Constructing a Comprehensive Events Database from the Web

Qifan Wang, Bhargav Kanagal, Vijay Garg, D. Sivakumar
Google Research, Mountain View, USA
{wqfcr,bhargav,vijaygarg,siva}@google.com

ABSTRACT

In this paper, we consider the problem of constructing a comprehensive database of events taking place around the world. Events include small hyper-local events like farmer's markets, neighborhood garage sales, as well as larger concerts and festivals. Designing a high-precision and high-recall event extractor from unstructured pages across the whole web is a challenging problem. We cannot resort overly to domain-specific strategies since it needs to work on all web pages, including on new domains; we need to account for variations in page layouts and structure across websites. Further, we need to deal with low-quality pages on the web with limited structure.

We have built an ML-powered extraction system to solve this problem, using schema.org annotations as training data. Our extraction system operates in two phases. In the first phase, we generate raw event information from individual web pages. To do this, an *event page classifier* predicts if a web page contains any event information; this is then followed by a *single/multiple classifier* that decides if the page contains a single event or multiple events; the first phase concludes by applying *event extractors* that extract the key fields of a public event (the title, the date/time information, and the location information). In the second phase, we further improve the extraction quality via three novel algorithms, *repeated patterns*, *event consolidation* and *wrapper induction*, which are designed to use the raw event extractions as input and generate events whose quality is significantly higher. We evaluate our extraction models on two large scale publicly available web corpus, Common Crawl and ClueWeb12. Experimental analysis shows that our methodology achieves over 95% extraction precision and recall on both datasets.

CCS CONCEPTS

• **Information systems → Data management systems**;

KEYWORDS

structure data, event data extraction, consolidation, wrapper

ACM Reference Format:
Qifan Wang, Bhargav Kanagal, Vijay Garg, D. Sivakumar. 2019. Constructing a Comprehensive Events Database from the Web. In *The 28th ACM International Conference on Information and Knowledge Management (CIKM '19), November 3–7, 2019, Beijing, China.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3357384.3357986

CIKM '19, November 3–7, 2019, Beijing, China

ACM ISBN 978-1-4503-6976-3/19/11.
https://doi.org/10.1145/3357384.3357986

1 INTRODUCTION

Large-scale structured data repositories are becoming increasingly important to empower new experiences in applications like web search as well as enable smart assistants (e.g., Amazon Alexa, Google Assistant) to do complex tasks (e.g., "Show me kid-friendly events in San Francisco this weekend"). The web contains billions of websites that contain rich event information, although it is mostly unstructured. In this paper, we consider the problem of constructing a comprehensive database of events taking place around the world, by automatically extracting from the web. We define an event as a public assembly which contains three crucial pieces of information — the *title* of the event, the *date/time* of the event and the *location* where the event will take place. We are interested in events that users may want to add to their calendars and potentially attend. This is in contrast to many other types of events, such as natural disasters, pandemics, political elections or social media news.

There are several major challenges in event extraction from an arbitrary web page. First, event pages identification. Before extraction, we need to know if a given page actually contains any event information. Due to the huge volume of web pages, a small page classification error would lead to a large number of mistakes ("false positives," or spurious events). Therefore, it is crucial to develop an event page classifier that achieves very high precision. Second, event page type classification. Knowing whether a page contains a single event or multiple events is important, since events have different presentations on these two types of pages. In the case of a single-event page, the different pieces of information about the event can appear anywhere on the page, whereas the information within multiple-event pages could appear in "factored form," that is, the common parts (e.g., the artist or the venue) could appear once but individual listings could appear in a table. Third, event information extraction. To scale to the entire web, our techniques cannot have domain-specific parsing such as template based extraction, and need to be fully general as new pages and domains are continuously being added. We need to handle a variety of web pages with differing structures. Moreover, information about an event can be distributed on different pages, or duplicated across sites, and we need the ability to handle both situations.

The general problem of information extraction from web pages has been studied extensively in recent years. Of these, template induction (or wrapper induction) [7, 24, 30] has proven to be successful for extracting relations from web pages. However, these techniques do not scale to the whole web as obtaining accurate ground truth for all the event domains is expensive. Moreover, the wrappers go out-of-date quickly because page structure changes frequently, and require periodic updating. Furthermore, as new (large) events occur, there is a steady stream of new domains that carry valuable event information. Recently, Foley et al. [9] proposed using machine-learned extractors with schema.org markup [39] to construct training data. We follow their lead but overcome two

significant problems in their work that makes it unusable for the Web: first, their method applies only to web pages with exactly one event, and second, the 76% precision they reported is unsuitable for real world applications.

In this paper, we develop a fully automated methodology to extract event information from arbitrary web pages. Our method proceeds in two phases: a raw event extraction phase that culminates in 84–85% precision, and a precision enhancement phase that improves the extraction quality to very high (and acceptable for wide usage) levels. We train an *event page classifier* to recognize event pages with high precision. To understand the layout of the page, we train another *single/multiple classifier* to predict if the page contains just one or several events. Based on the page layout, we design two *event extractors* to extract event information. All models are deep neural networks, trained using schema.org markup data as labeled examples. To further improve extraction quality, we propose three novel algorithms, *repeated patterns*, *event consolidation* and *wrapper induction* to post-process the raw event extractions from the model.

Our technical contributions are as follows:

(1) We develop an ML-powered system to do end-to-end event extractions from any page on the web. Our system detects if the page contains an event, predicts the page structure, and extracts events from the page.
(2) We utilize repeated sub-structures on web pages to improve extraction precision and recall. We develop a consolidation method to cluster event extractions from the model, remove duplicates, and identify high-precision events.
(3) We develop algorithms to automatically construct domain-level wrappers for extraction. We design fully automated algorithms to identify high precision templates without human raters, using statistical analysis of the template rules.

2 RELATED WORK

2.1 Information Extraction

There is a large body of work on information extraction from the web, whether directly, through template extraction [6, 7, 16, 30, 41], or through the more general idea of region extraction and classification [3, 5, 9–11]. Template extraction techniques have received a lot of attention recently to improve the performance of search engines, clustering, and classification of web pages. These methods learn desired patterns from the unstructured web data and construct templates for information extraction. Region extraction methods try to classify portions of a web page according to their specific purposes, i.e. navigation links, anchors and main content are studied in part to index only the content present on web pages. Part of our task could fit into this context: an attempt to classify regions as either containing an event or not. A detailed survey of region extractors in web pages, including those that leverage visual features can be found in [34]. As mentioned earlier, the work of Foley et al. [9] shares similar goals to this work. However, it uses simple naive-Bayes to classify the event page and SVM methods to get the score for each event field, and does not achieve the level of precision needed for real-world applications.

Additionally, there are a number of works that focus on repeated structure for extraction, using HTML tables [1, 12, 23], repetitive command tags or terms in general [40]. These techniques do not require supervision, but require repetitive structure and make the assumption that there will always be a multitude of records to extract. Finally, there are several recent works about extracting data from web pages that were generated from the same templates [14, 35]. In contrast to these works, we are aiming at extracting events at all scales (large events to hyper-local ones), which are usually not generated from a small number of pre-defined templates.

2.2 Relation Learning

The task of event extraction is essentially learning to extract event title, date and location which are correlated to each other. Therefore, relation extraction/learning research [28, 31, 37, 47] is also related to our work. Petrovski et al. use schema.org annotations in products to learn regular expressions that help identify various product attributes [29]. Lockard et al. [24] propose to generate training labels by aligning an existing knowledge base with a web page and leveraging the unique structural characteristics of semi-structured websites. A classifier is trained based on the labels to predict new relation instances. Recently, Li et al. [22] discuss the problem of discovering related events from web pages with a graph-based framework. Most recently, Wu et al. [44] design a machine-learning-based knowledge base construction system to extract relations conveyed jointly via textual, structural, tabular, and visual expressions.

2.3 Natural Language Processing

Related work within the natural language processing area [33, 42, 50] has focused on extracting information from a single sentence or a paragraph by formulating it as a sequence labeling task, relying on distant supervision with conditional random fields. Zheng et al. [50] propose an attention model to extract attributes from product pages. Seo et al. [33] develop a machine comprehension framework for answering a query from given context. However, these methods cannot be directly applied for event extraction since they do not work for extracting information from entire web pages, or when the information is not contained in well-formed sentences.

2.4 Event Detection in other Domains

There is also work on detecting events within microblogs or real-time social media updates [17, 18, 26, 36, 43, 49]. Yuan et al. [46] focus on recommending relevant merchandise for seasonal retail events, based on item retrieval from marketplace inventory. Zhang et al. [48] propose an event detection method that enables real-time local event extraction from geo-tagged tweet streams. Becker describes identification of unknown events and their content [2], but focuses on trending events on social media sites. Our work, in contrast, is aimed at finding events in an offline manner in order to present them in a search or recommendation system to users. There has been work on another topic in extraction of named events from news [21, 45]. Again, the definition of these news events is different from our event definition.

3 OVERVIEW

Our event extraction pipeline, as shown in Figure 1, consists of six key components: *event page classifier*, *single/multiple classifier*,

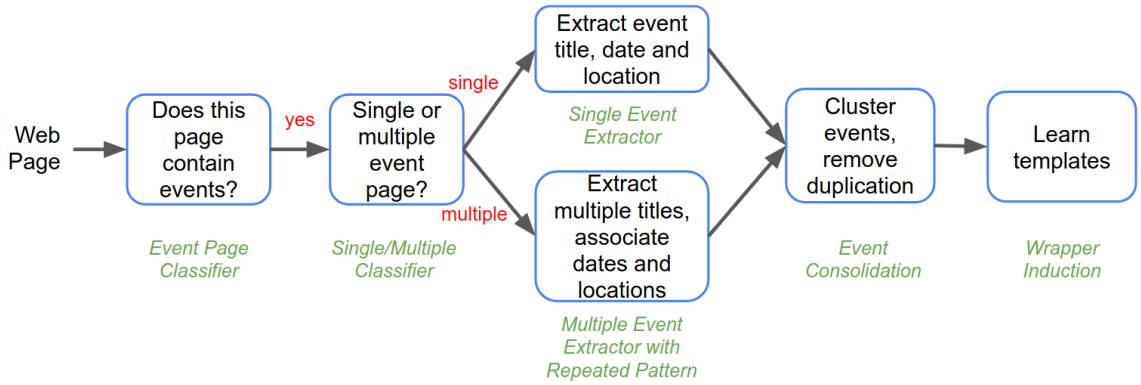

Figure 1: An overview of event extraction pipeline.

single event extractor, *multiple event extractor* with *repeated patterns*, *event consolidation* and *wrapper induction*.

The first component, the *event page classifier*, determines whether a given web page contains event information. If so, we have another classifier, *single/multiple classifier*, to determine whether the page talks about a single event or multiple events. The two types of pages are structurally different and thus two separate extraction models are developed, i.e., *single event extractor* and *multiple event extractor*. The extracted raw events are then fed into our *event consolidation* model in order to remove duplication and improve extraction quality. The last component, *wrapper induction*, learns automatic templates to produce near-perfect events.

Event page classifier, *single/multiple classifier*, *single event extractor* and *multiple event extractor* are all deep neural network models learned using training data from schema.org annotations, while *event consolidation* is a graph-based clustering model. *Wrapper induction* contains a clustering model and an aggregation framework. We will present the detail of each component in the next two sections.

4 EVENT EXTRACTION

4.1 Event Page Classifier

In order to extract events from web pages, a major problem is to determine whether a web page talks about an event. At the moment, there is not even a good estimate of what percentage of web pages are about events. On the other hand, event extraction requires extremely high precision classifiers, especially for the event page classifier. Therefore, it is important to develop an accurate event page classifier to select all event pages from the web corpus.

The event page classifier is built in a supervised learning framework. We use the part of the web that has semantic web annotations to learn about the rest of the web. The schema.org dataset contains, from our calculation, 1.5 billion annotated web pages. Around 16 millions of those annotations are event related. Our training set contains all web pages with schema.org event annotations as positive examples, with a 10% random sample of the remaining pages as negative training examples.

There exist several issues within the training data. It makes the assumption that schema.org annotations are used correctly, which is not totally true in practice. We have seen some examples where the annotations were used in unexpected ways. For example, a recipe website was marking recipes as schema.org/FoodEvent and event websites such as sf.funcheap.com that mark their events pages as schema.org/WebSite but not as schema.org/Event. Our way to deal with this problem is to identify large websites that do not use the annotations correctly and to remove them from our training set by adding them to a blacklist. We also remove pages with more than 10,000 words as these are unlikely to be event pages, and even if we could correctly identify them as event page, they would not be useful in later stages as it would be hard to extract events from them.

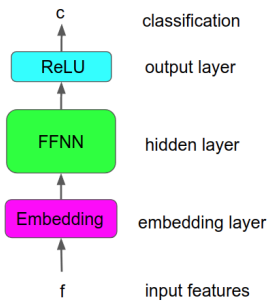

Figure 2: The architecture of the event page classifier.

We build the event page classifier model, as shown in Figure 2, using a deep neural network under the TensorFlow framework. In particular, we adopt a two layer feed forward neural network (FFNN) [38] to construct our model. An embedding layer is attached to the input features (we will discuss the features later), while a rectified linear unit (ReLU) is applied as the output layer to produce the final classification result:

$$C = \boldsymbol{ReLU}(\boldsymbol{FFNN}(embedding(f))) \quad (1)$$

We extract a set of features from the web page for the event page classifier, including *term frequency*, *web entity* [27], *anchors*, *URL segments* and *page title*. For the *term frequency* feature, terms are stemmed and normalized after removing all stopwords. Moreover,

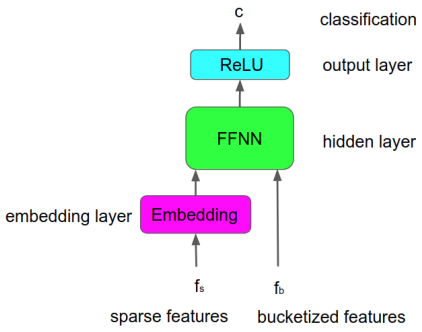

Figure 3: The architecture of the single/multiple classifier.

we remove features that occur on less than 100 different domains to make sure the features are general enough and not specific to our training set. We also remove features that occur on more than 30% of domains, because they are not specific enough to distinguish event from non-event pages. Due to the sparsity of the features, an embedding layer is added to the model.

4.2 Single/Multiple Classifier

Web pages may contain single or multiple events and we observe that these pages have largely differing HTML structures. Also, their content is geared towards different aspects of the event. In a single-event page[1], the primary focus is devoted to the event itself and its description. By contrast, a multiple-event page[2] is usually about details of a venue or it could be a list of disparate events hosted in an event aggregator domain. Motivated by this, we design separate extractors for single-event and multiple-event pages.

The single/multiple classifier takes an event page as input and classifies it into single-event page or multiple-event page. Knowing a page contains a single event or multiple events is important as events have different presentations on these two types of pages and thus different event extractors would be applied based on the page structure. Event information can appear anywhere on a single-event page, while events are often presented in a table with repeated structure on a multiple-event page.

The pages with schema.org event annotations are used as the training data to build our single/multiple classifier. These event pages are labeled as single-event pages or multiple-event pages, depending on the number of event annotations on the page. If there is more than one event annotation on the page, we label the page as a multiple-event page, otherwise it is labeled as a single-event page. However, in some single-event pages, the event is mentioned multiple times in different regions across the page with multiple event annotations. We solve this issue by grouping the markup events based on event title, date and location, and identify unique events on the page to remove duplicated annotations.

The features we extracted from the page to construct single/multiple classifier are: number of dates, number of locations (as multiple-event page usually contains multiple dates and locations), URL segments, number of HTML tables and lists, unigram and bigram

[1]Example: http://csinewsnow.com/?p=111711
[2]Example: https://www.eventbrite.com/d/ca--mountain-view/events/

Type	Features
sparse	unigram, bigram, trigram, x_path, entity, html_tag
boolean	is_bold, is_italic, is_underline, is_in_table, has_date, has_location
bucketized	url_match_rank, horizontal_pos, vertical_pos, anchor_match_rank, font_ratio

Table 1: Features used for title extraction.

texts. Note that for those numeric features (e.g., number of date), we bucketize them into several buckets using the logarithm function to remove noise. For the text-based features, we use the original text on the page instead of a stemmed representation, since the plural or singular form of the word is important. For example, "events", "concerts" and "shows" usually indicate multiple events, while "event", "festival" and "show" might talk about one single event.

We build the single/multiple classifier model using FFNN as shown in Figure 3. The model is similar to the one we employed for the event page classifier. The difference is that no embedding is applied for bucketized features:

$$C = \textbf{\textit{ReLU}}(\textbf{\textit{FFNN}}(concat\{embedding(f_s), f_b\})) \quad (2)$$

here *concat* is the concatenation operation. f_s and f_b represent the sparse features and bucketized features respectively.

4.3 Single Event Extractor

An event is composed of three main parts: title, date and location. For a single-event page, we first extract the event title, then event date and event location are jointly associated to the event title.

Title Extraction. We approach the task of extracting the event title as a supervised learning problem. All single-event pages in the training set are used for learning the event title extractor. All the text nodes on the page that contain less than 20 words are considered as candidates. In a single-event page, the event title usually appears multiple times on different regions of the page. Therefore, we calculate the Jaccard similarity between each text node and the event title, and label those candidates with high similarity values as positive examples. All other text nodes on the page are marked as negative examples.

The features we extracted from each candidate for building the title extractor are listed in Table 1. For example, is_bold means whether the candidate text node is in bold or not. $font\_ratio$ is the normalized font size. url_match_rank is the rank of the candidate among all candidates, based on the similarity with the URL. A feed forward neural network, similar to the architecture in Figure 3, is used to train the title extractor[3]. During inference, we select the text node with the highest prediction score as the event title.

Joint Date and Location Extraction. Extracting the event date and location from a single-event page is a very challenging task, as both event date and location can be mentioned once or multiple times on any region of the page depending on the information

[3]Boolean features are a special type of bucketized features with two buckets, i.e., {0, 1}.

Category	Features
date	is_in_title, is_in_url, group_size, nearby_texts, closeness_to_title, html_tag
location	is_in_title, is_in_url, group_size, nearby_texts, closeness_to_title, html_tag, anchor_rank, has_geo_info
pair	geo_distance, byte_distance

Table 2: Features used for joint date and location extraction.

Figure 4: Multiple event extraction with repeated pattern.

distribution. Many non-event dates and locations, considered as noisy data, usually co-exist with the desired event information, imposing additional difficulty for the extraction. Moreover, event date and location are highly correlated, and typically appear close to each other on the page. Traditional methods (e.g., [9]) that extract them independently might not be accurate. Therefore, we propose a joint date and location extraction model, which extracts the event date and location simultaneously for a given event title.

Pattern-based approaches to tagging of dates and addresses in text achieve reasonably good results, as evidenced by the approaches in [4, 25], as well as some prior work on address detection [8, 32]. Therefore, we use pattern-based baselines to obtain all dates and addresses on the page as the event date and location candidates for our extraction. We first cluster the same date together to form a date group, and represent the date groups as $\boldsymbol{D}_1, \boldsymbol{D}_2, ...\boldsymbol{D}_n$. Similarly, location groups are defined as $\boldsymbol{L}_1, \boldsymbol{L}_2, ...\boldsymbol{L}_m$. In our joint extraction model, each pair of date group and location group $(\boldsymbol{D}_i, \boldsymbol{L}_j)$ is a candidate example. From the schema.org annotation, we can easily label the date and location pair corresponding to the event date and location as positive. All other pairs are marked as negative training examples. A bunch of features are designed to represent the candidate pair as listed in Table 2, including date and location related features as well as features representing the correlation between date group and location group. For example, *is_in_title* is a boolean feature indicating whether the date or location is in the event title. *closeness_to_title* means the closeness of the candidate to the event title. *geo_distance* represents the spatial distance (in terms of pixels) between the date group and location group rendered on the page. Note that all distance related features are normalized. A two layer FFNN is used to train the model. In the inference process, the date and location pair with the highest prediction score is selected:

$$(\boldsymbol{D}, \boldsymbol{L}) = \text{argmax}_{(\boldsymbol{D_i}, \boldsymbol{L_j})} P(\boldsymbol{D}_i, \boldsymbol{L}_j) \quad (3)$$

5 ACHIEVING NEAR-PERFECT EXTRACTIONS

5.1 Multiple Event Extractor with Repeated Patterns

As aforementioned, a multiple-event page is usually about details of a venue or an artist, or it could be a list of events hosted in an event aggregator domain. Events on a multiple-event page naturally form a table, a list or more generally a repeated pattern. There is a number of works on repeated structure extraction [1, 40], which yields impressive results. Therefore, we develop our multiple event extractor on top of these repeated patterns.

We apply a similar strategy as shown above for single-event pages to extract titles of multiple events. We start with a corpus of multiple-event pages and use the same labeling process, i.e., mark all text nodes with event title annotation on the page as positive example, with others negative. In addition to the features used for the single event extractor, we add a feature, *is_in_repeated_table*, indicating whether the text node candidate is inside a repeated pattern. However, different from the single event extractor, during the inference process, directly picking the text node with the highest score is not suitable since there are multiple events. Instead, we select all text nodes whose prediction scores are larger than a predefined threshold, as the titles of multiple events.

Repeated patterns are then used to align the event titles, as well as to associate date and location for each of the events. In particular, we first search over all repeated patterns on the page, and find the one containing all the events by matching the extracted/predicted event titles. Then we identify the date and location columns in the matched repeated pattern to extract event date and location. Consider the example in Figure 4: Suppose that the model extracts titles for events 1, 3, 4, 6 (green check marks) but that it was unable to predict titles for the other events (title scores lower than threshold). In this case, we can search for the predicted titles in the repeated pattern – we can now establish that 4 out of 6 cells in column 1 correspond to event titles. From this we can infer that column 1 is really an event title column. Similarly, we can find the predicted dates and locations in the repeated pattern to establish that columns 2 and 3 correspond to dates and locations respectively. We can now simply extract information for all events directly from the repeated pattern. This approach increases the accuracy of multiple event extractor to 100% precision with 100% recall. Note that we can use this approach only if we can find a repeated pattern on the web page. If we do not find any repeated pattern, then we do not emit any extractions for the web page.

5.2 Event Consolidation

Event information might have different representations on different web pages, even for the same event. For example, Figure 5 shows a set of extracted events from the web corpus, where the first column is event title, the second column is event date, the third column is event location, the fourth column indicates whether it

Peninsula Symphony	2016-11-05	21250 Stevens Creek Boulevard, Cupertino 95014, United States	SINGLE	true	https://etrigg.com/event/penin
Conrad Tao	2016-11-05	21250 Stevens Creek Boulevard, Cupertino 95014, United States	SINGLE	true	https://etrigg.com/event/conra
Peninsula Symphony	2016-11-05	21250 Stevens Creek Boulevard, Cupertino, California, United States	SINGLE	true	http://events.sfgate.com/event
Peninsula Symphony in Cupertino	2016-11-05	Flint Center	MICRODATA	true	http://sanjose.eventful.com/ev
Peninsula Symphony	2016-11-05	Flint Center	MICRODATA	true	https://www.evensi.us/peninsul
Peninsula Symphony 11/05/2016	2016-11-05	Flint Center	MICRODATA	true	http://conradtao.com/shows/pen
Conrad Tao	2016-11-05	Flint Center	MICRODATA	true	https://evnnt.com/cupertino/21
Peninsula Symphony	2016-11-05	Flint Center for the Performing Arts	MICRODATA	true	https://eventseeker.com/event/
Peninsula Symphony	2016-11-05	Flint Center	SINGLE	false	https://www.facebook.com/event
Peninsula Symphony	2016-11-05	Flint Center	MICRODATA	true	http://conradtao.com/shows/
Peninsula Symphony	2016-11-05	Flint Center	MICRODATA	true	http://conradtao.com/shows/
Peninsula Symphony: Bremer, Schumann & Elgar	2016-11-05	Flint Center	MULTIPLE	false	http://www.chadgoodmanmusic.co

Figure 5: An example of one event extracted from different web pages.

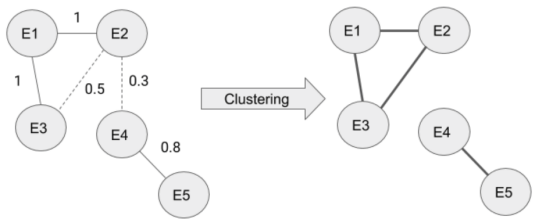

Figure 6: Event consolidation via graph based clustering.

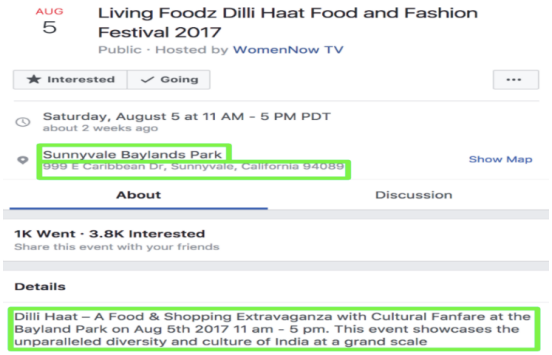

Figure 7: The event location appears in multiple nodes.

is extracted from a single or multiple event page or directly from the schema event annotation ("MICRODATA" is one of the markup sources), and the last column is the URL of the web page. As we can see in Figure 5, all rows actually talk about the same event but have slight variations in event details. In this case, it is clear that "Peninsula Symphony" would be a better event title than "Conrad Tao". On the other hand, we don't present all these duplicated events to users, but just one best extraction. Therefore, we design an event consolidation framework for removing duplication as well as improving the event extraction.

A straightforward solution is to perform a fuzzy match between every two events. However, this would require a quadratic number of matches which is non-trivial for hundreds millions of events. In this work, we propose a graph-based clustering algorithm for event consolidation. In particular, as shown in Figure 6, a graph $G = \{V, E\}$ is constructed over all extracted events $V = \{Events\}$, and the edges of the graph are assigned weights $W = \{W_{ij}\}$ representing the similarity between two events. Event title, date and location are fields extracted for any event and are used for computing similarity. Two nodes (events) in the graph are connected if and only if both their event date and location are same, resulting in a very sparse graph. Note that the event location can be venue name like "Google Headquarter", or street level address like "1600 Amphitheatre Parkway" or city name like "Mountain View, California". We use the Google Maps API for event location to disambiguate them. The Jaccard similarity of the event titles is used as the edge weight between two nodes. Then we apply the graph-based clustering algorithm [15] on the sparse graph G to obtain event clusters (more details are provided in the supplementary material due to space limitation). An event cluster with more than one event is marked as a high confidence event, while an event cluster with size one is considered as a low confidence event. This way, identical events are clustered together and we select the event title that appears most times in the cluster as the event title, e.g., it would be "Peninsula Symphony" in the above example.

5.3 Wrapper Induction

We make the observation that there are several event pages that have the same page structure, e.g., typically most event pages from the same domain have similar structure (generated by a computer program). We also observe that the event fields (e.g., event title) should be present in the same XPath on all pages. In this section, we present the details of a novel approach that allows us to exploit the combination of page structure along with the machine-learned extractions to achieve perfect event extractions for a large number of web pages. Note that the machine learned extractions may not be correct on all pages. But as long as at least a majority of the pages have the right extractions, we can aggregate the ML extractions based on their XPaths, and for each field, compute the XPath with the largest count. This will then enable us to turn a weak model (that is at least 50% accurate) into one which is 100% precise. Note that the recall also improves to 100% for pages in this domain. This approach has been called by various names in the literature including "Wrapper Induction" and was first proposed by Kushmerick [20].

Wrapper induction as described above requires two main steps, clustering and aggregation. The purpose of clustering is to detect event pages that have similar HTML structure. It is a challenging task to identify pages with the same HTML structure since no two pages are typically identical. Aggregation is aiming at aggregating relevant XPaths across pages within a cluster to generate an extraction template. In this work, the web pages are clustered based on their DOM[4] structure as well as by their URLs using a decision tree based model [13, 19]. The web pages in the same cluster share similar DOM structure. The aggregation component is mostly straightforward. For each event field, i.e., title, date and location, we count the number of occurrences of each of the XPaths corresponding to all pages in the domain. Since we expect the ML model to be correct in the majority, the XPath which is most frequent should be the right rule for extraction, for each of the fields. In many event pages, the same event information can be repeated in multiple nodes. For instance, on the page in Figure 7, the event location is repeated using venue name, a Google map link as well

[4]https://en.wikipedia.org/wiki/Document_Object_Model

Figure 8: Illustration of learned wrapper. The red box corresponds to the DOM node for event title. Similarly, the purple and the green boxes correspond to the location and date respectively.

as a street address. In this case, we aggregate all occurrences of the correct locations and select the best XPath based on empirical evaluation (e.g., top most mention or closest mention to event title). In our implementation, we require the best XPath to appear in at least 50% of the extractions. Figure 8 illustrates a wrapper learned for a specific page cluster, with all three field nodes.

The previously discussed wrapper induction algorithm primarily works for single-event pages. For multiple-event pages, it is difficult to directly generalize the position for each individual event. Since they generally occur in different positions. Instead, we generalize the start position of the repeated pattern from which the events are extracted. In addition to the start position, we also make note of the specific column indices from which we extract the event fields. For example, in Figure 4, the title is from column 1, the date is from column 2 and the location is from column 3.

6 EXPERIMENTS

In this section we describe the experimental setup, data sources and insights we have obtained. We build and test our event extraction models using the schema.org dataset, which contains roughly 16 million event pages. A random split of 90% data is used for training, and 10% holdout set is used for model validation. We use 2 hidden layers for all deep neural network models, with 1024 and 32 hidden units respectively. The embedding dimension for each sparse feature is set to log(V), where V is the feature vocabulary size. L1 and L2 regularization are used in model training, with both 0.001 weights.

As aforementioned, we train our models on the schema.org dataset, which contains 16 million event related pages with 2.5 million complete events. We evaluate the extraction models on two widely used benchmarks: the Common Crawl[5] corpus and the ClueWeb12[6] dataset. Both datasets are designed to support research on information retrieval and related tasks. Common Crawl contains more than 250 TiB of content from more than 3 billion web pages while ClueWeb12 consists of 0.7 billion web pages.

For the schema.org holdout set, it is easy to obtain the evaluation metrics since there are ground-truth labels/annotations. For the Common Crawl and ClueWeb12 datasets, we randomly sample 3000

[5]http://commoncrawl.org/connect/blog/
[6]https://lemurproject.org/clueweb12/

	Page Classifier		*Single Multiple*
Dataset	Precision	Recall	Accuracy
Schema.org	97.86	88.52	98.69
golden set	99.10	98.50	97.30
Common Crawl	99.23	90.48	98.36
ClueWeb12	99.06	89.75	98.19

Table 3: Evaluation results of event page classifier and single/multiple classifier on different test data.

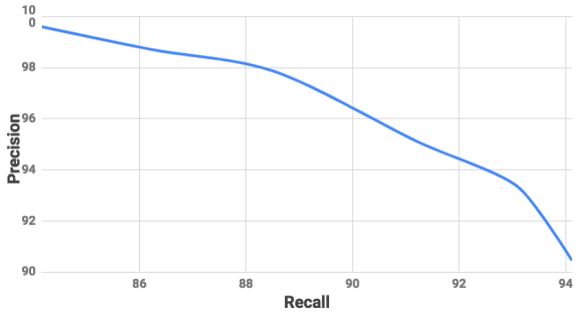

Figure 9: Precision-Recall curve of the event page classifier.

results of each task and obtain judgments from human raters. We only ask binary questions for all tasks, including "Is this an event page?", "Is this a single event page or multiple event page?", "Is the extracted event title correct?", etc. We pay raters $0.01 for each judgment. Each item is sent to three raters for judgments, and we mark it correct if at least two raters mark it correct.

6.1 Evaluation of Page Classification

The event page classifier and the single/multiple classifier are two key components in our event extraction pipeline. To achieve good model quality, we also manually create another golden test set containing 1000 event pages and 1000 non-event pages. For the event pages, there are 700 single event pages and 300 multiple event pages. There is no event annotation on any page of this golden set. The purpose of this set is for fast iteration on model tuning, as the holdout set might not be representative of pages without schema.org annotations. We want to train robust models that not only perform well on schema.org pages, but also behave well on non-schema.org pages.

Standard precision and recall metrics are used to evaluate the event page classifier. For single/multiple classifier, we use accuracy to measure the performance. The results are reported in Table 3. It can be seen that both classifiers achieve good performance, above 97% precision/accuracy, consistently across all datasets. The recall value of the page classifier is not very high, yet still reasonable, around 90%. We can always increase the recall by lowering the score threshold of the classifier, with some sacrifice in terms of precision. However, the event consolidation and wrappers handle most of the mistakes, which will be described in detail below. We also report the precision-recall curve on the holdout set in Figure 9.

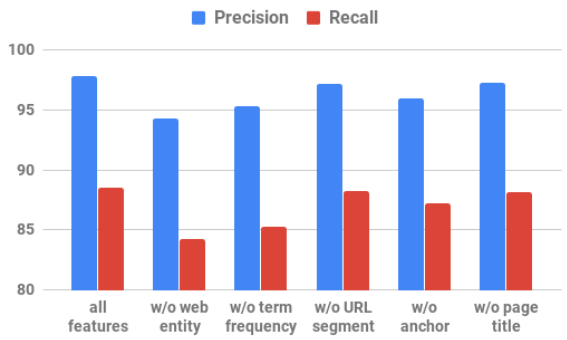

Figure 10: Feature importance chart of event page classifier.

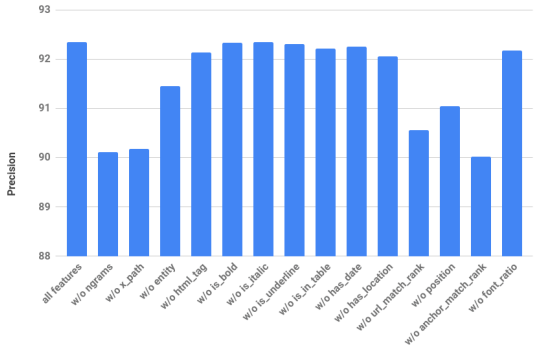

Figure 11: Feature importance chart of event title extractor.

	Title	Date	Location	Event
Schema.org	92.34	97.04	95.24	86.31
Common Crawl	88.27	96.33	91.47	83.84
ClueWeb12	87.83	95.96	92.61	84.07
ClueWeb12 [9]	36.00	32.00	66.00	76.00

Table 4: Precision results of event fields extraction on different datasets.

As shown in the figure, the model is able to achieve 90% precision with 95% recall.

In order to understand which features are important for constructing these classifiers, we conduct a set of feature importance experiments on the holdout set using leave-one-out cross validation. Specifically, we build and evaluate the model performance by removing one feature from the feature set. The precision and recall results of the event page classifier are shown in Figure 10. It is clear that web entity [27] is the most important feature for the event page classifier. Without it, both precision and recall decrease 3% to 4% in terms of absolute values. We conduct similar experiments on the feature importance for single/multiple classifier. The results indicate that number of dates/locations and number of HTML tables are the most useful features, which is intuitive as events on multiple event page usually form repeated patterns/tables. The feature performance chart is presented in the supplementary material.

There are several points that are worth mentioning here. First of all, by applying the event page classifier on the Common Crawl and ClueWeb12 datasets, we found that the event page ratios for both datasets are less than 1%, which is consistent with the event page ratio, around 1.1%, in the schema.org dataset. Secondly, the number of single event pages is significantly larger than the number of multiple event pages. The ratios are around 9.8 and 9.2 for Common Crawl and ClueWeb12 datasets respectively.

6.2 Evaluation of Event Extraction

We evaluate our event title, date and location extraction on all datasets. The event is marked as correct if all its extracted fields are correct. In order to get a full comparison, we also import the reported results from a recent work by Foley et al. [9] as a baseline. The precision results are shown in Table 4. We can observe that the proposed extraction models behave consistently over the schema.org holdout set, the Common Crawl and the ClueWeb12 datasets, where event date has about 96% precision, with title and location around 90% precision. Recall that we sent out each extraction to three raters for evaluation, and only count the extraction correct if at least two raters said so. We also observe from the raters' results that there are inconsistent judgments for certain pages. There are cases, for instance in event title evaluation, where only one rater marked the extracted title correct while the other two raters noted "title is close to the actual event title". One example is "Lady Gaga concert in SAP center" versus "concert in SAP center". Raters tend to mark the latter one as "close title". If we relax the criteria to: extraction is correct if at least one rater marks it as correct, the precision of event title would increase from 88% to 94%. It can be seen from Table 4 that our extractors perform much better than the ones in [9] on the ClueWeb12 dataset. This can be attributed to the deep neural network models developed in our method, compared to the linear extractors used in [9]. Moreover, their approach didn't pay attention to the page type. Different from their work, we employ separate extraction strategies for single and multiple event pages.

We conduct feature importance experiments on different event field extractors, similar to the one described above. The precision result of the event title extractor is reported in Figure 11. From the figure we can observe that *x_path*, *ngrams* and *anchor_match_rank* are the most important features for event title extraction. For instance, without *x_path* feature, the precision value drops about 2%. Similar patterns can be observed for *ngrams* and *anchor_match_rank* features. We also found that the most useful features for event date and location extraction are *group_size* (how many mentions of this date or location are there on the page), *closeness_to_title* and *nearby_texts*. These findings are consistent with our expectation. For example, if one date is mentioned in a number of places on the page, it is likely that this date is the event date. Similarly, if a particular date is very close to the event title, it also tends to be the event date.

We conduct error analysis over the extraction mistakes on event date and location (title mistakes are mostly due to "title close" or not exact event title as discussed above). We identify several main categories of mistakes. The largest group of mistakes is that event date/location is not mentioned or is in an image without HTML text. In other words, the event date/location is not even a candidate

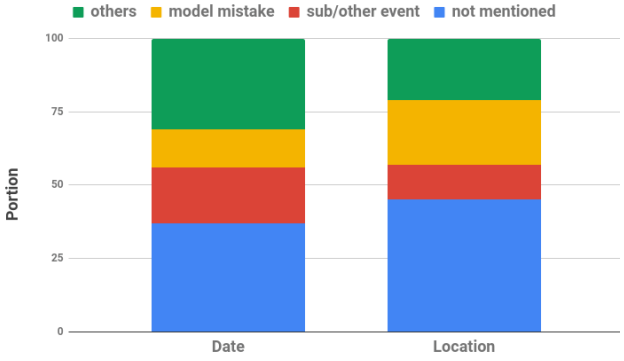

Figure 12: Error analysis of date and location extractions.

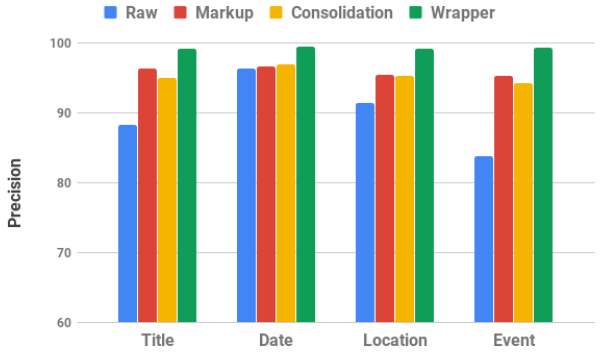

Figure 13: Event extraction comparison of raw extraction, markup extraction, consolidation and wrapper on Common Crawl dataset.

for our model to pick. The other type of error is that we extract date/location for a sub-event or other-event on the page. For example, our model extracts a sub-event location for a huge festival or a date of a presentation within a conference but not the main date of the conference. There are few other extraction errors made by the models. The complete analysis result is shown in Figure 12. However, most of these errors are handled by event consolidation and wrapper induction which will be described in the next section.

6.3 Evaluation of Event Consolidation and Wrapper Induction

Event consolidation clusters the same event together and selects the best event title, date and location from each cluster. We sample events from the event clusters with more than one event (high confidence) for evaluation. For wrapper induction, we randomly sample 3000 learned templates, and select one event from each template. We compare our raw extraction (pre-consolidation) with post-consolidation extraction and wrapper extraction. Note that no events that are sent out for evaluation contain any markup (event annotations). In addition, we also compare the extraction precision with markup precision from the schema.org event annotation. The

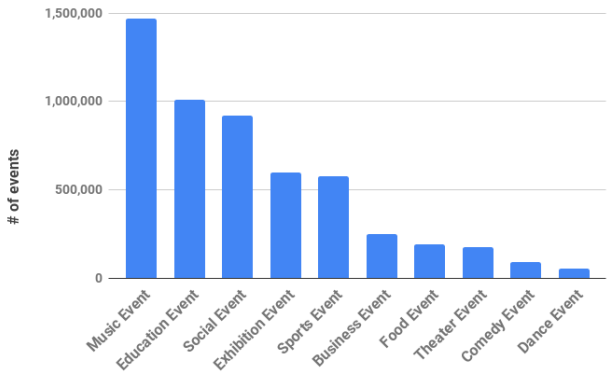

Figure 14: Event category distribution.

comparison results on Common Crawl dataset are shown in Figure 13. It can be seen that wrapper extraction achieves the best performance among all extractions with near perfect precision. The consolidation improves the extraction quality by 10%, from 84% to 94%, compared to the raw extraction, while wrappers further boost the performance to 99%. Moreover, we also found that wrappers is able to extract 28.6% additional events, which improves the recall of the extraction significantly. However, there are two limitations of the wrapper induction approach. First, the learned wrapper would fail to work if the page structure for the domain changes, and thus templates need to be updated periodically. Second, templates are learned for a set of domains but not all, as many event pages do not have fixed HTML structure and no rule could be extracted. We can also observe from the figure that the consolidated events achieve a similar precision value to markup events. It indicates that post-consolidation events reach a reasonable quality as those schema.org events that have annotations. We observe similar patterns on the ClueWeb12 dataset, the result is presented in the supplementary material due to space limitation.

Extraction Failures. There are two types of pages that our extraction pipeline would fail for. In the first group, the event information is not presented in the HTML text. For example, the event page is just an image containing event details. Our event extractors will not work on this type of pages, as they rely on the HTML texts on the page. The second group contains all multiple event pages with no repeated patterns/tables. Both our multiple event extractor and wrapper will remain silent since no event table is found. Although these failures would result in some recall losses, the overall precision of the event extraction remains the same.

6.4 Event Database Summarization

Our extraction system extracts about 5.4 million unique events through merging and deduplicating the events extracted from Common Crawl and ClueWeb12 datasets, with over 95% precision as shown above. In this database, 4.2 million events are obtained from the high confidence consolidated events while 1.2 million events are extracted with the wrappers. We build another multi-class event classifier to classify an event to one of the pre-selected categories.

The classifier is built using the schema.org event category annotation. The same feature sets and deep model structure from event page classifier are used to construct this classifier. The pre-selected categories are music event, education event, social event, exhibition event, sports event, business event, food event, theater event, comedy event and dance event. We then apply this category classifier over all events. The event category distribution is shown in Figure 14. As we can see from the figure, music event is the top-most category containing roughly 1.4 million events. Other categories such as sports event, education event and social event also contain a large number of events.

7 CONCLUSION

In this work, we develop a fully automated methodology to extract event information from arbitrary web pages. In order to build an end-to-end event extraction pipeline, we design several deep models leveraging schema.org event annotations as training examples. We further improve extraction quality by introducing three novel algorithms, *repeated patterns*, *event consolidation* and *wrapper induction*, to post-process the raw event extractions. Our large scale experiments demonstrate the effectiveness of the proposed method. In the future, we plan to develop algorithms to extract event meta-data such as event price, ticket link etc. We also plan to explore ranking models with extracted event features for better event recommendation.

REFERENCES

[1] M. D. Adelfio and H. Samet. Schema extraction for tabular data on the web. *PVLDB*, 6(6):421–432, 2013.

[2] H. Becker. *Identification and characteriaztion of events in social media.* PhD thesis, Czech Technical University, 2011.

[3] L. Bing, W. Lam, and Y. Gu. Towards a unified solution: data record region detection and segmentation. In *CIKM*, pages 1265–1274, 2011.

[4] R. Campos, G. Dias, A. M. Jorge, and A. Jatowt. Survey of temporal information retrieval and related applications. *ACM Comput. Surv.*, 47(2):15:1–15:41, 2014.

[5] S. Chakrabarti. Integrating the document object model with hyperlinks for enhanced topic distillation and information extraction. In *WWW*, pages 211–220, 2001.

[6] W. W. Cohen, M. Hurst, and L. S. Jensen. A flexible learning system for wrapping tables and lists in HTML documents. In *WWW*, pages 232–241, 2002.

[7] N. N. Dalvi, R. Kumar, and M. A. Soliman. Automatic wrappers for large scale web extraction. *PVLDB*, 4(4):219–230, 2011.

[8] J. Efremova, I. Endres, I. Vidas, and O. Melnik. A geo-tagging framework for address extraction from web pages. In *ICDM*, pages 288–295, 2018.

[9] J. Foley, M. Bendersky, and V. Josifovski. Learning to extract local events from the web. In *SIGIR*, pages 423–432, 2015.

[10] A. Galina. Sociopath: automatic local events extractor, 2017.

[11] S. Gottschalk and E. Demidova. Eventkg: A multilingual event-centric temporal knowledge graph. In *ESWC*, pages 272–287, 2018.

[12] R. Gupta and S. Sarawagi. Answering table augmentation queries from unstructured lists on the web. *PVLDB*, 2(1):289–300, 2009.

[13] I. Hernández, C. R. Rivero, D. Ruiz, and R. Corchuelo. A statistical approach to url-based web page clustering. In *WWW*, pages 525–526, 2012.

[14] J. L. Hong, E. Siew, and S. Egerton. Information extraction for search engines using fast heuristic techniques. *Data Knowl. Eng.*, 69(2):169–196, 2010.

[15] A. K. Jain. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31(8):651–666, 2010.

[16] C. Kim and K. Shim. TEXT: automatic template extraction from heterogeneous web pages. *IEEE Trans. Knowl. Data Eng.*, 23(4):612–626, 2011.

[17] D. Kim, J. Lee, D. Choi, J. Choi, and J. Kang. Learning user preferences and understanding calendar contexts for event scheduling. In *CIKM*, pages 337–346, 2018.

[18] A. Konovalov, B. Strauss, A. Ritter, and B. O'Connor. Learning to extract events from knowledge base revisions. In *WWW*, pages 1007–1014, 2017.

[19] H. S. Koppula, K. P. Leela, A. Agarwal, K. P. Chitrapura, S. Garg, and A. Sasturkar. Learning URL patterns for webpage de-duplication. In *WSDM*, pages 381–390, 2010.

[20] N. Kushmerick. *Wrapper Induction for Information Extraction.* PhD thesis, Seattle, WA, USA, 1997.

[21] E. Kuzey, J. Vreeken, and G. Weikum. A fresh look on knowledge bases: Distilling named events from news. In *CIKM*, pages 1689–1698, 2014.

[22] C. Li, M. Bendersky, V. Garg, and S. Ravi. Related event discovery. In *WSDM*, pages 355–364, 2017.

[23] G. Limaye, S. Sarawagi, and S. Chakrabarti. Annotating and searching web tables using entities, types and relationships. *PVLDB*, 3(1):1338–1347, 2010.

[24] C. Lockard, X. L. Dong, P. Shiralkar, and A. Einolghozati. CERES: distantly supervised relation extraction from the semi-structured web. *PVLDB*, 11(10):1084–1096, 2018.

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

[26] A. Mishra and K. Berberich. Event digest: A holistic view on past events. In *SIGIR*, pages 493–502, 2016.

[27] P. Pasupat and P. Liang. Zero-shot entity extraction from web pages. In *ACL*, pages 391–401, 2014.

[28] N. Peng, H. Poon, C. Quirk, K. Toutanova, and W. Yih. Cross-sentence n-ary relation extraction with graph lstms. *TACL*, 5:101–115, 2017.

[29] P. Petrovski, V. Bryl, and C. Bizer. Learning regular expressions for the extraction of product attributes from e-commerce microdata. In *Proceedings of the Second International Workshop on Linked Data for Information Extraction (LD4IE 2014), Riva del Garda, Italy, October 20, 2014.*, pages 43–54, 2014.

[30] J. Proskurnia, M. Cartright, L. G. Pueyo, I. Krka, J. B. Wendt, T. Kaufmann, and B. Miklos. Template induction over unstructured email corpora. In *WWW*, pages 1521–1530, 2017.

[31] S. Riedel, L. Yao, and A. McCallum. Modeling relations and their mentions without labeled text. In *PKDD*, pages 148–163, 2010.

[32] S. Schmidt, S. Manschitz, C. Rensing, and R. Steinmetz. Extraction of address data from unstructured text using free knowledge resources. In *13th International Conference on Knowledge Management and Knowledge Technologies, I-KNOW '13*, pages 7:1–7:8, 2013.

[33] M. J. Seo, A. Kembhavi, A. Farhadi, and H. Hajishirzi. Bidirectional attention flow for machine comprehension. *CoRR*, abs/1611.01603, 2016.

[34] H. A. Sleiman and R. Corchuelo. A survey on region extractors from web documents. *IEEE Trans. Knowl. Data Eng.*, 25(9):1960–1981, 2013.

[35] H. A. Sleiman and R. Corchuelo. TEX: an efficient and effective unsupervised web information extractor. *Knowl.-Based Syst.*, 39:109–123, 2013.

[36] A. Spitz and M. Gertz. Terms over LOAD: leveraging named entities for cross-document extraction and summarization of events. In *SIGIR*, pages 503–512, 2016.

[37] F. M. Suchanek, G. Ifrim, and G. Weikum. Combining linguistic and statistical analysis to extract relations from web documents. In *SIGKDD*, pages 712–717, 2006.

[38] D. Svozil, V. Kvasnicka, and J. Pospichal. Introduction to multi-layer feed-forward neural networks. *Chemometrics and Intelligent Laboratory Systems*, 39:43–62, 1997.

[39] N. Tempelmeier, E. Demidova, and S. Dietze. Inferring missing categorical information in noisy and sparse web markup. In *WWW*, pages 1297–1306, 2018.

[40] W. Thamviset and S. Wongthanavasu. Information extraction for deep web using repetitive subject pattern. *WWW*, 17(5):1109–1139, 2014.

[41] K. Vieira, A. S. da Silva, N. Pinto, E. S. de Moura, J. M. B. Cavalcanti, and J. Freire. A fast and robust method for web page template detection and removal. In *CIKM*, pages 258–267, 2006.

[42] W. Wang, Y. Ning, H. Rangwala, and N. Ramakrishnan. A multiple instance learning framework for identifying key sentences and detecting events. In *CIKM*, pages 509–518, 2016.

[43] F. Wu, P. Anchuri, and Z. Li. Structural event detection from log messages. In *SIGKDD*, pages 1175–1184, 2017.

[44] S. Wu, L. Hsiao, X. Cheng, B. Hancock, T. Rekatsinas, P. Levis, and C. Ré. Fonduer: Knowledge base construction from richly formatted data. In *SIGMOD*, 2018.

[45] Q. Yuan, X. Ren, W. He, C. Zhang, X. Geng, L. Huang, H. Ji, C. Lin, and J. Han. Open-schema event profiling for massive news corpora. In *CIKM*, pages 587–596, 2018.

[46] T. T. Yuan and Z. Zhang. Merchandise recommendation for retail events with word embedding weighted tf-idf and dynamic query expansion. In *SIGIR*, pages 1347–1348, 2018.

[47] D. Zeng, K. Liu, S. Lai, G. Zhou, and J. Zhao. Relation classification via convolutional deep neural network. In *COLING*, pages 2335–2344, 2014.

[48] C. Zhang, D. Lei, Q. Yuan, H. Zhuang, L. M. Kaplan, S. Wang, and J. Han. Geoburst+: Effective and real-time local event detection in geo-tagged tweet streams. *ACM TIST*, 9(3):34:1–34:24, 2018.

[49] C. Zhang, L. Liu, D. Lei, Q. Yuan, H. Zhuang, T. Hanratty, and J. Han. Triovecevent: Embedding-based online local event detection in geo-tagged tweet streams. In *SIGKDD*, pages 595–604, 2017.

[50] G. Zheng, S. Mukherjee, X. L. Dong, and F. Li. Opentag: Open attribute value extraction from product profiles. In *SIGKDD*, pages 1049–1058, 2018.