicate evaluates to 0. We call a matcher equipped with this strategy an *order-based* matcher.

6. EVALUATION

This section describes the evaluation methodologies and the experiments' results.

Table II. Base Concepts for Effectiveness Evaluation

	Ground Truth Relevant Events	Ground Truth Irrelevant Events
Matcher Relevant Events	TP (True Positive)	FP (False Positive)
Matcher Irrelevant Events	FN (False Negative)	TN (True Negative)

6.1. Evaluation Metrics

Evaluation metrics can be classified into two categories: effectiveness and efficiency metrics [Bellahsene et al. 2011]. Effectiveness metrics measure the quality of event matching. A fundamental requirement is the existence of a ground truth which divides events into relevant and irrelevant with respect to each approximate subscription. Table II shows the base concepts needed for evaluating effectiveness. For all these concepts to exist, the resulting events from the matcher must be divisible into two distinct sets of matcher-relevant and irrelevant events. In the case of the approximate matcher which assigns probabilities to events with respect to a subscription, the two sets can be achieved by ranking and cutting off using recall levels. *Precision*, *Recall*, and the combined F_1Score have been used for effectiveness evaluation.

Precision measures the proportion of relevant events discovered by the matcher with respect to all the discovered events such that $Precision = TP/(TP + FP)$. *Recall* measures the proportion of relevant events discovered by the matcher with respect to all the known relevant events from the ground truth such that $Recall = TP/(TP + FN)$. Precision and recall are calculated for the the whole set of subscriptions S by averaging the precision and recall achieved for all individual subscriptions, respectively. The F_1Score equally combines *Precision* and *Recall* such that $F_1Score = (2 \times Precision \times Recall)/(Precision + Recall)$. The metric used for evaluating time efficiency is the matcher *Throughput* defined as $Throughput = (Number\ of\ processed\ events)/(Time\ unit)$.

6.2. Methodology for Effectiveness Evaluation

The evaluation methodology for effectiveness is based on the methodologies of the schema matching/mapping community [Do et al. 2003]. The task of schema matching/mapping is to find the best mapping between a source schema S and a target schema T. The common evaluation methodology is based on a real-world workload of a relatively small number of schemata and manually-decided ground truth mappings for the baseline [Do et al. 2003]. However, due to the large-scale nature of the Internet of Things, it is preferable to evaluate with large sets of events and subscriptions. Thus, specifying the ground truth mappings becomes a challenge.

In recent years, there has been a trend towards synthetic evaluation [Bellahsene et al. 2011]. Two approaches can be recognized: a top-down approach and a bottom-up approach. In the top-down approach, a source schema S is used. Then, by systematically removing and transforming parts of S, it is possible to generate various target schemata and their corresponding ground truth mappings to S, as in eTuner [Lee et al. 2007]. In the bottom-up approach, pairs of relative small-source and target schemata with known ground truth mappings, are used. Systematic transformations are then applied to the schemata and the mappings to generate other pairs with corresponding ground truth mappings, as in STBenchmark [Alexe et al. 2008].

Within the context of event matching, we have approximate subscriptions and events instead of source and target schemata. Similarly to the idea in STBenchmark [Alexe et al. 2008], we start with pairs of exact subscriptions X and events E with known ground truths which are simply the result of exact matching of events to subscriptions. We then apply a semantic expansion transformation to the events and subscriptions

Table III. Sensor Capabilities

Sensor Capabilities
solar radiation, particles, speed, wind direction, wind speed, temperature, water flow, atmospheric pressure, noise, ozone, rainfall, parking, radiation par, co, ground temperature, light, no2, soil moisture tension, relative humidity, energy consumption, cpu usage, memory usage

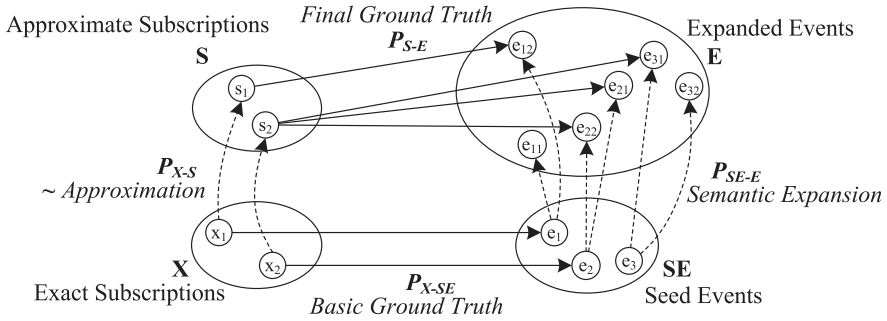

Fig. 5. Methodology for effectiveness evaluation.

based on thesaurus, similarly to the synonyms transformation based on the Merriam-Webster thesaurus [Merriam Webster's 2012] in eTuner [Lee et al. 2007]. Along with semantic expansion, the ground truth is updated accordingly. The methodology is outlined in Figure 5 and detailed in the following sections.

6.2.1. Generation of a Seed Event Set. The seed event set, SE in Figure 5, has been synthesized based on a set of IoT sensors identical to the ones deployed in the SmartSantander smart city project [Sanchez et al. 2011] and the Linked Energy Intelligence (LEI) dataspace [Curry et al. 2012]. The SmartSantander project proposes a city-scale experimental research testbed for IoT applications and services based on sensors deployed in a set of European cities. The LEI project targets sensing buildings for energy savings and management purposes. The used sensor capabilities of both sources are shown in Table III. A set of car brands from the Yahoo! directory [Yahoo! 2013] has been used to generate vehicle platforms for mobile sensors. A set of home-based appliances from the BLUED KDD dataset have been used as indoor platforms [Anderson et al. 2012]. For indoor locations, rooms from the LEI DERI Building has been used [Cyganiak 2013]. For geographical locations, the SmartSantander project locations as well as the LEI Galway city have been used. The seed event generation is done by randomly combining various attributes and values from the aforementioned datasets. A set of 165 seed events has been used to generate events for the experiments. Example 6.1 represents a resulting seed event generated at this stage.

Example 6.1 (*Seed Event*). {**type**: increased energy consumption event, **measurement unit**: kilowatt per hour, **device**: laptop, **desk**: desk 112c, **room**: room 112, **floor**: ground floor, **zone**: building, **city**: Galway, **country**: Ireland, **continent**: Europe}.

6.2.2. Generation of an Exact Subscription Set. Exact subscriptions are generated by randomly picking a number of tuples from the seed events and turning them into exact subscriptions, set X in Figure 5. Example 6.2 represents an exact subscription of length 3 generated from the seed event in Example 6.1.

Example 6.2 (*Exact Subscription*). {**type**=increased energy consumption event, **device**=laptop, **floor**=ground floor}.

6.2.3. Generation of Ground Truth for Exact Subscriptions and Seed Events. An exact matcher has been used to find the relevant and irrelevant seed events to exact subscriptions, function P_{X-SE} in Figure 5. An event is relevant to an exact subscription if every predicate in the subscription is exactly matched by at least one tuple from the event.

6.2.4. Semantic Expansion of Seed Events. The purpose of semantic expansion of seed events, transformation P_{SE-E} in Figure 5, is to generate a relatively large amount of events for evaluation, where the semantic heterogeneity property holds. Thus, the Merriam-Webster online thesaurus has been used [Merriam-Webster's 2012], as in eTuner [Lee et al. 2007]. A set of 50,000 expanded events of length up to 10 tuples has been generated starting from seed events by replacing one or more terms in an event's tuples by synonyms or related terms from the thesaurus. Example 6.3 represents an event resulting from semantically expanding the seed event in Example 6.1.

Example 6.3 (*Event Resulting from Expansion*). {**type**: power consumption fall event, **magnitude unit**: kilowatt per hour, **apparatus**: computer, **bureau**: bureau 112c, **place**: room 112, **level**: ground level, **area**: building, **metropolitan**: Galway, **homeland**: Ireland, **landmass**: Europe}.

6.2.5. Generation of an Approximate Subscription Set. An approximate subscription set, S in Figure 5, can be generated from an exact subscription set by introducing the *tilde* $\sim$ operator into one or more predicates in the exact subscription, the transformation P_{X-S} in Figure 5. This generation can also be guided by: the percentage of predicate parts to be relaxed by the *tilde* $\sim$ operator which is called the degree of approximation, and the semantic measure to be used at the attribute/value part of predicate tuples. Example 6.4 represents an approximate subscription resulting from relaxing 50% of the exact subscription in 6.2 using the *esa* semantic measure.

Example 6.4 (*Approximate Subscription*). {**type**=increased energy consumption event$\sim$ *esa*, **device**$\sim$ *esa*=laptop$\sim$ *esa*, **floor**=ground floor}.

6.2.6. Generation of Ground Truth for Approximate Subscriptions and Expanded Events. The goal of this stage is to find the resulting relevance function between approximate subscriptions and expanded events, function P_{S-E} in Figure 5. P_{S-E} is isomorphic to the basic exact relevance function P_{X-SE}, thus it is an exact relevance function. As a result, an expanded event is relevant to an approximate subscription if it exactly matches the subscription or a version of it which results from it by replacing the *tilde* $\sim$ approximated parts with related terms from the thesaurus.

6.3. Methodology for Efficiency Evaluation

Efficiency evaluation aims to position the proposed approach on the throughput scale with respect to an approach based on an exact matcher, and namely, rewriting of rules based on WordNet [Miller 1995] as a knowledge representation followed by an exact matching. Given a set of approximate subscriptions, each approximate subscription can be rewritten as a set of conjunctive statements, each of which is a set of attribute-value pairs resulting by replacing the approximate parts of a subscription with related terms from the WordNet [Miller 1995] dictionary. Example 6.5 shows a rewritten statement resulting from the approximate subscription in Example 6.4.

Example 6.5 (*Exact Rewritten Statement*). {**type** = increased energy use event, **appliance** = portable computer, **floor** = ground floor}.

6.4. Results

The following sections explain the experiments which study the top-k algorithm performance, the effects of optimization, the approximation degree, and the comparison

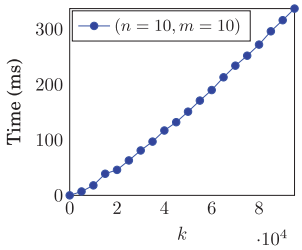

Fig. 6. Top-k time vs. k.

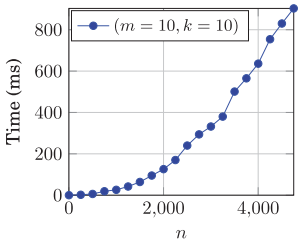

Fig. 7. Top-k time vs. n.

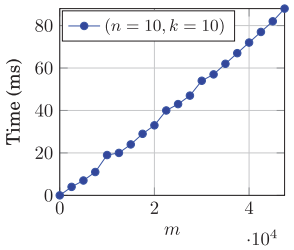

Fig. 8. Top-k time vs. m.

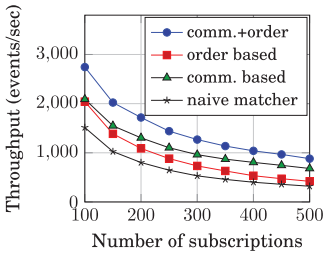

Fig. 9. Optimized matcher.

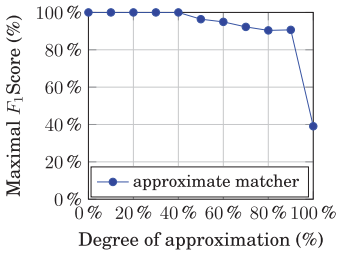

Fig. 10. Effectiveness vs. % of ∼.

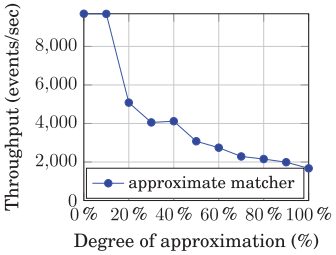

Fig. 11. Efficiency vs. % of ∼.

with the exact model. All experiments have been conducted on a Windows 7 machine, with an Intel Core i7-3520 2.90 GHz CPU and 8GB of RAM running JVM 1.7.

6.4.1. Top-k by an Evolving Frontier Algorithm Performance. Figures 6, 7, and 8 show that the algorithm time performance is polynomial and approximately linear with k and the number of event's tuples m, while it is polynomial and approximately quadratic with the number of subscription's predicates n. These findings confirm the anticipated contribution of n, m, and k to the algorithm complexity analyzed in Section 5.4. They also show that the proposed algorithm is quite efficient in finding the top-k mappings between a subscription and an event.

6.4.2. Optimization Strategies. This experiment has been conducted with Nine sets of 100–500 approximate subscriptions of 50% degree of approximation with *esa*. 43% of the predicates on average are unique in the subscriptions. Figure 9 shows that a matcher equipped with the commonalities and order optimization strategies outperforms a naive matcher for any number of subscriptions with an average optimization of 134%. The commonalities-based matcher and the order-based matcher both outperform the *naive* matcher. In the selected sample, the commonalities-based optimization outperforms the order-based one. That is caused by the relatively high number of shared predicates (about one shared predicate per each two subscriptions). Besides, 50% degree of approximation seems to leave little to do to the exact matchers for early elimination of subscriptions. The higher proportion of shared predicates and the lower degree of approximation, the better optimization that shall be achieved by commonalities-based and order-based strategies, respectively. These findings show that the proposed approximate matching model is naturally and effectively extendible by optimization strategies commonly used in event processing.

6.4.3. The Effect of the Degree of Approximation. This experiment has been conducted with 11 sets of increasing degrees of approximation of 100 approximate subscriptions with *esa*. Figure 10 shows that the matcher performs quite well with low degrees of approximation. Effectiveness slightly drops with medium degrees, 90%–100% F_1Score

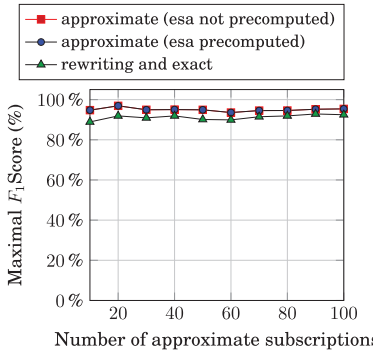

Fig. 12. Effectiveness vs. the number of subscriptions.

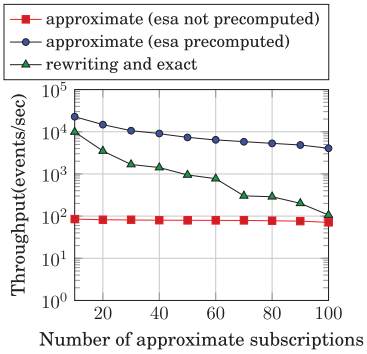

Fig. 13. Efficiency vs. the number of subscriptions.

with degrees up to 90%. It then sharply drops to 40% when the subscriptions become mostly or fully approximated, >90%, because exact predicates can better discriminate relevant events, and as they disappear in higher degrees of approximation, it becomes difficult for the matcher to decide on relevance, and F_1Score drops consequently.

Figure 11 shows that throughput decreases sharply from 9,700 events/sec to 5,100 events/sec when approximation starts to appear in subscriptions at around 20% degree of approximation. It then decreases almost linearly from 5,100 events/sec to 1,700 events/sec when the degree increases from 20% to 100%, because approximate predicates, which increasingly appear in higher degrees, are more time consuming to test than exact comparisons, and throughput decreases as a result. These results suggest that the best use cases for the proposed model are where small-to-medium degrees of approximation are expected, with the user having at least a partial knowledge of the event semantics. We think that this would be the case for many IoT applications.

6.4.4. Comparison to the Exact Model. This experiment has been conducted with Ten sets of 10–100 approximate subscriptions of 50% degree of approximation with *esa*. Figure 12 shows that the approximate matching model delivers 94%–97% matching quality, which is higher than the 89%–92% delivered by the WordNet-based rewriting approach equipped with an exact matching model. The rewriting approach outperforms the approximate model in throughput when the pairwise semantic relatedness scores are calculated at runtime. However, the approximate matching model based on precomputed *esa* scores outperforms in throughput with around 91,000 events/sec compared to around 19,100 events/sec on average.

In this experiment, around 16 million pairwise comparisons are needed: less than 100,000 of them, that is, less than 1% of them, need to be calculated just once. Precomputation is a valid assumption, as it can happen at the semantic measure side beforehand or when the system caches newly calculated scores so no recomputation is required. These results show the validity of an approximate model enhanced with a loosely coupled semantic model, such as the distributional semantic model, to achieve good effectiveness and efficiency as opposed to other approaches based on semantically coupled knowledge representations.

Finally, to achieve 100% of F_1Score and a throughput of an exact matcher, there is a need to manually write all the possible rules which are equivalent to the approximate rules. To quantify this situation, we measured how many exact rules are required to compensate for approximate rules, given that the rewriting is done with the ground truth thesaurus, which is Merriam-Webster. This showed that about 74,000 exact rules are needed to cover all events compared to a maximum of only 100 rules for the

approximate matcher. This is a nonfeasible situation in semantically heterogeneous environments. These figures show a tradeoff between effectiveness and efficiency, on the first hand, versus semantic loose coupling and ease of use, on the other hand. These results suggest that the proposed approximate event processing model is suitable for scenarios such as the IoT with a high level of semantic heterogeneity and where having complete prior semantic knowledge of events is unfeasible.

7. RELATED WORK

Related work to this article originates from the event processing and schema matching areas.

7.1. Semantic Event Processing

He et al. presented a model for Web services based on the context of things [2012]. The model uses ontologies and event-detection services to update the state of things and shows an example use of event processing as well as Web services for the Internet of Things. A-TOPSS [Liu and Jacobsen 2002] defines an approximate matching model based on fuzzy functions that specify the degree of membership between an event's value and a subscription's filter but without supporting schema approximation.

S-TOPSS [Petrovic et al. 2003] tackles schema and value semantic matching via the use of agreed-upon ontologies and a system architecture that generates events other than the original ones by replacing concepts with taxonomic concepts. S-TOPSS provides a generic architecture but no concrete model or empirical validation has been discussed. Besides, replicating events with new concepts has the downside of overwhelming the system with a large amount of events.

7.2. Uncertainty in Event Processing

A taxonomy for uncertainty in event processing proposed in Wasserkrug et al. [2006] suggests that uncertainty can be classified in two orthogonal dimensions: element uncertainty and origin uncertainty. *Element uncertainty* includes uncertainties about event *occurrence* and event *attributes*. In our model, *occurrence* uncertainty is represented by the mappings' probability space $\mathcal{P}$, and *attribute* uncertainties are represented by the correspondences probability space $\mathcal{P}_\sigma$. *Origin uncertainty* deals with the source of uncertainty which may originate from the event *source* or from event *inference*. Our model suggests another type of origin which is the matching of raw events. This uncertainty reflects the loose semantic coupling between sources and consumers. A model for complex event processing over uncertain events is proposed in Wasserkrug et al. [2008]. Resulting single-event matching with probabilities in our model can propagate to a complex event processing module.

7.3. Uncertain Schema Matching and Top-*k*

In recent years, uncertain schema-matching research has gained more attention with the realization that matchers are inherently uncertain [Gal 2011]. Statistically monotonic matchers may assign a slightly lower similarity than it should to mappings which may be of a specific precision, thus matching with top-k mappings becomes a potential solution to this [Gal 2006]. A proposed algorithm in Gal [2006] for the $n : 1$ mapping constraints has a similar complexity as our algorithm and is based on the bipartite graphs. Our algorithm follows another formalization based on an evolving Pareto frontier within a vector space, where every vector is equivalent to a valid mapping. Pareto optimum deals with the optimization of multiobjective functions and is known in database research as skyline query processing [Hose and Vlachou 2012].

8. FUTURE WORK

The uncertain model of event processing can be extended to allow various types of approximations to exist in event processing, such as matching events which lack some contextual information. In a previous work [Hasan et al. 2013], we investigated a model for improving incomplete events while matching. A suitable extension of the proposed approximate matcher which can tackle uncertainties about missing values is subject to future research. Approximate matching of events results in uncertain events, which then propagate to complex event processing. The pattern matcher should then propagate uncertainties appropriately to the derived events. Top-k mappings can also be used for complex event matching, and thus a reasoner with top-k inputs is required and potentially could deliver a better quality of inference. Future work aims also at investigating the effect of different types of mapping semantics, such as $1:1$ and $n:n$ between subscription predicates and event tuples. It also includes optimizing the time performance of the matcher through different optimization strategies.

9. CONCLUSIONS

Content-based and concept-based event processing assume a high level of semantic shared agreement between event producers and consumers. This limits scalability in highly heterogeneous environments due to the need for a large number of event processing rules and the difficulty to guarantee large scale agreements on event semantics. This article investigates this challenge to the Internet of Things (IoT). It proposes a model of approximate matching of events based on distributional semantics which requires loose semantic coupling to a large textual corpus such as Wikipedia. The model uses ensembles and top-k matching for uncertainty management. Experiments show that the proposed model achieves about 95% F_1Score of effectiveness and 1,000 events/sec of throughput in environments with a high level of semantic heterogeneity. The model can contribute to the middleware layer of IoT to support application developers and to users specifically with low-to-medium prior knowledge of event semantics.

REFERENCES

Bogdan Alexe, Wang-Chiew Tan, and Yannis Velegrakis. 2008. STBenchmark: Towards a benchmark for mapping systems. *Proc. VLDB Endow.* 1, 1 (2008), 230–244.

Kyle Anderson, Adrian Ocneanu, Diego Benitez, Derrick Carlson, Anthony Rowe, and Mario Berges. 2012. BLUED: A fully labeled public dataset for event-based non-intrusive load monitoring research. In *Proceedings of the 2nd KDD Workshop on Data Mining Applications in Sustainability (SustKDD)*.

Luigi Atzori, Antonio Iera, and Giacomo Morabito. 2010. The Internet of Things: A survey. *Comput. Netw.* 54, 15 (2010), 2787–2805.

Zohra Bellahsene, Angela Bonifati, Fabien Duchateau, and Yannis Velegrakis. 2011. On evaluating schema matching and mapping. In *Schema Matching and Mapping*, Springer, Bertin, 253–291.

Antonio Carzaniga, David S. Rosenblum, and Alexander L. Wolf. 2000. Achieving scalability and expressiveness in an Internet-scale event notification service. In *Proceedings of the 19th Annual ACM Symposium on Principles of Distributed Computing*. ACM, 219–227.

Gianpaolo Cugola and Alessandro Margara. 2012. Processing flows of information: From data stream to complex event processing. *ACM Comput. Surv.* 44, 3, Article 15 (2012).

Edward Curry, Souleiman Hasan, and Sean O'Riain. 2012. Enterprise energy management using a linked dataspace for energy intelligence. In *Proceedings of Sustainable Internet and ICT for Sustainability (SustainIT)*. IEEE, 1–6.

Richard Cyganiak. 2013. Rooms in the DERI building. http://lab.linkeddata.deri.ie/2010/deri-rooms.

Hong-Hai Do, Sergey Melnik, and Erhard Rahm. 2003. Comparison of schema matching evaluations. In *Proceedings of the Revised Papers from the NODe Web and Database-Related Workshop on Web, Web-Services, and Database Systems.* Akmal B. Chaudhri, Mario Jeckle, Erhard Rahm, and Rainer Unland (Eds.), Lecture Notes in Computer Science, vol. 2593, Springer, Berlin Heidelberg, 221–237.

EsperTech. 2013. Esper complex event processing engine. http://esper.codehaus.org/.

Patrick Th. Eugster, Pascal A. Felber, Rachid Guerraoui, and Anne-Marie Kermarrec. 2003. The many faces of publish/subscribe. *ACM Comput. Surv.* 35, 2 (2003), 114–131.

Françoise Fabret, François Llirbat, Joao Pereira, I. Rocquencourt, and Dennis Shasha. 2000. Efficient matching for content-based publish/subscribe systems. In *Proceedings of the International Conference on Cooperative Information Systems (CoopIS)*.

Evgeniy Gabrilovich and Shaul Markovitch. 2007. Computing semantic relatedness using Wikipedia-based explicit semantic analysis. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence (IJCAI'07)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, 1606–1611.

Avigdor Gal. 2006. Managing uncertainty in schema matching with top-K schema mappings. In *Journal on Data Semantics VI*, Stefano Spaccapietra, Karl Aberer, and Philippe Cudr-Mauroux (Eds.), Lecture Notes in Computer Science, vol. 4090, Springer, Berlin Heidelberg, 90–114.

Avigdor Gal. 2011. Uncertain schema matching. *Synthesis Lect. Data Manage.* 3, 1 (2011), 1–97.

Zellig S. Harris. 1954. Distributional structure. *Word* 10 (1954), 146–162.

Souleiman Hasan, Sean O'Riain, and Edward Curry. 2012. Approximate semantic matching of heterogeneous events. In *Proceedings of the 6th ACM International Conference on Distributed Event-Based Systems (DEBS'12)*. 252–263.

Souleiman Hasan, Sean O'Riain, and Edward Curry. 2013. Towards unified and native enrichment in event processing systems. In *Proceedings of the 7th ACM International Conference on Distributed Event-Based Systems (DEBS'13)*. 171–182.

Jing He, Yanchun Zhang, Guangyan Huang, and Jinli Cao. 2012. A smart Web service based on the context of things. *ACM Trans. Internet Technol.* 11, 3, Article 13 (2012).

Gregor Hohpe and Bobby Woolf. 2004. *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. Addison-Wesley Professional.

Katja Hose and Akrivi Vlachou. 2012. A survey of skyline processing in highly distributed environments. *VLDB* 21, 3 (2012), 359–384.

Danh Le-Phuoc, Minh Dao-Tran, Josiane Xavier Parreira, and Manfred Hauswirth. 2011. A native and adaptive approach for unified processing of linked streams and linked data. In *Proceedings of the 10th International Semantic Web Conference (ISWC)*. Lecture Notes in Computer Science, vol. 7031, Springer, Berlin Heidelberg, 370–388.

Yoonkyong Lee, Mayssam Sayyadian, AnHai Doan, and Arnon S. Rosenthal. 2007. eTuner: Tuning schema matching software using synthetic scenarios. *VLDB* 16, 1 (2007), 97–122.

Haifeng Liu and H.-Arno Jacobsen. 2002. A-TOPSS: A publish/subscribe system supporting approximate matching. In *Proceedings of the 28th International Conference on Very Large Data Bases (VLDB'02)*. VLDB Endowment, 1107–1110.

Zhen Liu, Srinivasan Parthasarathy, Anand Ranganathan, and Hao Yang. 2008. Near-optimal algorithms for shared filter evaluation in data stream systems. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*. ACM, 133–146.

Merriam-Webster. 2012. *Merriam-Webster's Collegiate Thesaurus*. http://www.dictionaryapi.com/products/api-collegiate-thesaurus.htm.

George A. Miller. 1995. WordNet: A lexical database for English. *Commun. ACM* 38, 11 (1995), 39–41.

Gero Mühl, Ludger Fiege, and Peter Pietzuch. 2006. *Distributed Event-Based Systems*. Vol. 1. Springer.

OECD. 2012. Machine-to-machine communications: Connecting billions of devices. http://www.oecd-ilibrary.org.

Milenko Petrovic, Ioana Burcea, and Hans-Arno Jacobsen. 2003. S-ToPSS: Semantic Toronto publish/subscribe system. In *Proceedings of the 29th International Conference on Very Large Data Bases (VLDB'03)*. VLDB Endowment, 1101–1104.

Luis Sanchez, José Antonio Galache, Veronica Gutierrez, J. M. Hernandez, J. Bernat, Alex Gluhak, and Tomás Garcia. 2011. SmartSantander: The meeting point between future Internet research and experimentation and the smart cities. In *Proceedings of the Future Network & Mobile Summit (FutureNetw)*. IEEE, 1–8.

Jinling Wang, Beihong Jin, and Jing Li. 2004. An ontology-based publish/subscribe system. In *Proceedings of the 5th ACM/IFIP/USENIX International Conference on Middleware (Middleware'04)*. Springer-Verlag, Berlin Heidelber, 232–253.

Segev Wasserkrug, Avigdor Gal, and Opher Etzion. 2006. A taxonomy and representation of sources of uncertainty in active systems. In *Proceedings of the 6th International Conference on Next Generation Information Technologies and Systems (NGITS)*. Opher Etzion, Tsvi Kuflik, and Amihai Motro (Eds.), Lecture Notes in Computer Science, vol. 4032, Springer, Berlin, 174–185.

Segev Wasserkrug, Avigdor Gal, Opher Etzion, and Yulia Turchin. 2008. Complex event processing over uncertain data. In *Proceedings of the 2nd International Conference on Distributed Event-Based Systems (DEBS'08)*. ACM, New York, NY, 253–264.

Yahoo!. 2013. Yahoo! directory: Automotive - makes and models. http://dir.yahoo.com/recreation/automotive/makes_and_models/.

Liangzhao Zeng and Hui Lei. 2004. A semantic publish/subscribe system. In *Proceedings of the IEEE International Conference on E-Commerce Technology for Dynamic E-Business*. 32–39.

Received October 2013; revised March 2014; accepted April 2014