OpenCeres: When Open Information Extraction Meets the Semi-Structured Web

Colin Lockard
Paul G. Allen School
University of Washington
lockardc@cs.washington.edu

Prashant Shiralkar
Amazon
shiralp@amazon.com

Xin Luna Dong
Amazon
lunadong@amazon.com

Abstract

Open Information Extraction (OpenIE), the problem of harvesting triples from natural language text whose predicate relations are not aligned to any pre-defined ontology, has been a popular subject of research for the last decade. However, this research has largely ignored the vast quantity of facts available in semi-structured webpages. In this paper, we define the problem of OpenIE from semi-structured websites to extract such facts, and present an approach for solving it. We also introduce a labeled evaluation dataset to motivate research in this area. Given a semi-structured website and a set of seed facts for some relations existing on its pages, we employ a semi-supervised label propagation technique to automatically create training data for the relations present on the site. We then use this training data to learn a classifier for relation extraction. Experimental results of this method on our new benchmark dataset obtained a precision of over 70%. A larger scale extraction experiment on 31 websites in the movie vertical resulted in the extraction of over 2 million triples.

1 Introduction

Knowledge extraction is the problem of extracting *(subject, predicate, object)* triples from unstructured or semi-structured data, where the subject and object are entities and the predicate indicates the relationship between them. In conventional information extraction (which we call "ClosedIE"), a closed set of potential predicates and their semantics are pre-defined in an ontology. Open Information Extraction (OpenIE) is an alternative approach that has no pre-defined ontology and instead represents the predicate with a string extracted from the source data. These extractions can capture a much vaster array of semantic relationships than ClosedIE and have been used to support many downstream use-cases, including question-answering, ontology discovery, embedding generation, fact checking, and summarization (Mausam, 2016). Previous OpenIE work has concentrated on raw text, with the aim to extract "open" triples from natural language sentences (Niklaus et al., 2018), with another line of work focused on extracting from webtables (Cafarella et al., 2008).

Semi-structured websites (e.g. IMDb) contain many pages displaying information in stand-alone fields in relatively consistent locations on each page, with entities and the relationship between them indicated via formatting features such as section headers and lists of key-value pairs. Figure 1 shows an example page and the triples it conveys. Semi-structured websites have recently been shown to be a rich target for IE; the Knowledge Vault large-scale web extraction experiment, which extracted from semi-structured websites, natural language text, webtables, and Semantic Web annotations, found that semi-structured websites contributed 75% of total extracted facts and 94% of high-confidence extractions (Dong et al., 2014); the Ceres system showed that one can automatically extract from semi-structured sites with a precision over 90% using distant supervision (Lockard et al., 2018). These works, however, all build on the tradition of semi-structured ClosedIE techniques (Kushmerick et al., 1997; Soderland, 1999; Gulhane et al., 2011; Furche et al., 2014).

Interestingly, we are not aware of any work that applies OpenIE on semi-structured sources, despite the great potential to identify new relationships and new knowledge triples. We investigated 8 movie websites from the Structured Web Data Extraction (SWDE) corpus (Hao et al., 2011) and found that the IMDb ontology can cover only 7% of semantically unique predicates on these sites.

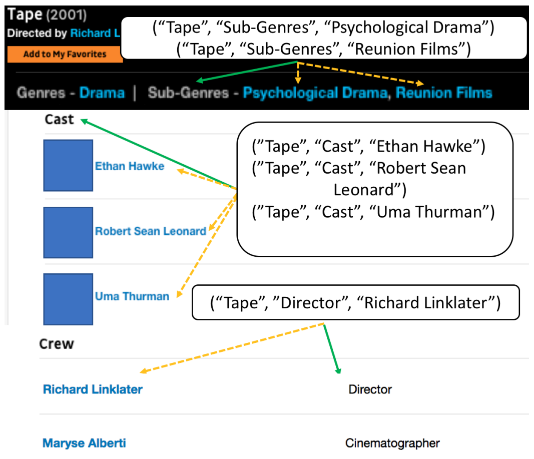
The major challenges that distinguish natural language text and semi-structured data are the basic unit and the inherent structure of the data. In natural language text, each sentence is a unit; it typically consists of a subject, a verb, and an object, which corresponds naturally to the subject, predicate, and object of a knowledge triple. Similarly, in webtables, each table is a unit; its rows, columns, and cells also naturally correspond to subjects, predicates, and objects in triples.

In the semi-structured setting, the basic unit is the webpage, which may contain hundreds or thousands of entity mentions. There is no fixed layout between the subject entity, object entity, and their relation, which may be far apart on the webpage. For example, Figure 1 contains object strings that are below, to the left, and to the right of their corresponding predicates; even trickier, for object string "Uma Thurman", the correct predicate string "Cast" is much farther away than the incorrect one "Crew". Despite the challenges, semi-structured pages do provide inherent visual structure to help distinguish the subject of a page, and *(predicate, object)* pairs for the subject. In this paper we answer the following question: *Given semi-structured webpages in a website, how can we tell which field contains the subject, and which fields contain the (predicate, object) pairs for the subject through any visual or DOM-structured clue?*

This paper makes three contributions. Our first contribution is to formally define a new problem of OpenIE from semi-structured websites (Section 2). We created a benchmark[1] for this problem by enhancing the SWDE corpus (Hao et al., 2011); our benchmark contains a high accuracy and coverage set of ground truth extractions for 21 websites spanning three domains, comprising 855,748 labels across 27,641 pages (Section 4).

Our second contribution is OpenCeres, a solution for OpenIE on semi-structured websites (Section 3). Our solution is novel in three aspects. First, whereas ClosedIE techniques on semi-structured data focus on extracting objects for given predicates, we also identify predicate strings on the website that represent the relations. Second, while ClosedIE techniques can only learn extraction patterns for predicates where there exists seed knowledge, we identify unseen predicates by applying semi-supervised label propagation. Third, whereas most existing extraction techniques on semi-structured sites leverage only DOM patterns as evidence, we use visual aspects of the webpage

[1]https://homes.cs.washington.edu/~lockardc/expanded_swde.html



("Tape", "Sub-Genres", "Psychological Drama")
("Tape", "Sub-Genres", "Reunion Films")

("Tape", "Cast", "Ethan Hawke")
("Tape", "Cast", "Robert Sean Leonard")
("Tape", "Cast", "Uma Thurman")

("Tape", "Director", "Richard Linklater")

Figure 1: A cropped portion of the detail page from allmovie.com for the film *Tape* with some triples indicated. Solid green and dashed yellow arrows indicate predicate strings and objects respectively.

for distant supervision and label propagation.

Our final contribution is a comprehensive evaluation on our new benchmark dataset and online websites (Section 5). Our proposed method obtained an F1 of 0.68, significantly higher than baseline systems, while extracting 7 times as many predicates as were present in the original ontology. In addition, we evaluate on a set of 31 movie websites, yielding 1.17 million extractions at a precision of 0.70. Our results inspire new directions for improvement as discussed in Section 7, and serve as a good baseline for future work.

2 Overview

2.1 Problem Definition

We propose the problem of OpenIE from semi-structured websites. A *semi-structured website W* consists of a set of *detail pages* that each contains information about a particular entity, the *topic entity* of the page. This information is typically populated from an underlying database into an HTML template to create the detail pages. The goal of semi-structured OpenIE is to recover all *(subject, predicate, object)* triples represented in these templatized fields, including the extraction of the string that semantically represents each predicate.

Relation objects are sometimes present without an explicit predicate string defining the relationship; since OpenIE requires extraction of a predicate string, we only consider the case where a meaningful predicate string exists on the page.

Definition 2.1 (Semi-Structured OpenIE) *Let W be a semi-structured website, with each page*

w_i containing facts about a topic entity t_i. Semi-Structured OpenIE extracts a set of triples from W such that the subject, predicate, and object of each triple is a string value on a page in W, with the subject representing t_i and the predicate string semantically representing the relation between the subject and object as asserted by the webpage.

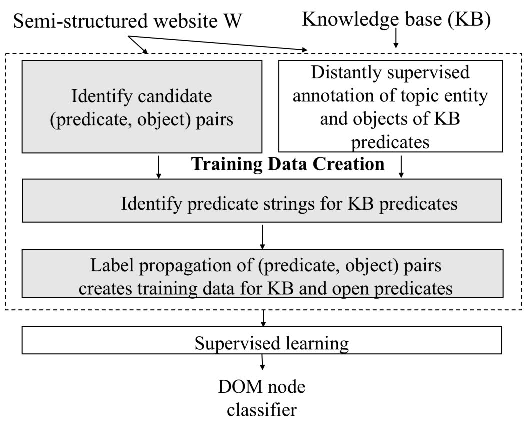
Following the tradition of relation extraction, which considers only binary relationships, we do not consider extraction of compound value types (CVTs) (Freebase, 2018), which express multi-way relationships. In this work, we narrow our focus to Semi-structured OpenIE from a given domain, since we will rely on pre-existing knowledge about that domain to provide us with seed annotations. We leave the extension to the general semi-structured OpenIE problem for future work.

2.2 From ClosedIE to OpenIE

We first summarize the Ceres techniques proposed in (Lockard et al., 2018), which is the state-of-the-art for ClosedIE from semi-structured websites. Ceres learns a model capable of generalizing across variations in a website from training labels automatically generated by the distant supervision technique. The automatic annotation contains two steps. First, *topic annotation* annotates the topic name on the page. Second, *relation annotation* annotates each object field, where the relation is guessed as a relationship in the seed ontology that is valid between the topic and the object.

OpenIE needs to go beyond existing relations in the ontology, identifying relations not existing in seed knowledge. As such, it raises two challenges for the relation annotation step. First, in addition to annotating the objects, we also need to be able to identify the predicate fields in order to extract predicate strings. Second, in addition to annotating the predicates already in the seed knowledge, we also need to identify new predicates on a webpage.

Figure 2 shows the infrastructure of our OpenIE solution, OpenCeres. We propose a relation annotation method that is suitable for OpenIE (shown in the shaded blocks), and inherit other components from Ceres (Lockard et al., 2018). Our key intuition to solve this problem is that different predicates often share some visual features, such as being aligned vertically or horizontally, sharing the same font, size or color, and so on. Thus, if we can identify at least one *(predicate, object)* pair on the page, we can look for other similarly formatted pairs of nodes and assume that they also represent *(predicate, object)* pairs. Accordingly, we propose a three-stage process that combines distant supervision and semi-supervised learning.



Semi-structured website W
Knowledge base (KB)
Identify candidate (predicate, object) pairs
Distantly supervised annotation of topic entity and objects of KB predicates
Training Data Creation
Identify predicate strings for KB predicates
Label propagation of (predicate, object) pairs creates training data for KB and open predicates
Supervised learning
DOM node classifier

Figure 2: An overview of our proposed semi-structured OpenIE model learning process. Shaded areas indicate contributions of our process.

1. Predicate-object candidate generation: We first generate potential candidate *(predicate, object)* pairs, as described in Section 3.1. The search for these candidate pairs is quasilinear in the number of DOM nodes, thereby avoiding examination of every pair of DOM nodes.
2. Distant supervision: We then use a seed knowledge base (KB) to identify instances of *(predicate, object)* pairs appearing in the seed, where the predicate exists in a pre-defined ontology, as described in Section 3.2. For example, for the page in Figure 1, we would hope to identify *("Director", "Richard Linklater")* assuming that fact was in our KB.
3. Semi-supervised label propagation: We perform a semi-supervised label propagation step to identify pairs of nodes that are formatted similarly to the known *(predicate, object)* pairs as described in Section 3.3; these new *(predicate, object)* pairs give us training labels for new predicates. For example, in Figure 1, we should identify the pair *("Cinematographer", "Maryse Alberti")* since it is formatted similarly to our seed pair, even though the concept of *cinematographer* does not exist in our seed ontology.

3 Approach

We now describe the three key steps we use to generate training labels.

3.1 Candidate pair generation

Recall that for ClosedIE we only need to annotate objects; for OpenIE we need to additionally identify the predicates, which will then allow us to find other predicates not found in the seed KB. Our first step is thus to find all potential *(predicate, object)* pairs on a webpage for further annotation.

Determining which nodes should be checked as potential predicates for a given object is not trivial. On the one hand, there may be hundreds of nodes on the page, so considering all potential pairs of nodes would be computationally expensive. On the other hand, consider that a webpage about a movie may contain a long list of actors under the section header "Cast"; in Euclidean distance, the actor at the bottom of the list may be quite far away from the "Cast" string at the top of the list, so only searching nearby nodes may miss the predicate node.

To identify potential predicate nodes, we use our intuition that predicate strings should be more common across a website than their related object strings. Consider *("Language", "English")* as an example. Even though the object string "English" might be quite common on a site, it should be less frequent than its corresponding predicate string "Language". According to the site-wide frequency rankings, we consider a *(predicate, object)* pair a candidate pair only if the predicate appears more frequently than the object[2].

Following this intuition, we start by computing the frequency of each string found across all pages in the website, and create a ranked list. Second, for each DOM node, we create candidate pairs consisting of the node as object and the k-nearest higher ranking nodes as predicate . Third, to identify potential predicates that may be farther away but still represent a section header for the region of the page containing the object, we recursively travel up the DOM tree, and at each level we find the k-highest ranked candidate predicates paired with any candidate object in that DOM subtree. We create additional candidate pairs pairing those candidate predicates with all candidate objects in that subtree. Thus in total each candidate object is paired with up to dk candidate predicates, where d is the depth of the DOM.

For example, consider the object strings in the Cast section of Figure 1, with a k value of 1. Since "Cast" appears on every page, an initial candidate pair would be created for ("Cast", "Ethan Hawke"). If Hawke is mentioned more frequently across the site than the other actors, he would be paired as the potential predicate for their strings since they are closer to his name than "Cast", such as ("Ethan Hawke", "Robert Sean Leonard"). However, since Hawke and Leonard are in the same `<div>`, the recursive process would add the candidate pair ("Cast", "Robert Sean Leonard") since "Cast" would be the most highly ranked string associated with any object in that section.

[2]We consider strings consisting of numeric values only as potential objects and not predicates, regardless of their frequency.

3.2 Seed labels

In the second stage, given a webpage, the subject and objects we have identified on the page, and the candidate *(predicate, object)* pairs, we are now ready for distant supervision annotation to generate seed labels that are *(predicate, object)* pairs for the subject appearing in the seed knowledge.

We start with the Ceres object identification to generate a list of nodes containing object strings corresponding to KB predicates, and look up *(predicate, object)* pairs in the candidate list that contain the object node. We use lexical clues to we filter a candidate *(predicate, object)* pair if the predicate name is not semantically similar to the predicate in the ontology. There are multiple ways of doing this. One way is to compare the cosine similarity of word embeddings (such as FastText (Bojanowski et al., 2017)) representing the predicate string and the ontology predicate name and filter using a threshold. Another way is to manually compile a few terms for each predicate in our ontology, and filter a predicate if it does not contain any of the terms as a substring. Empirically we found using a manually compiled list, which takes about a minute per predicate, gives higher precision than using embeddings, though it limits us to the particular language of those terms. After the filtering step, we can fairly safely choose the *(predicate, object)* pair where the predicate is closest to the object in Euclidean distance.

3.3 Semi-supervised Label Propagation

In the third stage, given a set of *(predicate, object)* pairs on a webpage generated in the first stage, we aim at following visual clues to find other *(predicate, object)* pairs on the same page. These new candidate pairs serve as training labels for predicates that may not occur in the seed knowledge.

We apply semi-supervised learning, which typically resorts to a similarity graph where similar instances are connected by edges in the graph, and propagates existing labels to neighbor nodes. Our intuition is that *(predicate, object)* pairs should share similar formatting; we capture this intuition as we construct the graph.

Graph construction: Each vertex in the similarity graph represents a candidate pair, an edge connecting two vertices indicates that the two candidate pairs are similar, and the edge weight gives the level of similarity. We compute similarity between candidate pairs by visual clues[3], creating an edge between them if they have similar predicate formatting and simliar object formatting. Formatting similarity requires having the same font, font size, font weight, and text alignment, and being either vertically or horizontally aligned.

We then weight the edges by adding up similarities of the horizontal, vertical, and DOM relationship between predicate and object. Similarity of DOM relationship is 1 for exact match and 0 otherwise. Similarity of horizontal relationship is computed by measuring the distance between the predicate and the object in a (pred, obj) pair, and then taking the ratio of minimum distance and maximum distance[4]. We compute similarity of vertical relationship in a similar way, giving:

$$w_{i,j} = \mathbb{1}_{r_d(i)=r_d(j)} + \max\left(0, \frac{\min\left(r_h\left(i\right), r_h\left(j\right)\right)}{\max\left(r_h\left(i\right), r_h\left(j\right)\right)}\right) + \max\left(0, \frac{\min\left(r_v\left(i\right), r_v\left(j\right)\right)}{\max\left(r_v\left(i\right), r_v\left(j\right)\right)}\right)$$

where i and j are candidate pairs, r_d calculates the DOM path, r_h calculates the horizontal distance between candidate predicate and candidate object, and r_v calculates the vertical distance.

A sample graph for the webpage in Figure 1 is shown in Figure 3. The pair ("Director", "Richard Linklater") is connected to the pair ("Cinematographer", "Maryse Alberti") with a weight of 3, since they have identical values for all three relationships, while ("Sub-Genres:", "Psychological Drama") and ("Sub-Genres", "Reunion Films") have an edge weight of 2.1 since the latter's horizontal distance is ten times greater than the former.

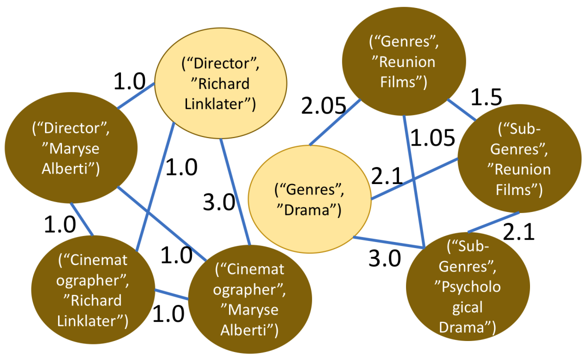

("Director", "Richard Linklater")
("Genres", "Reunion Films")
("Director", "Maryse Alberti")
("Sub-Genres", "Reunion Films")
("Genres", "Drama")
("Cinematographer", "Richard Linklater")
("Cinematographer", "Maryse Alberti")
("Sub-Genres", "Psychological Drama")
1.0
2.05
1.5
1.05
1.0
3.0
2.1
1.0
2.1
1.0
3.0
1.0

Figure 3: A portion of the graph corresponding to the webpage in Figure 1. Lighter nodes indicate seed pairs labeled by the Ceres process.

To speed up propagation, we keep only the 10 top-weighted edges for each pair. On average, on the dataset in Section 4, pages have 1,142 text fields resulting in 2,813 candidate pairs connected by 14,733 edges, far less than the 1.3 million candidate pairs (and corresponding increase in edges) that would result from a naive pairwise matching.

Label propagation: We use the MultiRankWalk label propagation algorithm (Lin and Cohen, 2010), which has been shown to be successful in very low-data situations. This allows us to propagate even when we only have a single seed label on a page. MultiRankWalk adapts the Personalized PageRank (PPR) algorithm for a classification setting, conducting a PageRank run for each class, with the personalization vector set to equally divide the restart probability among positive labeled examples of the class. The PageRank runs are conducted over the weighted graph constructed in the prior step. Each unlabeled vertex is assigned to the class whose PageRank run gives it the highest score. In our case we have two PPR runs: a positive run for labeled *(predicate, object)* candidates and a negative run for unlabeled candidates.

The results of this process are then used as training data to train a supervised Ceres (Lockard et al., 2018) extraction model.

4 Extended SWDE Benchmark

The Structured Web Data Extraction (SWDE) dataset has served as a benchmark for semi-structured web extraction, with webpages and ground truth extractions from 10 websites in each of 8 domains (Hao et al., 2011). However, the

[3]To harvest these features, we render the page using the headless Chrome browser and access element attributes with Selenium (https://www.seleniumhq.org/).

[4]In practice, there are multiple ways to calculate horizontal distance: Left side to left side, left side to right side, right to right, and right to left. The same holds for vertical distance. We calculate each possible ratio and use the one that gives the highest weight. In the case that the ratio is negative (e.g. one pair had predicate to the left of the object while the other pair had it to the right), we set it to 0.

ground truth in SWDE only covers a subset of the predicates found on each site, typically 3-4 predicates per domain.

We extend it as follows: Of the 8 domains, we kept 4 domains whose page topics are named entities. We extended their gold set to include extractions identifying all key-value semi-structured fields on the websites. Since not all SWDE websites can still be rendered in the browser (due to missing resources), we eliminated websites that we were unable to successfully render in the Chrome browser, resulting in 30 websites. We then attempted to create ground truth via a combination of wrapper induction based on manually-labeled training data (with an extractor implementation based on (Gulhane et al., 2011)), hand-crafted extraction rules, and manual cleanup of remaining errors. Eventually, we generated accurate labels for 21 sites in 3 domains.

This new extended benchmark includes both extracted object values as well as the predicate string that accompanies each value on the page. The statistics of the augmented dataset are shown in Table 1. We enhanced SWDE in two ways. First, SWDE on average contains 4,480 triples for 3 predicates from these 21 websites, whereas we have an average of 41K triples for 36 predicates. The number of predicates per website ranges from 5 to 272 (Hollywood features very fine-grained relationships like "Assistant Art Director"). Second, when multiple predicate strings may apply on the webpage, we list all of them in order of specificity. Taking Figure 1 as an example, we include both "Director" and "Crew" for a relation, considering the former to be more specific. To our knowledge, this is the first dataset that represents all key-value pairs found in semi-structured web data.

5 Experimental Evaluation

5.1 Experimental Setup

Datasets: Our primary dataset is the augmented SWDE corpus described in Section 4. In addition, we used the set of 31[5] movie websites (comprising 433,000 webpages) found in CommonCrawl[6] from Lockard et al. (2018). To generate seed KBs for the distant supervision, we relied on the methodology from Lockard et al. (2018), using the IMDb database for the Movie domain, and using the original SWDE ground truth for websites CollegeBoard and ESPN to create a KB for the University and NBAPlayer domains respectively.

[5]We removed the two sites on which Lockard et al. (2018) reported Ceres made no annotations.

[6]www.commoncrawl.org

Domain	Site	# Predicates	# Labels
Movie	AllMovie	65	104,303
Movie	AMCTV	20	85,916
Movie	Hollywood	272	77,047
Movie	iHeartMovies	9	21,253
Movie	IMDb	36	152,880
Movie	Metacritic	18	43,450
Movie	RottenTomatoes	12	65,524
Movie	Yahoo	12	28,354
University	CollegeProwler	26	40,707
University	ECampusTours	17	18,448
University	Embark	67	46,431
University	MatchCollege	68	107,763
University	USNews	22	21,269
NBAPlayer	ESPN	22	6,757
NBAPlayer	FanHouse	16	6,656
NBAPlayer	FoxSports	15	6,157
NBAPlayer	MSNca	12	5,208
NBAPlayer	SI	12	6,082
NBAPlayer	Slam	13	5,453
NBAPlayer	USAToday	5	2,178
NBAPlayer	Yahoo	9	3,912

Table 1: Statistics of the augmented SWDE dataset.

Implementations: We compared OpenCeres with two baselines. The three algorithms apply the same method to extract topic subjects but differ in how they extract *(predicate, object)* pairs.

1. *WEIR:* Proposed by Bronzi et al. (2013), the Web Extraction and Integration of Redundant data (WEIR) approach takes as input a set of websites in the same subject domain and makes use of overlap in observed entities across sites to learn extraction rules for predicates. The system is unsupervised, though it does require a dictionary of potential page topic entities for the domain to align pages between sites. WEIR also contains a method for automatically identifying predicate strings for the extraction rules it learned by finding strings that frequently occur nearby extracted objects in the HTML templates of sites in the domain.
2. *Colon Baseline:* Semi-structured pages frequently represent a *(predicate, object)* pair via a set of adjacent DOM nodes, with the predicate string ending in a colon and the object string either to its right or below it. This baseline starts with the *(predicate, object)* candidate pairs generated in Section 3.1,

identifies those where the predicate field ends with a colon, and extracts them as a predicate along with their closest candidate object either to their right or below.

3. *OpenCeres:* This implements our system exactly as described in Section 3, using the generated training data to train a Ceres extractor.

In addition, to understand the uper bound of OpenCeres, we implemented two versions using ground truth data for training seeds:

4. *OpenCeres-Gold:* This implements our system, but skips the label propagation step and replaces noisy seed labels (Section 3.2) with samples from ground truth triples. We sampled 25% of triples for each predicate, so this method is essentially ClosedIE Ceres with incomplete but clean training labels, giving an upper bound on the system's performance when no errors are introduced during training data generation and label propagation.
5. *OpenCeres-GoldProp:* This implements OpenCeres-Gold, but adds the label propagation step described in Section 3.3. Rather than sampling 25% of ground truth triples from all predicates, we instead sample $p\%$ of ground truth predicates for a site (with p varying from 10 to 100) and then sample 25% of the corresponding triples for each page. The process is run five times for each setting of p and the results are averaged.

Evaluation: Evaluation is tricky for semi-structured OpenIE because a page may contain multiple valid predicates for a relation. Recall that the SWDE benchmark data we generated (Section 4) lists all predicate strings that are valid, ranked in their order of specificity. We thus define two scores for an extracted triple.

- A *strict score* that requires an exact match between the extracted predicate and the most-specific predicate string in the ground truth.
- A *lenient score* that counts an extraction as correct if the extracted predicate matches any of the predicate strings in the ground truth.

For the SWDE dataset, where we have complete ground truth, we compute precision, recall, and F1. For the CommonCrawl dataset, where no ground truth exists, we sampled 500 extractions at each confidence threshold (giving a 4% margin of error) and manually scored them; since we cannot calculate true recall, we report precision and yield.

System	**Movie**		**NBA**		**University**	
	P	R	P	R	P	R
WEIR (Bronzi et al., 2013)	0.23	0.17	0.08	0.17	0.13	0.18
Colon Baseline	0.63	0.21	0.51	0.33	0.46	**0.31**
OpenCeres	**0.77**	**0.68**	**0.74**	**0.48**	**0.65**	0.29
OpenCeres-Gold	0.99	0.74	0.98	0.80	0.99	0.60

Table 2: Extraction precision and recall (lenient) on SWDE domains. OpenCeres on average improves over baseline by 36% on precision and by 88% on recall.

5.2 Results on SWDE

Overall results: Table 2 shows the precision and recall obtained via lenient scoring. Our results show that OpenCeres outperformed both baselines, achieving an average precision of 72% across the three domains, with an average recall of 48%. Comparing with OpenCeres-Gold on Movie, our precision is 22% lower, while recall is only 5% lower, showing that our label propagation is fairly effective in preserving recall, but introduces errors reducing precision. WEIR does not perform as well as ColonBaseline, showing that our *(predicate, object)* candidate identification technique works well.

Our recall is a robust 68% in the Movie domain, but is much lower in the other two domains. This is because we failed to make any extraction in 3 of the 5 University sites and 2 of the 8 NBA sites due to the inability to find a predicate string for the seed predicates. In some cases no predicate string existed, but in others the string was not in our lexicon. In fact, if we skip those websites where we extract nothing, our recall increases to 58% for NBA and 44% for University. Other recall misses occur when a page has some semi-structured fields that differ significantly in format from those found in our seed ontology, so they were too dissimilar for the label propagation to extend to them.

Details: We now deep dive to results of OpenCeres, shown in Table 3. First, our scoring under the "strict" rules is only slightly lower than under "lenient" rules, because the case that multiple predicates apply is not common and we are often able to find the most specific ones. Across all triples, the overall lenient F1 is 0.68 and strict F1 is 0.61. Second, at predicate-level, OpenCeres has an average precision of 74% and recall of 39%, showing that our method attains high precision for the new predicates it identifies. Third, through the label propagation technique, we are able to extract an average of 10.5 new predicates for every predicate in our seed ontology.

A sample of 100 erroneous OpenCeres extractions shows that 33% of errors are due to the presence of CVTs on the page. For example, the movie site Rotten Tomatoes contains a “Full Review” predicate that contains review date, writer, publication, and text; we extracted only the date, which arguably is not useful. Considering these extractions as correct will increase the precision to 81%. Among the errors, 22% were due to incoherent predicates such as “See More”, while 20% were due to incorrect extraction of a template string as an object of a correct predicate.

Label propagation: Figure 4 shows how the label propagation process successfully creates new training examples from a small number of seeds. While propagation does introduce some precison errors, when only 10% of predicates are given to OpenCeres-GoldProp as seeds (and only 25% of triples sampled for each predicate), training data recall is already nearly 50%. As the percentage of seed predicates rises, the seeds become more likely to capture the full variety of predicate formatting, and recall rises.

There are a number of reasons why the recall upper bound demonstrated by OpenCeres-Gold (and OpenCeres-GoldProp) is less than perfect. First, a small number of relations in the dataset have predicate or object strings that span multiple text fields (particularly in the University vertical); the implemented system can only extract a single text field, so these will be missed. Second, the Candidate Pair Identification algorithm has imperfect recall. Finally, because only 25% of ground truth triples were used for each page of training data, some true positive examples were sampled as negative examples for training, thereby lowering classification recall.

Parameter setting: Table 4 shows that Candidate Pair Identification has increasing recall in capturing true candidate pairs in the SWDE-Movie vertical with more neighbors considered, with a tradeoff in increased runtime due to the creation of more pairs; we used $k = 5$ in our experiments.

5.3 Results on CommonCrawl

We now report results of ClosedIE and OpenIE extractions on the 31 CommonCrawl websites; the ClosedIE implementation is a subset of the OpenIE system, without the shaded components in Figure 2. Of these 31 websites, we successfully extracted from 22 sites using OpenIE, and failed to extract from 9 sites because of our inability to match their predicate strings to our lexicon for seed predicates (4 sites were in foreign languages while our lexicon is in English, on 3 sites the pages had no predicate strings labeling the seed object, and 2 sites used terms outside our lexicon).

Figure 5 shows the precision-yield curve of our ClosedIE and OpenIE extractions as we vary the confidence threshold. At a 0.5 confidence threshold, we extracted 2.3M triples at a precision of 0.58, where 1.17M (51%) have new predicates. A higher threshold of 0.8 yielded 1.17M extractions at a precision of 0.70, with 50% of extractions representing new predicates. The high percentage of extractions with new predicates shows the big promise of our method in enriching existing knowledge bases not only with new entities and new facts, but also with new relationships.

	Movie	**NBA Player**	**University**
Triple-level F1	0.72 (0.65)	0.58 (0.58)	0.41 (0.36)
Pred-level Prec	0.55 (0.52)	0.86 (0.86)	0.81 (0.76)
Pred-level Rec	0.35 (0.32)	0.46 (0.46)	0.37 (0.35)
Pred-level F1	0.43 (0.40)	0.60 (0.60)	0.51 (0.48)
New:Existing-pred ratio	4.4 : 1	4.3 : 1	23.0 : 1

Table 3: Detailed results of OpenCeres using lenient scoring, with strict scoring results shown in parentheses.

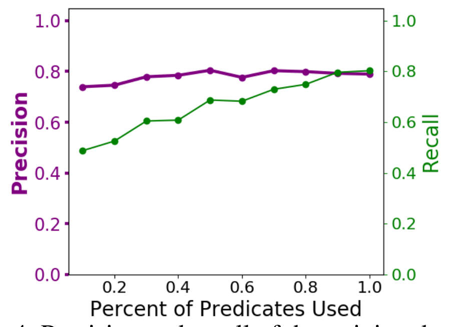


Figure 4: Precision and recall of the training data automatically created for the Movie domain by OpenCeres-GoldProp, after label propagation from seed data created by first sampling varying percents of the ground truth predicates for a site and then sampling a constant 25% of ground truth objects for each predicate.

6 Related Work

In unstructured text, OpenIE was originally proposed by Banko et al. (2007), an approach extended by ReVerb (Fader et al., 2011) and Ollie (Mausam et al., 2012), which relied on syntactic

k	# candidates	recall
3	1,863	0.92
7	3,044	0.95
11	4,104	0.98

Table 4: Average number of candidate pairs produced by considering k-nearest higher ranking text fields for each candidate object on the SWDE-Movie dataset, along with recall over true pairs.

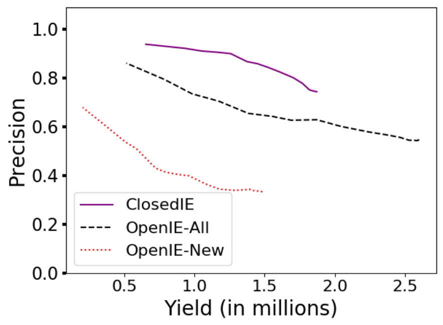


Figure 5: Extraction precision vs. number of extractions on the CommonCrawl dataset at various confidence thresholds; ClosedIE is the implementation of the Ceres system. OpenCeres-All shows all extractions from OpenCeres, while OpenCeres-New shows only the OpenCeres extractions with new predicates.

constraints to identify relation patterns. Our approach is influenced by Wu and Weld (2010), who aligned Wikipedia infobox contents to article text to automatically create training data for an extractor. Recent work on neural extraction models (Cui et al., 2018) has explored entirely supervised models learned from a modified version of the QA-SRL dataset (Stanovsky et al., 2018).

The line of research that has most closely examined the prospect of OpenIE-style extractions using webpage structure is the work on Webtables (Cafarella et al., 2008; Dalvi et al., 2012; Balakrishnan et al., 2015; Cafarella et al., 2018). This work specifically examines the identification of subjects, predicate, and object strings but is limited to fields in rows and columns created using HTML `<table>` tags. Extractions from webtables have recently been harnessed as a source of facts for question-answering systems (Pasupat and Liang, 2015; Krishnamurthy et al., 2017).

In extraction from semi-structured websites, the traditional approach is *wrapper induction*, in which a rule learning algorithm is applied to a set of labeled training examples (Kushmerick et al., 1997). Work in this line of research has achieved high accuracy from only a few labeled examples, but requires manually-annotated examples for each website (Gulhane et al., 2011). To remove this bottleneck, researchers have explored alternative ways to automatically create labeled data and learn models from such potentially noisy labels (Dalvi et al., 2011; Gentile et al., 2015; Furche et al., 2014; Lockard et al., 2018). However, these approaches cannot find triples for predicates that are not in the seed ontology.

The Roadrunner project (Crescenzi et al., 2001) does attempt to identify the objects of all relations represented on a site, but does not extract predicate strings. However, the WEIR project (Bronzi et al., 2013) extended this framework with a heuristic to harvest predicate strings based on words found in DOM nodes that form part of the path of the learned XPath extraction rule. This is the first work that could truly be considered an OpenIE approach to semi-structured extraction. However, as we show in our experiments, the constraints of their heuristic limit the recall of this approach.

7 Conclusions

We presented a new problem of Open Information Extraction from semi-structured websites, and are releasing a new set of over 855,000 ground truth extractions for 21 websites available in the SWDE corpus. We also proposed an algorithm for OpenIE that employs semi-supervised label propagation to discover new predicates based on a set of seed predicates in a known ontology. This method attained a 68% F1 score in OpenIE extractions on our benchmark. In addition, a large-scale evaluation on 31 CommonCrawl movie websites yielded extractions of over two million triples.

In the future, we would like to improve extraction by training a model to extract *(predicate, object)* pairs directly without having to train on particular predicates. Such a model could potentially be based on visual clues common across websites, so a single model could be applied to many sites.

Acknowledgments

We wish to thank Hannaneh Hajishirzi for helpful advice, Paolo Merialdo and Valter Crescenzi for help in running WEIR, and Andrew Bridges, Marc Landers, and Alexander Macdonald for their help in annotating data.

References

Sreeram Balakrishnan, Alon Y. Halevy, Boulos Harb, Hongrae Lee, Jayant Madhavan, Afshin Rostamizadeh, Warren Shen, Kenneth Wilder, Fei Wu, and Cong Yu. 2015. Applying webtables in practice. In *CIDR*.

Michele Banko, Michael J. Cafarella, Stephen Soderland, Matthew G Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *IJCAI*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *TACL*, 5:135–146.

Mirko Bronzi, Valter Crescenzi, Paolo Merialdo, and Paolo Papotti. 2013. Extraction and integration of partially overlapping web sources. *PVLDB*, 6:805–816.

Michael J. Cafarella, Alon Y. Halevy, Hongrae Lee, Jayant Madhavan, Cong Yu, Daisy Zhe Wang, and Eugene Wu. 2018. Ten years of webtables. *PVLDB*, 11:2140–2149.

Michael J. Cafarella, Alon Y. Halevy, Daisy Zhe Wang, Eugene Wu, and Yonghui Zhang. 2008. Webtables: exploring the power of tables on the web. *PVLDB*, 1:538–549.

Valter Crescenzi, Giansalvatore Mecca, Paolo Merialdo, et al. 2001. Roadrunner: Towards automatic data extraction from large web sites. In *VLDB*, volume 1, pages 109–118.

Lei Cui, Furu Wei, and Ming Zhou. 2018. Neural open information extraction. In *ACL*.

Bhavana Bharat Dalvi, William W. Cohen, and James P. Callan. 2012. Websets: extracting sets of entities from the web using unsupervised information extraction. In *WSDM*.

Nilesh N. Dalvi, Ravi Kumar, and Mohamed A. Soliman. 2011. Automatic wrappers for large scale web extraction. *PVLDB*, 4:219–230.

Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge vault: a web-scale approach to probabilistic knowledge fusion. In *KDD*.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *EMNLP*.

Freebase. 2018. Compound value types in Freebase API.

Tim Furche, Georg Gottlob, Giovanni Grasso, Xiaonan Guo, Giorgio Orsi, Christian Schallhart, and Cheng Wang. 2014. Diadem: Thousands of websites to a single database. *PVLDB*, 7:1845–1856.

Anna Lisa Gentile, Ziqi Zhang, and Fabio Ciravegna. 2015. Early steps towards web scale information extraction with lodie. *AI Magazine*, 36:55–64.

Pankaj Gulhane, Amit Madaan, Rupesh R. Mehta, Jeyashankher Ramamirtham, Rajeev Rastogi, Sandeepkumar Satpal, Srinivasan H. Sengamedu, Ashwin Tengli, and Charu Tiwari. 2011. Web-scale information extraction with vertex. *ICDE*, pages 1209–1220.

Qiang Hao, Rui Cai, Yanwei Pang, and Lei Zhang. 2011. From one tree to a forest: a unified solution for structured web data extraction. In *SIGIR*.

Jayant Krishnamurthy, Pradeep Dasigi, and Matt Gardner. 2017. Neural semantic parsing with type constraints for semi-structured tables. In *EMNLP*.

Nicholas Kushmerick, Daniel S. Weld, and Robert B. Doorenbos. 1997. Wrapper induction for information extraction. In *IJCAI*.

Frank Lin and William W. Cohen. 2010. Semi-supervised classification of network data using very few labels. *2010 International Conference on Advances in Social Networks Analysis and Mining*, pages 192–199.

Colin Lockard, Xin Dong, Prashant Shiralkar, and Arash Einolghozati. 2018. Ceres: Distantly supervised relation extraction from the semi-structured web. *PVLDB*, 11:1084–1096.

Mausam. 2016. Open information extraction systems and downstream applications. In *IJCAI*.

Mausam, Michael Schmitz, Stephen Soderland, Robert Bart, and Oren Etzioni. 2012. Open language learning for information extraction. In *EMNLP-CoNLL*.

Christina Niklaus, Matthias Cetto, André Freitas, and Siegfried Handschuh. 2018. A survey on open information extraction. In *COLING*.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *ACL*.

Stephen Soderland. 1999. Learning information extraction rules for semi-structured and free text. *Machine Learning*, 34:233–272.

Gabriel Stanovsky, Julian Michael, Luke S. Zettlemoyer, and Ido Dagan. 2018. Supervised open information extraction. In *NAACL-HLT*.

Fei Wu and Daniel S. Weld. 2010. Open information extraction using wikipedia. In *ACL*.