Open-Schema Event Profiling for Massive News Corpora

Quan Yuan[1], Xiang Ren[2], Wenqi He[3], Chao Zhang[4], Xinhe Geng[4], Lifu Huang[5], Heng Ji[5], Chin-Yew Lin[6], Jiawei Han[4]
[1]Facebook Inc., Menlo Park, CA, USA
[2]Department of Computer Science, University of Southern California, Los Angeles, CA, USA
[3]Snap Inc., San Francisco, CA, USA
[4]Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL, USA
[5]Computer Science Department, Rensselaer Polytechnic Institute, Troy, NY, USA
[6]Microsoft Research, Beijing, China
[1]qyuan@fb.com [2]xiangren@usc.edu [3]maggiehwq@gmail.com
[4]{czhang82, xgeng2, hanj}@illinois.edu [5]{huangl7, jih}@rpi.edu [6]cyl@microsoft.com

ABSTRACT

With the rapid growth of online information services, a sheer volume of news data becomes available. To help people quickly digest the explosive information, we define a new problem – schema-based news event profiling – profiling events reported in open-domain news corpora, with a set of slots and slot-value pairs for each event, where the set of slots forms the schema of an event type. Such profiling not only provides readers with concise views of events, but also facilitates various applications such as information retrieval, knowledge graph construction and question answering. It is however a quite challenging task. The first challenge is to find out events and event types because they are both initially unknown. The second difficulty is the lack of pre-defined event-type schemas. Lastly, even with the schemas extracted, to generate event profiles from them is still essential yet demanding.

To address these challenges, we propose a fully automatic, unsupervised, three-step framework to obtain event profiles. First, we develop a Bayesian non-parametric model to detect events and event types by exploiting the slot expressions of the entities mentioned in news articles. Second, we propose an unsupervised embedding model for schema induction that encodes the insight: an entity may serve as the values of multiple slots in an event, but if it appears in more sentences along with the same set of more entities in the event, its slots in these sentences tend to be similar. Finally, we build event profiles by extracting slot values for each event based on the slots' expression patterns. To the best of our knowledge, this is the first work on schema-based profiling for news events. Experimental results on a large news corpus demonstrate the superior performance of our method against the state-of-the-art baselines on event detection, schema induction and event profiling.

*This work was done when the first 3 authors were at University of Illinois at Urbana-Champaign.

ACM ISBN 978-1-4503-6014-2/18/10.
https://doi.org/10.1145/3269206.3271674

ACM Reference Format:
Quan Yuan, Xiang Ren, Wenqi He, Chao Zhang, Xinhe Geng, Lifu Huang, Heng Ji, Chin-Yew Lin, Jiawei Han. 2018. Open-Schema Event Profiling for Massive News Corpora. In *The 27th ACM International Conference on Information and Knowledge Management (CIKM '18), October 22–26, 2018, Torino, Italy.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3269206.3271674

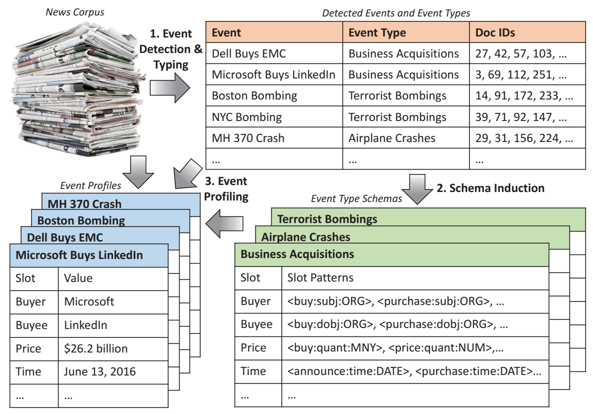
1 INTRODUCTION

The number of news articles is growing at an explosive rate. As of 2015, about 1,200 pieces of news were published daily on average by only a single new source, the Washington Post[1]. That is more than 1 piece of news was published in every 2 minutes. If we consider all news agencies around the globe, the number will be tremendous. News is the major source of how people learn about events throughout the world; however, it is virtually impossible to read through such enormous content in a very short time. As an example, investment bankers need to stay up-to-date with many essential yet trivial events so that they can make strategic decisions in time. Numerous summarization studies [11, 22, 24, 36, 37, 57] have been developed to extract or to generate representative sentences in the hope of allowing people to quickly get the gists of news articles. Nevertheless, such unstructured text is still up people to read through and is also not friendly for further analysis.

In this paper, we propose to solve this problem by generating schema-based structured profiles for news events from open-domain news corpora. A `schema` is a template that defines a specific type of events, associated with `slots` (key aspects) held by entities as `slot values` [29, 44]. For a specific event, each slot has a specific slot value (a slot may have more than one slot value, but for simplicity we only consider the single-value case). For example, the schema of the event type "Business Acquisitions" has slots such as *buyer*, *buyee*, and *price*; the value of *buyee* in the *Acquisition of LinkedIn* event is *LinkedIn*. Our goal is to extract the *slot* and *slot value* pairs to profile each event in an open-domain news corpus.

Event profiling is an important task, as the generated event profiles not only help people quickly identify the key aspects of any event without having to go through the massive news articles (*e.g.,* Figure 1), but also facilitate a number of applications, such as knowledge-base construction, information retrieval, and question answering (QA) systems. For example, based on the event profiles, a

[1]http://www.wsj.com/articles/bezos-takes-hands-on-role-at-washington-post-1450658089



Event	Event Type	Doc IDs
Dell Buys EMC	Business Acquisitions	27, 42, 57, 103, ...
Microsoft Buys LinkedIn	Business Acquisitions	3, 69, 112, 251, ...
Boston Bombing	Terrorist Bombings	14, 91, 172, 233, ...
NYC Bombing	Terrorist Bombings	39, 71, 92, 147, ...
MH 370 Crash	Airplane Crashes	29, 31, 156, 224, ...
...	...	...

Microsoft Buys LinkedIn

Slot	Value
Buyer	Microsoft
Buyee	LinkedIn
Price	$26.2 billion
Time	June 13, 2016
...	...

Business Acquisitions

Slot	Slot Patterns
Buyer	<buy:subj:ORG>, <purchase:subj:ORG>, ...
Buyee	<buy:dobj:ORG>, <purchase:dobj:ORG>, ...
Price	<buy:quant:MNY>, <price:quant:NUM>,...
Time	<announce:time:DATE>, <purchase:time:DATE>...
...	...

Figure 1: Framework Overview

QA system can easily answer the questions like "which companies have been acquired by Microsoft in the past", and "the fatality of which terrorist bombing is greater than 100 since 2005", *etc.*

However, event profiling on open-domain news corpora is challenging. This is not only because events and event types are unknown, but also due to the lack of pre-defined schemas. There are only few event-type schemas existing and they were all hand-coded by domain experts. When it comes to our task, for all possible event types in open-domain corpora, it is cost- and labor-prohibitive to hire experts to create all the schemas. Even if we already know the schemas, it is still non-trivial to extract event profiles from them.

We propose to obtain the event profiles in three steps: (1) detect events and their event types from the news corpus by clustering news articles; (2) induce schemas for each event type by discovering the schema slots—since a slot can be expressed in a number of ways (*e.g., buyee* may be an object of *buy* or a subject of *sell*), we extract and cluster the expressions from the news articles of the same event types as slot patterns for slot value extraction; and (3) extract slot values using slot patterns and profile each event by *slot* and *slot value* pairs. The framework overview is illustrated in Figure 1.

The key challenge lies in generating (*i.e.,* inducing) schemas. Existing schema induction methods can be categorized into supervised and unsupervised. The supervised methods rely on domain-specific labeling data and human guidance [10, 16, 21, 46, 51, 62, 68]. However, it is too expensive to apply these manual-labor intensive supervised methods to *massive* news corpus. The unsupervised ones deal with sentence- [1] or document- [23] level events. But in real-life open-domain news corpus, we are dealing with events reported in multiple news articles which collaboratively offer valuable, mutually complementary information for schema induction at a corpus level. Thus sentence- or document-based analysis may not be beneficial. Sentence-level studies [28] find only common arguments for events triggered by predicates or nouns within a sentence as their schemas, but such events are at too fine-grain level to summarize news events. Document-level ones treat news articles independent of each other and overlook their relations. Worse still, document-level methods are all built upon simplistic assumptions (*e.g.,* an entity can serve as the value for one slot in a news article [12, 13, 15, 44]). Our event analysis is conducted at the corpus-level [2], rather than at the sentence [1] or document-level [17]. We believe that there is no prior study on schema induction for corpus-level events.

In this paper, we develop a fully automatic unsupervised approach to schema induction. Our key insight is that an entity may be the values of multiple slots in an event, but if it appears in multiple sentences along with the same set of entities in an event, then the entity tends to have semantically similar slot expressions across these sentences. Further, with more sentences and larger same set of entities, the expressions of the entity are more similar. Take *LinkedIn* as an entity example, its slots could be *buyee* or *new services launcher* and so on in different sentences. However, if *LinkedIn* appears with other entities such as *Microsoft* and *$26.2 billion* in multiple sentences (see the example sentences in Section 4.2), these sentences are likely to talk about the same activity—the purchase, a component of the event—the acquisition. Thus, the slots of *LinkedIn* in these sentences should be the same, namely, *buyee*. Although rare in one news article, the co-occurrences are easy to be found across news articles about the same events. We encode this insight in an embedding method that captures the similarities between slot expressions based on entity co-occurrences. The similarities are then used to cluster expressions to generate slots and slot patterns.

In order to induce schemas for corpus-level events, we need to find events and their types from the news corpus. However, to the best of our knowledge, there is no integrated work on this task. In open news corpus, event and their types are both unknown, so if the accuracy in event detection improves, the accuracy in event-type detection increases, and vice versa. The studies that focus only on event detection discover events by clustering news articles based on the similar word [18, 67] or entity distributions [38, 54]. They all rely on ad-hoc methods to guess the number of events and cannot detect the prerequisite to schema induction—event types. We observe that the slot expressions of entities which are overlooked in existing studies are important to event and event type detection, *e.g.,* we can better distinguish between the news about *the LinkedIn Acquisition* and about *LinkedIn Launches Lynda 'Learning Path'* with the slot expressions of *LinkedIn*, i.e., the object of acquisition and the subject of launching new services, respectively. We propose a model that can exploit slot expressions and automatically detect a reasonable number of events and their event types in an integrated manner.

In summary, the contributions of the paper are as follows.

- We define a new problem – open-domain schema-based events profiling. It aims to generate structured representations for open-domain corpus-level news events.
- We propose the first integrated model for event and event type detections. Our model utilizes slot expressions of entities and extracts events and event types under the framework of Nonparametric Bayesian Model.
- We propose the first schema induction framework for corpus-level events. It exploits entity co-occurrence information to induce event schemas and extract slot patterns.
- We conduct extensive experiments to evaluate the effectiveness of our models in event detection and typing, schema induction, and event profiling. The results demonstrate its superior performance over the state-of-the-art baselines.

2 RELATED WORK

In this section, we review the existing studies on three related topics, namely, event extraction, event detection and schema induction.

2.1 Event Extraction

Our event profiling task is related to the problem of event extraction. A number of studies [27, 28, 30, 32, 35, 40, 42, 43, 45, 49, 56, 66] focus on extracting *atomic events* defined by the Automatic Content Extraction (ACE) program [1]. These studies extract events triggered by predicates or nouns and their arguments within a sentence, but the extracted events are too fine-grained to summarize the events at high level. Another line of work [16, 17, 34, 47, 50, 59] follows the paradigm of the Message Understanding Conferences (MUC) program [23], and extracts slot values of document-level events by classification or pattern matching. However, since an event is often reported in multiple news articles, it is important to aggregate the knowledge of multiple related articles rather than a single article to obtain good extraction results. Worse still, the extraction relies on manually created rules. These studies only consider a few types of events, *e.g.,* only 6 event types are studied in MUC 4. Unlike these studies, we aim to extract the slot values for corpus-level events, which are defined in the Topic Detection and Tracking (TDT) program [2] as a particular thing that happens at a specific time and place, along with all necessary pre-conditions and unavoidable consequences. In addition, the model should be generic enough to profile events for open-domain news corpora without supervision.

2.2 Event Detection

Our task is related to retrospective event detection [3], *i.e.,* to assign news articles in a given corpus to their respective previously unknown events. Yang *et al.* [67] propose a hierarchical agglomeration clustering (HAC) based approach, clustering news articles on words as events. Dai *et al.* first get preliminary event clusters by Affinity Propagation [18] and then perform HAC to get the final events. Li *et al.* propose a probabilistic model that considers words, time, person and location entities. None of the existing solutions can find event types — the prerequisite to schema induction. In addition, they all rely on ad-hoc methods by guessing the number of events and overlook slot expressions of entities. Our topic is also related to event detection on social media [8, 48, 52, 53, 69]. However, due to the differences in data length, quality and problem definition, these cited methods are not applicable to news data. For example, Ritter *et al.* detect events for each day from tweets, and their model requires human labeled data to extract event phrases. In contrast, we aim to detect events from newswire text, and our model has no constraints on event time and is fully unsupervised.

2.3 Schema Induction

Schemas are often hand-coded by human experts with domain knowledge [23], but manually creating schemas for large open-domain news corpus is infeasible. Early attempts on schema induction take special training resources as input [10, 16, 21], or call for human judgments [51] or bootstrapping seeds [46, 62, 68]. Some automated methods discover schema slots by extracting and clustering expressions from dependency trees [20, 60], but how to differentiate expressions with same entity types [60] or with different verbs [20] remains unsolved. Cheung *et al.* [15] propose a generative model for frame (correlated events) induction with account event transitions, but neither event causality nor evolution is our focus.

Several entity-driven models were proposed in recent years. Chambers *et al.* [13] extract domain-specific relation expressions, and cluster relations agglomeratively with the mentions of the same entities into slots. In their follow-up work [12], they employ topic models to cluster entity mentions of news articles as slots, assuming the expressions of the same entity mentions belong to the same slot. Nguyen *et al.* [44] exploit the immediate entity context to disambiguate entities and discover schema slots. Sha *et al.* [55] employ normalized cut to cluster entities into schemas and slots by exploiting entity heads, predicates and dependency paths. In summary, these methods are built based on an untenable premise that the slots of an entity are constant. Besides, they all fail to exploit data redundancy across corpus. Huang *et al.* [28] propose the first schema induction method for sentence-level events, but as detailed in Section 2.1, the sentence-level events are too fine-grained to summarize general news events. Although some studies [12, 13, 15, 44, 55] involve profiling tasks, they can only profile at the document-level. Large-scale learning of scripts and narrative schemas for frequent topics [5, 31], slot-filling for knowledge base population [4, 61] and relation discovery [7, 14, 19, 25, 41, 58, 60] are also related to our topic. The goals are different from ours, but their outputs can help boost the performance of schema induction.

3 PRELIMINARIES

Instead of dealing with raw text, we extract the following information from news articles.

Entities and Entity Labels: A news article contains entities with different entity labels (types). These entities may serve as the values of key event aspects such as who, where and when. We consider entities of labels person (*PER*), organization (*ORG*), location (*LOC*), date (*DATE*), money (*MNY*), number (*NUM*), and percent (*PCT*).

Keywords: We sort words in each news article based on TF-IDF, and use the top 5% ranked words as the set of keywords. We then augment keyword set with key phrases extracted by an existing phrase mining tool, SegPhrase [38]. Note that other keywords extraction techniques [70] can also be employed.

Slot Expressions and Patterns: The semantic role of an entity is often unveiled by its context. We convert the context into slot expression ⟨predicate:role:label⟩ ($\langle p{:}r{:}l\rangle$ for short). Predicate is a general verb or non-entity noun; role is the relation of an entity to its predicate, and label is the entity label. The following roles are considered in this paper: subject (*subj*), direct object (*dobj*), indirect object (*iobj*), location, source location (*source*), destination, path, time, and quantity (*quant*). A slot may have multiple expressions, among which the representative ones serve as the slot patterns for value extraction. We obtain slot expressions from the parsing graphs generated by Abstract Semantic Representation (AMR) [6]. Although the AMR parsing result for one sentence may not be always perfect, it can be mitigated by aggregating the results of numerous sentences.

Slot Tuples: Slot expressions, together with their entities v, contextual keywords $\boldsymbol{w}$ and time t, form slot tuples $tp = \langle\langle p{:}r{:}l\rangle, v, \boldsymbol{w}, t\rangle$. We convert each news article into a set of slot tuples, where t is defined as the publication date for simplicity.

Example: consider the sentence "Microsoft buys LinkedIn – the largest tech company acquisition" in a news article about the **event**

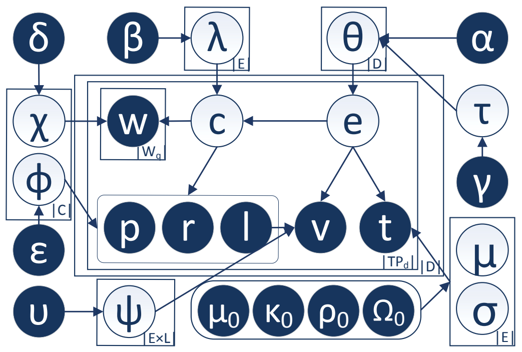


Figure 2: Graphical model for event detection and typing

Microsoft's LinkedIn purchase on June 13 2016. The **event type** is Business Acquisitions. The **entity** LinkedIn is of **entity label** ORG, and its **slot tuple** is $\langle\langle$*buy*:*dobj*:*ORG*$\rangle$, *LinkedIn*,'*tech company*', 2016-06-13$\rangle$, where $\langle$*buy*:*dobj*:*ORG*$\rangle$ is the **slot expression** and 'tech company' is a keyword. The **slot** of LinkedIn is buyee, and the **patterns** of the slot include $\langle$buy:dobj:ORG$\rangle$ and $\langle$sell:subj:ORG$\rangle$.

4 PROPOSED MODEL

This section introduces our framework for event profiling.

4.1 Event Detection and Typing

We build our probabilistic model for event detection and typing based on the following intuitions:

- Each news article has a distribution over events where each event has distributions over entities and event types. For example, a news article may be more related to the event of Microsoft's LinkedIn purchase than its Skype purchase. The former event involves entities such as Microsoft, LinkedIn, \$26.2 billion, *etc.*, which is more related to the event type "Business Acquisitions" than any other type, such as "Product Announcements".
- Each event type is related to a set of slot expressions and keywords (*e.g.,* business acquisitions often involve slot expressions $\langle$*buy*:*dobj*:*ORG*$\rangle$, $\langle$*price*:*quant*:*MNY*$\rangle$ and keywords *purchase*, *stock market*, *etc.*).
- Slot expressions determine the label of entities. Events determine the specific entities of the labels (*e.g.,* the entities of $\langle$*buy*:*dobj*:*ORG*$\rangle$ must be organizations, *e.g.,* in *the Acquisition of LinkedIn*, the entity is the organization *LinkedIn*).
- News articles on an event often appear together and form a cluster on timeline.

We model both event e and event type c as latent variables. For each slot tuple tp in document d, we sample an event e based on d's event distribution θ_d and sample an event type c based on e's event type distribution λ_e (Intuition 1). c further generates a slot expression $q = \langle$p:r:l$\rangle$ and a set of keywords $\boldsymbol{w}$ based on its multinomial distributions ϕ_c and χ_c (Intuition 2). Finally, an entity value v is drawn from multinomial distribution $\psi_{e,l}$ conditioned on event e and entity label l of q (Intuition 3); a time stamp t is sampled from e's time distribution (Intuition 4). Since the news articles often appear around the peak time of events (*i.e.,* the number of news articles increases with the event's evolution, and then decreases afterwards), we model time distribution by a Gaussian distribution $N(t|\mu, \sigma^2)$ with μ and σ^2 as the mean and the variance, respectively.

Identifying the number of events is crucial to event detection. We employ Hierarchical Dirichlet Process (HDP) [64] where there

Algorithm 1: Generative Process

```
Draw global event distribution $\tau \sim GEM(\gamma)$;
for each event type $c \in C$ do
    Draw slot expression distribution $\phi_c \sim Dir(\epsilon)$;
    Draw keyword distribution $\chi_c \sim Dir(\delta)$;
for each news article $d \in D$ do
    Draw a document event distribution $\theta_d \sim DP(\alpha\tau)$;
    for each slot tuple $tp_i \in d$ do
        Draw an event $e \sim Dir(\theta_d)$;
        if $e \notin E$ then
            Draw event type distribution $\lambda_e \sim Dir(\beta)$;
            Draw time distribution
              $\mathcal{N}(\mu_e, \sigma_e) \sim Normal\text{-}Gamma(\mu_0, \kappa_0, \rho_0, \Omega_0)$;
            for each entity label $l \in L$ do
                Draw entity distribution $\psi_{e,l} \sim Dir(\upsilon)$;
            Add $e$ to $E$;
        Draw a slot pattern $q_i \sim Dir(\phi_c)$;
        Draw an entity $v_i \sim Dir(\psi_{l_q})$;
        Draw a time $t_i \sim \mathcal{N}(\mu_e, \sigma_e)$;
        for each keyword associated with $tp_i$ do
            Draw a word $w_i \sim Dir(\psi_c)$;
```

is a global event distribution τ that is drawn from a stick-breaking process $GEM(\gamma)$. The article-specific event distribution θ_d is a variation of τ with precision α controlling how similar θ_d is to τ. If a slot tuple is less likely to be generated from existing events, a new event will be created. We assume the number of event types is given, though we can estimate it in a similar way. The graphical model is shown in Figure 2 where ψ, and $\mu_0, \kappa_0, \rho_0, \Omega_0$ are the parameters of Normal-Gamma prior; β, θ, ϵ, υ are the Dirichlet priors for λ, χ, ϕ. The generative process is presented in Algorithm 1.

4.2 Learning Slot Expression Embeddings

After obtaining events and event types, we propose a graph-based model to learn the embedding of slot expressions for each event type. The model is built based on the following two observations:

Observation 1: If one entity appears in multiple sentences along with the same set of entities in an event, then it tends to have semantically similar slot expressions across these sentences. Further, the more sentences, the larger the same set of entities, the more similar slot expressions it has.

Entities act as participants of activities. Although an entity alone may be involved in many activities, if it appears together with other entities in multiple sentences, the activity they participate together is relatively fixed. Consider the following 3 sentences:

Microsoft buys LinkedIn for \$26.2 billion.

LinkedIn is purchased by Microsoft, and the price is \$26.2 billion.

LinkedIn is going to launch online learning paths.

We observe that *LinkedIn* itself is involved in two activities, namely, purchase and new services launch. When it appears together with the entities *\$26.2B* and *Microsoft* in the first two sentences, it is more likely to participate in the same event—acquisition. Thus, the slot expressions of *LinkedIn*, *e.g.,* $\langle$buy:dobj:ORG$\rangle$ and $\langle$purchase:dobj:ORG$\rangle$, can be assumed semantically similar (which also holds for *\$26.2B* and *Microsoft*). Further, if the two sentences share more entities, *e.g.,* date *2016-06-13*, then we can be more confident in the similarity of the slot expressions of *LinkedIn* across these sentences. Unfortunately, existing studies assume that one entity is involved in only one activity and therefore will undesirably make ⟨launch:subj:ORG⟩ similar with ⟨buy:dobj:ORG⟩.

Observation 2: Two predicates are more semantically similar if they are synonyms and share many common entities as arguments. Synonym predicates often have similar semantics, but this is not always guaranteed because they may be synonyms in terms of some less important meanings (senses). Nevertheless, if they share many common entities in an event, they are more likely to be of similar meanings, *e.g., buy* is the synonym of both *purchase* and *bribe* in WordNet, but in the *Acquisition of LinkedIn* event, *buy* is semantically more similar with *purchase* as they share common values like *Microsoft*, *LinkedIn*, *$26.2B*.

We propose a VAlue-cooccurrence-based Slot Expression (VASE) Graph to encode these observations. To prevent overfitting, we decompose a slot expression $q = \langle p{:}r{:}l \rangle$ into predicate p and role:label pair $r{:}l$. The embedding u_q^Q of the slot expression q is the sum of the embeddings of its predicate u_p^P and its role:label pair $u_{r:l}^{RL}$:

$$u_q^Q = u_{<p:r:l>}^Q = u_p^P + u_{r:l}^{RL} \tag{1}$$

A VASE graph (Figure 3) consists of a node set N and an edge set E. The graph is constructed as follows: 1) each sentence has a sentence node (rounded rectangle) $n^S \in N^S$; 2) each entity is represented as a value node (squares); 3) the slot expression of each entity in a sentence is denoted as a slot expression node (a circle) $n^Q \in N^Q$; 4) the slot expression nodes altogether constitute the corresponding sentence node; 5) a slot expression node for <$p{:}r{:}l$> is connected with both predicate node (diamonds) $n_p^P \in N^P$ and role:label node (triangles) $n_{r:l}^{RL} \in N^{RL}$ by expression-predicate edge $e_{q,p}^{QP} \in E^{QP}$ and expression-role:label edge $e_{q,r:l}^{QRL} \in E^{QRL}$, respectively; 6) if two sentences share common values (squares), then they are connected by a sentence-sentence edge $e^{SS} \in E^{SS}$ (a grey curve); 7) if two synonym predicates share more than 3 values, then they will be connected by a predicate-predicate edge $e^{PP} \in E^{PP}$ (a black curve). We ultimately want to induce schema by assembling slots. Slots are generated by clustering slot expressions. To cluster slot expressions, we need to learn the embeddings of predicates and role:label pairs. The embeddings are calculated as follows. We sample one pair of sentence nodes (*e.g.,* sentences 1 & 2 in Figure 3) at a time by using the weights of e^{SS}. Then for each of their common value nodes (*e.g., $26.2B*), we first find its slot expressions in the two sentences (⟨buy:iobj:MNY⟩ and ⟨purchase:quant:MNY⟩) respectively. Next we update the embeddings of their predicates (buy and purchase) and role:label pairs (iobj:MNY and quant:MNY), so that the difference between the embeddings of the two slot expressions decreases. Also, we sample a pair of synonym predicates (buy and purchase) and close up their embeddings.

The key problem is how to assign the weights of $e^{SS} \in E^{SS}$. Using the number of common values as the weights cannot model **Observation 1**. This is because as the number of common value increases, the number of sentence pairs drops dramatically. For example, in a sample of our news data, there are only 15 sentence pairs sharing 3 entities and 785 pairs sharing 2 entities, but there are 18,483 pairs sharing 1 common entity. If we weigh edges in E^{SS}

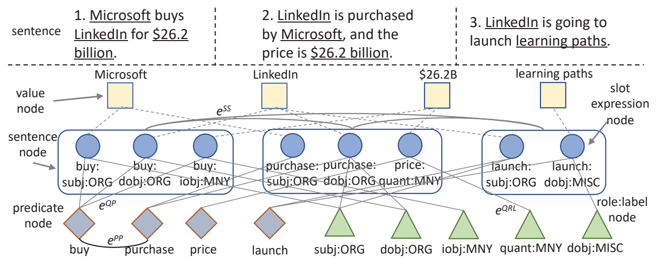


Figure 3: VASE Graph for Example Sentences

using the number of common values, the graph will be dominated by the weight-1 edges. As a result, the probability of getting a weight-3 edge is only $0.0022 = 15 * 3/(15 * 3 + 785 * 2 + 18,483)$, much smaller than 0.9196, the probability of weight-1 edges. To remedy this, we normalize the edge weights which share the same number of common entities. This results in larger probability of sampling sentence pairs with more common entities (*e.g.,* 1/15 for each weight-3 edge, 1/18, 483 for each weight-1 edge).

4.3 Schema Induction

We employ spectral clustering to group slot expressions into slots. The similarity between two expressions is defined as the cosine similarity between their embeddings. The key task is to determine the number k^* of clusters that can maximize both intra-cluster cohesion and inter-cluster separation. We define cohesion and separation w.r.t the values of slot tuples. Specifically, we calculate scores s_k for different k's and select the k with greatest score as the number of clusters.

$$s_k = \sum_{tp_i, tp_j, i \neq j, ev_{tp_i} = ev_{tp_j}} I(v_{tp_i} = v_{tp_j})^{I(slot(tp_i) = slot(tp_j))} \cdot I(v_{tp_i} \neq v_{tp_j})^{I(slot(tp_i) \neq slot(tp_j))} \tag{2}$$

where $slot(tp)$ is the slot of tuple tp, and $I(\cdot)$=1 if $\cdot$ is true and 0 otherwise. The rationale is, if two slot tuples tp_i, tp_j of the same event (ev_{tp_i}=ev_{tp_j}) belong to the same slot ($slot(tp_i)$=$slot(tp_j)$), their associated values should be the same (v_{tp_i}=v_{tp_j}), otherwise their values should be different. Lastly, we select a subset of the candidates slots to build schemas. Two criteria are considered: 1) representativeness: a slot must appear in more than half of the events; 2) uniqueness: slots must not semantically duplicate. To make sure they are met, we represent each slot by a distribution over slot values for each event and select slots incrementally. Each time we select the slot with maximum support, we calculate the cosine similarity between the value distribution of selected slot and that of every previously-added ones. If any similarity is greater than the threshold ω (empirically set $\omega = 0.25$), we discard the slot. We iterate the process until no slots can be added. Those added slots constitute the schema of an event type.

4.4 Event Profiling

For each event, we have detected its event type and its related news articles 4.1. For each event type, we now have its schema. In order to profile an event, for each slot in its schema, we use its slot expressions as patterns to extract values from the news corpus. The entity with maximum support is used as its slot value. Finally, the slot and slot value pairs together build the even profile.

5 PARAMETER ESTIMATION

In this section, we introduce how to estimate the parameters of the event detection and schema induction models.

5.1 Event Detection and Typing

We estimate the unknown parameters $\{\boldsymbol{\theta}, \boldsymbol{\lambda}, \boldsymbol{\tau}, \boldsymbol{\chi}, \boldsymbol{\phi}, \boldsymbol{\psi}, \boldsymbol{\mu}, \boldsymbol{\sigma}\}$ by employing collapsed Gibbs sampling to generate samples of event e and event type c assignments. A Gibbs sampler computes the full conditional probability for the event and event type assignments e_i and c_i for slot tuple i conditioned on all the other assignments $\boldsymbol{e}_{\neg i}$ and $\boldsymbol{c}_{\neg i}$. We only present the updating equations here.

$$\begin{aligned}
&P(e_i = e, c_i = c | \boldsymbol{e}_{\neg i}, \boldsymbol{c}_{\neg i}, .) \propto \frac{P(\boldsymbol{e}, \boldsymbol{c}, .)}{P(\boldsymbol{e}_{\neg i}, \boldsymbol{c}_{\neg i}, .)} \\
= \quad & \frac{n^{DE}_{d_i, e, \neg i} + \alpha\tau_e}{\sum_{e' \in E} n^{DE}_{d_i, e', \neg i} + \alpha} \cdot \frac{n^{EC}_{e, c, \neg i} + \beta}{\sum_{c' \in C} n^{EC}_{e, c, \neg i} + \beta|C|} \cdot \\
& t_{\rho-1}(t_i | \mu_{e, \neg i}, \frac{\Omega_{e, \neg i}(\kappa + 1)}{\kappa(\rho - 1)}) \cdot \\
& \frac{n^{CQ}_{c, q_i, \neg i} + \epsilon}{\sum_{q' \in Q} n^{CQ}_{c, q_i, \neg i} + \epsilon|Q|} \cdot \frac{n^{ELV}_{e, l_i, v, \neg i} + v}{\sum_{v' \in V_{l_i}} n^{ELV}_{e, l_i, v', \neg i} + v|V_{l_i}|} \cdot \\
& \frac{\prod_{w \in W} \prod_{y=0}^{c_{\boldsymbol{w}_i, w-1}} (n^{CW}_{c, w, \neg i} + \delta + y)}{\prod_{y=0}^{c_{\boldsymbol{w}_i}-1} \sum_{w \in W} (n^{CW}_{c, w, \neg i} + \delta|W| + y)},
\end{aligned} \tag{3}$$

where d_i, q_i, v_i, l_i and $\boldsymbol{w}_i$ are the document, slot expressions, value, entity label and keywords of slot tuple i, $n^{AB}_{a, b, \neg i}$ is the number of times $b \in B$ is assigned to $a \in A$ after excluding slot tuple i, where A can be either a single set (*e.g., E*) or multi-set (*e.g.,* event-label *EL*), $c_{\boldsymbol{w}}$ is the length of $\boldsymbol{w}$, and $c_{\boldsymbol{w}, w}$ is the count of w in $\boldsymbol{w}$. $t_\rho(t|\mu_e, \frac{\Omega_e(\kappa+1)}{\kappa\rho})$ is the likelihood time t is generated by the student-t distribution, where ρ is degree of freedom and

$$\begin{aligned}
\mu_e &= \frac{\kappa\mu_0 + n_e \bar{t}_e}{\kappa + n_e - 1}, \ \kappa = \kappa_0 + n_e - 1, \ \rho = \rho_0 + \frac{n_e - 1}{2} \\
\Omega_e &= \Omega_0 + \frac{1}{2} \sum_{t' \in \boldsymbol{t}_e} (t' - \bar{t}_e)^2 + \frac{\kappa_0(n_e - 1)(\bar{t}_e - \mu_0)^2}{2(\kappa_0 + n_e - 1)}
\end{aligned} \tag{4}$$

Here n_e is the number of times slot tuples assigned to e, and $\bar{t}_e$ is the average time of the slot tuples assigned to e.

When the number of events is changed or when an iteration is finished, we update the global event distribution $\boldsymbol{\tau}$ as follows:

$$\begin{aligned}
&\boldsymbol{\tau} \sim Dir(d_{e\,e}, \gamma), \ d_e = \sum_d \sum_e d_{d, e, n} \\
&d_{d, e, n} \sim Bern(\frac{\alpha\tau_e}{\alpha\tau_e + n - 1}), \ \forall n \in [1, n^{DE}_{d, e}], \ d \in D, \ e \in E
\end{aligned} \tag{5}$$

We iteratively generate assignments based on Equation 3, and update model parameters based on Equations 4 and 5, until a stable state is reached. Then we read out the assignments of e and c for all slot tuples as their events and event types.

5.2 Schema Induction

We estimate the embeddings for predicates and role:label pairs by fitting value co-occurrences between sentences (embedded in edge set E^{SS}), and fitting synonym relations between predicates (embedded in edge set E^{PP}).

Fitting E^{SS}: Let $V_{i,j}$ be the set of common values of sentences s_i and s_j. We model the probability $P(s_i, s_j)$ of observing the edge $e^{SS}_{s_i, s_j}$ by the product of the joint probabilities $P(q_{i,v}, q_{j,v})$ of the pairs of slot expressions $q_{i,v}$ and $q_{j,v}$ for all value $v \in V_{i,j}$ in sentences s_i and s_j, where $P(q_{i,v}, q_{j,v})$ is defined by a logistic function on their embeddings $u^Q_{q_{i,v}}$ and $u^Q_{q_{j,v}}$:

$$P(s_i, s_j) = \prod_{v \in V_{i,j}} P(q_{i,v}, q_{j,v}), \tag{6}$$

$$\text{where } P(q_a, q_b) = \frac{1}{1 + exp(-u^{Q\,tr}_{q_a} \cdot u^Q_{q_b})}$$

We can get the estimated distribution of weights for edges in E^{SS} by normalizing the weights of all edges in E^{SS} calculated by Equation 7. We approximate it to the empirical distribution w.r.t. KL-divergence, where the empirical one is obtained by normalizing the assigned weights of edges in E^{SS}. Having omitted some derivations, our goal is to minimize the objective function:

$$O^{SS} = -\sum_{e^{SS}_{i,j} \in E^{SS}} w^{SS}_{i,j} \sum_{v \in V_{i,j}} logP(q^Q_{i,v}, q^Q_{j,v}) \tag{7}$$

Fitting E^{PP}: Similarly, we fit the edges in E^{PP} by approximating the estimated and empirical distributions of weights for edges in E^{PP}. Let $w^{PP}_{i,j}$ be the assigned edge weights for synonym predicates p_i and p_j, and $P(p_i, p_j)$ be the logistic function on their embeddings. Our goal is to minimize the objective function:

$$O^{PP} = -\sum_{e^{PP}_{i,j} \in E^{PP}} w^{PP}_{i,j} logP(p_i, p_j) \tag{8}$$

The overall objective function is as follows:

$$O = O^{SS} + O^{PP} \tag{9}$$

We employ edge sampling to minimize the objective function. Specifically, in each step, we draw two positive edges $e^{SS}_{i,j}$ and $e^{PP}_{x,y}$ from E^{SS} and E^{PP}, respectively, and then draw K noisy edges for each of them. Based on negative sampling procedure, we update the embeddings of predicates and role:label pairs involved in $e^{SS}_{i,j}$ and $e^{PP}_{x,y}$ to get the embeddings of slot expressions. Instead of directly optimizing based on edge weight, we treat all edges in the two graphs are with weight 1, but the sampling probability of edges are proportional to their original weights. Comparing with original method, this sampling method enables us to set the updating rate more easily. In addition, the final results are less likely to be dominant by edges with larger weights. More details can be found in [63].

6 EXPERIMENTS

In this section, we evaluate the effectiveness of our model on event detection and typing, schema induction and event profiling on a news corpus. The corpus consists of 50 events under 5 event types, namely, *business mergers & acquisitions*, *airplane crashes*, *product announcements*, *terrorist bombings*, and *earthquakes*. For each event, we issued key words to NewsBank and crawled the retrieved news articles. In total, we got 5400 news articles. The number of news articles for an event ranges from 38 to 146 with the average number being 108. After collecting the corpus, we convert news articles into slot tuples by Stanford CoreNLP toolkit [39] and C-AMR parser [65][2]. Note that the MUC data that were used in previous schema induction papers [12, 13, 15, 44, 56] cannot be used in our paper because MUC events are document-level rather than corpus-level (Section 1). Our dataset shares a similar number of event types with MUC, but triples the number of news articles.

[2]the dataset and code are available at http://www.quan-yuan.com/datacode.html

6.1 Event and Event Type Detection

6.1.1 Experimental Setups. We compare the following methods in this experiment.

- Group Average Clustering with Bucketing and Re-clustering (GAC-BR) [67]. It divides documents into buckets based on time and clusters them iteratively by HAC on word vectors until a pre-defined number of clusters is obtained.
- Two Layer Cluster (TLC) [18]. It first obtains preliminary event clusters by Affinity Propagation on documents' word vectors, and then performs HAC on these clusters' word vectors to get a pre-set number of clusters as events.
- Multi-modal Retrospective Event Detection (MM-RED) [33]. It models each event by 3 multinomial distributions over keywords, person entities, location entities and 1 Gaussian distribution over time.
- Event Detection and Typing (EDT). Our method proposed in Section 4.1.
- EDT without Slot Expression (EDT$_{\neg q}$). To examine the contribution of slot expressions to event detection and typing, we evaluate a simplified version of our method by removing slot and all associated edges.

Our method uses the common default parameter values [26]: β=50/C, δ=ϵ=υ=μ_0=κ_0=0.01, α=γ=ρ_0=Ω_0=1.0. The number of iterations is 50 (we tried more iterations but got similar results). The parameters of baselines are set as their suggested values.

When using keywords to query news articles, we know the true event and event type of each news article. Then different methods will be used to detect the event of each news article. Following the settings in [33, 67], we map each generated event to its closest true event and evaluate the performance by precision (*pre*), recall (*rec*), micro F_1 (*mic* F_1) and macro F_1 (*mac* F_1). *mic* F_1 and *mac* F_1 are computed based on all events' and an event's *pre* & *rec*, respectively. To evaluate the performance of event typing, we map the generated event types to the true events, and check whether the generated event type of a generated event is the true event type of its corresponding true event. The same metrics are used for performance measurement.

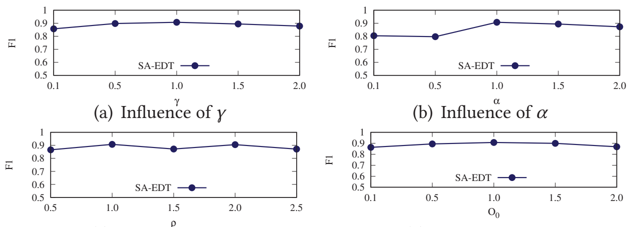
6.1.2 Performance Study.
Tables 1 summarizes the comparison results of event detection. Among the baselines, GACB has the best precision but the worst recall, because it extracts 575 events, much larger than the true number of events. After checking the results, we find GACB splits the news articles of one event into several over fine-grained clusters, because it relies on ad-hoc threshold to cluster news articles into events, which is very difficult to set. Extracting a larger number of events increases precision, but makes it much harder to retrieve all relevant documents for a true event, leading to worse recall. MM-RED, which detects 79 events, has the best recall among baselines, owing to its exploitation of person and location entities. Comparing with the baselines, our simplified model EDT$_{\neg q}$ achieves much better performance, probably because it makes use of entities of various types. Our full model EDT generates even better results, demonstrating the importance of slot expressions to event detection. Our detected number of events matches with the true number, 50. It shows that our model is effective in inferring the number of events from real-life data.

Table 1: Event Detection Results

Method	*pre*	*rec*	*mic* F_1	*mac* F_1
GAC-BR	**0.9866**	0.5810	0.7313	0.6957
TLC	0.9073	0.6417	0.7517	0.7280
MM-RED	0.7901	0.8436	0.8160	0.7412
EDT$_{\neg q}$	0.8381	0.9146	0.8747	0.8439
EDT	0.8735	**0.9444**	**0.9176**	**0.8952**

Table 2: Event Typing Results

Method	*pre*	*rec*	*mic* F_1	*mac* F_1
Our$_{\neg q}$	0.6226	0.6711	0.6459	0.6219
Our	**0.8601**	**0.8560**	**0.8580**	**0.8524**



(a) Influence of γ (b) Influence of α
(c) Influence of ρ (d) Influence of Ω_0

Figure 4: Influence of Hyper-parameters

The performance of event typing is shown in Table 2. Since other baselines cannot find event types, we only report the results of our two models. The table shows our full model EDT is all-round winner w.r.t. all metrics. The results again show the importance of slot expression to event detection and typing.

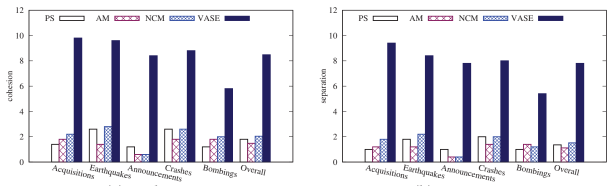
Our model involves a set of hyper-parameters. While β, δ, ϵ, υ and κ are set as their widely accepted values, we want to know if our model is sensitive to the remaining hyper-parameters. We plot the *microF*$_1$ of our model on event detection under different hyper-parameter settings in Figure 4. It shows our model can consistently achieve high detection performance. The results of other metrics or of event typing show similar trends and are thus omitted.

6.2 Schema Induction

6.2.1 Experimental Setups. The following schema induction methods are evaluated in this experiment.

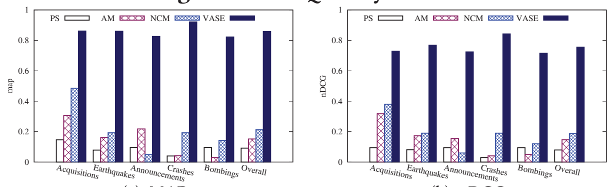
- ProbSchemas (PS) [12]: A Latent Dirichlet Allocation (LDA) [9]-based method that models schema type and slot as latent variables. Schema types generate predicates; slots generate head words, labels and typed dependencies of entities.
- AttributeModel (AM) [44]: A generative model which exploits attributes (adjective, verbal or nominal modifiers of entities) as additional observations to cluster heads and trigger relations of co-referring entities into slots.
- NormalizeCutModel (NCM) [55]: It exploits word embeddings and PMI and generates schemas by normalized cut.
- VASE: Our proposed model.

The codes of PS and NCM are obtained from the authors. The parameters of all baselines are set to their suggested values. For each event type, we select 10 slots with the largest supports to build its schema. The schema of "Business Acquisitions" can be found in Table 3, except for slots 7 and 10. Due to the limited space, we are not able to illustrate all schemas in the paper, but we are going to publish them online later. We evaluate the effectiveness on two aspects, namely, slot quality and schema quality.



(a) Cohesion (b) Separation

Figure 5: Slot Quality Results



(a) MAP (b) nDCG

Figure 6: Schema Induction Results

Slot Quality: A good method should be able to generate schemas with high-quality slots. As slots are modeled as clusters of slot expressions, we evaluate the quality of slots by 1) cohesion: semantic relatedness of the slot patterns of a slot; 2) separation: distinctness of a slot from others. Since schema-induction for corpus-level events has not been studied before, we don't have any ground-truth data for evaluation and we therefore have to rely on manual annotations. Given a schema, we ask five 3rd-party volunteers to examine the 10 slots. If the expressions of one slot are semantically cohesive (different), we add 1 to its cohesion (separation) score. We take the averages as the final scores.

Schema Quality: We ask an expert to generate schemas for different event types as ground-truths. Then we compare the generated schemas with them. An effective method should be able to generate schemas that well match the manually created ones. To measure the degree of matching, we verify whether each slot of the schema can match a slot in the ground-truth schema, given an event type and a generated schema. Since in a quality generated schema, more slots should be mapped to the slots of the ground-truth with high rankings, we evaluate the performance by Mean Averaged Precision (MAP) and Normalized Discounted Cumulative Gain (nDCG).

6.2.2 *Performance Study.*

Slot Quality: The results of different methods are shown in Figure 5. Our VASE is the all-round winner. Among the baselines, NCM achieves the best cohesion and separation because 1) it employs normalized cut to maximize the intra-slot similarity and minimize the inter-slot similarity; and 2) it exploits semantic information of entities and predicates to estimate the template-level and slot-level connectivities between two entities. In contrast, PS and AM ignore the inter-slot similarity and semantics. The cohesion and separation of PS are better than those of AM, because 1) AM relies on attributes when clustering entities, despite the fact that only a small portion of entities has attributes; and 2) PS requires the value of the same slot be the same as the entity label, imposing constraints on the semantics of the slot expressions. The cohesions of the baselines are much lower than ours, because they assume that the slot expressions of same entities tend to belong to the same slots. For example, they tend to put the slots expressions with predicates *have*, *say*, *expect* into the *buyer* slot, because buyers often make announcements. In contrast, our method puts those slot expressions into different slots because the co-occurrences of their values are often low. Our model

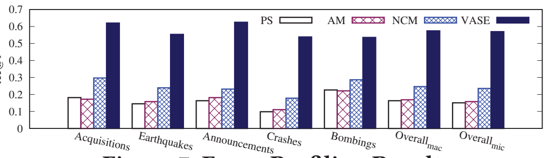


Figure 7: Event Profiling Results

has the best separation because 1) it employs spectral clustering to separate slots based on semantics, where spectral clustering is to find the partitions that minimizes normalized cut; and 2) it filters out duplicate slots based on slot values when selecting slots to build schemas. Note that the performance of VASE is relatively low on Bombing type. This is because VASE only generates 6 slots. In fact, there are only 7 slots in ground-truth schema. Thus, the results of Bombing type is at most 7/10 of other event types. To examine the consistency of labeling, we divide scores into 4 classes, {0-2}, {3-5}, {6-8}, {9-10}, and calculate the Cohen's Kappa Coefficient between the scores given by each pair of volunteers. The averaged Kappa Coefficients for separation and cohesion are 0.749 and 0.711, suggesting substantial agreement among our volunteers.

Schema Quality The MAP and nDCG of different methods on different event types are plotted in Figure 6. First, our model always achieves superior MAP and nDCG values across all event types, showing its capabilities in discovering meaningful slots and ranking them higher. Here are the reasons why our model achieves unparalleled performance. 1) Value co-occurrences are used to generate cohesive slot expression clusters which in turn benefit the slot matching; 2) we remove the duplicate slots when generating schemas to prevent slots of the same semantics from appearing multiple times. Otherwise they will bring down the ranking results; and 3) only the slots that frequently appear are included in schemas, so the slot representativeness is guaranteed. Unlike our method, none of the baselines has the slot selection process. They simply rank all generated slots based on frequency and select the top ones to build schemas. Among the baselines, NCM performs the best, probably because its exploitation of semantic embeddings and normalized cut, whereas the other two only consider entity mentions and use LDA-like model to perform clustering.

6.3 Event Profiling

We are interested in whether a method can generate quality event profiles. To evaluate this, for each event, we first identify the slot value for each slot of the ground-truth schema. Given a generated profile, we then find the number of ground-truth slot values that are identified as the top-1 slot values. Next, we calculate the number of slot values that are assigned to the correct slots. Last, we use $rec@1$ to measure the percentage of the ground-truth slot values.

We plot the results of the methods on different event types in Figure 7. In the same figure, we also show the macro and micro average $rec@1$ of all event types as $Overall_{mac}$ and $Overall_{mic}$. Among the baselines, NCM achieves the best performance, followed by AM. Such rank is the same as the rank of schema induction. Due to poor quality of generated schemas, baselines can seldom find semantically meaningful slots for slot values and thus perform not so well. Although our method outperforms the baselines significantly, our model still has a drawback when generating event profiles in the following scenarios. 1) The slot is not identified in the generated schemas, *e.g.,* our schema induction model fails to find the economic loss slot for Earthquakes; 2) the detected slot expressions

Table 3: Generated Schema for Business Acquisitions

Slot ID	Slot Name	Slot Patterns	Values
1	Buyer	⟨acquire:subj:ORG⟩, ⟨buy:subj:ORG⟩, ⟨pay:subj:ORG⟩	Facebook, Intel, Dell, Pfizer
2	Buyee	⟨buy:dobj:ORG⟩, ⟨acquire:dobj:ORG⟩, ⟨share:iobj:ORG⟩	Whatsapp, EMC, Sprint, Skype, PCC
3	Announce time	⟨announce:time:DATE⟩	2014-02-19, 2015-08-10, 2012-10-15
4	CEO	⟨CEO:subj:PER⟩, ⟨executive:subj:PER⟩	Zuckerberg, Brian, Liveris, Joe, Steve
5	Total price	⟨rate:quant:MNY⟩, ⟨value:iobj:MNY⟩, ⟨earn:quant:MNY⟩	\$1.02B, \$37.2B, ~\$67B, \$.345B
6	Close time	⟨close:time:DATE⟩, ⟨end:time:DATE⟩	2015-06-30, 2015-12-22, 2012-09-30
7	Price time	⟨price:time:DATE⟩	2014-02-24, 2015-08-07, 2012-10-02
8	Stack percent	⟨stack:quant:PCT⟩, ⟨share:quant:PCT⟩	8.0%, 20.0%, 75%, 80%, 76%
9	Price per share	⟨share:quant:MNY⟩, ⟨pay:dobj:MNY⟩	\$17.5, \$24.05, \$17.0, \$1.27
10	Stoke rise	⟨rise:iobj:PCT⟩, ⟨increase:iobj:PCT⟩	>2.0%, 2.1%, 1.68%, 5.5%, 1.55%
11	Location	⟨base:loc:LOC⟩, ⟨headquaters:loc:LOC⟩	India, Portland, Germany, Hopkinton
12	Company size	⟨employ:dobj:NUM⟩, ⟨employee:quant:NUM⟩	~3,0000, ~7,0000, ~1,600, >5,000

are incorrect, *e.g.,* the close time slot of Business Acquisitions; 3) the slot value is changing over time, *e.g.,* the number of deaths in an earthquake; and 4) the slot value is not identified as a named entity, *e.g.,* the company ARM is not recognized as an organization, probably because it is misclassified as a body part. Though our generated profiles are not perfect, they can serve as a good start point for various tasks, like knowledge graph construction and QA systems.

6.4 Case Study

We use business acquisition as an event type to illustrate our results. Table 3 shows our model identifies 12 slots, together with their frequent slot patterns (⟨predicate:role:label⟩ defined in Section 3) and slot values. To ease reading, we manually generate slot names based on the slot patterns. Most of the slots are meaningful and the slot patterns are indicative of value extraction. Our model even finds several slots beyond the ground-truth (*e.g.,* the rise of stock price after the purchase, and the size of the company). However, there are a few meaningless slots, *e.g.,* price time. The original sentences mention stake price on specific days. Although this slot appears frequently, it is meaningless as the date information is not modeled.

We use the event of Dell acquiring EMC as an example to illustrate the event profiling results of our model. The profile is shown in Figure 4 where the support of each value is in the parenthesis. The profile first gives a reasonable summary for the event. The slots then cover most key aspects. All slot values match the facts except for the price time and the close time. As discussed earlier, the price time is not a meaningful slot. In original news, the close time means when market closes rather than when the acquisition closes. Therefore, the close time is also meaningless.

Table 4: Profile for Dell's Acquisition of EMC

Slot Name	Values
Buyer	Dell (69), EMC (13), AOL (1)
Buyee	EMC (145), VMWare (11), Broadcom (3)
Announce time	2015-10-13 (16), 2015-05-13 (1)
CEO	Joe (8), Michael (4), Pat Gelsinger (2)
Total Price	>\$67B (4), \$24.05 (2)
Close time	2015-10-07 (1), 2015-10-09 (1)
Price time	2015-10-07 (7), 2015-10-07 (1)
Stack percent	80% (5), 98% (1)
Price per share	\$24.05 (2)
Stake rise	5.5% (1)
Location	Hopkinton (5), MA (1), Austin (1)
Company size	~70000 (1)

7 CONCLUSIONS

Event profiling not only provides readers with concise views of news events, but also facilitates various applications such as information retrieval, knowledge graph construction, QA systems, *etc.* In this paper, we propose the first framework on schema-based event profiling for large open-domain news corpora. To achieve the goal, we first detect events and event types by exploiting slot expressions of entities, and then learn the embeddings of slot expressions by leveraging entity co-occurrence information. After that, we cluster slot expressions as the slots of schemas, and select a subset of slots to build schemas. In the end, these schemas are used as the templates for event profiling. Extensive experimental results on a large open-domain news corpus demonstrate superior effectiveness of our model against the state-of-the-art baselines.

This is the very first work to profile events for open-domain news corpora. Consequently, there are still some very interesting directions waiting for further exploration. First, some slots may have multiple slot values, *e.g.,* more than one products may be launched in an announcement, so we need to find ways to extract a proper number of values for these slots. Second, the values of some slots may change over time, *e.g.,* the number of deaths in an earthquake may increase as time passes. Also, the reported values in different news articles around the same time may differ. Thus, it is interesting to develop truth-finding methods to find slot values more accurately. The last but not least, we need to improve the efficiency when our algorithm comes to dealing with larger datasets, as a larger dataset usually provides us with more data redundancy and therefore allows us to generate more accurate and comprehensive profiles.

Acknowledgements: This work was sponsored in part by U.S. Army Research Lab. under Cooperative Agreement No. W911NF-09-2-0053 (NSCTA), DARPA under Agreement No. W911NF-17-C-0099, National Science Foundation IIS 16-18481, IIS 17-04532, and IIS-17-41317, DTRA HDTRA11810026, and grant 1U54GM114838 awarded by NIGMS through funds provided by the trans-NIH Big Data to Knowledge (BD2K) initiative (www.bd2k.nih.gov). Any opinions, findings, and conclusions or recommendations expressed in this document are those of the author(s) and should not be interpreted as the views of any funding agencies.

REFERENCES

[1] The ace 2005 (ace05) evaluation plan. https://goo.gl/6GWc8j.

[2] Tdt 2004: Annotation manual. https://goo.gl/1RUcXB.

[3] J. Allan. *Topic detection and tracking: event-based information organization*, volume 12. Springer Science & Business Media, 2012.

[4] G. Angeli, S. Gupta, M. Jose, C. D. Manning, C. Ré, J. Tibshirani, J. Y. Wu, S. Wu, and C. Zhang. StanfordâĂŹs 2014 slot filling systems. *TAC KBP*, 695, 2014.

[5] N. Balasubramanian, S. Soderland, O. E. Mausam, and O. Etzioni. Generating coherent event schemas at scale. In *Proc. EMNLP*, pages 1721–1731, 2013.

[6] L. Banarescu, C. Bonial, S. Cai, M. Georgescu, K. Griffitt, U. Hermjakob, K. Knight, P. Koehn, M. Palmer, and N. Schneider. Abstract meaning representation (amr) 1.0 specification. In *Proc. EMNLP*, pages 1533–1544, 2012.

[7] M. Banko, M. J. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. Open information extraction from the web. In *Proc. IJCAI*, pages 2670–2676, 2007.

[8] H. Becker, D. Iter, M. Naaman, and L. Gravano. Identifying content for planned events across social media sites. In *Proc. WSDM*, pages 533–542, 2012.

[9] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *JMLR*, 3(Jan):993–1022, 2003.

[10] R. Bunescu and R. J. Mooney. Collective information extraction with relational markov networks. In *Proc. ACL*, 2004.

[11] Z. Cao, F. Wei, L. Dong, S. Li, and M. Zhou. Ranking with recursive neural networks and its application to multi-document summarization. In *Proc. AAAI*, pages 2153–2159, 2015.

[12] N. Chambers. Event schema induction with a probabilistic entity-driven model. In *Proc. EMNLP*, volume 13, pages 1797–1807, 2013.

[13] N. Chambers and D. Jurafsky. Template-based information extraction without the templates. In *Proc. ACL*, pages 976–986, 2011.

[14] H. Chen, E. Benson, T. Naseem, and R. Barzilay. In-domain relation discovery with meta-constraints via posterior regularization. In *Proc. ACL*, pages 530–540, 2011.

[15] J. C. K. Cheung, H. Poon, and L. Vanderwende. Probabilistic frame induction. In *Proc. NAACL-HLT*, pages 837–846, 2013.

[16] H. L. Chieu, H. T. Ng, and Y. K. Lee. Closing the gap: Learning-based information extraction rivaling knowledge-engineering methods. In *Proc. ACL*, pages 216–223, 2003.

[17] N. Chinchor, D. D. Lewis, and L. Hirschman. Evaluating message understanding systems: an analysis of the third message understanding conference (muc-3). *Computational linguistics*, 19(3):409–449, 1993.

[18] X. Dai, Y. He, and Y. Sun. A two-layer text clustering approach for retrospective news event detection. In *Proc. AICI*, volume 1, pages 364–368, 2010.

[19] A. Fader, S. Soderland, and O. Etzioni. Identifying relations for open information extraction. In *Proc. EMNLP*, pages 1535–1545, 2011.

[20] E. Filatova, V. Hatzivassiloglou, and K. McKeown. Automatic creation of domain templates. In *Proc. COLING/ACL poster sessions*, pages 207–214, 2006.

[21] D. Freitag. Toward general-purpose learning for information extraction. In *Proc. ACL*, pages 404–408, 1998.

[22] J. Goldstein, V. Mittal, J. Carbonell, and M. Kantrowitz. Multi-document summarization by sentence extraction. In *Proc. NAACL-ANLPWorkshop on Automatic summarization*, pages 40–48, 2000.

[23] R. Grishman and B. Sundheim. Message understanding conference-6: A brief history. In *Proc. COLING*, pages 466–471, 1996.

[24] A. Haghighi and L. Vanderwende. Exploring content models for multi-document summarization. In *Proc. NAACL*, pages 362–370, 2009.

[25] T. Hasegawa, S. Sekine, and R. Grishman. Discovering relations among named entities from large corpora. In *Proc. ACL*, page 415, 2004.

[26] G. Heinrich. Parameter estimation for text analysis. Technical report, 2008.

[27] Y. Hong, J. Zhang, B. Ma, J. Yao, G. Zhou, and Q. Zhu. Using cross-entity inference to improve event extraction. In *Proc. ACL*, pages 1127–1136, 2011.

[28] L. Huang, T. Cassidy, X. Feng, H. Ji, C. R. Voss, J. Han, and A. Sil. Liberal event extraction and event schema induction. In *Proc. ACL*, pages 1797–1807, 2016.

[29] L. Jean-Louis, R. Besançon, and O. Ferret. Text segmentation and graph-based method for template filling in information extraction. In *IJCNLP*, pages 723–731, 2011.

[30] H. Ji and R. Grishman. Refining event extraction through cross-document inference. In *Proc. ACL*, pages 254–262, 2008.

[31] N. Kasch and T. Oates. Mining script-like structures from the web. In *Proc. NAACL-HLT Workshop on Formalisms and Methodology for Learning by Reading*, pages 34–42, 2010.

[32] Q. Li, H. Ji, and L. Huang. Joint event extraction via structured prediction with global features. In *Proc. ACL*, pages 73–82, 2013.

[33] Z. Li, B. Wang, M. Li, and W.-Y. Ma. A probabilistic model for retrospective news event detection. In *Proc. SIGIR*, pages 106–113, 2005.

[34] S. Liao and R. Grishman. Filtered ranking for bootstrapping in event extraction. In *Proc. ACL*, pages 680–688, 2010.

[35] S. Liao and R. Grishman. Using document level cross-event inference to improve event extraction. In *Proc. ACL*, pages 789–797, 2010.

[36] C.-Y. Lin. Improving summarization performance by sentence compression: a pilot study. In *Proc. IRAL*, pages 1–8, 2003.

[37] H. Lin and J. Bilmes. A class of submodular functions for document summarization. In *Proc. ACL*, pages 510–520, 2011.

[38] J. Liu, J. Shang, C. Wang, X. Ren, and J. Han. Mining quality phrases from massive text corpora. In *Proc. SIGMOD*, pages 1729–1744, 2015.

[39] C. D. Manning, M. Surdeanu, J. Bauer, J. R. Finkel, S. Bethard, and D. McClosky. The stanford corenlp natural language processing toolkit. In *ACL (System Demonstrations)*, pages 55–60, 2014.

[40] D. McClosky, M. Surdeanu, and C. D. Manning. Event extraction as dependency parsing. In *Proc. ACL*, pages 1626–1635, 2011.

[41] B. Min, S. Shi, R. Grishman, and C.-Y. Lin. Ensemble semantics for large-scale unsupervised relation extraction. In *Proc. EMNLP*, pages 1027–1037, 2012.

[42] M. Miwa, R. Sætre, Y. Miyao, and J. Tsujii. A rich feature vector for protein-protein interaction extraction from multiple corpora. In *Proc. EMNLP*, pages 121–130, 2009.

[43] J. Mott, A. Bies, Z. Song, and S. Strassel. Parallel chinese-english entities, relations and events corpora. In *LREC*, 2016.

[44] K.-H. Nguyen, X. Tannier, O. Ferret, and R. Besançon. Generative event schema induction with entity disambiguation. In *Proc. ACL*, 2015.

[45] T. H. Nguyen and R. Grishman. Event detection and domain adaptation with convolutional neural networks. In *Proc. ACL*, pages 365–371, 2015.

[46] S. Patwardhan and E. Riloff. Effective information extraction with semantic affinity patterns and relevant regions. In *Proc. EMNLP-CoNLL*, pages 717–727, 2007.

[47] S. Patwardhan and E. Riloff. A unified model of phrasal and sentential evidence for information extraction. In *Proc. EMNLP*, pages 151–160, 2009.

[48] A.-M. Popescu, M. Pennacchiotti, and D. Paranjpe. Extracting events and event descriptions from twitter. In *Proc. WWW*, pages 105–106, 2011.

[49] S. Riedel and A. McCallum. Fast and robust joint models for biomedical event extraction. In *Proc. EMNLP*, pages 1–12, 2011.

[50] E. Riloff. Automatically generating extraction patterns from untagged text. In *Proc. AAAI*, pages 1044–1049, 1996.

[51] E. Riloff, J. Wiebe, and W. Phillips. Exploiting subjectivity classification to improve information extraction. In *Proc. AAAI*, volume 20-3, page 1106, 2005.

[52] A. Ritter, O. Etzioni, S. Clark, et al. Open domain event extraction from twitter. In *Proc. KDD*, pages 1104–1112, 2012.

[53] T. Sakaki, M. Okazaki, and Y. Matsuo. Earthquake shakes twitter users: real-time event detection by social sensors. In *Proc. WWW*, pages 851–860, 2010.

[54] V. Setty, A. Anand, A. Mishra, and A. Anand. Modeling event importance for ranking daily news events. In *Proc. WSDM*, pages 231–240, 2017.

[55] L. Sha, S. Li, B. Chang, and Z. Sui. Joint learning templates and slots for event schema induction. In *Proc. NAACL-HLT*, pages 428–434, 2016.

[56] L. Sha, J. Liu, C.-Y. Lin, S. Li, B. Chang, and Z. Sui. Rbpb: Regularization-based pattern balancing method for event extraction. In *Proc. ACL*, volume 1, pages 1224–1234, 2016.

[57] D. Shen, J.-T. Sun, H. Li, Q. Yang, and Z. Chen. Document summarization using conditional random fields. In *Proc. IJCAI*, pages 2862–2867, 2007.

[58] Y. Shinyama and S. Sekine. Preemptive information extraction using unrestricted relation discovery. In *Proc. HLT-NAACL*, pages 304–311, 2006.

[59] M. Stevenson and M. A. Greenwood. A semantic approach to ie pattern induction. In *Proc. ACL*, pages 379–386, 2005.

[60] K. Sudo, S. Sekine, and R. Grishman. An improved extraction pattern representation model for automatic ie pattern acquisition. In *Proc. ACL*, pages 224–231, 2003.

[61] A. Sun, R. Grishman, W. Xu, and B. Min. New york university 2011 system for kbp slot filling. In *TAC*, 2011.

[62] M. Surdeanu, J. Turmo, and A. Ageno. A hybrid approach for the acquisition of information extraction patterns. In *Proc. ATEM*, pages 48–55, 2006.

[63] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei. Line: Large-scale information network embedding. In *Proc. WWW*, pages 1067–1077, 2015.

[64] Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei. Hierarchical dirichlet processes. *Journal of the American Statistical Association*, 2012.

[65] C. Wang, N. Xue, S. Pradhan, and S. Pradhan. A transition-based algorithm for amr parsing. In *Proc. NAACL*, pages 366–375, 2015.

[66] B. Yang and T. Mitchell. Joint extraction of events and entities within a document context. In *Proc. NAACL*, 2016.

[67] Y. Yang, T. Pierce, and J. Carbonell. A study of retrospective and on-line event detection. In *Proc. SIGIR*, pages 28–36, 1998.

[68] R. Yangarber, R. Grishman, P. Tapanainen, and S. Huttunen. Automatic acquisition of domain knowledge for information extraction. In *Proc. ACL*, pages 940–946, 2000.

[69] C. Zhang, G. Zhou, Q. Yuan, H. Zhuang, Y. Zheng, L. Kaplan, S. Wang, and J. Han. Geoburst: Real-time local event detection in geo-tagged tweet streams. In *Proc. SIGIR*, pages 513–522, 2016.

[70] W. Zhang, W. Feng, and J. Wang. Integrating semantic relatedness and words' intrinsic features for keyword extraction. In *Proc. IJCAI*, pages 2225–2231, 2013.