Open-Schema Event Profiling for Massive News Corpora

Anonymous EMNLP submission

Abstract

The availability of news articles in a sheer volume calls for an effective way to represent event information. To this end, we define a new problem – schema-based news event profiling, which aims to profile events reported in open-domain news corpora by sets of slot and slot value pairs, where the slots constitute the schema of an event type. In this paper, we first develop a probabilistic model to detect events and event types by exploiting the semantic roles of entities, and then propose an unsupervised embedding model to induce schemas based on the co-occurrences of values in different sentences. Experimental results on a large news corpus demonstrate our superior performance against state-of-the-art baselines on event detection, schema induction and event profiling.

1 Introduction

News event extraction is an important task that has been investigated for years. Some studies [36, 20, 12] extract *topic-level events*, *i.e.,* "a particular thing that happens at a specific time and place, along with all necessary pre-conditions and unavoidable consequences" [2] together with their relevant news articles. These studies can detect events like "Acquisition of LinkedIn", but they do not try to extract key aspects to represent events. Other studies [17, 19, 15] focus on extracting *atomic events*, *i.e.,* "a specific occurrence which involves in some participants" [1]. These studies can extract events and their participants from sentences, but creating a high-level event summaries is not their focus. In this paper, we combine their merits, and define the problem of schema-based event profiling, which aims to generate profiles for topic-level events from open-domain news corpora. The event profiles can not help people quickly digest the huge volume of news available online, but also facilitate a number of applications, such as knowledge-base construction, information retrieval, question answering systems, *etc.*

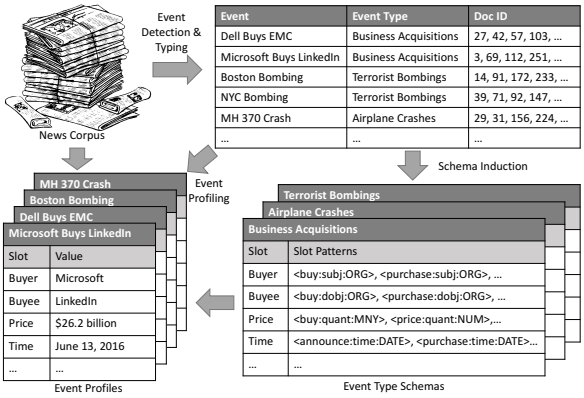


Figure 1: Framework Overview

`Schema` is a template that defines a specific type of events, associated with `slots` (key aspects) held by entities as `slot values` [16, 22]. For a specific event, each slot has a specific slot value. For example, the schema of business acquisition events involves slots like *buyee*, *buyer*, *price*, *etc.*, and the slot value of *buyee* in the *Acquisition of LinkedIn* event is *LinkedIn*. Given an open-domain news corpus, our goal is to extract slot – slot value pairs to profile each reported event. To achieve this, we first extract events together with their event types and relevant news articles. As a slot can be expressed in a number of ways (*e.g., buyee* may be the object of *buy* or the subject of *sell*), for each event type, we extract expressions from the news articles of the same events and cluster them as slot patterns. In the end, we extract slot values using slot patterns, and profile each event by slot – slot value pairs. The overview of the framework is illustrated in Figure 1.

The key challenge is schema induction. Existing schema induction methods either rely on

domain-specific labeling data and human guidance [14, 10, 6, 25, 37, 26, 32, 23], or are built on simplistic assumptions, *e.g.,* an entity can serve as the value for one slot in a news article [8, 7, 9, 22]. Worse still, existing studies treat news articles independently and overlook their relations. However, an event is always reported by multiple news articles, which together offer valuable redundant data for schema induction.

In this paper, we develop a fully automatic unsupervised approach to schema induction. Our key insight is, in an event, an entity may be the values of multiple slots, but if a set of entities appear together in multiple sentences, their respective slots in these sentences tend to be the same. Take *LinkedIn* as an entity example, its slots could be *buyee* or *new services launcher* and so on. However, if *LinkedIn* appears with other entities such as *$26.2 billion* and *Microsoft* in multiple sentences, these sentences are likely to talk about the same activity – the purchase. Thus, the slots of *LinkedIn* in these sentences should be the same, namely, *buyee*. Although rare in a news article, the co-occurrences are easy to find across the news articles about the same events. We encode the insight in an embedding method, which can capture the similarities between slot expressions according to entity co-occurrences. The similarities are then used to generate slots and their slot patterns.

To make the paper self-contained, we propose a novel approach to event detection and typing. Existing studies discover events by clustering news articles with similar words vectors [36, 12, 11] or entities [20]. They all rely on ad-hoc methods to guess event numbers, and are unable to detect event types, which are the prerequisite to schema induction. We observe that the slot expressions of entities overlooked in existing studies are important to event and event type detection, *e.g.,* based on the slot expressions of *Microsoft*, we can better separate the news about *Microsoft Windows 10 launch* and *the LinkedIn acquisition*. We propose a model that can exploit slot expressions and generate a proper number of events with event types.

In summary, the contributions of the paper are:

- We exploit slot expressions of entities and propose a generative event extraction and typing model that can automatically determine the number of events in a news corpus.
- We propose a schema induction framework, which exploits entity co-occurrence information in a large news corpus to induce event schemas and extracts slot patterns.
- We conduct extensive experiments to evaluate the effectiveness of our models in event detection, schema induction, and event profiling. The results demonstrate its superior performance over the state-of-the-art baselines.

2 Related Work

2.1 Event Detection

Our task is related to the subtask of retrospective event detection defined in TDT [3] project, which aims to assign news articles in a given corpus to their respective previously unknown events. Yang *et al.* [36] propose a hierarchical agglomeration clustering (HAC) based approach to cluster news articles on words as events. Dai *et al.* first get preliminary event clusters by pairwise clustering [12] or Affinity Propagation [11], and then perform HAC to get the final events. Li *et al.* propose a probabilistic model that considers words, person, location entities and time. None of existing solutions can find event types and determine event numbers in a principled way. In addition, they rely on ad-hoc methods to guess the number of events, and overlook slot expressions of entities. Event detection from social media data [28, 24, 27, 5, 38] is also related. However, due to the differences in data length and quality, those methods are not applicable to news data.

2.2 Schema Induction

Schemas are often hand-coded by human experts with domain knowledge, but manually creating schemas for large open-domain news corpus is infeasible. Early attempts on schema induction take special training resources as input [14, 10, 6], or call for human judgments [25, 26] or bootstrapping seeds [37, 32, 23]. Some automated methods discover schema slots by extracting and clustering expressions from dependency trees [31, 13], but how to differentiate expressions with the same entity types [31] or with different verbs [13] are unsolved. Cheung *et al.* [9] propose a generative model for frame (correlated events) induction with the account event transitions, but neither the event causality nor evolution is our focus.

Several entity-driven models are proposed in recent years. Chambers *et al.* [8] extract domain-specific relation expressions, and agglomeratively cluster relations with the mentions of the same entities into slots. In the follow-up work [7], they

employ topic models to cluster entity mentions of news articles as slots, assuming the expressions of the same entity's mentions belong to the same slot. Nguyen *et al.* [22] exploit the immediate entity context to disambiguate entities and discover schema slots. Sha *et al.* [29] 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 is constant, and they all fail to exploits the redundant data across corpus. Although some studies [8, 7, 9, 22, 29] involve profiling tasks, they can only profile on document level. Note that event extraction and schema induction for atomic events [17, 19, 15, 30] are also related. However, different from them, we induce schemas for topic-level events by aggregating the information across corpus, rather than analyzing on sentence level.

3 Preliminaries and Data Preparation

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

Entities and Entity Labels: a news article always contains entities with different entity labels. These entities may serve as the values of an event's key aspects such as who, where and when. We consider entities of various labels, including person (*PER*), organization (*ORG*), location (*LOC*), date (*DATE*), money (*MNY*), number (*NUM*), and percent (*PCT*). We also normalize date into the form of the TIMEX3 by SUTime.

Keywords: we sort the terms of each news article based on TF-IDF, and use the top 5% terms as the set of keywords. We also augment keyword set with key phrases extracted by SegPhrase [20].

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> (<p:r:l> 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. For example, the slot expression of *LinkedIn* in the sentence "Microsoft buys LinkedIn" is <*buy*:*dobj*:*ORG*>, indicating that the organization *LinkedIn* is the direct object of *buy*. 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 (*e.g.,* <buy:dobj:ORG>, <sell:subj:ORG> for the slot *buyee*), and the representative ones serve as the slot patterns for value extraction. We obtain slot expressions from the parsing graphs of Abstract Semantic Representation (AMR) [4]. Although the AMR parsing result for one sentence may not be always perfect, it can be mitigated by aggregating the results of massive sentences.

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

Data Preparation: We selected 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 the end, we got 5400 news articles. The number of news articles for an event ranges from 38 to 146, and the average number is 108. After crawling, we convert each news article into slot tuples by Stanford CoreNLP toolkit [21] and C-AMR parser [35].

4 Proposed Model

In this section we introduce our proposed model for event profiling. We first detect events and event types (Section 4.1), and then for each event type, we learn the embeddings for slot expressions (Section 4.2). Based on the embeddings, we induce the schema of each event type by clustering similar expressions as slots (Section 4.3). In the end, we build profiles for extracted events based on the induced schemas (Section 4.4).

4.1 Event Detection and Typing

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

1. Each news article has a distribution over events, while each event has distributions over entities and event types.
2. Each event type is related to a set of slot expressions and keywords, *e.g.,* business acquisitions often involve expressions <*buy*:*dobj*:*ORG*>, <*price*:*quant*:*MNY*> and keywords *purchase*, *stock market*, *etc.*
3. Slot expressions determine the label of entities, and events determine the specific entities of that labels, *e.g.,* the entities of <*buy*:*dobj*:*ORG*> must be organizations, and in *the Acquisition of LinkedIn*, the entity is the organization *LinkedIn*.

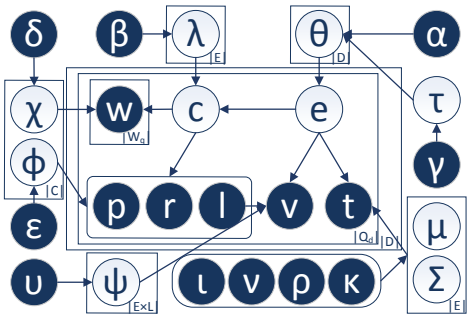


Figure 2: Graphical Model

4. 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. c further generates a slot expression q=<p:r:l> and a set of keywords $\boldsymbol{w}$ based on its multinomial distributions ϕ_c and χ_c. In the end, a entity value v is drawn from multinomial distribution $\psi_{e,l}$ conditioned on event e and entity label l of q, and a time stamp t is sampled from e's time distribution. Here we model time distribution by a Gaussian distribution $N(t|\mu, \Sigma)$ with μ and Σ as the mean and the variance, respectively.

Identifying the number of events is crucial to event detection. We employ Hierarchical Dirichlet Process (HDP) [34], in which there 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, in which ψ, and $\mu_0, \kappa_0, \rho_0, \Omega_0$ are the parameters of Normal-Gamma prior, and $\beta, \theta, \epsilon, \upsilon$ are the Dirichlet priors for λ, χ, ϕ. Gibbs sampling is used to fit the model.

Table 1 and 2 summarize the comparison results of our model and its simplified version (Our$_{\neg q}$) with slot expression removed against baselines Group Average Clustering with Bucketing and Re-clustering (GACB) [36], Two Layer Cluster (TLC) [11] and Multi-modal RED (MM) [18]. When using keywords to query for news articles, we know the true event and event type of each news article. Following the settings in [36, 18], we map each generated cluster to its closest true event, and evaluate the performance

Table 1: Event Detection Results

Method	*pre*	*rec*	*mic* F_1	*mac* F_1
GACB	**0.9866**	0.5810	0.7313	0.6957
TLC	0.9073	0.6417	0.7517	0.7280
MM	0.7901	0.8436	0.8160	0.7412
Our$_{\neg q}$	0.8381	0.9146	0.8747	0.8439
Our	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**

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-event and per-event *pre* & *rec*, respectively. The parameters of baselines are set as their suggested values. Our method uses the common default parameter values: β=$50/C$, δ=ϵ=υ=μ_0=κ=0.01, α=γ=ρ=Ω_0=1.0[1].

Among the baselines, GACB has the best precision, but its recall is low, because it extracted too many events. MM has the best recall among baselines, owing to the exploitation of person and location entities. Comparing with the baselines, the simplified version of our model achieves much better performance, probably because it makes use of entities of various types. Our full model generates the best results, demonstrating the importance of slot expressions to event detection. Similar results can be observed on the event typing task. The number of events detected by our model is 50 which is the true number, where the numbers of the three baselines are 575, 79 and 86, respectively. It shows that our model is effective in inferring event numbers from data.

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 hypotheses:

Hypothesis 1: Two slot expressions are more semantically similar if they are associated with the same entity and this entity appear with the same set of other entities in multiple sentences. □

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

[1]the dataset and all implementations of the paper will be publicly available online

the following sentences in news articles about LinkedIn purchase:

Microsoft buys LinkedIn for $26.2 billion.
LinkedIn is purchased by Microsoft with the price of $26.2 billion.
LinkedIn is going to launch online learning paths.

We observe that *LinkedIn* itself is involved in two activities, namely, acquisition and launching new services. However, when it appears together with entities *$26.2 billion* and *Microsoft* in the first two sentences, it is more likely to participate in the same activity – acquisition. Thus, the slot expressions of *LinkedIn*, namely, *e.g.,* <buy:dobj:ORG> and <purchase:dobj:ORG>, should be semantically similar (it also holds for those of *$26.2 billion* and *Microsoft*). Existing studies assume one entity is involved in only one activity, and thus will undesirably make <launch:subj:ORG> similar with <buy:dobj:ORG>.

Hypothesis 2: Two slot expressions are more semantically similar if they are associated with the same set of entities in more sentences. □

Although a set of entities may be involved in multiple activities, in an event there are only a few dominant ones. Thus, the slot expressions of frequently appearing entities are likely to describe the same activities, and thus they are semantically similar. To see this, we collected the predicates with *Microsoft* and *LinkedIn* as the arguments, and found the most frequent 3 are *buy*, *purchase* and *acquire*, all of which are about acquisition. In contrast, those less frequent predicates are diverse in semantics, such as *agree*, *contact*, *etc.*

Hypothesis 3: Two predicates are more semantically similar if they are synonyms and share many common entities as arguments. □

Synonym predicates often have similar semantics, but it is not always guaranteed because they may be synonyms in terms of some less relevant 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 semantically similar with *purchase* as they share common values like *Microsoft*, *LinkedIn*, *$26.2*.

We propose a VAlue-cooccurrence-based Slot Expression (VASE) graph to encode these hypotheses. To prevent overfitting, we decompose a slot expression q=<p:r:l> 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 $u_{r:l}^{RL}$:

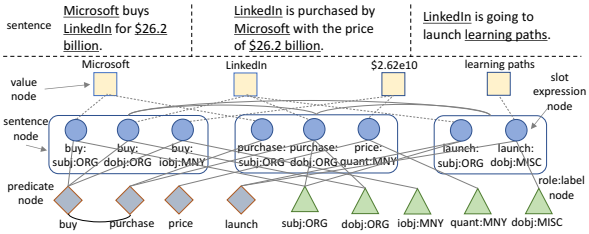


Figure 3: VASE Graph for Example Sentences

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

A VASE graph (Figure 3) consists of node set N and edge set E. Each sentence has a sentence node (rounded rectangle) $n^S \in N^S$, which contains a set of slot expression nodes N^Q (circles) for the expressions mentioned in it. If two sentences share common values (squares), they are connected by a sentence-sentence edge $e^{SS} \in E^{SS}$. A slot expression node for <p:r:l> is connected to its 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. In addition, if two synonym predicates share more than 3 common values, they will be connected with a predicate-predicate edge $e^{PP} \in E^{PP}$.

We learn the embeddings as follows: each time we sample a pair of sentence nodes (*e.g.,* the first two sentences in Figure 3) based on the weights of e^{SS}. Then for each of their common values (*e.g., $26.2e10*), we check its expressions in the two sentences (<buy:iobj:MNY> and <purchase:quant:MNY>) and update the embeddings of their predicates (buy and purchase) and role:label pairs (iobj:MNY and quant:MNY) to make their sum closer in the embedding space. Each time we also sample a pair of synonym predicates (*e.g.,* buy and purchase) and make their embeddings closer.

The key problem is how to set the weights for $e^{SS} \in E^{SS}$. A simple solution is to use the number of common values as the weights. Although such setting can meet Hypothesis 2, it violates Hypothesis 1. This is because as the number of common value increases, the number of sentence pairs drops dramatically. For example, in 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} by the number of common values, the graph will be dominant 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+18483)), against 0.9196 for weight-1 edges. To remedy this, we normalize the weights for E^{SS} with the same number of common entities. By doing this, we give much more importance to sentence pairs with more common entities (*e.g.,* $1/15$ for each weight-3 edge) than those with fewer common entities (*e.g.,* $1/18,483$ for each weight-1 edge).

Model Fitting 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 $u^Q_{q_{j,v}}$:

$$P(s_i, s_j) = \prod_{v \in V_{i,j}} P(q_{i,v}, q_{j,v}), \quad (2)$$

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

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 3. We approximate it to the empirical distribution w.r.t. KL-divergence, where the empirical one is obtained by normalizing the assigned weights w^{SS} of edges in E^{SS}. Omitted some derivations, our goal is to minimize the objective function:

$$O^{SS} = - \sum_{e^{SS}_{i,j} \in E^{SS}} w^{SS}_{i,j} \sum_{v \in V_{i,j}} logP(q^Q_{i,v}, q^Q_{j,v}) \quad (3)$$

Fitting E^{PP}: Similarly, we fit they 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:

$$O^{PP} = - \sum_{e^{PP}_{i,j} \in E^{PP}} w^{PP}_{i,j} logP(p_i, p_j) \quad (4)$$

The overall objective function is as follows:

$$O = O^{SS} + O^{PP} \quad (5)$$

We employ negative sampling on edges to minimize the objective function by following [33].

4.3 Schema Induction

We employ spectral clustering to group slot expressions into slots, where the similarity between two expressions is defined as the cosine similarity between their embeddings. The key problem is to determine the number of clusters k, which should be able to maximize the within-cluster cohesiveness and cross-cluster separateness. As embeddings have been used for similarity estimation, we employ the values of slot tuples as another kind of information to defined the cohesiveness and separateness. Specifically, we calculate the score s_k for different k, and select the k with greatest score as the final number of clusters.

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

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 ($slot(tp_i) = slot(tp_j)$), we hope their associated values to be the same ($v_{tp_i} = v_{tp_j}$), otherwise we hope their values to be different.

Then, we select a subset of the candidates slots to build schema. Two criteria are considered: 1) representativeness. A slot must appear in more than half of the events; 2) distinguishness. Slots should not duplicate. To guarantee distinguishness, 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, and check the cosine similarity between its value distribution with that of every added slot. If any similarity is greater than threshold ω (we 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.

4.4 Event Profiling

Now we have a set of events, and for each event, we have its event type and news articles about it. For each event type, we have its schema. To profile an event, we first find its schema, and for each slot in the schema, we use its slot expressions as the patterns to extract values from the news articles. The entities with the maximum supports are used as the values of the slot. In the end, the slot and slot value pairs serve as the profile of an event.

5 Experiments

In this section, we evaluate the effectiveness of our model on schema induction and event profiling.

5.1 Experimental Setups

This experiment evaluates how well a method can generate meaningful schema for each event type.

Methods for Comparison

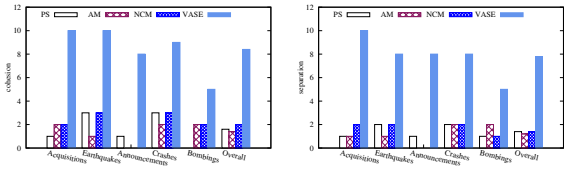
- ProbSchemas (PS) [7]. An LDA-based method that models schema type and slot as latent variables, where slots generate the head words, labels and typed dependencies of entities, and schema types generate predicates.
- AttributeModel (AM) [22]. A generative model which exploits attributes as additional observations to cluster the heads and trigger relations of co-referring entities into slots.
- NormalizeCutModel (NCM) [29]. The model exploits word embeddings and PMI, and generates schemas by employing normalized cut.
- VASE. Our proposed model.

The code of PS and NCM is obtained from the authors. The parameters of all baselines are set to their suggested values. We use 0.25 as the learning rate in our model, and for each positive sample, we generate 5 negative samples for updating.

Tasks and Evaluation Metrics

Slot Quality A good method should be able to generate schemas with high-quality slots. As a slot is modeled as a cluster of relevant expression patterns, we evaluate the quality of slots by annotating the quality of the 3 aspects: 1) cohesion: how semantically related are the expression patterns of a slot; 2) separation: how distinct a slot is from other slots; and 3) extractiveness: how effective the expression patterns can help extract slot values. Given an event type, we ask 5 students to compare between every pair of schemas on the 3 aspects, and assign score 1 to the winner schema. The score of a method on an aspect is the sum of scores it receives from the annotators, averaged by the numbers of annotators and event types. The annotators do not know by which method a schema is generated.

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



(a) Cohesion (b) Separation

Figure 4: Slot Quality Results

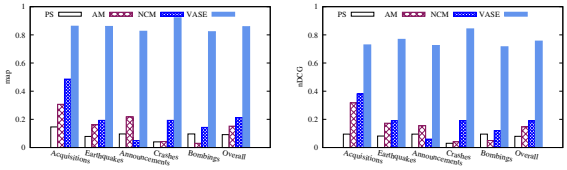
5.2 Experimental Results on Slot Quality

The results of different methods are shown in Figure 4. We can find that our method 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; 2) it exploits semantic information for schema induction. 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 PS additionally considers entity labels when clustering slot patterns. The cohesions of the baselines are much lower than ours, because they assume that the slot patterns of the same entities tend to belong to the same slots. For example, they tend to put the slots patterns with predicates *have*, *say*, *expect* into the slot of *buyer*. Comparing with the baselines, our method exploits value co-occurrences to disambiguate slot patterns. Our model also enjoys the best separation, because 1) it employs spectral clustering to separate slots based on semantics; 2) it filters out duplicate slots based on values when selecting slots to build schemas.

The extractiveness of the three methods is not satisfying, because 1) they do not require the values must be named entities, as a result, less meaningful values will be extracted, such as *we*, *deal*, *company*; 2) only PS exploits entity labels, but it assume each slot has a distribution over entity labels. Thus, a slot may takes entities of different labels (*e.g.,* PER, LOC) as its values. In contrast, we impose more strict constraints on values: the values must be named entities, and the values of a slot must be of the same label.

Table 3: Generated Schema for Business Acquisitions

Slot Name	Slot Patterns	Values
Buyer	<acquire:subj:ORG>, <buy:subj:ORG>	Facebook, Intel, Dell
Buyee	<buy:dobj:ORG>, <acquire:dobj:ORG>	Whatsapp, EMC, Sprint
Announce time	<announce:time:DATE>	2014-02-19, 2015-08-10
CEO	<CEO:subj:PER>, <executive:subj:PER>	Zuckerberg, Brian, Liveris
Total price	<rate:quant:MNY>, <value:iobj:MNY>	$1.02e9, $3.72e10, ~$6.7e10
Close time	<close:time:DATE>, <end:time:DATE>	2015-06-30, 2015-12-22
Price time	<price:time:DATE>	2014-02-24, 2015-08-07
Stack percent	<stack:quant:PER>, <share:quant:PER>	8.0%, 20.0%, 75%, 80%, 76%
Price per share	<share:quant:MNY>, <pay:dobj:MNY>	17.5%, 24.05%, 17.0%, 1.27%
Stake rise	<rise:iobj:PCT>, <increase:iobj:PCT>	>2.0%, 2.1%, 1.68%, 5.5%
Location	<base:loc:LOC>, <headquaters:loc:LOC>	India, Portland, Germany
Company size	<employ:dobj:NUM>, <employee:quant:NUM>	~3,0000, ~7,0000, ~1,600



(a) MAP (b) nDCG

Figure 5: Schema Induction MAP and nDCG

5.3 Experimental Results on Schema Quality

The MAP and nDCG of different methods on different event types are plotted in Figure 5. First, we find our model always achieves superior MAP and nDCG values on all event types, showing its capabilities in discovering meaningful slots and ranking them higher. The values of other methods are much worse. Among the baselines, NCM always performs the best, probably because its exploitation of semantic embeddings and normalized cut, while the other two methods only consider entity mentions and only use LDA-like model to perform clustering.

5.4 Case Study

We use business acquisitions as example to illustrate our results. As shown in table 3, our model identifies 12 slots, together with their frequent slot patterns and values (slot names are annotated by us based on the slot patterns). Most of the slots are quite meaningful, and the slot patterns are indicative for value extraction. Our model even find several slots beyond the groundtruth, *e.g.,* the rise of stock price after the purchase, and the size of the company. However, there are also a few slots that are less meaningful, *e.g.,* price time. We checked the original sentences, and found most of them talk about the stake price on specific days. Although the slot appears frequently in acquisition news articles, since the price is not included in the slot patterns, the slot is not meaningful any more.

We use the event of Dell acquires EMC as an example to illustrate the event profiling results of our model. The profile is shown in Figure 4, where the slot patterns of each slot can be found in Figure 3, and the support of each value is in the brace. First of all, we can find that the profile gives a quite reasonable summary for the event. The slots cover most key aspects, and most of the values match the facts. Two exceptions are the close time and price time. As discussed earlier, price time is not a meaningful slot, while the value of close time is not correct. We checked the original news articles, and found these sentences report the closing price, and the values are just those date, rather than the closing date of the purchase.

Table 4: Profile for Dell's Acquisition of EMC

Slot Name	Values
Buyer	Dell (69), EMC (13)
Buyee	EMC (145), VMWare (11)
Announce time	2015-10-13 (16)
CEO	Joe (8), Michael (4)
Total Price	>$6.7e10 (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)
Company size	~70000 (1)

6 Conclusion

Profiling events can not only provide readers with concise overviews of news events, but also facili-

tate a number of applications such as information retrieval, knowledge graph construction, question-answering systems, *etc.* In this paper, we propose a framework that can detect events from massive news corpora, and profile events based on event type schemas. To detect events, we exploit slot information of various types of entities, and develop a probabilistic model that can extract a proper number of events from news articles. To induce schemas, we make use of entity-cooccurrence information to learn the embeddings of slot patterns, based on which we cluster slot patterns into the slots of schemas. The schemas are used as the template for event profiling. Experimental results on a large news corpus show that our proposed model achieve superior effectiveness in event detection and typing, schema induction and event profiling.

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] 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.

[5] 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.

[6] R. Bunescu and R. J. Mooney. Collective information extraction with relational markov networks. In *Proc. ACL*, page 438, 2004.

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

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

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

[10] 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.

[11] 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.

[12] X.-Y. Dai, Q.-C. Chen, X.-L. Wang, and J. Xu. Online topic detection and tracking of financial news based on hierarchical clustering. In *Proc. ICMLC*, volume 6, pages 3341–3346, 2010.

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

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

[15] 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.

[16] 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.

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

[18] 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.

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

[20] 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.

[21] 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.

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

[23] S. Patwardhan and E. Riloff. Effective information extraction with semantic affinity patterns and relevant regions. In *Proc. EMNLP-CoNLL*, volume 7, pages 717–727, 2007.

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

[25] E. Riloff and M. Schmelzenbach. An empirical approach to conceptual case frame acquisition. In *Proc. Workshop on Very Large Corpora*, pages 49–56, 1998.

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

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

[28] 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.

[29] 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.

[30] 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.

[31] 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.

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

[33] 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.

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

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

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

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

[38] 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.