Learning to Extract Local Events from the Web

John Foley*
Center for Intelligent Information Retrieval
University of Massachusetts
jfoley@cs.umass.edu

Michael Bendersky
Google
bemike@google.com

Vanja Josifovski*
Pinterest
vanja@pinterest.com

ABSTRACT

The goal of this work is extraction and retrieval of local events from web pages. Examples of local events include small venue concerts, theater performances, garage sales, movie screenings, etc. We collect these events in the form of retrievable calendar entries that include structured information about event name, date, time and location.

Between existing information extraction techniques and the availability of information on social media and semantic web technologies, there are numerous ways to collect commercial, high-profile events. However, most extraction techniques require domain-level supervision, which is not attainable at web scale. Similarly, while the adoption of the semantic web has grown, there will always be organizations without the resources or the expertise to add machine-readable annotations to their pages. Therefore, our approach bootstraps these explicit annotations to massively scale up local event extraction.

We propose a novel event extraction model that uses distant supervision to assign scores to individual event fields (event name, date, time and location) and a structural algorithm to optimally group these fields into event records. Our model integrates information from both the entire source document and its relevant sub-regions, and is highly scalable.

We evaluate our extraction model on all 700 million documents in a large publicly available web corpus, ClueWeb12. Using the 217,000 unique explicitly annotated events as distant supervision, we are able to double recall with 85% precision and quadruple it with 65% precision, with no additional human supervision. We also show that our model can be bootstrapped for a fully supervised approach, which can further improve the precision by 30%.

In addition, we evaluate the geographic coverage of the extracted events. We find that there is a significant increase in the geo-diversity of extracted events compared to existing explicit annotations, while maintaining high precision levels.

*Work done while at Google.

SIGIR'15, August 09-13, 2015, Santiago, Chile.
ACM 978-1-4503-3621-5/15/08.
DOI: http://dx.doi.org/10.1145/2766462.2767739 .

Categories and Subject Descriptors

H3.3 [**Information Storage And Retrieval**]: Information Search and Retrieval

Keywords

Information Retrieval; Information Extraction

1. INTRODUCTION

With the increasing trend toward personalized mobile applications and user experiences, there is a need for information systems that react to user preferences and location. In the past few years, this challenge has gathered more attention in the research community. Lagun et al. find that not only is local context useful in search, but that users are interested in explicit feedback in locality-sensitive tasks [16]. The Contextual Suggestion Track in the Text Retrieval Conference (TREC) presents the task of recommending establishments or venues to users given the preferences of a user in another city [5]. In this work, we explore a similar task, presenting users with *events* near them, rather than locations. Unlike the contextual suggestion track, we move away from whole-page relevance judgments toward extracting relevant atomic event records.

We define an event as an object having three mandatory properties, keeping in mind our goal: to recommend, display, and make searchable all events that can be extracted from the web.

Definition An event occurs at a certain location, has a start date and time, and a title or description. In other words, to be useful to a user, an event must be able to answer the questions: *What?*, *When?*, and *Where?*

Succintly, we are interested in events that users may want to add to their calendars to be reminded of and potentially attend. This is in contrast to many other definitions of an event, such as those in works discussing real-time detection of natural disasters, riots, pandemics or terrorist attacks in microblog streams [26, 36, 20], or the classic event definition in computational linguistics, which can be as broad as "a situation that occurs" [25].

Before a recommendation system for events can be created and evaluated, there is the information extraction challenge of discovering and collecting all available events in all areas.

Simple approaches to this problem include:

- Mining and recommending events from social media [14].

- Leveraging semantic web annotations like `Schema.org`[1].
- Traditional wrapper induction and data mining.

Unfortunately, both semantic web and social media approaches require organizations to maintain their data in a particular format. With social media, this means an updated organization page, and with semantic web technologies, this means marking up the page with microdata (e.g., `Schema.org`). Unfortunately, smaller businesses, charities, and truly local organizations will lack the funding or the expertise required to fully and correctly adopt semantic web technologies.

Similarly, most existing approaches to information extraction require supervision at either the page or domain level, or some sort of repeated element structure [34]. As it would be infeasible and costly to annotate all such pages - or even a single page per domain, existing systems that mine for events or other structured data fall short of our goal of local event extraction from *all* web pages.

Examples of events that we consider local and that are unlikely to have existing markup, or sufficient social media presence are farmer's markets, poetry readings, library book sales, charity dinners, garage sales, community band concerts, etc. These events are of interest to a smaller, local community, and are unlikely to be selling tickets on high-profile sites or paying for advertisement.

In this work, our goal is to leverage the well-advertised, high-profile events to learn to extract a greater variety and depth of events, including the kinds of local events described above. Specifically, we leverage the existing `Schema.org` microdata annotations (there is an example of how these annotations appear in Figure 1) as a source of data for distant supervision, allowing us to learn to extract events that do not have semantic web annotations, including local events, without actually collecting judgments specifically for our task.

We introduce a model for scoring event field extractions and an algorithm that groups these fields into complete event records. We scale up our technique to the entire ClueWeb12 corpus (700 million pages), extracting 2.7 million events. We evaluate our precision at various recall-levels, and show that we can double event coverage of a system with respect to the available semantic web annotations at an 85% precision level. We briefly explore using our judgments for a supervised approach to this task and are able to improve precision by 30% on another million events with only 30 annotator-hours.

We also explore the geographic diversity of our dataset, finding that existing markup is heavily biased toward large cities (e.g. New York, Chicago, and Los Angeles) and that the results of our extraction cover a wider variety of locations. We validate this hypothesis via visual mapping, and by showing that we have good precision in a random sample of 200 cities across the world.

In the next section, we introduce related work in detail. In Section 3, we introduce our event field extraction and scoring model, and our event record grouping algorithm. In Section 4, we discuss our corpus and our judgments in more detail, and we present the results of our experiments in Section 5. We end with our conclusions in Section 6.

[1]`http://schema.org/Event`

Figure 1: Example Microdata adapted from `Schema.org`

```
<div itemscope itemtype=“http://schema.org/Event”>
  <span itemprop=“name”>
    Miami Heat at Philadelphia 76ers
  </span>
  <meta itemprop=“startDate”
        content=“2016−04−21T20:00”>
    Thu, 04/21/16 8:00 p.m.
  <div itemprop=“location” itemscope
       itemtype=“http://schema.org/Place”>
    Wells Fargo Center
    <div itemprop=“address” itemscope
         itemtype=“http://schema.org/PostalAddress”>
      <span itemprop=“addressLocality”>
        Philadelphia
      </span>,
      <span itemprop=“addressRegion”>PA</span>
    </div>
  </div>
</div>
```

2. RELATED WORK

Our work is characterized by using the explicitly annotated `Schema.org` as training data to learn to extract local events from the web. While there is work looking at events on social media, work leveraging semantic web annotations, and work on extraction in general, to our knowledge, our work is the first to leverage this data in this way, and the first to attempt this task.

2.1 Similar Tasks

The Contextual Suggestion Track [5] considers the task of a known user spending time and looking for entertainment in a new city. Evaluation is done on the snippet and page level, with users judging sites as interesting or not, effectively making the task about retrieving venues or establishments. In a similar motivation, we would like to consider the task of finding events relevant to a user in their current location, but because no large corpora of events exist, we consider first the task of extracting local events.

There are numerous works that identify the availability of semantic web information [27, 21, 3] but there is very little prior work on using this information as a source of distant supervision. Petrovski et al. use `Schema.org` annotations for products to learn regular expressions that help identify product attributes such as CPU speed, version, and product number [22]. Gentile et al. work on dictionary-based wrapper induction methods that learn interesting XPaths using linked data [7, 8]. Because Linked Data websites like DBPedia and Freebase are not typical web pages as those with `Schema.org` data, the structural features we are able to learn are not available in these works. We also attempt to learn about less structured fields, in particular, our *What?* aspect of events.

Another similar work comes from the historical domain. Smith worked on detecting, and disambiguating places and dates within historical documents in a digital libraries setting. He looked at collocations between these places and dates as events, and ranked them to identify significant date-place collocations that might merit further study by historians [32]. In a follow-up work, he looked at associating terms with these collocations and building interfaces for browsing [31].

2.2 Event Detection in other Domains

There is an entire class of work on detecting events within microblogs or real-time social media updates [26, 36, 20]. Becker describes identification of unknown events and their content in her thesis, but focuses on trending events on social media sites, and classification is used to separate event from non-event content clusters [2]. Our work, in contrast, is looking to find events in an offline manner that are explicitly described so as to present them in a search or recommendation system to users.

Kayaalp et al. discuss an event-based recommendation system integrated with a social media site, but focus on events already available through Facebook and Last.fm [14]. In this work, we consider these social media sites as two examples of the larger concept of "head domains" which are likely to have or adopt semantic web technologies, or be worthwhile candidates for supervised wrapper induction techniques.

Extraction of named events in news is another topic in which there has been work, e.g., [15]. Again, the definition of these news events is different from our local event definition.

2.3 Information Extraction

In the extraction domain, there is an immense amount of work on reducing boilerplate, whether directly, through template extraction [10, 23, 18], or through the more general idea of region extraction and classification. Sleimen and Corchuelo provide a detailed survey of region extractors in web documents, including those that leverage visual features [29]. Region extractors, in general, attempt to classify portions of a page into their purpose, i.e. navigation links, main content, and are studied in part to index only the content present on web pages. At least part of our task could be phrased within this context: an attempt to classify regions as either containing an event or not.

Work on temporal knowledge bases and extraction needed for automatic construction is similar to our task [11], except in a different domain and context. Most popular knowledge bases, have some ties to Wikipedia, which requires articles to be notable. By definition, we are seeking to extract events that do not have global relevance.

Additionally, there are a number of works that focus on repeated structure for extraction. Taking cues from HTML tables [9, 17, 4, 1, 39], or repetitive command tags or terms in general [34], these techniques do not require microdata, but require repetitive structure and make the assumption that there will always be a multitude of records to extract [35]. Recently, there has been a number of works about extracting data from web documents that were generated from the same templates [12, 30, 28]. In contrast to these works, we are interested in extracting local event records, which may or may not be generated from a pre-specified template.

3. EVENT EXTRACTION MODEL

We approach the task of extracting structured event information as a scoring or ranking method. We do this because we do not expect our target data – the listings of events on the web – to be particularly clean or perfectly aligned with any schema. Therefore, we wish to have a technique that joins several field scoring functions, which together give a multi-faceted score of how well the extracted structure fits our model of an event. The scoring functions take into account the source web page (Section 3.1), extraction sub-region (Section 3.2), and the individual extracted fields (Section 3.3).

The fact that there may be multiple extractions from overlapping regions on the page, poses an additional challenge of grouping the extracted fields into disjoint event records that contain event name, date, time and location. To this end, we propose a greedy selection and grouping algorithm that provides an efficient way to optimize the quality of extracted event records. This algorithm is described in detail in Section 3.4.

Formally, we represent each extracted event record as a set of fields ($\boldsymbol{\mathcal{F}} = \{f_1, f_2 \dots f_n\}$), their enclosing region ($\boldsymbol{\mathcal{R}}$), and the source document ($\boldsymbol{\mathcal{D}}$).

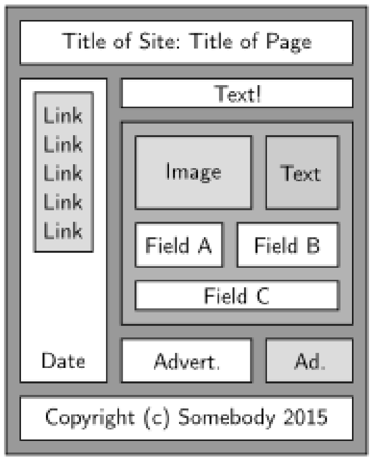

Figure 2: Example of an HTML document structure, presented visually. The boxes present represent nodes, the white boxes are the ones that are plausibly detected as fields in this example. Subsets of these fields represent possible event extractions.

An example HTML document is shown in Figure 2. If we consider the case where this document represents only one event, it is likely that Fields A,B, and C ($\boldsymbol{\mathcal{F}}$) are the key fields in our event, and that the date in the navigation bar and in the copyright are false-positive fields. An event extraction only including the copyright field should get a low score, and similarly, the event containing all the fields should be considered too big and also receive a low score. We define the region of a fieldset to be the first shared parent of its fields, so for fields A and B, the $\boldsymbol{\mathcal{R}}$ should be the box that contains all lettered fields, the image and the text node.

We formulate our scoring function, ϕ, which relates all our available context, as follows

$$\phi(\boldsymbol{\mathcal{F}}, \boldsymbol{\mathcal{R}}, \boldsymbol{\mathcal{D}}) = \alpha(\boldsymbol{\mathcal{D}})\beta(\boldsymbol{\mathcal{R}})\gamma(\boldsymbol{\mathcal{F}})$$

We discuss the scoring components in order, and note where we adapted this object-general model to our event domain. Our document scoring, $\alpha(\boldsymbol{\mathcal{D}})$, our region scoring $\beta(\boldsymbol{\mathcal{R}})$, and our field scoring $\gamma(\boldsymbol{\mathcal{F}})$ are discussed in depth in the following sections.

3.1 Document Scoring

We want our event scoring function to take into account the page's likelihood of discussing an event, therefore we include a document-level score ($\alpha(\boldsymbol{\mathcal{D}})$).

We take a naive-Bayes approach to this problem, and examine the probability of a web page discussing an event (belonging to the event class). We denote the candidate document $\boldsymbol{\mathcal{D}}$ and the event class E, and consider the following ratio

$$\frac{P(E|\boldsymbol{\mathcal{D}})}{P(\overline{E}|\boldsymbol{\mathcal{D}})} > 1.$$

In other words, we consider a page worth investigating if the probability of it belonging to the event class (E) is higher than the probability that it does not belong to the event class ($\overline{E}$). In practice, the division of small fractions can lead to underflow, so we consider the equivalent relationship, asking whether the so-called log-odds of a page belonging to an event class is greater than zero.

$$logP(E|\boldsymbol{\mathcal{D}}) - logP(\overline{E}|\boldsymbol{\mathcal{D}}) > 0$$

We estimate these probabilities based upon the commonly-used language modeling framework introduced by Ponte and Croft [24]. In the language modeling framework, we estimate the probability of a word given a model X (which will be one of $\{E, \overline{E}\}$) as the probability of drawing it randomly from the bag of words that is that model.

$$P(w \in X) = \frac{tf(w, X)}{tf(*, X)}$$

The bag of words assumption means that we can treat all our terms as being independent, and we estimate the probability of a whole document by the probability of all its component terms ($w \in \boldsymbol{\mathcal{D}}$) under the model.

$$P(\boldsymbol{\mathcal{D}} \in X) = \prod_{w \in \boldsymbol{\mathcal{D}}} \frac{tf(w, X)}{tf(*, X)}$$

Because our event class may be sparse in comparsion to any given document, we apply linear smoothing [13] to our positive class to avoid zero probabilities.

$$P(E|D) = \prod_{w \in \boldsymbol{\mathcal{D}}} \lambda P(w \in E) + (1 - \lambda) P(w \in C)$$

In contrast, because we approximate our non-event class by the language model of the entire collection, $\overline{E} = C$, no smoothing is needed because all terms are present by construction.

$$P(\overline{E}|\boldsymbol{\mathcal{D}}) = \prod_{w \in \boldsymbol{\mathcal{D}}} P(w \in C)$$

Since we run our algorithm on millions of pages, we chose to use this page scoring mechanism as a first pass filter, restricting our calculation of other scoring components to those whose $\alpha(\boldsymbol{\mathcal{D}}) > 0$, where α is defined as follows:

$$\alpha(\boldsymbol{\mathcal{D}}) = \begin{cases} 1 & logP(E|\boldsymbol{\mathcal{D}}) - logP(\overline{E}|\boldsymbol{\mathcal{D}}) > 0 \\ 0 & \text{otherwise} \end{cases}$$

What remains now is identifying an initial set of documents that are used to construct the language model for the event class E. For this, we bootstrap existing semantic annotations on the web. Specifically, we consider all the pages that contain any `http://schema.org/Event` annotations (see Figure 1 for an example) as belonging to the event class E, since mark-up by their creators suggests that they discuss events. Overall, we have close to 150,000 such pages, which allows creating a robust event model.

3.2 Region Scoring

Region scoring ($\beta(\boldsymbol{\mathcal{R}})$) is considered on the enclosing region $\boldsymbol{\mathcal{R}}$ in the document. Since this region encloses all of event potential fields, it is a good unstructured representation of the extracted event. In fact, we present this enclosing region to annotators to understand our event prediction performance separately from our field prediction.

Therefore, we decided on a simple region filtering approach, which simply removed from consideration regions above certain length

$$\beta(\boldsymbol{\mathcal{R}}) = \begin{cases} 1 & |\boldsymbol{\mathcal{R}}| < \tau \\ 0 & \text{otherwise} \end{cases}$$

τ is set to 2^{12} in all the subsequent experiments. Our empirical evaluation have shown that this simple approach effectively removed the majority of bad regions. In fact, we considered a number of more sophisticated approaches, including learning probability distributions of the size of the region and enclosing region tags. However, in experiments not listed here due to space constraints, we found the effects of such additional information negligible, especially since region features were included on a per-field basis within our field-scoring functions (see Section 3.3).

3.3 Field Set Scoring

We explore field scoring in a way that requires at least one of each required field to be part of our extracted field set ($\boldsymbol{\mathcal{F}}$). Therefore our formulation for $\gamma(\boldsymbol{\mathcal{F}})$ includes both a scoring function γ_S and an indicator function that tests for required fields, γ_R.

$$\gamma(\boldsymbol{\mathcal{F}}) = \begin{cases} \gamma_S(\boldsymbol{\mathcal{F}}) & \gamma_R(\boldsymbol{\mathcal{F}}) \\ 0 & \text{otherwise} \end{cases}$$

We will discuss the breakdown of the γ_S and γ_R functions below. Generally, we jointly score field occurrences, and this joint scoring considers the independent scores, $\delta_k(f)$, assigned to each field f of type $k \in \{\text{What}, \text{When}, \text{Where}\}$.

3.3.1 *Independently Scoring Fields*

In this work, we consider a number of ways to assign independent scores to fields that allows us to define $\delta_k(f)$ for any f and $k \in \{\text{What}, \text{When}, \text{Where}\}$. Formally, we define f as a tuple with a $(\text{begin}, \text{end})$ index range within the source document $\boldsymbol{\mathcal{D}}$. Because the document itself has some HTML structure, the offsets within the field allow us to understand where in the structure of the page it occurs and leverage that as part of our scoring features.

Pattern-based approaches to tagging of dates, times and postal addresses in text yields results of reasonable accuracy, as evidenced by the approaches in HeidelTime [33], StanfordNLP [19], as well as some prior work on address detection [38]. Therefore, we consider pattern-based baselines for our extraction of event 'When' and 'Where' fields. While dates, times, and places have inherent and sometimes obvious structure, patterns do not make sense as a baseline for the 'What' of an event, so we assign an equal score to all candidates.

Table 1: Features used in Field Classification.

Category	Feature	Description
Text	Unigrams	Stopped and stemmed unigrams, hashed to 10,000 count-features.
	Bigrams	Stopped and stemmed bigrams, hashed to 10,000 count-features.
NLP	Capitalization	Ratio of terms capitalized, first term capitalized.
	Address overlap	Number of address fields that overlap the target span.
	Date overlap	Number of date fields that overlap the target span.
	Time overlap	Number of time fields that overlap the target span.
Structural	Size	Ratio of size to parent and to page.
	Location	Ratio of start and end locations to page.
	Attribute Text	Unigrams present in attributes; can capture style information.
	Parent tag	Hashed vocabulary of size 10 of lower-case parent tag.
	GrandParent tag	Hashed vocabulary of size 10 of lower-case parent's parent tag.
	Sibling tags	Hashed vocabulary of size 1000 sibling tags.
	Reverse XPath	10 features for each XPath entry going backwards toward HTML.

No classification uses baseline approaches for all fields.

$$\delta_{\text{What}}(f) = 0.5$$
$$\delta_{\text{Where}}(f) = \text{matches}(f, \text{Address})$$
$$\delta_{\text{When}}(f) = \text{matches}(f, \text{Date/Time})$$

What classification uses baseline approaches except for What fields.

$$\delta_{\text{What}}(f) = \vec{W}^T_{\text{What}} \cdot \vec{X}_f$$
$$\delta_{\text{Where}}(f) = \text{matches}(f, \text{Address})$$
$$\delta_{\text{When}}(f) = \text{matches}(f, \text{Date/Time})$$

What-When-Where classification uses multiclass classification to rescore the boolean baseline approaches for all fields.

$$\delta_{\text{What}}(f) = \vec{W}^T_{\text{What}} \cdot \vec{X}_f$$
$$\delta_{\text{Where}}(f) = \text{matches}(f, \text{Address}) \cdot \vec{W}^T_{\text{Where}} \cdot \vec{X}_f$$
$$\delta_{\text{When}}(f) = \text{matches}(f, \text{Date/Time}) \cdot \vec{W}^T_{\text{When}} \cdot \vec{X}_f$$

Our baseline methods are implemented by the function matches(f, r_k) where f is a field, and r is a set of regular expressions for field type k, returning 1 if a field f is considered a match for field type k, and 0 otherwise.

Our classification methods leverage features $\vec{X}_f$ extracted from the candidate field f and weights $\vec{W}_k$ learned using LIBLINEAR [6]. The features used encompass textual, natural-language, and structural features that are more fully described in Table 1. Evaluation of these prediction methods is discussed in Section 5.1. All other evaluations consider only the What-When-Where classification method for independent field scoring.

In order to train our classification methods we turn once again to the pages in the event class E, described in Section 3.1. Using these pages, we label all the HTML tags with semantic mark-up related to one of our three target fields ('What', 'When', 'Where') with their respective field type. In addition, we label all other textual HTML tags on these pages as 'Other'. We then use this bootstrapped training data to construct a multiclass classifier with label set

$$\mathcal{K} = [' What', ' When', ' Where', `Other'],$$

and learn a weight vector $\vec{W}_k$, for each $k \in \mathcal{K}$. See Section 4.3 for more details on this process.

3.3.2 Jointly Scoring Fields

Given $\boldsymbol{\mathcal{F}}^R = \{\text{What}, \text{When}, \text{Where}\}$ as the set of required fields for an event, a field set $\boldsymbol{\mathcal{F}}$ has all its required fields if and only if $\gamma_R(\boldsymbol{\mathcal{F}})$ is true. We make the assumption that the field type k, with the maximum score $\delta_k(f)$ is the PredictedType of the given field f. Formally:

$$\text{PredictedType}(f) = \operatorname*{argmax}_{k \in \boldsymbol{\mathcal{F}}^R} \delta_k(f)$$

We test for the presence of all required fields by containment; the required fields should be a subset of the predicted types of the entire field set we are scoring. As an example, a set of fields may comprise an event if it has an extra date or time, but it may not if it doesn't have a date or time at all.

$$\gamma_R(\boldsymbol{\mathcal{F}}) = \boldsymbol{\mathcal{F}}^R \subseteq \{\text{PredictedType}(f) | f \in \boldsymbol{\mathcal{F}}\}$$

We combine the individual field scores within $\gamma_S(\boldsymbol{\mathcal{F}})$, our field set scoring function, using the same notation as before for per-field-type scorers ($\delta_k(f)$).

$$\gamma_S(\boldsymbol{\mathcal{F}}) = \prod_{f \in \boldsymbol{\mathcal{F}}} \max_{k \in \boldsymbol{\mathcal{F}}^R} \delta_k(f)$$

The individual score for each field is still labeled with a function, $\delta_k(f)$, which computes the score for the maximally-likely class for each field, reusing $\boldsymbol{\mathcal{F}}^R$ to describe the set of required classes.

In this formulation, we expect the output range of δ_k to be $[0, 1]$. Since our independent scores are all in this range, it means that our function for γ_S will tend to prefer field sets with fewer higher-scored fields, with γ_R ensuring that we do not consistently predict incomplete events.

3.4 Event Record Grouping Algorithm

We consider all HTML tags that contain any baseline (regular expression pattern-based) matches on the page to be candidates for field scoring, rather than exhaustively iterating over all the subsets of text fields on the page. Even with this relatively-smaller number of candidate fields, the prediction algorithm is computationally difficult. Recall that we have formulated our field scoring as a ranking problem. Therefore, grouping these ranked fields into complete event records is a variation of the general subset selection problem, which is known to be NP-hard in most cases [37].

To ameliorate this problem, we choose to add a constraint that no field may be used twice (predicted events should not overlap) and to use a greedy approach to assigning fields to event records, so that we at least are able to predict the best events correctly. This greedy algorithm for event record grouping is presented in Algorithm 1. We rank our predicted nodes by score, and greedily select the highest scoring events, whose enclosing regions ($\mathcal{R}$) do not overlap.

Algorithm 1 Greedy event record grouping algorithm.

```
# All tags and all regex matches on the page:
candidate_fields = {span, ...}
# All tags on the page containing candidate_fields.
candidate_tags = {span, ...}

# Score and collect candidates
possible_events = ∅
for $\mathcal{R}$ in each candidate_tag:
  $\mathcal{F}$ = set(candidate_fields inside $\mathcal{R}$)
  score = $\phi(\mathcal{F}, \mathcal{R}, \mathcal{D})$
  if score > 0:
    add (score, $\mathcal{F}, \mathcal{R}, \mathcal{D}$) to possible_events

# Greedily select highest scoring, non−overlapping tags
output = ∅
for event in sorted(possible_events):
  if event fields do not overlap with any fields in output:
    add event to output

return output;
```

As another nod to efficiency, we implement ϕ by considering the parts independently, in terms of cost. Because $\alpha(\mathcal{D})$, $\beta(\mathcal{R})$, and $\gamma_R(\mathcal{F})$ are simple to calculate and may return zero scores, we calculate these parts first and skip the relatively-more expensive independent field calculations ($\gamma_S(\mathcal{F})$). Additionally, this reduces the number of field sets that must be considered, lowering the overall cost of the algorithm in practice.

An example HTML structure is shown in Figure 3. Nodes D,E,F, and G are textual nodes, which contain fields. D has been detected as a *What* field, E has been detected as a *When* field, F has both a *Where* and a partial *When* (only a relative date) and maybe a *What* field. Node G is an example of a date that is nearby but irrelevant to the event itself.

Running on this example, our algorithm would walk through the nodes, assigning ϕ scores to each node. Node D would receive a score of zero, because it is missing required field types (γ_R). Similarly, nodes E,F, and G would be removed from consideration due to missing fields. The *best* selection for this event would either be Node B or Node C, depending on whether Node D is required to describe the event or if Node F is considered sufficient *What* explanation. Because of our joint field scoring, even if we do not apply a *What* field classifier, Node A will receive a lower score since it has an extra date field, which will lower its γ_S score and overall score.

3.4.1 Deduplication

During development, our annotators reported seeing numerous instances of duplicate events, particularly of high-profile events that were advertised on many sites. Therefore, we

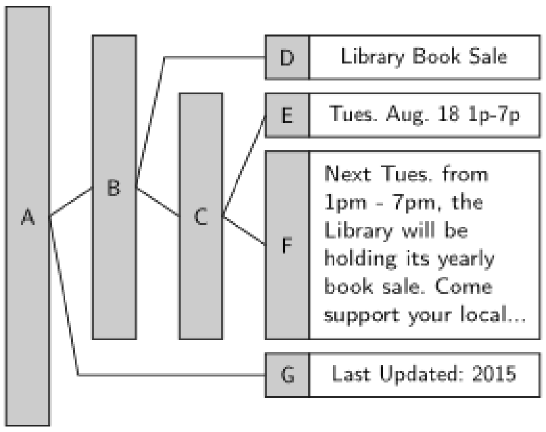
A
B
C
D
Library Book Sale
E
Tues. Aug. 18 1p-7p
F
Next Tues. from 1pm - 7pm, the Library will be holding its yearly book sale. Come support your local...
G
Last Updated: 2015

Figure 3: Example document structure handled by our algorithm. The boxes labeled with letters represent nodes in the HTML structure. Nodes D-G are textual nodes whose text is present. The lines display the parent-child relationships in the tree.

applied an exact-string matching duplicate detection method that ignored differences in capitalization, HTML tags and whitespace as a baseline and found that it was difficult to improve upon this baseline as our annotators no longer reported finding duplicates in our sampling of events. In all following analyses, we deduplicate all the extracted events using this method. We leave using extracted date/time and place information to perform a more domain-specific duplicate detection for future work.

4. EXPERIMENTAL SETUP

We evaluate our technique for extracting events on the ClueWeb12[2] corpus, which contains about 733 million pages. We extract `http://schema.org/event` records as microdata and rDFa in a first pass. This generates 145,000 unique web pages with events, and about 900,000 events on those pages. After deduplication, there are 430,000 unique events. Of these unique events, we map `Schema.org` properties to our three fields: What, Where, and When, and we are left with 217,000 complete events without any further processing.

We use incomplete events to train our field classifiers, but we will later consider recall in respect to the number of unique, complete `Schema.org` events, and we remove all pages with `Schema.org` events from our processing of the corpus moving forward, except for training data generation purposes.

4.1 Collecting Judgments

During the development of our system, we collected 2,485 binary judgments for understanding precision of event predictions. Each of these judgments contains the ClueWeb12 id, the start and end offsets of the events (in bytes after the WARC header), and a judgment: 1 if it is an event, and 0 otherwise.

For a large number of events in our best method, and in our method comparsion, we evaluate at the field level as well as at the event level, but only if the extraction actually is an event. For these judgments, as well as the agreement judgments, we turn to a crowdsourcing platform (Amazon

[2]`http://lemurproject.org/clueweb12/`

Mechanical Turk) to rapidly accumulate high-quality per-field judgments. We paid annotators \$0.05 for event-level judgments and \$0.10 for field-level judgments. We allowed annotators to mark *I can't tell from the available information* for any field or event, and removed these from our evaluation (Only 46 events in total out of the 2500).

For the fields, we asked annotators if they believed the selected snippet answered the given question about the event. As an example, we asked annotators to be picky about the "When" field in particular: two of our options were *No, it only includes a date* and *No, it only includes a time.*

4.2 Annotator Agreement

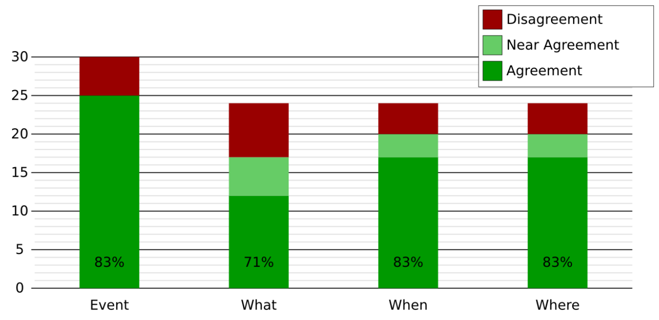

Figure 4: Agreement on Events and Fields

We calculated agreement on a set of thirty events randomly selected from our best-performing technique. Results are displayed in Figure 4. Because we asked our annotators for reasons when a field was inaccurate, we have some data on which the annotators agreed the field was incorrect, but disagreed on the reason: for example, one reviewer said the field was too short and the other said that they could not tell.

Our agreement is relatively high (>80%) on all but the "What" field, and our local reviewers found that agreement tended to be correlated with quality. The relative field performance here displays the ambiguity inherent in deciding what should describe an event in comparison to the stricter definitions of location and time.

4.3 Training Field Classifiers

Earlier, in Section 3.3.1, we discuss our approach to assigning scores to fields, and our classification methods. We list the features that we extract in Table 1, and we have already mentioned that we were able to parse 430,000 unique events from websites with `Schema.org` markup.

The question remains - how do we use this data as training for *What Classification* and *What-When-Where Classification*? Our first attempt to learn from this data involved a cross-fold evaluation setup, where we split events by domain into train, validate and test splits, extract features, and try to predict the fields in our validate and test sets. We split by domain here to better prepare ourselves for new data in our cross-validation setup, to avoid overtfitting to head domains that would have been present in all cross-validation sets.

While `Schema.org` gives us plenty of positive examples for each field, we don't necessarily have obvious negative examples. In the end, we found that taking all the tags, including other `Schema.org` properties, gave us the richest, most accurate representation of our problem space. We want our field classifiers to be able to select tags that actually correspond to an event from all the other tags on a page that has events. In early experiments, we found that using a single-class classifier was a mistake - the best negative examples for *What* include the positives for *When* and *Where*, as judged by the way our *What* field initially predicted dates and times and places. We moved to a multiclass setup, and this seemed to improve on this problem. Even for our *What Classification* setup, we emitted training, validation, and test data. We used our validation data to tune the C and ϵ parameters for LIBLINEAR and to choose between L1 and L2 normalization.

Even with the care taken to select negative examples, our cross-validation estimate of field classifier performance was still extremely high ($F_1 > 75\%$) for our three fields, which as we discuss later, is not the case when evaluated on pages without `Schema.org`.

5. RESULTS

5.1 Evaluating our Field Prediction Methods

Table 2: Fields Classified by Method. (N,W) denote significant improvements, where 99% of bootstrap samples of the target method have higher precision than 99% of samples from the None and What methods, respectively.

	None	What	What-Where-When
Event	0.54	0.51	0.76 (N,W)
What	0.09	0.30 (N)	0.36 (N)
When	0.17	0.20	0.32
Where	0.32	0.32	0.66 (N,W)

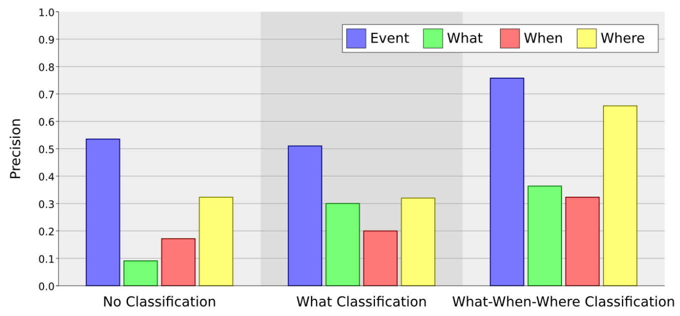

Figure 5: Comparison of Precision of Field Detection Methods

A key part of our overall event extraction system is the ability to assign meaningful independent scores to fields. We introduced three field scoring systems in Section 3.3.1 based on our baseline ability to detect Where/When aspects of events through regular expressions, and combining these baselines with classification.

In the *No Classification* system, we don't use any training data, and end up leveraging text near our Where/When fields for our What predictions. In our *What Classification* system, we use classification on only the What field, for which the baseline is weakest, and in our *What/When/Where Classification* system, we combine our baseline and classification approaches for all fields, leveraging the semi-supervision of our `Schema.org` extracted fields to learn weights for features as described earlier.

Table 3: Precision Levels used in evaluation.

	V. High	High	Medium	Low
Predicted	25,833	201,531	452,274	1,575,909
Increase	12%	93%	208%	725%
Judgments	155	567	482	491
Precision	0.92	0.85	0.65	0.55

In order to compare these methods, we first attempted to use our semi-supervised data - the `Schema.org` events and fields. Unfortunately, on such an event-focused dataset, all of our methods performed well. We believe that because of the format of microdata, and how it requires that each field be represented succinctly and separately in its own tag, these training instances were actually too simple. As a result, we were not able to leverage the `Schema.org` data to compare methods, and we had to create our own judgments.

We chose the top 30,000 pages by our document scoring method: $\alpha(\boldsymbol{\mathcal{D}})$, and ran our algorithm three times on each page, once with each of our classification methods, and generated a pool of events from each. From each pool we sampled 100 random events (any attempt at a pairwise comparison would not have accurately compared the different methods).

We used bootstrap sampling with 10,000 samples to obtain 99% confidence intervals for each of our evaluations. We mark improvements where the 1% (worst-case) of the treatment is greater than the 99% (best-case) of the baseline. The improvements are marked in terms of the *None* baseline (N) and the *What* baseline (W), and the raw, mean values are presented in the Table 2.

5.2 Exploring Precision and Recall

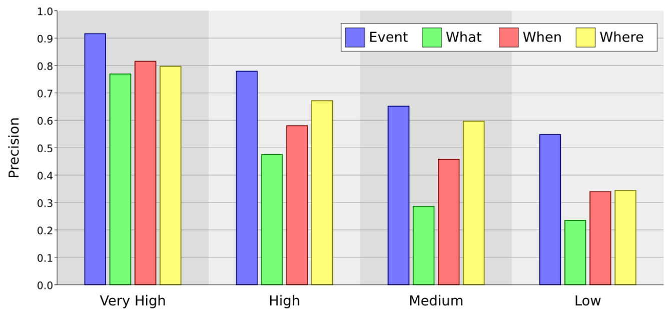

Figure 6: Precision Evaluated at Recall Levels

As we discussed earlier, there were only about 217,000 unique, complete events used in our semi-supervised approach for training. We cannot evaluate our recall perfectly because we simply do not know how many events are on the internet, or even in ClueWeb12's 700 million pages. Additionally, we think the most interesting way to evaluate recall is to consider a number of interesting recall points (precision categories) for evaluation based on the ratio of events predicted to the size of our semi-supervised input (`Schema.org`) events).

We consider a "Very High" precision category, chosen because it was a cutoff found to have over 90% precision. Our "High" precision category is chosen because it represents about a 1:1 prediction to training ratio. Our "Medium" precision category captures the case where we predict two events for each training instance, and our "Low" precision category contains the lowest score range in which we collected judgments, at about a 7:1 prediction ratio. These pseudo-recall levels are listed in Table 3.

In Figure 6, precision for each component of the event is illustrated. As overall precision drops, we can see how the weakest part of our detection is still the Event title, or the *What* field. Intuitively, this makes sense, as it is less structured and even our annotators had trouble agreeing on *What* field with respect to the other fields. Another finding that's interesting here is that even though the field precision degrades rapidly, (partly because it is conditioned on the event being correct) our event precision is significantly higher, particularly in the "Low" precision level. This suggests that even our noisy field detection has value, particularly to a selection algorithm like ours.

5.3 Supervised Evaluation

As a qualitative observation, when we had false-positives for event classification, they were often near-misses, in the form of album releases, band histories, obituaries, or biographies. They tended to be rich in field information, but from the wrong domain. In general, our technique introduced some false positives as the domain broadened from the focus on commercial music events of the `Schema.org` events.

Since we had collected a large number of judgments, we decided to briefly explore a human-supervised approach. In all other sections, we discuss distant supervision results, where we had no gold-truth human assigned labels, but only the silver-truth of the `Schema.org` data. This evaluation serves two purposes: (1) determining whether our impression of easily-detected domain mismatches is true, and (2) evaluating the amount of effort needed to apply a supervised approach to this problem.

Since we were interested in understanding annotator effort, we only used our 1100 Mechanical Turk annotations for this part, as they were automatically timed. We break our judgments into two parts, taking 800 events from medium to very-high precision, as our training and validation data, and 300 events in our low-precision bracket (original Precision of 0.55). We build a simple unigram classifier with 10,000 hashed term features as well as document, region, and individual field scores for the best field of each kind from those predictions, again using LIBLINEAR.

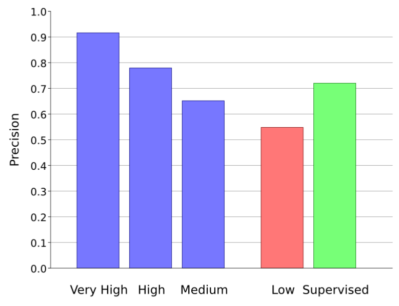

Figure 7: Supervised Event Prediction of Low-precision data based on other judgments. On the left, we have the training/validation data, and their precision represented (Very High, High, and Medium), and on the right, we have the test data, and its original precision (Low) and its new precision under supervision (Supervised).

This classifier works extremely well over our 300 evaluation events from the low range, boosting the precision from 0.55 to 0.72, with a 92% relative recall (the new classifier rejected another 8% of the 300 events). This experiment is displayed in Figure 7, where the blue bars correspond to the training data, the red bar with the original performance of the low-precision events, and the green bar with the supervised precision of the low-precision events.

This experiment supports our claim that a large amount of false-positives from our technique are simple to eliminate with minimal training data and annotation effort (~30 total hours of annotator time for the 1100 train/validate/evaluation judgments used).

5.4 Geographic Evaluation

We use a publicly-available geocoding and address API[3] to convert our extracted "Where" fields into full street addresses, and latitude/longitude information. The street addresses are used to evaluate our performance on future retrieval tasks, and the latitude and longitude pairs are used to generate maps to better understand our coverage and that of the `Schema.org` data.

5.4.1 *City Coverage around the World*

Table 4: Retrieval Metrics for 200 random cities.

MRR@5	P@1	P@2	P@3	P@4	P@5
0.78	0.71	0.70	0.69	0.70	0.70

The TREC Contextual Suggestion Track considers the task of recommending venues or destinations to users in new cities [5]. In this work, we briefly evaluate the suitability of our dataset to be applied to a similar task: the task of recommending events to users based on their location. Because the work we present is not about the recommendation task itself, but rather the extraction of events, we evaluate our ability to provide event coverage in a random city.

We group our events by city, and select events from 200 cities at random. We then judge, using Amazon Mechanical Turk, the 5 highest scoring events from each city. We evaluate as a retrieval task, where our query is the random city, and our ranking function for now is our confidence in the event prediction. The mean reciprocal rank (MRR) of finding an event is 0.78 on this data: on average there is an event at either rank 1 or rank 2. Our event precision at 1 is 0.71. Full results are listed in Table 4. The precision numbers here seem flat because there tend to be five or so valid events for each location. It is nice to see this consistency, because it suggests that our system is able to find many events for randomly selected locations.

5.4.2 *Geographic Display*

In Figure 8, we display a small world map where our events are plotted to understand our international coverage. Since the dataset we used, ClueWeb12 is biased toward English pages, it is understandable that the events we have extracted are mostly found in the United States, and north-western Europe. There is also some token coverage in other parts of the world, including India and Australia, but we focus on areas that we expect to have good coverage based on the ClueWeb12 English-speaking bias.

In Figure 9, you can see our events plotted on a US Map in two colors: green for high and very-high confidence $P >= 0.65$, and yellow for $0.55 < P < 0.65$. The blue dots are events parsed from `Schema.org` pages. The key takeaway from this

[3]`https://developers.google.com/maps`

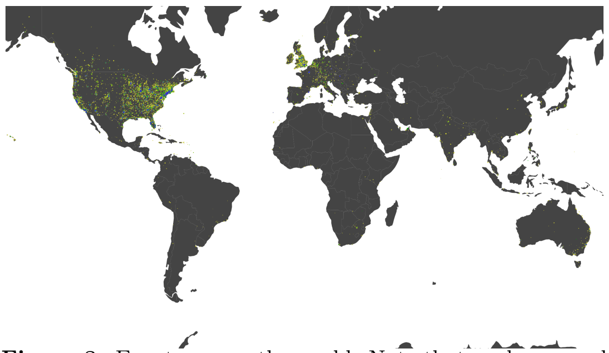
Figure 8: Events across the world. Note that we have good coverage of countries where English is the dominant language, in a reflection of our corpus, ClueWeb12.

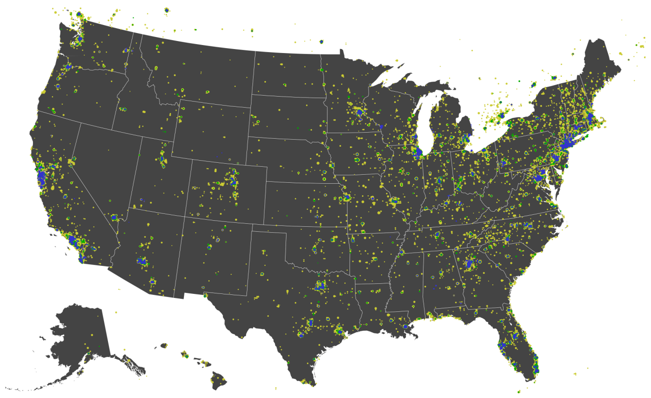
Figure 9: Events within the United States. Blue data points indicate `Schema.org` data, Green represents high confidence (P=0.65..0.85+), and Yellow represents low to medium confidence points (P=0.55..0.65). All points have more than 5 unique events, and are logarithmically scaled according to weight.

map is that `Schema.org` events are definitely more prevalent in large cities, specifically in New York, Los Angeles, and Chicago, and visually, our technique does quite a good job of expanding coverage geographically. Note that we only plot a point if there are more than five unique event occurrences at that location, and we have restricted our sampled precision to be above 0.55, and there are many more events at the larger dots, as their size is logarithmically scaled.

6. CONCLUSION

In this work, we introduce a new task inspired by the TREC Contextual Suggestion Track and the increasing population of mobile users: to retrieve and recommend events that users might want to attend. Because we are particularly interested in local events, our work focuses on the identification and extraction of events on the open internet.

We show that using semantic web technologies, and specifically `Schema.org` microdata to collect distant supervision data can be useful for training semi-supervised approaches to extraction problems like ours. This setup has the advantage that as more organizations adopt such technologies, the performance of our extractions will increase over time without any additional labeling effort.

We propose a novel model of event extraction based on independent field scoring and a greedy algorithm for grouping these fields into complete event records, with confidence scores assigned to them. Our features for field classification allow us to combine elements commonly seen in region extraction approaches with both information retrieval-based and linguistically-inspired features. As the proposed model depends heavily upon the ability to assign individual field scores, we evaluate a number of score assignment methods, and find that classification that learns from using the distant supervision of the `Schema.org` data is significantly beneficial.

7. ACKNOWLEDGEMENTS

This work was supported in part by the Center for Intelligent Information Retrieval. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.

8. REFERENCES

[1] M. D. Adelfio and H. Samet. Schema extraction for tabular data on the web. *VLDB'13*, 6(6):421–432, 2013.

[2] H. Becker. *Identification and characterization of events in social media*. PhD thesis, Columbia University, 2011.

[3] C. Bizer, K. Eckert, R. Meusel, H. Mühleisen, M. Schuhmacher, and J. Völker. Deployment of rDFa, microdata, and microformats on the web–A quantitative analysis. *ISWC*, pages 17–32, 2013.

[4] M. J. Cafarella, A. Halevy, and J. Madhavan. Structured data on the web. *CACM'11*, 54(2):72–79, 2011.

[5] A. Dean-Hall, C. L. Clarke, J. Kamps, P. Thomas, N. Simone, and E. Voorhees. Overview of the trec 2013 contextual suggestion track. Technical report, DTIC Document, 2013.

[6] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. Liblinear: A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874, 2008.

[7] A. L. Gentile, Z. Zhang, I. Augenstein, and F. Ciravegna. Unsupervised wrapper induction using linked data. In *K-CAP '13*, pages 41–48, New York, NY, USA, 2013. ACM.

[8] A. L. Gentile, Z. Zhang, and F. Ciravegna. Self training wrapper induction with linked data. In *Text, Speech and Dialogue*, pages 285–292. Springer, 2014.

[9] R. Gupta and S. Sarawagi. Answering table augmentation queries from unstructured lists on the web. *VLDB'09*, 2(1):289–300, 2009.

[10] S. Gupta, G. Kaiser, D. Neistadt, and P. Grimm. Dom-based content extraction of html documents. In *WWW'03*, pages 207–214. ACM, 2003.

[11] J. Hoffart, F. M. Suchanek, K. Berberich, and G. Weikum. Yago2: a spatially and temporally enhanced knowledge base from wikipedia. *Artificial Intelligence*, 194:28–61, 2013.

[12] J. L. Hong, E.-G. Siew, and S. Egerton. Information extraction for search engines using fast heuristic techniques. *Data & Knowledge Engineering*, 69(2):169–196, 2010.

[13] F. Jelinek and R. L. Mercer. Interpolated estimation of markov source parameters from sparse data. In *Proceedings of the Workshop on Pattern Recognition in Practice*, 1980.

[14] M. Kayaalp, T. Ozyer, and S. Ozyer. A collaborative and content based event recommendation system integrated with data collection scrapers and services at a social networking site. In *ASONAM'09*, pages 113–118. IEEE, 2009.

[15] E. Kuzey, J. Vreeken, and G. Weikum. A fresh look on knowledge bases: Distilling named events from news. In *CIKM'14*, pages 1689–1698. ACM, 2014.

[16] D. Lagun, A. Sud, R. W. White, P. Bailey, and G. Buscher. Explicit feedback in local search tasks. In *SIGIR'13*, pages 1065–1068. ACM, 2013.

[17] G. Limaye, S. Sarawagi, and S. Chakrabarti. Annotating and searching web tables using entities, types and relationships. *VLDB'10*, 3(1-2):1338–1347, 2010.

[18] R. Manjula and A. Chilambuchelvan. Extracting templates from web pages. In *Green Computing, Communication and Conservation of Energy (ICGCE), 2013 International Conference on*, pages 788–791. IEEE, 2013.

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

[20] D. Metzler, C. Cai, and E. Hovy. Structured event retrieval over microblog archives. In *ACL'12*, pages 646–655. Association for Computational Linguistics, 2012.

[21] H. Mühleisen and C. Bizer. Web data commons-extracting structured data from two large web corpora. *LDOW*, 937, 2012.

[22] P. Petrovski, V. Bryl, and C. Bizer. Learning regular expressions for the extraction of product attributes from e-commerce microdata. 2014.

[23] J. Pomikálek. Removing boilerplate and duplicate content from web corpora. *Disertacnı práce, Masarykova univerzita, Fakulta informatiky*, 2011.

[24] J. M. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *SIGIR'98*, pages 275–281. ACM, 1998.

[25] J. Pustejovsky, P. Hanks, R. Sauri, A. See, R. Gaizauskas, A. Setzer, D. Radev, B. Sundheim, D. Day, L. Ferro, et al. The timebank corpus. In *Corpus linguistics*, volume 2003, page 40, 2003.

[26] T. Sakaki, M. Okazaki, and Y. Matsuo. Earthquake shakes twitter users: real-time event detection by social sensors. In *Proceedings of the 19th international conference on World wide web*, pages 851–860. ACM, 2010.

[27] M. Schmachtenberg, C. Bizer, and H. Paulheim. Adoption of the linked data best practices in different topical domains. *ISWC*, pages 245–260, 2014.

[28] H. Sleiman and R. Corchuelo. Trinity: on using trinary trees for unsupervised web data extraction. 2013.

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

[30] H. A. Sleiman and R. Corchuelo. Tex: An efficient and effective unsupervised web information extractor. *Knowledge-Based Systems*, 39:109–123, 2013.

[31] D. A. Smith. Detecting and browsing events in unstructured text. In *SIGIR'02*, pages 73–80. ACM, 2002.

[32] D. A. Smith. Detecting events with date and place information in unstructured text. In *Proceedings of the 2nd ACM/IEEE-CS joint conference on Digital libraries*, pages 191–196. ACM, 2002.

[33] J. Strötgen and M. Gertz. Heideltime: High quality rule-based extraction and normalization of temporal expressions. In *Workshop on Semantic Evaluation, ACL*, pages 321–324, 2010.

[34] W. Thamviset and S. Wongthanavasu. Information extraction for deep web using repetitive subject pattern. *WWW'13*, pages 1–31, 2013.

[35] W. Thamviset and S. Wongthanavasu. Bottom-up region extractor for semi-structured web pages. In *ICSEC'14*, pages 284–289. IEEE, 2014.

[36] K. Watanabe, M. Ochi, M. Okabe, and R. Onai. Jasmine: a real-time local-event detection system based on geolocation information propagated to microblogs. In *CIKM'11*, pages 2541–2544. ACM, 2011.

[37] W. J. Welch. Algorithmic complexity: three np-hard problems in computational statistics. *Journal of Statistical Computation and Simulation*, 15(1):17–25, 1982.

[38] Z. Yu. High accuracy postal address extraction from web pages. Master's thesis, Dalhousie University, Halifax, Nova Scotia, 2007.

[39] Z. Zhang. Towards efficient and effective semantic table interpretation. In *ISWC'14*, pages 487–502. Springer, 2014.